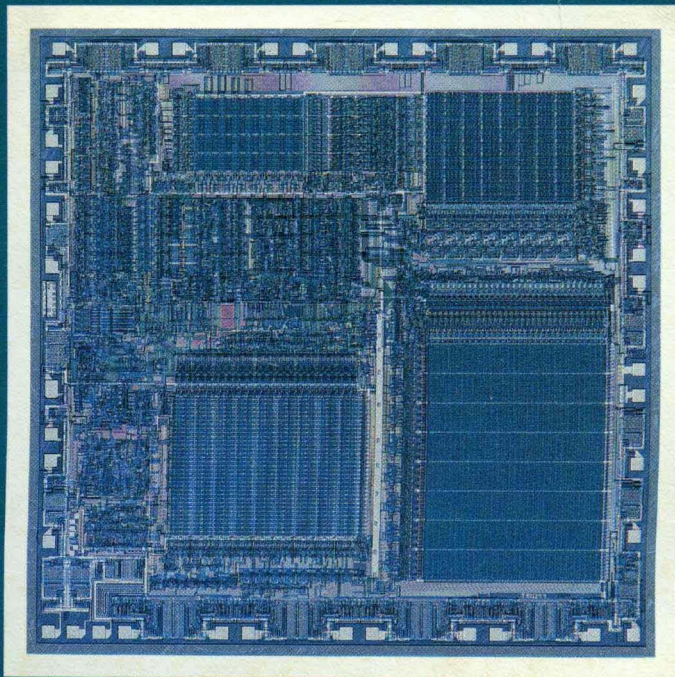


RCA LSI Products — Applications



- **Microprocessors** • **Memories** • **Peripheral Circuits**
- **Microboards** • **Microsystems**

RCA LSI Products — Applications

The RCA 1800 series of CMOS LSI products—microprocessors, memories, and peripheral support circuits—are used in many widely diverse applications covering a broad range of end markets. This series may be used, in some cases in combination with other industry-compatible peripherals, to provide the complete component requirements for a host of exciting and innovative systems, limited only by the ingenuity of the system designer. The 1800 series also encompasses a fully coordinated line of ready-to-use development and support systems (hardware and software) and Microboard computer modules for designers of microprocessor-based equipment.

RCA provides extensive literature support for the 1800-series products. Technical data bulletins contain detailed descriptive information on each type. Application notes provide helpful user information on most major types. This book is a collection of the current application notes on 1800-series products. Other technical literature used to support the 1800 series, including a comprehensive DATABOOK, system product descriptions, hardware and software user manuals, and product selection guides, are listed at the end of this book.

The application notes on 1800-series products describe specific hardware and software interface schemes of memory or peripheral components with an 1800-series central processing unit (CPU) and provide tutorial information on basic concepts, general device architecture and characteristics, and features and advantages of the 1800-series family. This information serves as a useful design guide to the system designer in determining the required hardware and software for his microcomputer-based design.

RCA Solid
State

Brussels • Buenos Aires • Hamburg • Madrid • Mexico City • Milan
Montreal • Paris • Sao Paulo • Somerville NJ • Stockholm
Sunbury-on-Thames • Taipei • Tokyo

Information furnished by RCA is believed to be accurate and reliable. However, no responsibility is assumed by RCA for its use; nor for any infringements of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent rights of RCA.

When incorporating RCA Solid State Devices in equipment, it is recommended that the designer refer to "Operating Considerations for RCA Solid State Devices", Form No. 1CE-402, available on request from RCA Solid State Division, Box 3200, Somerville, N.J. 08876.

Copyright 1982 by RCA Corporation
(All rights reserved under Pan-American
Copyright Convention)

Trademark(s)®Registered
Marca(s) Registrada(s)

Printed in USA/1-82

Table of Contents

	Page
Application Note Classification Chart	
By Category	4
By Device Type	4
Application Notes	
ICAN-6315 COS/MOS Interfacing Simplified	6
ICAN-6416 An Introduction to Microprocessors and the RCA COSMAC COS/MOS Microprocessor	13
ICAN-6525 Guide to Better Handling and Operation of CMOS Integrated Circuits	21
ICAN-6536 Use of CMOS ROM's CDP1831 and CDP1832 with the RCA Microprocessor Evaluation Kit CDP18S020	26
ICAN-6537 Use of CMOS RAM CDP1824 with Microprocessor Evaluation Kit CDP18S020	30
ICAN-6538 Use of the CDP1852 8-Bit I/O Port with RCA Microprocessor Evaluation Kit CDP18S020	33
ICAN-6539 Use of CMOS-SOS RAM CDP1822 with RCA Microprocessor Evaluation Kit CDP18S020	37
ICAN-6562 Register-Based Output Function for RCA COSMAC Microprocessors	40
ICAN-6565 Design of Clock Generators for Use with RCA COSMAC Microprocessor CDP1802	42
ICAN-6581 Power-On Reset/Run Circuits for the RCA CDP1802 COSMAC Microprocessor	45
ICAN-6595 Interfacing Analog and Digital Displays with CMOS Integrated Circuits	47
ICAN-6602 Interfacing COS/MOS with Other Logic Families	59
ICAN-6611 Keyboard Scan Routine for Use with COSMAC Microterminal CDP18S021	71
ICAN-6632 Use of the CDP1854 UART with RCA Microprocessor Evaluation Kit CDP18S020 or EK/Assembler-Editor Design Kit CDP18S024	74
ICAN-6635 Use of CMOS ROM's CDP1833 and CDP1834 with the RCA Microprocessor Evaluation Kit CDP18S020 and the EK/Assembler-Editor Design Kit CDP18S024	79
ICAN-6657 Use of the CDP1856 and CDP1857 Buffer/ Separators in CDP1802 Microprocessor Systems	82
ICAN-6677 Software Control of Microprocessor-Based Realtime Clock	85
ICAN-6693 CDP1802-Based Designs Using the 8253 Programmable Counter/Timer	94
ICAN-6704 Optimizing Hardware/Software Trade-Offs in RCA CDP1802 Microprocessor Applications	97
ICAN-6834 Microprocessor Control for Color-TV Receivers	107
ICAN-6842 16-Bit Operations in the CDP1802 Microprocessor	117
ICAN-6847 Programming 2732 PROM's with the CDP18S480 PROM Programmer	122
ICAN-6883 Simplified Design of Astable RC Oscillators Using the CD4060B or Two CMOS Inverters	125
ICAN-6889 Using Slower Memories with the VIS Display System	127
ICAN-6901 CDP1802 Microprocessor-Based Setback Thermostat	130
ICAN-6918 A Methodology for Programming COSMAC 1802 Applications Using Higher-Level Languages	137
ICAN-6925 Understanding and Using the CDP18U42 EPROM	141
ICAN-6928 Interfacing PLM Code to CDOS System Functions	149
ICAN-6934 Cassette Tape I/O for COSMAC Microprocessor Systems	155
ICAN-6943 Designing Minimum/Nonvolatile Memory Systems with CMOS Static RAM's	159
ICAN-6948 Parallel Clocking of Sequential CMOS Devices	184
ICAN-6953 An Introduction to the Video Interface System (VIS) Devices—CDP1869 and CDP1870	186
ICAN-6955 Using the COSMAC Microboard Battery-Backup RAM, CDP18S622	197
ICAN-6957 CDP1804 and CDP1805 Processors Improve System Performance and Lower Chip Count	202
ICAN-6968 New CMOS CDP1800-Series Processors Reduce Chip Count	208
ICAN-6970 Understanding and Using the CDP1855 Multiply/Divide Unit	212
ICAN-6971 New CMOS CDP1800-Series Processors Enhance System Performance	223
ICAN-6991 A Slave CDP1802 Serial Printer Buffer System	229
ICAN-7009 New CDP1805 Microprocessor Upgrades CDP1800-Based Systems	237
ICAN-7020 Multimicroprocessor-Based Transistor Test Equipment	243
ICAN-7023 CDP1800-Series Peripherals—Building Blocks of a Complete Processor Family	251
ICAN-7026 Microboard Equipment Control	258
ICAN-7029 Low-Power Techniques for Use with CMOS CDP1800-Based Systems	263
ICAN-7032 CDP1800-Based Video Terminal Using the RCA Video Interface System VIS	269
ICAN-7038 A CDP1800-Based CRT Controller	305
ICAN-7063 Understanding the CDP1851 Programmable I/O	313
ICAN-7067 VIS—A Commercially Competitive CRT Controller Chip Set	319
Other Related Technical Publications	323

Application Note Classification Chart

By Category

Category	Relevant Application Note
General	ICAN-6416 ICAN-6525 ICAN-6704 ICAN-6883 ICAN-6943 ICAN-6948 ICAN-6957 ICAN-6968 ICAN-6971 ICAN-7023 ICAN-7029
Complete System Designs	ICAN-6834 ICAN-6901 ICAN-6991 ICAN-7020 ICAN-7026 ICAN-7032
Microprocessor Interfacing	ICAN-6315 ICAN-6595 ICAN-6602 ICAN-6934
Software	ICAN-6677 ICAN-6704 ICAN-6918 ICAN-6928
Development Support Applications	ICAN-6611 ICAN-6847 ICAN-6955 ICAN-6928
Miscellaneous	ICAN-6693

By Device Type

Type	Relevant Application Note
CDP1802	ICAN-6562 ICAN-6565 ICAN-6581 ICAN-6677 ICAN-6842 ICAN-6991 ICAN-7020 ICAN-7026 ICAN-7029
CDP1804	ICAN-6957 ICAN-6968 ICAN-6971
CDP1805	ICAN-6957 ICAN-6968 ICAN-6971 ICAN-7009
CDP1806	ICAN-6957 ICAN-6968 ICAN-6971 ICAN-7009
CDP1821	ICAN-6943
CDP1822/ MWS5101	ICAN-6539 ICAN-6943
CDP1823	ICAN-6943
CDP1824	ICAN-6537 ICAN-6943
CDP1825/ MWS5114	ICAN-6943
CDP1831	ICAN-6536
CDP1832	ICAN-6536
CDP1833	ICAN-6635
CDP1834	ICAN-6635
CDP18U42	ICAN-6925
CDP1851	ICAN-7023 ICAN-7063
CDP1852	ICAN-6538
CDP1854	ICAN-6632 ICAN-6991 ICAN-7023
CDP1855	ICAN-6970 ICAN-7023
CDP1856	ICAN-6657
CDP1857	ICAN-6657
CDP1869	ICAN-6889 ICAN-6953 ICAN-7032 ICAN-7038 ICAN-7067
CDP1870	ICAN-6889 ICAN-6953 ICAN-7032 ICAN-7038 ICAN-7067
CDP1876	ICAN-6889 ICAN-6953 ICAN-7032 ICAN-7038 ICAN-7067
CDP1878	ICAN-7023
CDP1879	ICAN-7023

Application Notes

ICAN-6315

COS/MOS Interfacing Simplified

by D. Blandford and A. Bishop

COS/MOS with its wide range of operating supply voltages, low input current, and low power consumption, interfaces easily with many electronic devices. In addition, COS/MOS circuitry can easily be added to a system and can often be operated from the existing power supply. Examples of practical circuits for a wide variety of interfacing situations are given in this Note; design constraints are included in each case.

Note that the CD4000 Series type numbers are followed by a suffix letter, A or B, which specifies the maximum operating voltage for the device: A, 3 to 15 volts; B, 3 to 18 volts. The outputs of all B-type devices are buffered and have the same output drive current and equal source and sink capabilities. Table I shows some characteristics of B-type devices.

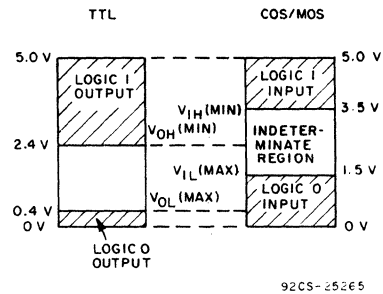


Fig. 1—TTL to COS/MOS voltage levels.

Table I — Output Drive Current—B-Type Devices

Output Drive Current	Symbol	V _{DD} Volt	V _O Volt	BD, BK, BF, BH				BE				Units
				-55°C Min.	+25°C Min.	+125°C Min.	-40°C Min.	+25°C Min.	+85°C Min.	+25°C Typ.		
Sink	I _{DN}	5	0.4	0.5	0.4	0.3	0.45	0.4	0.36	0.8	mA	
		10	0.5	1.1	0.9	0.65	1.0	0.9	0.75	1.8	mA	
Source	I _{DP}	5	4.6	-0.5	-0.4	-0.3	-0.45	-0.4	-0.36	-0.8	mA	
		10	9.5	-2.0	-0.9	-1.15	-1.8	-1.6	-1.3	-3.2	mA	

COS/MOS to TTL

In interfacing TTL with COS/MOS with a common power supply of between 4.5 and 5.5 volts, the guaranteed active-pull-up TTL output voltage of 2.4 volts is lower than the minimum COS/MOS input voltage required to guarantee switching, 3.5 volts, Fig. 1. This difference is overcome by the use of an external resistor, R_X in Fig. 2, which is also the resistor to be used for open-collector-output TTL at a V_{DD} of 5 volts. The minimum value of R_X is fixed by the maximum sink current, e.g., 1.6 milliamperes for 74-series TTL, its maximum value by I_{OH}, the off leakage of the output sink transistor. As shown in Table II, the values of R_X between 1.5 and 4.7 kilohms are suitable for all the TTL families under worst-case conditions. The COS/MOS input impedance is

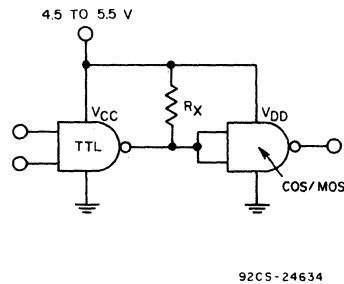


Fig. 2—TTL to COS/MOS interface.

Table II—Values of R_X for TTL—COS/MOS Interface

Characteristic	74	74H	74L	74LS	74S
R_X min. (ohms)	390	270	1.5k	820	270
R_X max. (kilohms)	4.7	4.7	27	12	4.7

essentially “capacitive”, which means that many COS/MOS inputs may be driven by a single TTL output. The actual number depends on the frequency of operation.

In the COS/MOS to TTL interface, Fig. 3, the requirement is to sink sufficient current in the low output state at a maximum output voltage of 0.4 volt. Table III gives the current

To gain improvements in speed and noise immunity in a system using a COS/MOS supply voltage greater than +5 volts, high-voltage open-collector TTL circuits such as the 7416, 7417 or 7426 may be used, as shown in Fig. 4. The value of the pull-up resistor R_X will depend on the actual value of V_{DD} ; at 10 volts, 39 kilohms would be suitable.

COS/MOS to HNIL

The wide operating-voltage range and low power consumption of COS/MOS circuitry enables it to operate from the HNIL power supply. Most CD4000A circuits will drive the HNIL input directly; for example, in Fig. 5, the CD4081B output sinks the required 1.4 milliamperes at an output voltage

Table III—Minimum Current-Sinking Capability of COS/MOS Devices

COS/MOS Type	Description	Sink Current (mA at 25°C $V_O = 0.4$ Volt, $V_{DD} = 5$ Volt)	
		Ceramic	Plastic
CD4000A	Dual 3-Input NOR Gate Plus Inverter	0.4	0.3
CD4001A	Quad 2-Input NOR Gate	0.4	0.3
CD4002A	Dual 4-Input NOR Gate	0.4	0.3
CD4007A	Dual Complementary Pair Plus Inverter	0.6	0.3
CD4009A/49A	Inverting Hex Buffer	3.0	3.0
CD4010A/50A	Non-Inverting Hex Buffer	3.0	3.0
CD4011A	Quad 2-Input NAND Gate	0.2	0.1
CD4012A	Dual 4-Input NAND Gate	0.1	0.05
CD4041A	Quad True/Complement Buffer	0.4	0.2
CD4031A	64-Stage Static Shift Register	1.3	1.3
CD4048A	Expandable 8-Input Gate	1.6	1.6
CD4XXXB	Any B-Type Device Output	0.4	0.4

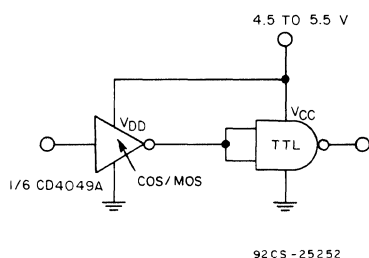


Fig. 3—COS/MOS to TTL interface.

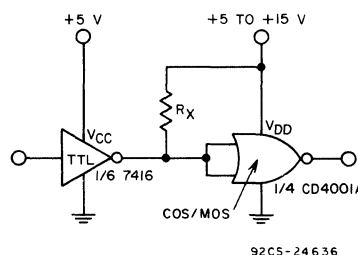


Fig. 4—TTL to COS/MOS at a V_{DD} greater than 5 volts.

sinking capability of some CD4000-series devices. Note that all B-type devices have the same standard output drive and are capable of sinking two low-power TTL loads, worst case. For the higher power types of TTL, the CD4049A and CD4050A buffers may be used. Table IV shows the minimum and typical fanout for each TTL family. The buffer takes its power from the 5-volt TTL supply and has an additional advantage in that it can accept input voltage swings of 5 to 15 volts from the preceding COS/MOS system.

Table IV—Fanout of CD4049A and CD4050A Buffers to TTL

Buffer Fanout	TTL Family				
	74	74H	74L	74LS	74S
Minimum	1	1	14	7	1
Typical	3	2	28	14	2

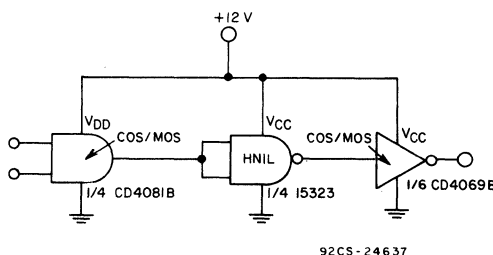


Fig. 5—COS/MOS to HNIL to COS/MOS interface.

ICAN-6315

typically less than 0.5 volt. The HNIL output-voltage levels, 0.8 volt and 10 volts, enable it to interface directly with the COS/MOS input with good noise immunity.

COS/MOS to DTL

The COS/MOS to DTL interface requires a buffer, such as the CD4049A shown in Fig. 6, to sink the DTL input current of 1.5 milliamperes at 0.4 volt. Fanout to DTL circuits depends on the sink-current capability of the COS/MOS buffer used. For the CD4049A and CD4050A, typical fanout is 3.

The DTL to COS/MOS interface requires no special consideration because the internal pull-up resistor in DTL circuits and the extremely low input current of COS/MOS circuits ensures a high logic level almost equal to the power-supply voltage.

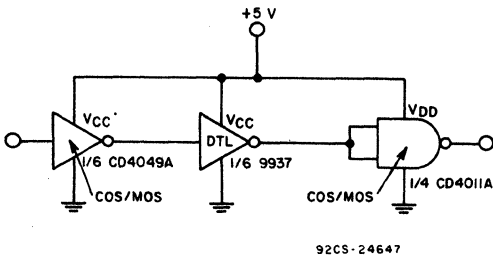


Fig. 6—COS/MOS to DTL to COS/MOS interface.

COS/MOS to 10k ECL

COS/MOS and 10k ECL are not normally interfaced, but they can be readily by using the 10124 and 10125 devices which are intended for conversion between ECL and TTL. This interface requires that the COS/MOS device be operated at a 5-volt V_{DD} , as shown in Fig. 7. Where greater speed is required of the COS/MOS system, it can be operated with V_{DD} at the ECL ground and V_{SS} at -12 volts. In the latter case, a 1N914 diode clamps the COS/MOS output to V_{EE} as shown in Fig. 8. At supply voltages greater than 6 volts, a COS/MOS buffer should not be used, as over-dissipation will occur in the buffer.

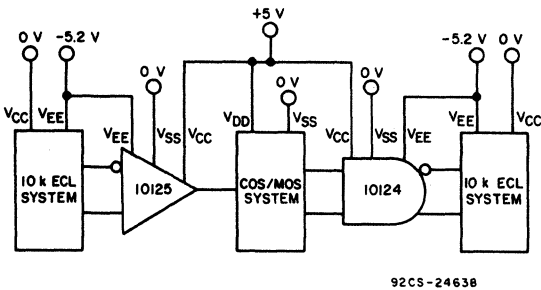


Fig. 7—10k ECL to COS/MOS and COS/MOS to 10k-ECL interface.

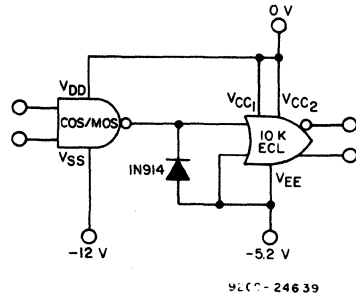


Fig. 8—COS/MOS at 12 volts to 10k-ECL interface.

COS/MOS to NMOS

The increasing use of n-channel MOS memories means that interfaces between COS/MOS and NMOS are now common. In a system of 1k memories, such as the type 2102, which employ peripheral COS/MOS circuitry for address, read/write, chip select and data handling, the COS/MOS circuitry can be supplied from the 5-volt power supply of the memory. Inputs to the memory are then COS/MOS compatible, and direct interface is permitted. The data output requires only a single pull-up resistor, R_X , as shown in Fig. 9, to ensure an acceptable high-state output voltage.

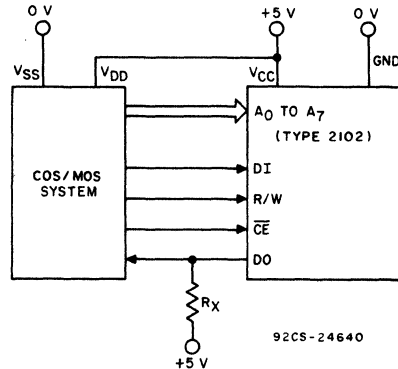


Fig. 9—Direct interface between COS/MOS and a 1k memory, type 2102.

A 4k-bit, dynamic, n-channel RAM, such as the 2107A, has +12-volt and -5-volt supplies as well as the +5-volt V_{CC} supply, as shown in Fig. 10. The COS/MOS peripheral circuitry in this system is probably best operated from the +12-volt supply, ensuring good speed characteristics and noise immunity. The 5-volt input signals to the memory are provided by CD4050A buffers powered by the 5-volt V_{CC} supply. The 12-volt-swing chip-enable signal is directly compatible with the 12-volt COS/MOS system. The data output uses a single transistor to generate the required 12-volt logic swing; memories added to provide an increase in word capacity are wire-OR'ed at the data output pin of the memory.

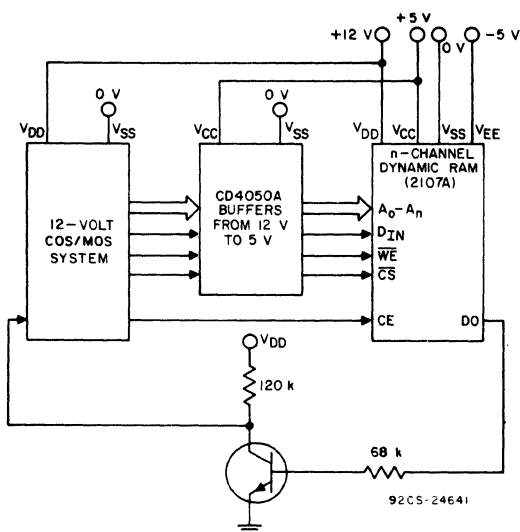


Fig. 10—COS/MOS to n-channel dynamic-RAM interface.

COS/MOS to PMOS

Silicon-gate PMOS static shift registers operating from +5-volt and -12-volt supplies are directly compatible with a COS/MOS system operating from the +5-volt supply with V_{SS} at zero volts. The only additional component required is a clamp diode to V_{SS} on the data output, as shown in Fig. 11, because the unloaded PMOS output voltage will go negative in the low output state.

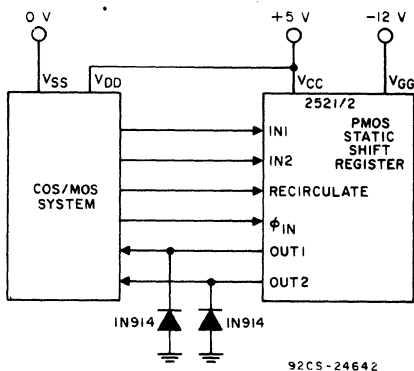


Fig. 11—COS/MOS to PMOS static-shift-register interface.

COS/MOS to Industrial and Power-Control Circuits

Industrial control systems employ greater logic swings than IC logic systems, such as COS/MOS, to achieve high noise immunity and to enable them to operate from readily available high-voltage supplies and to interface with electro-mechanical equipment.

Fig. 12 shows a simple, resistive-divider circuit used to interface a system with a 24-volt logic swing to COS/MOS; the circuit could readily be modified for even higher voltage swings. The capacitor filter enhances the excellent noise

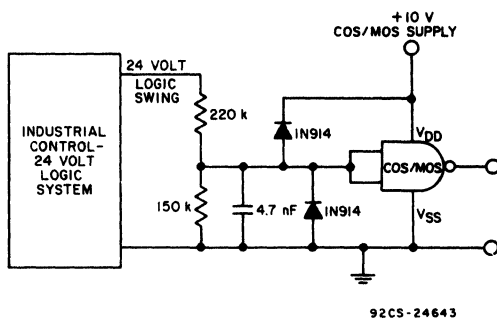


Fig. 12—Industrial control to COS/MOS interface.

immunity of the COS/MOS logic, and the two clamp diodes ensure that the input signal voltage is between V_{DD} and V_{SS} . An alternative circuit using a zener diode is shown in Fig. 13.

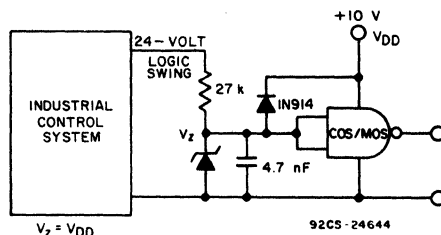


Fig. 13—Zener diode industrial control interface.

A single-transistor level-converter interfaces a COS/MOS device to an industrial control system, as shown in Fig. 14. The transistor is driven directly from the COS/MOS device output (Fig. 23 describes the method of calculating the values of the resistors needed in Fig. 14).

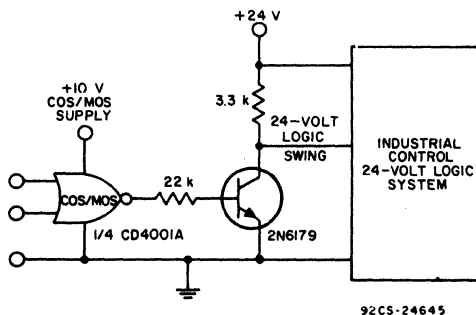


Fig. 14—COS/MOS to industrial-control interface.

The slow pulse edges typically found in an industrial control system can be speeded up in the COS/MOS system by a Schmitt-trigger circuit, the CD4093B, Fig. 15(a). At a V_{DD} of 5 volts, V_H is typically 0.6 volt, Fig. 15(b).

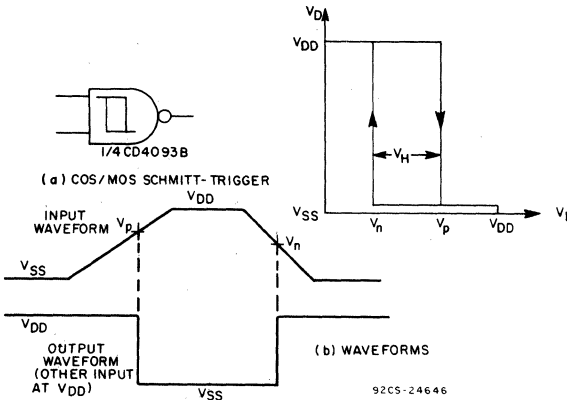


Fig. 15—(a) COS/MOS Schmitt-trigger, (b) typical waveforms for Schmitt-trigger.

A high-power coil, such as the solenoid of a printer hammer, which requires about 1 ampere at 70 volts, may be driven from a COS/MOS system by using a Darlington transistor as shown in Fig. 16. A typical value of V_{BE} for a type 2N6385 transistor is 1.5 volts at a collector current of 1 ampere and a minimum gain of 1000, so that the output source transistor of the CD4073B has to supply 1.5 milliamperes. The value of resistor R is chosen so that V_{DS} is sufficient to guarantee this output current. Suitable values of R for use with a B-type device are given in Fig. 16 for a V_{DD} of 5, 10, and 15 volts.

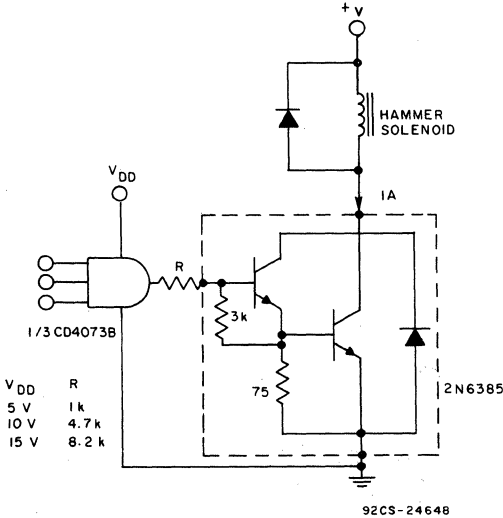


Fig. 16—COS/MOS system driving a printer-hammer solenoid with the aid of a Darlington transistor.

Power-control SCR's and triacs may also be driven directly by COS/MOS outputs. A sensitive-gate SCR, such as the 106B1, may be controlled directly by a COS/MOS gate, such as the CD4069B, and thus be able to control directly 2.5 amperes at reverse voltages up to 600 volts, as shown in Fig. 17.

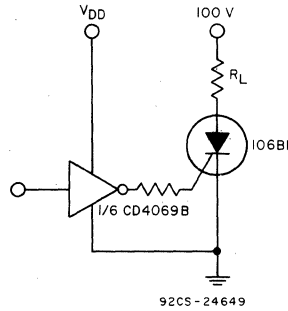


Fig. 17—COS/MOS directly driving a sensitive-gate SCR.

SCR's and triacs with gate currents in the milliampere region may be controlled by a buffer, such as the CD4049A. This buffer could, in turn, be controlled by a COS/MOS system or, as in Fig. 18, by an opto-coupler to provide greater isolation.

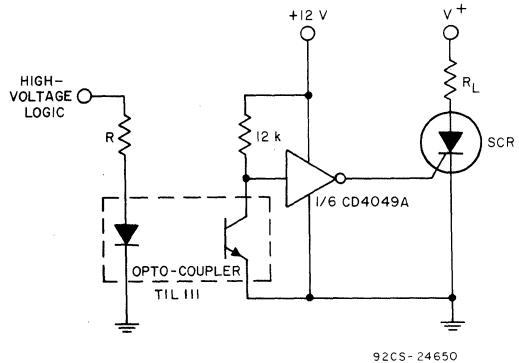


Fig. 18—High-voltage logic to COS/MOS driving an SCR.

In cases where a single-gate output source or sink current proves insufficient, it is possible to parallel the inputs and outputs of the gates on the same chip, as in Fig. 19. Gates not on the same chip and buffer circuits should not be operated in parallel as over-dissipation may result.

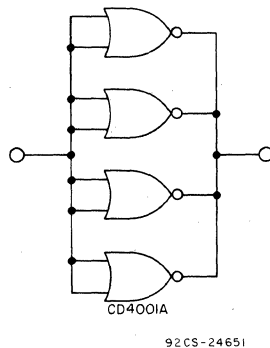


Fig. 19—Paralleling inputs and outputs.

Interfacing Op-Amps to COS/MOS

COS/MOS circuits may be connected directly to the output of an op-amp operating between the normal ±15-volt supply rails, as in Fig. 20, provided clamp diodes to V_{DD} and V_{SS} are used to ensure that the COS/MOS input voltage does not go outside the range V_{SS} to V_{DD}. Resistor R3 limits the op-amp output current should the op-amp output voltage tend toward the negative rail.

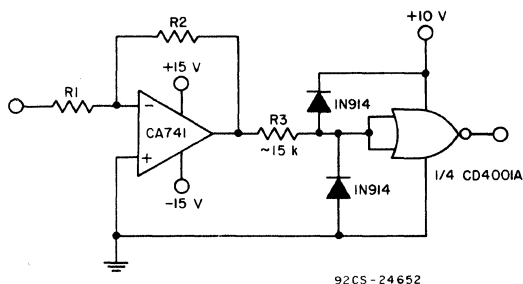


Fig. 20—Split-rail op-amp to COS/MOS interface.

Fig. 21 shows a CA741-type op-amp operated between V_{DD} and V_{SS} with a resistive divider on the non-inverting op-amp input.

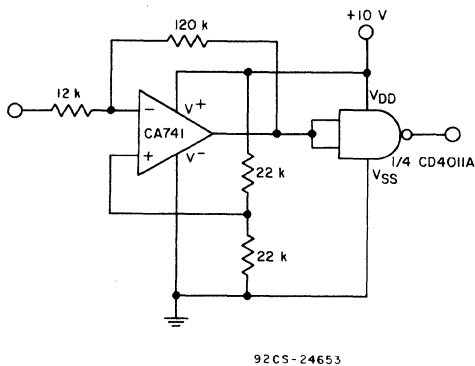


Fig. 21—Interface of op-amp and COS/MOS with common supply rail.

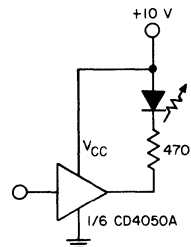
COS/MOS Driving Displays

Digital systems now employ a great variety of digital displays, so that their interface to COS/MOS is a common requirement.

COS/MOS TO LED'S

LED's may be driven directly from a COS/MOS buffer, such as the CD4050A shown in Fig. 22, at a drive current of 15 milliamperes if a power supply of approximately 10 volts is available.

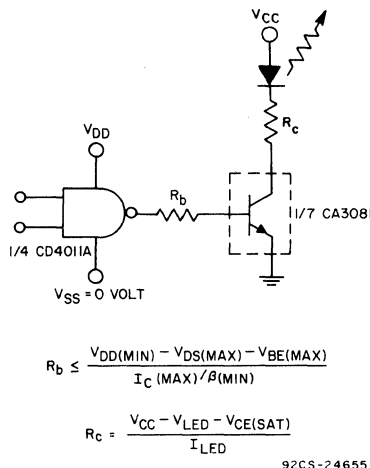
Seven-segment LED displays connected in either common anode or common cathode configurations may be driven at supply voltages as low as +5 volts by the seven-transistor



92CS-24654

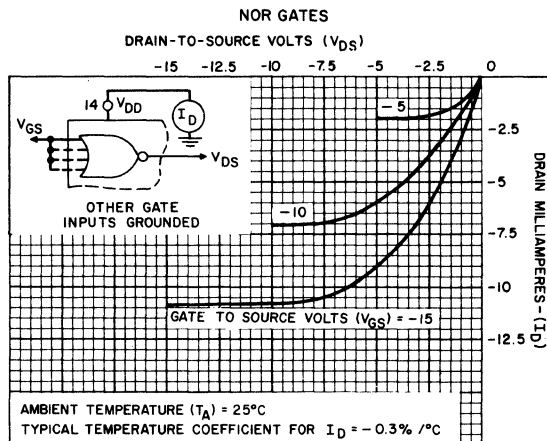
Fig. 22—COS/MOS buffer driving an LED.

arrays CA3081 and CA3082. Fig. 23 shows one of the seven transistors of the CA3081 with an LED load. The figure also shows the method of calculating R_b and R_c. The base drive current available depends on the CD4000A Series device used and the values of V_{DD} and V_{DS}. As shown in Fig. 24, the base drive current increases with both V_{DD} and V_{DS}. Fig. 25 shows one of the seven transistors of the CA3082 driving a common-cathode LED. The method of calculating the value of emitter resistor R_e is also shown in Fig. 25.



92CS-24655

Fig. 23—COS/MOS driving a transistor that has an LED load.



92CS-17776

Fig. 24—CD4001A—typical p-channel drain characteristics.

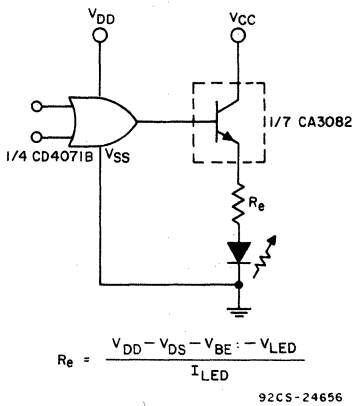


Fig. 25—COS/MOS driving a transistor with a common-cathode-connected LED load.

COS/MOS TO LCD

Seven-segment liquid-crystal displays may be driven directly by COS/MOS circuits CD4054A, CD4055A or CD4056A, as shown in Fig. 26. These circuits contain the internal level-shifting circuitry needed to convert the typically 5-volt input logic-level swing to the 30-volt peak ac signal required to drive the dynamic-scattering LCD.

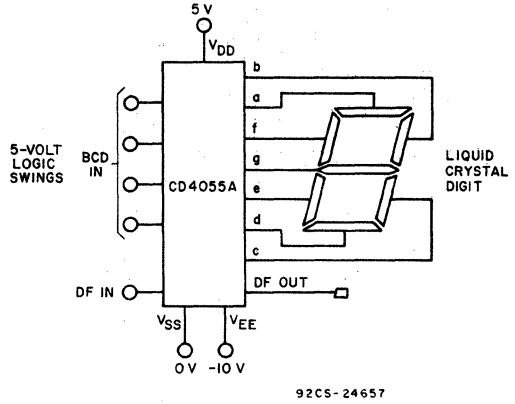


Fig. 26—Using the CD4055A to drive a liquid crystal.

COS/MOS TO GAS-DISCHARGE DISPLAY

The popular seven-segment gas-discharge display requires a cathode drive current that varies from segment to segment. Manufacturers supply drivers which are COS/MOS compatible at their inputs so that they can interface a COS/MOS system to the gas-discharge display without additional circuitry.

REFERENCE

1. "COS/MOS Digital Integrated Circuits", RCA DATABOOK Series SSD-203B, 1975.

An Introduction to Microprocessors and the RCA COSMAC COS/MOS Microprocessor

by H. Tweddle

The advent of LSI technology has not as yet brought about the revolution in digital circuit design of which it is, in principle, capable. An LSI circuit, in comparison to its hardwired SSI/MSI equivalent, is more reliable, more compact, and less expensive; however it is less expensive only in high volume, and because of its complexity it tends to be a specialized device with a limited range of applications. As a result of these two factors, LSI has rarely been suitable for standard parts and has found outlets mainly in the custom parts area, which is only economically viable for a large-volume user.

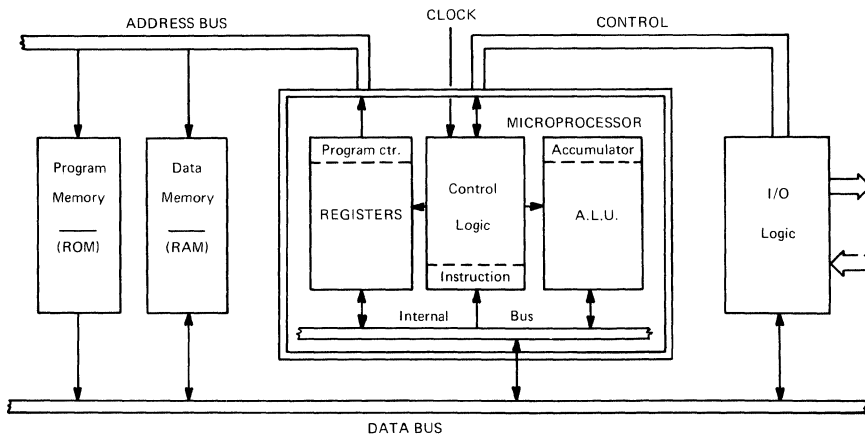
The problem, then, is to produce an LSI circuit which is versatile enough to be made available economically as a standard part. One solution would be to design it with logic elements separately accessible to the user, to be wired in whatever way suited a particular application, but this would result in an impractical pin count. So another approach is taken, that of an LSI circuit whose function is defined by the state of a number of control inputs. This approach has the added advantage that the control inputs need not necessarily be hardwired to a particular function, but may be set by other devices when the system is actually in operation.

Consider, as an example, the CD4057 COS/MOS 4-bit arithmetic logic unit.¹ It has four "function select" inputs on which four bits (the instruction) may be placed to select one of sixteen different arithmetic and logical functions. By supplying it with a series of instructions (which might be described as an elementary program), it can be made to perform more complex functions, such as multiplication and division.

A microprocessor takes this idea several stages further in that, as well as performing arithmetic and logical functions, it can address memory, input, output, and store data, and make program branch decisions. This Note is an introduction to the fundamentals of microprocessors and to the specific capabilities of the RCA COSMAC microprocessor.

AN ELEMENTARY MICROPROCESSOR SYSTEM

Fig. 1 shows a generalized block diagram of a microprocessor together with the three additional functions usually required in a system: program memory, data memory and I/O electronics. It is instructive to divide the microprocessor itself into three main areas: the arithmetic and logic unit (ALU), the registers and the control logic.



92CS-26810

Fig. 1—An elementary microprocessor system.

The Arithmetic and Logic Unit

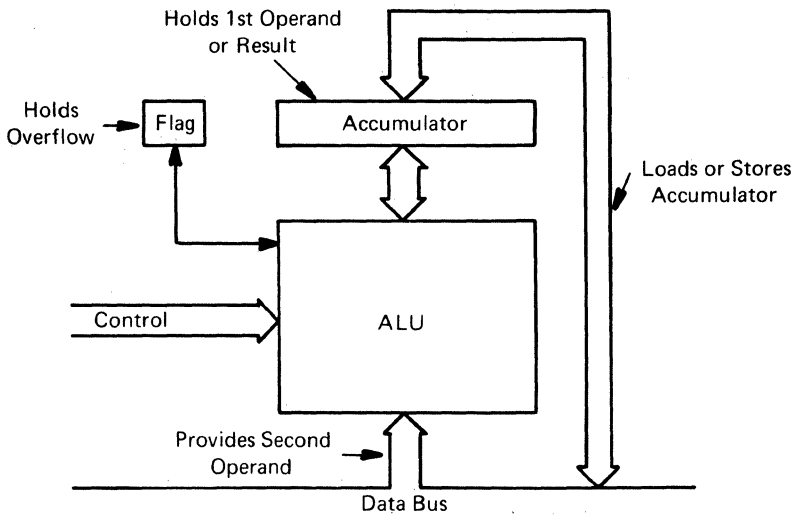
The ALU, shown in Fig. 2, performs arithmetic and logical functions on words presented to it via the internal data bus. This bus comprises a number of lines (usually 4, 8, 12, or 16) on which data words may be placed. The word length upon which the ALU operates is an important characteristic of a microprocessor: a longer word, by allowing more data to be processed at one time, provides potentially higher speed but results in a more complex, and thus more expensive, machine. The number of different ALU functions available to the user is again a characteristic of a particular microprocessor.

Associated with the ALU is the accumulator, which is a register for temporary storage of operands and results of ALU operations. Some microprocessor architectures employ several accumulators; in the simplest case, however, one operand is loaded

into the accumulator and the other is presented on the data bus, while the result appears in the accumulator, overwriting the first operand. An overflow of the accumulator sets a flip-flop called a flag. There may be a number of other flags, both to indicate overflow or underflow of registers and to show certain internal- and external-state information.

Registers

A microprocessor stores memory addresses and data in a number of registers, as shown in Table I. One register holds the current program counter, which addresses a location in program memory. A group of registers is usually provided for temporary data storage; these registers are referred to as scratchpad memory. In most applications, some additional read/write data memory is required; this memory is addressed by a register containing the data pointer.



92CS-26808

Fig. 2—The arithmetic and logic unit, the ALU.

Table I – Typical Microprocessor Register Usage

PROGRAM COUNTER	Addresses current instruction in program memory
DATA POINTER	Addresses current location of interest in data memory
STACK POINTER	Addresses next vacant location in external stack
DMA POINTER	Addresses location in data memory for DMA transfer
(INTERNAL) STACK	LIFO data/address storage
SCRATCHPAD	Random access data/address storage

A register block may also be set aside as a "stack", which provides last-in first-out storage of return addresses and data. A stack is needed when the program includes sub-routines. A limited stack size, however, limits the subroutine nesting capability of a microprocessor. A more versatile approach is to form a stack in external memory, in locations addressed by the stack pointer.

The word length of the address registers is important as it defines the number of memory locations that may be directly addressed. The word length of the data registers is usually defined by the word length of the machine.

The Control Logic

The control logic is responsible for defining the operation of the ALU, data movements within the microprocessor and data transfer to and from external devices. It also provides control signals to help external logic to interface with the microprocessor, and can be instructed to change the program counter, and thus the microprocessor operation sequence, in response to the state of control inputs, flags or internal registers. The control logic derives its timing from one or more clock inputs; its function is defined by the instruction, a word presented to it from program memory and stored in the instruction register.

Program Execution

The sequence of operations that the microprocessor is required to perform is stored as a string of instructions, called a program, in the program memory. The first of these instructions is addressed by the program counter and fetched into the instruction register. The control logic then acts upon this instruction to execute the operation it specifies. The program counter is then incremented, the next instruction is fetched, and another fetch-and-execute sequence is carried out. In this way, the microprocessor steps through the instructions to perform the task defined in the program.

Certain instructions, usually called branch instructions, change the program counter to a location other than the immediately subsequent one, allowing program jumps and the use of subroutines. Subroutines are frequently used groups of instructions, which may be stored away from the main program stream and called into use whenever needed by the main program. They help conserve memory space, since a subroutine which may be used several times in the course of a program need be stored only once.

Conditional branch instructions test the state of a particular register or flag and implement the branch if the required condition is met. If the condition is not met, the microprocessor continues with the next instruction in the main program. In this way, the microprocessor may be made to respond differently to different external or internal conditions.

Memory Requirements

Several different types of semiconductor memories are commonly used with microprocessors, as shown in Table II. Data memory is written into in the course of a program; therefore, it is implemented in a random access memory (RAM), a read/write memory that is volatile (i.e., it loses its data when the power supply is removed). Program memory, on the other hand, must usually be non-volatile and is not altered during normal microprocessor operation, so that program storage, in a production system, is normally a read-only memory (ROM). However, if the application allows or requires reloading of the program, the program may be stored in a RAM. This memory may also be used during initial development of a system, where frequent program changes are required. Another possible form of program storage is the programmable ROM (PROM), which may be used in prototype systems or in small-quantity production systems where a mask-programmed ROM is not economical. It may be found convenient in a prototype system to use an erasable PROM (EPROM) for program storage since, although non-volatile, an EPROM may be reprogrammed as the system is modified.

Input/Output

The input/output², or I/O, portion of the microprocessor provides information as words on the data bus with additional controls available separately. The I/O electronics is responsible for interfacing these signals with whatever input/output devices the system uses. Broadly, the functions of the I/O electronics include synchronization of data transfer, selection and activation of one of a number of I/O devices, and the formatting of data so that it is compatible with the device selected. The functions available on the control lines have a marked effect on the complexity of the I/O electronics. The I/O electronics may also be required to do logic-level conversion as part of the interfacing function if this facility is not provided within the microprocessor itself.

Table II – Types of Memory Commonly Used with Microprocessors

RAM	Random Access Memory	Volatile	Data storage, and program storage in some applications
ROM	Read Only Memory	Non-Volatile	Program storage in production systems
PROM	Programmable ROM	"	Program storage in small quantity production and prototypes
EPROM	Erasable PROM	"	"

An I/O data transfer may be initiated either by the microprocessor in the course of its program execution (programmed mode I/O), or by one of the I/O devices. In the latter case, the I/O device requests an interrupt by raising the interrupt input of the microprocessor. The current program execution is then halted, data and addresses required for eventual resumption of the program are stored in the stack as for a normal subroutine, and the microprocessor starts performing the interrupt service routine defined by the user. The interrupt service routine first establishes which I/O device has requested the interrupt and then performs the appropriate instructions to deal with the request.

The input or output of data may also be performed directly between the I/O device and data memory; this mode is called direct memory access (DMA). DMA may be externally or internally controlled. In the first case, raising the DMA request pin causes the microprocessor, after completing its current instruction, to detach itself from the data and address buses. The I/O device is then permitted to communicate with data memory at a speed limited only by the access time of the memory, rather than by the cycle time of the microprocessor. This form of DMA can be made very fast by using fast memory, but in this form the I/O electronics is required to provide address and control as well as data inputs to the memory, thereby increasing interface complexity. In the second form of DMA, the microprocessor itself provides address and control signals in response to a DMA request; the I/O electronics need only present the data to the I/O bus. This second form is the more convenient form of DMA, and is considerably faster than a standard programmed or interrupt mode I/O. However, the speed of data transfer is a function of the cycle time of the microprocessor, and therefore no advantage is gained from using a very fast memory.

SOFTWARE

The preceding text has established broadly what a microprocessor is and what it can do. Its associated software³ (its instructions and their manipulation) and software support (the computer assistance which may be called upon to assist in software handling) must now be considered.

Hexadecimal Code

An instruction word as understood by the microprocessor is, of course, composed of a number of binary digits or bits, which can only take one of the two values 0 or 1. However, it would be very laborious for the user to write his programs in this form, so a shorthand notation is used.

Decimal, or base 10, notation is not a convenient form of shorthand for binary numbers since 10 is not an integral power of 2 and hence conversion from decimal to

binary notation is awkward. A more practical shorthand is hexadecimal, or base 16, notation, which uses the 16 symbols 0 to 9 and A to F to represent the binary numbers 0000 to 1111. An eight-bit instruction is therefore represented by two hexadecimal or hex digits, which is considerably more convenient. Conversion from binary to hex notation is straightforward and involves direct translation of each block of four bits into a hex digit, as shown in Table III. The only points to remember are that the division into blocks must start from the "least significant" end of the binary number, and that leading zeros must be added to fill up the "most significant" block if necessary. Conversion from hex to binary is simply a reversal of this process. For example, hex 2A represents binary 101010.

Table III – Binary/Octal/Decimal/Hex Conversion Table

Binary	Octal	Decimal	Hex
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B
1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

Assembly Language

Although hex notation is convenient for converting instructions to binary code for the microprocessor, it is not very easy for the user to read. For this reason an assembly language is used in which instructions are represented by mnemonic names which bear some relation to the instruction in plain English and which can, therefore, be read easily. In addition, it is convenient to label branch destinations with mnemonic names as an alternative to keeping track of the actual addresses of such destinations in program memory (which may change as the program is modified and instructions are inserted or deleted). It should be noted that assembly-language instructions bear a one-to-one correspondence to machine-code instructions, so the designer still has close control of his instruction usage, and is, therefore, in a position to optimize his program for minimum program storage space and execution time. Conversion to machine code, or assembly, may be done on a computer using a program called an assembler. The assembler is

usually accessible through computer time-share companies as one of a microprocessor manufacturer's software support programs.

On assembly, certain diagnostic messages will be output indicating "grammatical" errors in the program. For instance, a branch to a non-existent label will be objected to by the assembler. Any changes to the program may be made by using another program resident in the computer, the editor.

Higher-Level Languages

It is sometimes convenient to write programs in a higher-level language (e.g., PL/M), in which case a support program called a compiler is required to convert the high-level instructions to machine code. One high-level instruction gives rise to a number of machine-code instructions on compilation: this feature distinguishes it from an assembly language instruction. The many-for-one transformation means that the programmer is no longer in direct control of his machine code and cannot easily optimize his program for minimum storage space and execution time. However, higher-level languages are useful since they represent a quick and convenient way of writing microprocessor programs. A well-written compiler will minimize the inefficiency of the many-for-one transformation.

Simulation

Once a grammatically correct program has been written, it is necessary to ascertain whether it performs the desired function. For this purpose, a simulated version of the microprocessor may be set up within the host computer by calling the simulator program. The simulated machine is put through its paces by applying inputs and observing outputs on the keyboard of the host computer.

Debugging

If the program run on the simulated machine has bugs (i.e., errors) it is necessary to trace the exact nature of these bugs in order to make the appropriate modifications to the program. A program called the debugger facilitates this process by allowing the designer to observe or alter registers, memory locations, and flags in the simulated machine. After removal of a bug by use of the editor, the simulator is called back and the simulate/debug/edit sequence continues until program operation is satisfactory.

Fig. 3 compares the method of programming by hand with that of using software support programs.

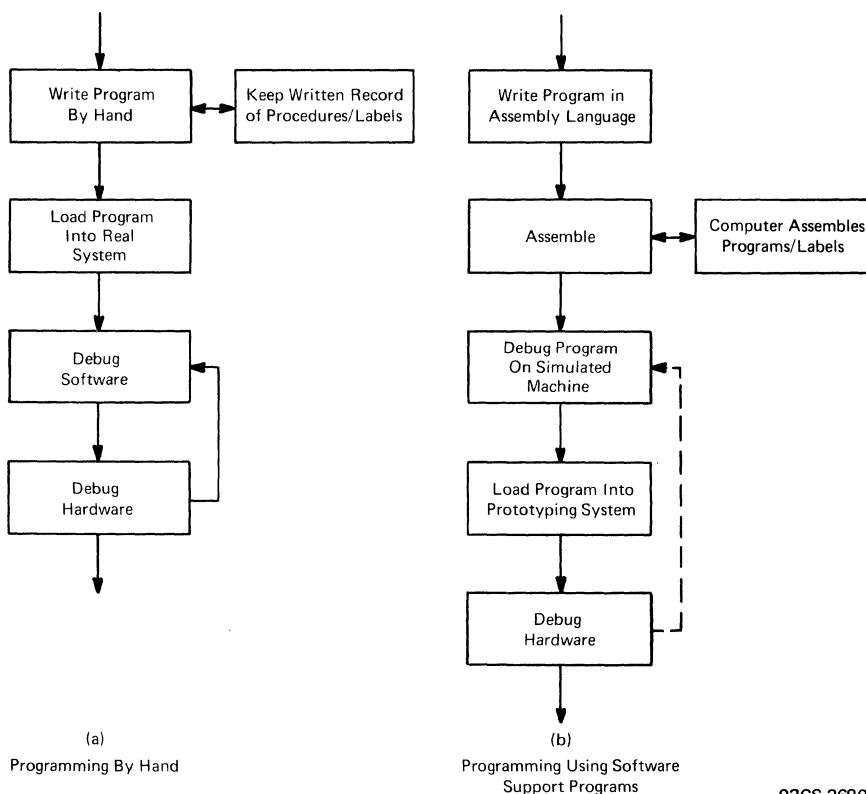


Fig. 3—Two ways of writing a program.

HARDWARE

The program produced as previously described must now be put through its paces in a real system, and for two main reasons it may still contain bugs. First, the simulated machine does not, of course, operate in real time, so that certain timing problems may not be apparent at the simulation stage. Second, the simulation does not include the I/O devices and their associated electronics, and therefore these too must be checked out in actual hardware.

Part of the applications backup provided for microprocessors is the hardware support kit. This kit contains the bare essentials of a microprocessor system including the microprocessor itself, some memory, and some interface and control electronics. It is intended as an aid to prototyping microprocessor designs. The designer builds his prototype system using this kit as a basis, and can therefore prove its operation before embarking upon his production design. The kit may then be reused to prototype the next system.

To further aid interfacing and memory expansion, standard cards implementing commonly used functions may be made available for use in the prototyping systems; support chips provide similar support for production systems.

COSMAC

The first commercially available RCA microprocessor, COSMAC, is, at the time of writing, a two-chip machine consisting of the CDP1801U microprocessor control IC and the CDP1801R microprocessor register IC. The COSMAC microprocessor is described in detail elsewhere^{4,5,6}, but it is instructive to draw attention to some of its main features here.

Technology

The COSMAC microprocessor makes use of COS/MOS devices, with their well-known advantages of low power consumption, high noise immunity and wide power supply tolerance. Only one clock input is required, and all registers employ static memory cells, so that there is no minimum clock frequency. Therefore, the user can choose slow-speed operation for ultra-low power consumption or implementation of inexpensive slow memory. The clock may, in fact, be stopped at any time without any loss of information, which is a useful feature for synchronization with other system components.

The COSMAC microprocessor operates over the full military temperature range (-55 to 125°C) and can be run from a single power supply anywhere in the range of 4 to 12 volts. It can also supply the output buffers independently of the main supply voltage if required, making external logic level conversion unnecessary.

Architecture

The COSMAC microprocessor is an eight-bit machine: its I/O bus has eight lines and the ALU acts upon eight-bit words. The architecture⁷, shown in Fig. 4, is based on a 16-word by 16-bit register matrix that can be used to store multiple program counters, data pointers, etc. This register matrix can also be used as scratch-pad memory because the high- and low-order eight bits of each register may be accessed separately by means of the data bus. This register-based architecture is extremely flexible; the designer is allowed free choice of the way in which he uses the registers to suit his particular application. The 16-bit program counters and data pointers allow direct addressing of up to 64K bytes of memory.

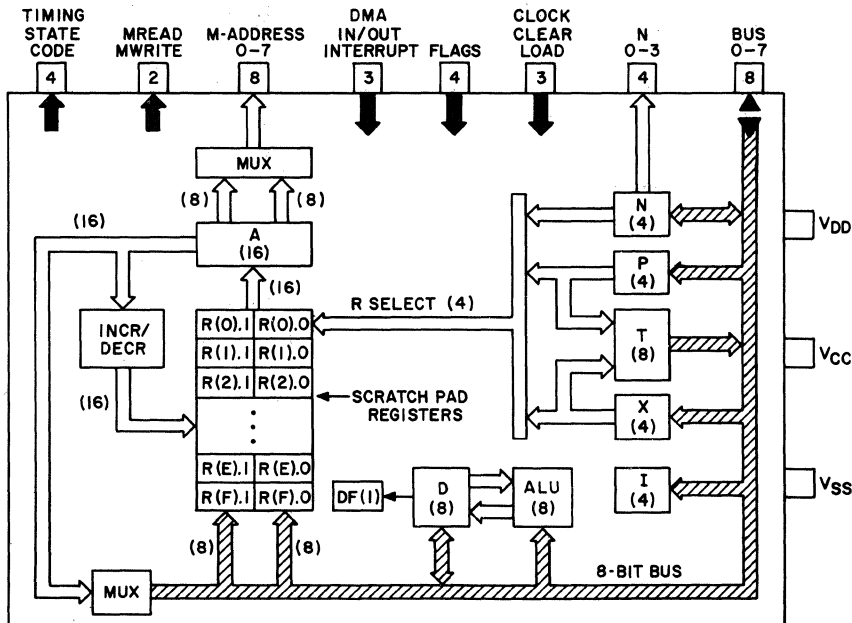


Fig. 4—Architecture of the COSMAC microprocessor.

92CS-26557

Instruction Set

The instruction set is a simple but powerful one and, again, flexible. It contains ALU, branch, register transfer, memory transfer, interrupt handling, and I/O instructions. Each instruction is eight bits long and has the same instruction cycle time (i.e., two machine cycles, or 16 clock periods). This uniformity of format and speed is significant in making the COSMAC microprocessor easy to use, especially in real-time applications. The total number of instructions recognized by the microprocessor is 208, but this total subdivides into 59 easily understood generic instructions.

Input/Output Structure

The cost of the microprocessor itself is, of course, only a small part of the total system cost; memory, input, output, power supply, system control, and programming costs are all major considerations. The advantages of the COSMAC microprocessor in some of these areas have already been described. The I/O structure especially has been evolved with a view to minimizing system cost and complexity, as shown in Fig. 5.

In addition to the eight-bit bidirectional data bus, there are four external flags (i.e., inputs which may be tested by branch instructions) and a four-bit I/O command bus which is set by the programmer as part of each I/O instruction. Two timing-pulse outputs further aid interface design.

There is an interrupt request line and two lines to request DMA input and output. The COSMAC microprocessor provides its own addressing for DMA transfers ("on-chip DMA"), again reducing I/O interface complexity. The built-in program load facility

takes advantage of this on-chip DMA capability, allowing loading of a program starting at memory location number 1 by simply setting the "load" input; no external addressing is required to load a program.

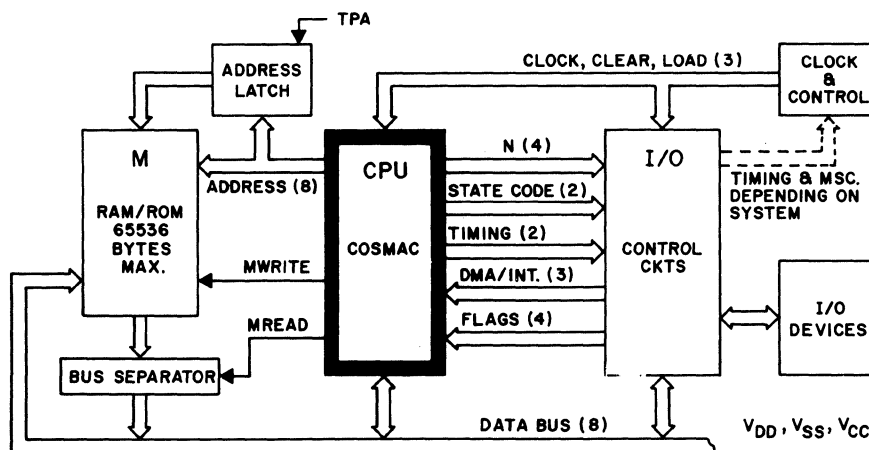
Software Development Package

Although the register-based architecture makes machine-code programming particularly easy, software support programs⁸ are available on timeshare or on tape for running on an in-house computer to speed up program development. These programs include an assembler, a simulator, and a debugger, as shown in Fig. 6. At the time of writing, a compiler is in preparation.

A batch assembler for use on an IBM 360/370 is also available. A "stand-alone" editor/assembler will fulfill the needs of those users who do not wish to depend upon either in-house or timeshare computing.

COSMAC Microkit

The COSMAC microkit⁹ is a prototyping system for the development of systems based on the COSMAC microprocessor. A teletype or similar terminal can be attached to the microkit to form an elementary but complete microcomputer system. Physically, the microkit comprises a 19-inch rack-mountable card nest with power supply and basic controls with a COSMAC CPU, control and interface circuitry, 512 bytes of PROM containing utility programs, and 1,000 bytes of RAM into which the user can load his own programs. Spare positions are provided into which the user can plug cards carrying extra memory, I/O electronics, or whatever circuitry is required to build his prototype system. Another function of the microkit will be as a vehicle for the "stand-alone" software support described above.



92CS-26554

Fig. 5—System block diagram of the COSMAC microprocessor.

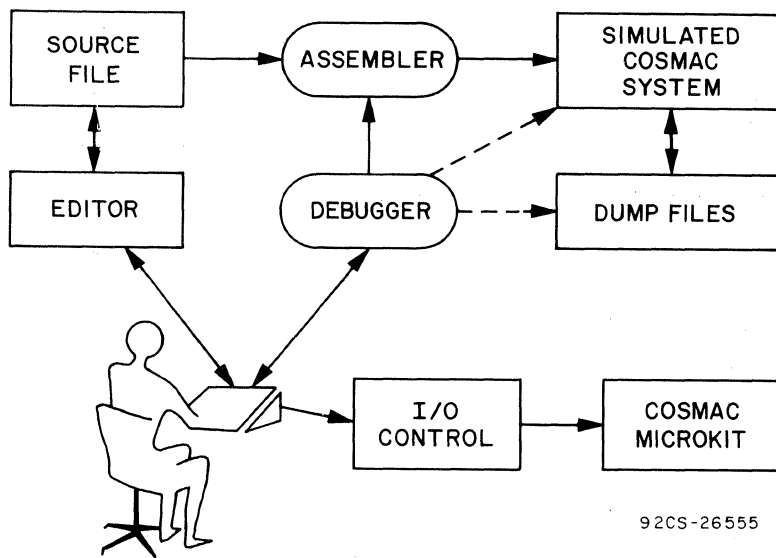


Fig. 6—Software support system for the COSMAC microprocessor.

APPLICATIONS

The range of applications of the microprocessor is extremely wide and includes most data-handling and control applications, where it provides an attractive alternative not only to hardwired logic, but also to custom LSI and in some cases minicomputers. Such applications include instrumentation, computer peripherals, automobile electronics, process and traffic control, home entertainment¹⁰, and educational and business systems.

The COSMAC microprocessor finds applications in areas where microprocessors using other technologies may not be suitable. For example, its low power consumption and wide power-supply tolerance reduce power-supply costs and allow its use in battery-operated systems or systems with standby battery supplies. The wide operating-temperature range opens up applications in the military field and in automotive and harsh industrial environments, where high noise immunity is an added advantage. In addition, since the COSMAC microprocessor has been specifically designed to reduce system cost and complexity, it promises to be economically viable in a very wide spectrum of applications.

CONCLUSION

This Note has discussed some of the characteristics of microprocessors with the intention of providing a basic introduction and definition of terms; a more detailed treatment of the subject may be found in references 11 through 14. A very brief description of the COSMAC microprocessor has been provided. It is suggested that this machine, because of its unusually simple and symmetrical architecture, provides an ideal introduction to microprocessors as well as,

for both economic and technical reasons, representing new and interesting aspects of microprocessor technology.

REFERENCES

1. RCA COS/MOS Arithmetic Logic Unit, CD4057, RCA data sheet, File No. 635.
2. "Case History: Store and Forward," P. M. Russo and M. D. Lippman, IEEE Spectrum, Sept. 1974.
3. "Basic Microcomputer Software," C. D. Weiss, Electronic Design, April 26, 1974.
4. "COSMAC: A Microprocessor for Minimum Cost Systems," N. P. Swales and J. A. Weisbecker, IEEE Intercon, 1974.
5. RCA COS/MOS Microprocessor (COSMAC), CDP1801, RCA data sheet, File No. 900.
6. "User Manual for the COSMAC Microprocessor," RCA MPM-101.
7. "A Simplified Microcomputer Architecture," J. Weisbecker, Computer, March 1974.
8. "Program Development Guide for the COSMAC Microprocessor," RCA MPM-102.
9. "COSMAC Microkit: Operator's Manual," RCA MPM-103.
10. "A Practical, Low-cost, Home/School Microprocessor System," J. Weisbecker, Computer, August 1974.
11. "Focus on Microprocessors," E. A. Torrero, Electronic Design, Sept. 1, 1974.
12. "Microprocessors," Electronics Weekly, January 29, 1975.
13. "Microprocessors: the Minicomputer/Random Logic Alternative?," B. Francis, Electronic Engineering, March 1975.
14. "Preparation: the Key to Success with Microprocessors," R. Lowandowski, Electronics, March 20, 1975.

Guide to Better Handling and Operation of CMOS Integrated Circuits

by J. Flood and H. L. Pujol

This Note recommends specific handling and operating practices that minimize the probability of damage to CMOS integrated circuits in the manufacturing operation and the field environment.

A description of various gate-oxide networks that protect against electrostatic discharge in both A-series and B-series RCA COS/MOS product is provided. A practical explanation of the SCR latch-up mechanism and its associated failure mode is given. In addition, operating procedures that help prevent device malfunction are described.

HANDLING CONSIDERATIONS

All CMOS devices are susceptible to damage by the discharge of electrostatic energy between any two pins. The gate input is equivalent to a small, low-leakage capacitor (5 picofarads typical) in parallel with a very high resistance (10^{12} ohms typical). This extremely high input impedance lends itself readily to the buildup of electrostatic charges. Therefore, because the gate-oxide breakdown of a CMOS device is typically 80 volts, damage by high levels of electrostatic discharge can occur.

To protect the gate oxide against high levels of electrostatic discharge, protective networks are implemented on all RCA CMOS (COS/MOS) devices, as described below.

Standard Protection Networks

Fig. 1 shows the standard protection network incorporated on all A-series devices

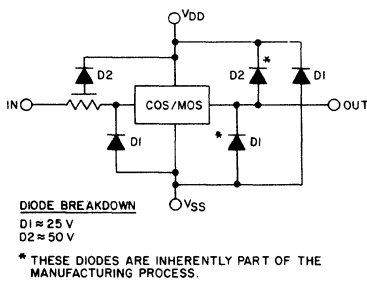


Fig. 1 - Standard protection network.

and some B-series devices. Input-diode D_2 is a distributed resistor-diode network that appears as two diodes to V_{DD} .

Improved Protection Network

Fig. 2 shows the improved protection network incorporated on all new B-series devices as well as on all A-series, B-converted types.

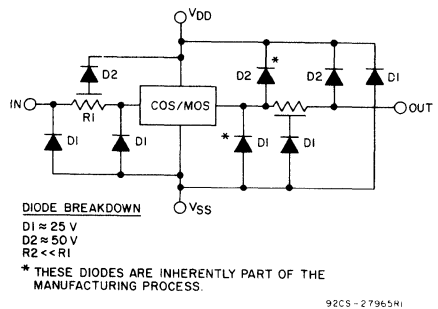


Fig. 2 - Improved protection network.

Other Protective Networks

Fig. 3 shows the modified protective network for a CD4049/4050 buffer. The input diode to V_{DD} is not incorporated so that the level-shifting function can occur.

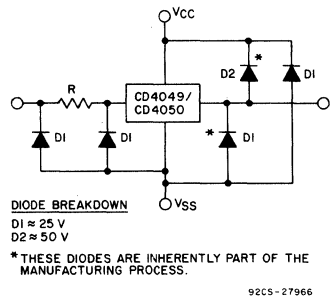


Fig. 3 - Modified protection network.

Fig. 4 shows a transmission gate with the intrinsic diodes that protect against electrostatic discharge.

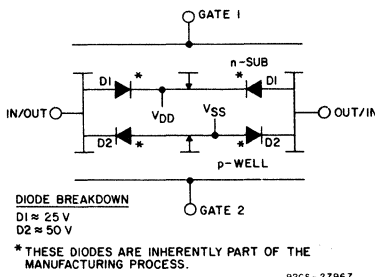


Fig.4 — Transmission gate with intrinsic diodes that protect against electrostatic discharge.

The protection networks described in this Note were characterized by using the equivalent body discharge network of Fig. 5. There are 12 possible combinations by which a device can be damaged. A discussion of the combinations is beyond the scope of

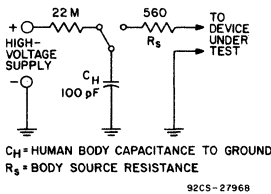


Fig.5 — Equivalent-body discharge network.

this Note; however, Table I shows worst-case protection levels for the above networks. Additional protection can be obtained by adding external series resistors at device inputs. The value of this resistance should be 10 kilohms for gate inputs and 1 kilohm for transmission gate inputs, where applicable. In addition, zener diodes at the output pins can clamp the voltage at a safe level. The zener value should not exceed the absolute maximum rating of the part.

On-chip protection networks are not used on transmission gates to maintain low on-resistance. The 800-volt worst case capability is provided by the intrinsic diodes shown in Fig. 4.

TABLE I — Worst-Case Capability of Protective Networks

Protective Network	Worst-Case Capability
Standard (inc. CD4049, CD4050)	1 kV to 2 kV
Improved	4 kV
Transmission Gate	< 800 V

General Handling Rules

Table I indicates the typical, worst-case voltage levels that the above networks can

withstand. Because every manufacturing environment is different, levels above those shown in Table I should be anticipated and protected against by following the handling recommendations of Table II.

Basic protection starts with personnel and materials all at the same or ground potential.

Dry weather (relative humidity less than 30 percent) tends to multiply the accumulation of static charges on any surface. Conversely, higher humidity levels (40 to 50 percent) tend to reduce the magnitude of the static voltage generated. In a low-humidity environment, the handling precautions listed above take on added importance and should be adhered to without exception.

HANDLING OF UNMOUNTED CHIPS

In handling unmounted chips, differences in voltage potential should be avoided. A conductive carrier or a carrier having a conductive overlay should be used. Another important consideration is the sequence in which bonds are made; the VDD (device supply) connection should always be made before the VSS (ground) bond.

HANDLING OF SUBASSEMBLY BOARDS

After COS/MOS units have been mounted on circuit boards, proper handling precautions should still be observed. Until these subassemblies are inserted into a complete system in which the proper voltages are applied, the board is no more than an extension of the leads of the device mounted on the board.

It is good practice to put conductive clips or conductive tape on the circuit-board terminals. This precaution prevents static charges from being transmitted through the board wiring to the devices mounted on it.

AUTOMATIC HANDLING EQUIPMENT

When automatic handling equipment is used, it may not always be possible to eliminate static electricity through grounding techniques alone. Automatic feed mechanisms must be insulated from the devices under test at the point where the devices are connected to the test set. The anvil transport portion of the automatic handling mechanism can generate very high levels of static electricity as a result of the continuous flow of devices over and then separating from the anvil. Total control of these static voltages is critical because of the high throughputs associated with automatic handling.

Fortunately, the resolution of this problem is simple, practical, and inexpensive. Ionized-air blowers, which supply large volumes of ionized air to objects that are to be charge neutralized, are commercially available from many supply sources. Field experience with ionized-air techniques reveals this method to be extremely effective in eliminating static electricity when grounding techniques cannot be used.

TABLE II — General Handling Recommendations

	Should be conductive	Should be grounded to common point
Handling equipment	X	
Metal Parts of Fixtures and Tools		X
Handling Trays	X	
Soldering Irons		X
Table Tops	X	X
Transport Carts	X	
Manufacturing Operating Personnel		Utilize grounded, metal or conductive plastic wrist straps with 1-megohm series resistor
General Handling of Devices		Utilize grounded, metal or conductive plastic wrist straps with 1-megohm series resistor

Failure Mechanisms

Electrical damage resulting from handling is usually caused by either of the two following failure mechanisms:

1. Low-level static electricity (voltage of 1 kV to 4 kV). Input diode protection may be overstressed and input leakage currents as high as 1 milli-ampere across diodes may cause a malfunction.
2. High-level static electricity (voltages greater than 4 kV). Gate oxides may become short-circuited. Inputs to V_{DD} or V_{SS} terminals will be low-impedance inputs.

The presence of these types of device malfunction can be detected by curve-tracer checks of the input protection diodes described above. Diode degradation resulting from static electricity is observable in the low-reverse-breakdown characteristics shown on the curve tracer. On the other hand, damage resulting from high levels of static electricity are observed as a resistive short to V_{DD} or V_{SS} .

Typical Manufacturing Area Procedure

The example below illustrates all of the above recommendations for handling CMOS devices in a typical manufacturing environment. Although existing protective networks offer a high level of protection against electrostatic discharge, this example emphasizes specific precautions that can help eliminate damage.

Receiving Area

Devices should not be removed from their conductive or antistatic carriers. If devices are not received in conductive or antistatic packing material, they should be returned to the supplier.

Incoming Inspection

Physical — Parts should be counted without removing them from their containers.

Storage — Devices should remain in carriers. Even a partial removal of IC's from a carrier should only be done by a grounded operator. Devices removed should be placed in a conductive tray.

Electrical — All testing should be performed by a grounded operator. Devices should be reinserted in conductive carriers after completion of a test.

PC Board Assembly

It is desirable that PC boards have shorting bars installed prior to assembly (soldering). Where possible, CMOS IC's should be the last component installed on the PC board.

Boards should be transported to the wave-solder area in conductive carriers. Flux removal should be done with an acceptable solvent. Examples of specific, acceptable alcohols are isopropanol, methanol and special denatured alcohols such as SDA1, SDA30, SDA34 and SDA44. The removal of flux from non-hermetic and molded-plastic devices by means of soap and water in a dishwasher is NOT recommended as this procedure will adversely affect the long-term life of the device.

OPERATING CONSIDERATIONS

Proper operating procedures are as important as proper handling techniques. A review of RCA COS/MOS A-series and B-series operating characteristics and ratings is given in Table III.

Operating Voltage

When devices are operated near the maximum supply-voltage range, power-supply turn-on or turn-off transients, power-supply ripple or regulation, and ground noise should be suppressed; any of the above conditions must not cause ($V_{DD} - V_{SS}$) to exceed the absolute maximum rating. A good practice is to use a zener protection diode in parallel with the power bus. The zener value should be above the expected maximum regulation

**TABLE III — Maximum Ratings of RCA COS/MOS Devices
(Voltages referenced to V_{SS})**

DC Supply Voltage Range	3 to 15 V (A Series); 3 to 20 V (B Series)
Recommended Operating Voltage	3 to 12 V (A Series); 3 to 18 V (B Series)
DC Input Voltage Range	-0.5 to V _{DD} + 0.5 V
Dissipation per Package	500 mW
Device Dissipation per Output Transistor	100 mW
Storage Temperature Range	-65 to +150°C
Operating Temperature Range	
Ceramic Package Types	-55 to +125°C
Plastic Package Types	-40 to +85°C
Lead Temperature (during soldering) at a distance 1/16 ± 1/32 inch (1.59 ± 0.79 mm) from case for 10 seconds max.	+ 265°C

excursion, but should not exceed the maximum supply voltage. Fig. 6 illustrates a practical zener shunt circuit. A current-limiting resistor is included if the supply-current compliance is higher than the zener power-dissipation rating for a given zener voltage. The shunt capacitor value is chosen to supply required peak-current switching transients.

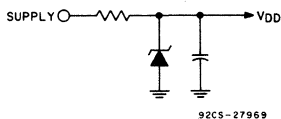


Fig. 6 — Zener-diode short circuit.

Unused Inputs

All unused input leads must be connected to either V_{SS} or V_{DD}, whichever is appropriate for the logic circuit involved. A floating input on a high-current type (such as the CD4009A, CD4010A, CD4041A, CD4049A, CD4050A) can result not only in faulty logic operation, but can cause the maximum power dissipation of 500 milliwatts to be exceeded; the result may be damage to the device. Another consideration with high-current devices is the need for a pull-up resistor between the inputs and V_{SS} or V_{DD} should there be any possibility that the device terminals may become temporarily open or unconnected (e.g., if the printed circuit board driving the high-current types is removed from the chassis). A useful range of values for such resistors is from 0.2 to 1 megohm.

Input Signals

Signals should not be applied to the inputs while the device power supply is off

unless the input current is limited to a steady-state value of typically less than 10 milliamperes. Input-signal interfaces that are the allowable 0.5 volt above V_{DD} or below V_{SS} should be current-limited to typically 10 milliamperes or less.

Whenever the possibility of exceeding 10 milliamperes of input current exists, a resistor in series with the input is recommended. The value of this resistor can be as high as 10 kilohms without affecting static electrical characteristics. However, speed will be reduced because of the added RC delay. Particular attention should be given to long input-signal lines where high inductance can increase the likelihood of large-signal pickup in noisy environments. In these cases, series resistance with shunt capacitance at the IC input terminals is recommended. The shunt capacitance should be made as large as possible consistent with the system speed requirements.

Fan-Out — COS/MOS to COS/MOS

All RCA COS/MOS devices have a dc fan-out capability of greater than 50. The reduction in COS/MOS switching speed caused by added capacitive loading should, however, be consistent with high-speed system design. The input capacitance is typically 5 picofarads for most types; the CD4009 and CD4049 buffers have a typical input capacitance of 15 picofarads.

Maximum Clock Rise and Fall Time

All COS/MOS clocked devices show maximum clock rise- and fall-time ratings (normally 5 to 15 microseconds). With longer rise or fall times, a device may not function properly.

Parallel Clocking

When two or more different CMOS devices use a common clock, the clock rise time must be kept at a value less than the sum of the propagation delay time, the output transition time, and the setup time. Most flip-flop and shift-register types are included in this rule and are so noted in the individual data sheets.

Output Short Circuits

Shorting of outputs to VSS or VDD can cause the device power dissipation to exceed the safe value of 500 milliwatts. In general, outputs of these types can all be safely shorted when the device is operated with $(V_{DD} - V_{SS}) \geq 5$ volts, but the 500 milliwatt dissipation ratings may be exceeded at higher power-supply voltages. For cases in which a short-circuited load, such as the base of a p-n-p or n-p-n bipolar transistor, is directly driven, the device output characteristics given in the published data should be consulted to determine the requirements for safe operation below 500 milliwatts. Note that a single output transistor short must be limited to 100 milliwatts.

SCR Latch-Up

Operation above maximum ratings can force CMOS devices into a p-n-p-n SCR "latch-up" mechanism, which can be destructive. Any transients should be avoided and any large loads occurring during operation near the maximum rating should be avoided.

"Latch-up" is considered to be the creation of a low-resistance path between the power supply and ground on a circuit during an electrical pulse; the path remains a low-resistance path after the pulse. In CMOS circuits, several parasitic bipolar transistors exist, as shown in Fig. 7. The p-n-p transistor

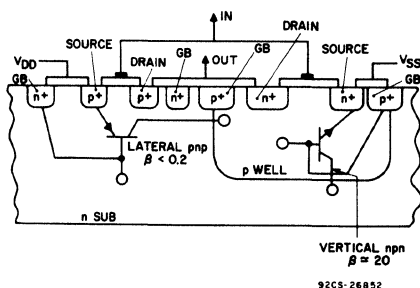


Fig. 7 — Parasitic bipolar transistors in CMOS circuits.

is a wide-base lateral structure whose β , normally less than 0.2, is a function of device geometry. The conditions for SCR turn-on are as follows:

1. $\beta_{n-p-n} \times \beta_{p-n-p} \geq 1$
(vert.) (lat.)
2. The lateral p-n-p and vertical n-p-n base emitter junctions are forward biased.

3. The bias circuit that applies power to VDD and to the input must be capable of supplying current equal to the holding current of potential SCR's.

Fig. 8 shows the equivalent circuit for the SCR structure present in CMOS circuits.

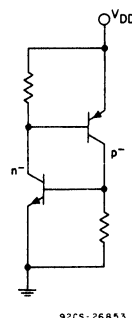


Fig. 8 — Equivalent circuit for the SCR structure present in CMOS circuits.

Fig. 9 shows a curve of I_{DD} as a function of V_{DD} , which illustrates the effect of secondary breakdown and SCR latch-up.

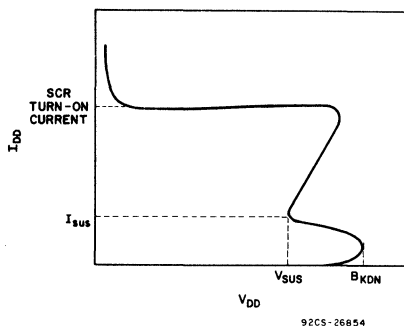


Fig. 9 — Curve illustrating effect of secondary breakdown and SCR latch-up.

Table IV shows typical values of breakdown voltage and sustaining voltage and current for RCA COS/MOS A-series and B-series devices. The table shows that B-series devices are much harder to latch than A-series types because of the higher breakdown voltage.

TABLE IV — Breakdown Voltage and Sustaining Voltage and Current Values

Characteristic	A-Series	B-Series
$V_{BKDN_{min}}$	17 V	25 V
V_{SUS}	15 V	22 V
I_{SUS}	Type-Dependent	50–100 mA 2–40 mA

Use of CMOS ROM's CDP1831 and CDP1832 With the RCA Microprocessor Evaluation Kit CDP18S020*

The CDP1800 family of microprocessor products includes two 4096-bit static CMOS mask-programmable read-only memories, the CDP1831 and CDP1832. Each is organized as 512 8-bit words, but they differ in addressing structure. The CDP18S020 Evaluation Kit is designed to accept the two ROM's in prewired locations and will also accommodate expanded ROM systems in the User I/O area. The Evaluation Kit is provided with a factory-programmed CDP1832 which contains the Utility Program UT4.

This application note describes the operation and design of CDP1831-and CDP1832-based read-only memory systems in the CDP18S020 Evaluation Kit.

DESCRIPTION OF CDP1831 FEATURES

The CDP1831 interfaces directly with the CDP1802 without additional components. The CDP1831 responds to a 16-bit address multiplexed on 8 address lines (MA0-MA7) 8 bits at a time. Fig. 1

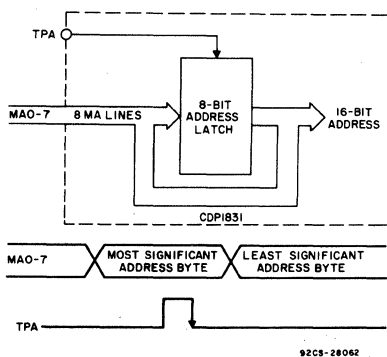


Fig. 1 - CDP1831 ROM address-line multiplexing.

shows how the address lines are multiplexed and latched to form the 16-bit address. The most significant address byte is latched internally by either a positive or negative (user mask programmable) level of TPA. For compatibility with the CDP1802, this level should be programmed negative, as also indicated in Fig. 1.

Fig. 2 shows the internal allocation of the 16 address lines. The seven most significant address lines select one of 128 512-byte sectors of the 64K microprocessor memory space. The sector address is equivalent to a ROM select and is user-programmable for a specific CDP1831. The seven-bit ROM select code is "ANDed" with the three chip selects (CS1, CS2, and MRD) to enable a given ROM. This signal is provided on an output pin (CEO) of the CDP1831 to indicate when the ROM is enabled. The nine least significant address lines are decoded to select one of 512 bytes in the ROM (sector).

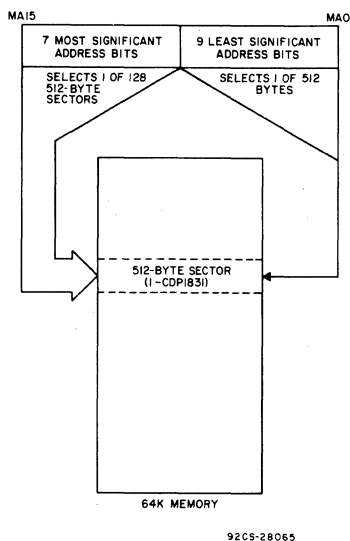


Fig. 2 - Internal allocations of 16 address lines.

*Note: The information in this Application Note is useful here; the Evaluation Kit, however, is no longer available.

In summary, the CDP1831 responds to a 16-bit address; this address specifies one of 128 memory sectors, which is fixed for a given ROM, and one of 512 contiguous bytes within the specified ROM.

There are three important features of the CDP1831 which result from this addressing structure. First, it is directly compatible with the CDP1802 multiplexed address bus. The specific connections are shown in Fig. 3. Second, a ROM

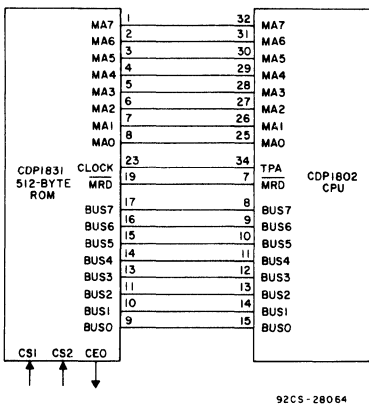


Fig. 3 - Interconnections of ROM CDP1831 to microprocessor CDP1802.

system of from 512 to 64K bytes can be configured which is directly compatible with the CDP1802 and requires no address decoding. This system is shown in Fig. 4. Third, when the ROM is selected,

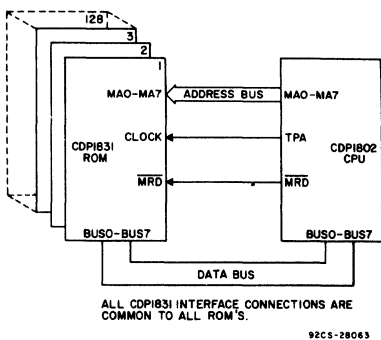


Fig. 4 - Multiple CDP1831 ROM system.

the CEO output goes high and can be used directly to disable a RAM system. A minimal system consisting of the CDP1802 CPU, CDP1824 32-byte RAM, and CDP1831 512-byte ROM is shown in Fig. 5.

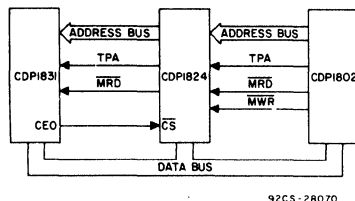


Fig. 5 - Minimal microprocessor memory system consisting of ROM CDP1831 and RAM CDP1824.

Reading data out of the CDP1831 follows a standard procedure. The device is selected by clocking the proper sector address into the address latch. The tri-state output buffers are enabled by the proper combination of the three chip selects, CS1, CS2, and MRD. The chip-select enable polarities are user mask programmable. Valid data will appear at the output within one access time (t_{AA}) from the last address change.

Operating conditions and static and dynamic characteristics for the CDP1831 are given in the device data sheet. The CDP1831 is available in two versions. The CDP1831D has a recommended operating voltage range of 3 to 12 volts; the CDP1831CD has a recommended operating voltage range of 4 to 6 volts. Both versions are functionally identical and are supplied in 24-lead dual-in-line packages.

APPLICATION OF CDP1831 IN EVALUATION KIT ROM SYSTEMS

The CDP18S020 Evaluation Kit has been prewired to accept the CDP1831 ROM in position U1. Fig. 6 shows the U1

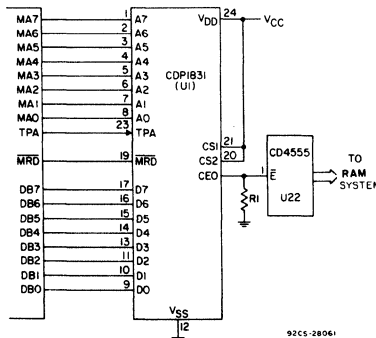


Fig. 6 - Pin connections for CDP1831 ROM in U1 location of Evaluation Kit.

pin connections. The eight CDP1802 memory address lines (MA0-MA7) and TPA are wired to the appropriate ROM pins. The CDP1831 data outputs are con-

nected directly to the data bus. The CS1 and CS2 inputs are connected permanently high, and the MRD input is connected to the CDP1802 MRD output. As a result, the ROM puts data on the bus when MRD = 0. The chip enable output CEO is connected to the \bar{E} (pin 1 of U22) control input of the RAM system decoder. When the ROM is addressed (enabled), the CEO output goes high and disables the RAM system. The CEO output can be used to disable the CDP1832 Utility Program ROM (U2) as shown in Fig. 7. Link 3 must be

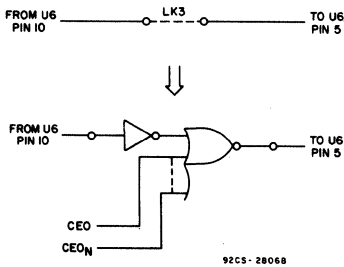


Fig. 7 - Use of CDP1831 CEO output to disable the CDP1832 ROM (U2) utility program.

broken and the indicated gating inserted. The effect of the extra gating is to "OR" the CEO output with the existing CDP1832 \bar{CS} signal. These controls assure the ability to place the CDP1831 sector address in any of the possible 128 locations regardless of the RAM or ROM memory space allocation. Note that a pull-down resistor ($R1 = 22K\Omega$) is on the CEO line to assure proper operation of the RAM system when the CDP1831 ROM is removed. When the system operates with a ROM in the U1 location, this resistor may be removed.

The CDP1831 VDD (pin 24) is connected to the Evaluation Kit VCC. This connection permits maximum memory-system flexibility with or without the CDP1831.

To use the U1 position for the CDP1831, the device is simply inserted in the location as indicated on the PC card. (Optionally, R1 can be removed). Proper operation can be verified by use of the ?M[Starting Address][A][FF] command of UT4 to list the memory contents. This check assumes that the CDP1831 does not occupy location 8000₁₆ to 81FF₁₆.

Expanding the ROM System on the Evaluation Kit is especially easy. In the User I/O section of the PC card, the CDP1802 memory address bus, data bus, MRD, and TPA signals are provided on the left side. Multiple CDP1831's (up to 12) can be inserted in the pre-drilled locations and interfaced to the address and data buses. Multiple CEO outputs can be combined with a standard AND gate to

generate a ROM system CEO. Example devices for this function are the CD4081, CD4082, and CD4085.

It should be noted that memory pattern generation and verification is possible in the CDP18S020 RAM system by using the write-protect and battery hold-up features. Once a pattern has been verified, it can be permanently stored by the custom masking of one or more CDP1831 ROM's.

DESCRIPTION OF CDP1832 FEATURES

The CDP1832, like the CDP1831, is a static 4096-bit mask-programmable COS/MOS read-only memory organized as 512 8-bit words. It is conventionally organized, requiring nine address lines and a chip select \bar{CS} to read one of 512 8-bit words onto a bidirectional data bus. Data is available within one access time (t_{AA}) from the last address change. The chip-select control \bar{CS} functions as an output disable. The CDP1832 is pin-compatible with the 2704 512-word x 8-bit erasable and electrically reprogrammable ROM. It can be directly inserted into a 2704 socket without any PC board changes. Additional details on the CDP1832/2704 compatibility and design considerations can be found in the RCA Application Note entitled "Use of Erasable and Electrically Reprogrammable ROM 2704 With RCA Microprocessor Evaluation Kit CDP18S020", ICAN-6540.

A functional diagram of the CDP1832 is shown in Fig. 8. Operating conditions and

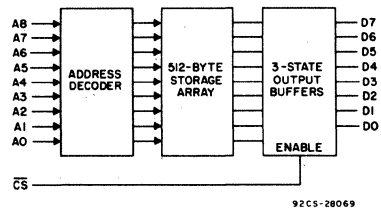


Fig. 8 - Functional diagram of CDP1832 512-word x 8-bit static read-only memory.

static and dynamic characteristics are given in the device data sheet. The CDP1832 is available in two versions. The CDP1832D has a recommended operating voltage range of 3 to 12 volts; the CDP1832CD has a recommended operating voltage range of 4 to 6 volts. Both versions are functionally identical and are supplied in 24-lead dual-in-line packages.

APPLICATION OF CDP1832 IN EVALUATION KIT ROM SYSTEMS

The CDP18S020 Evaluation Kit has been prewired to accept the CDP1832 ROM in position U2. Fig. 9 shows the U2

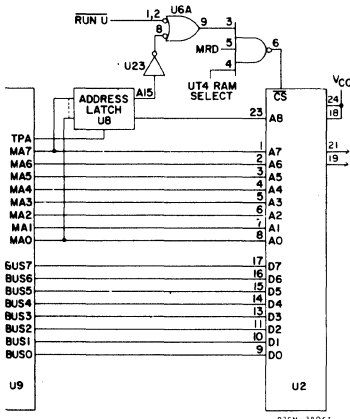


Fig. 9 - Pin connections for CDP1832 ROM in U2 location of Evaluation Kit.

pin connections. The basic Evaluation Kit is supplied with a CDP1832 factory-programmed with the Utility Program UT4. Because of the desire to have UT4 in the upper half of memory, the circuitry which interfaces with the CDP1832 in position U2 selects the ROM when either A15

8000₁₆ or RUN U are true. The effect is as follows: When RUN U is activated, the ROM (UT4) is selected and the first 4 locations of the Utility Program initialize the most significant byte of the CDP1802 program counter to 80₁₆. By the time the RUN U pushbutton is released, A15 = 1, and the ROM continues to be selected. As shown in Fig. 9, gates U6A and U6B control the CDP1832 chip select CS. U6A performs the logic for starting the Utility Program in 8000₁₆. U6B AND's the enable request with MRD which assures that the ROM puts data on the data bus at the proper time. In addition, a UT4 RAM-select-input disables the ROM when UT4 writes data in the CDP1824 Utility RAM.

Because the CDP1832 is compatible with the 2704 EPROM, location U2 has been prewired for the necessary V_{BB} and V_{DD} voltages required by the 2704. These pin connections are available at the SYSTEM CONNECTOR (P1). The CDP1832 does not require any connections to these pins; however, if they have been connected for 2704 operation they need not be disconnected for proper operation of the CDP1832. The CDP1832 V_{DD} supply (pin 24) has been connected to the Evaluation Kit V_{CC}.

Expanded CDP1832 ROM-based memory systems are easily constructed in the User I/O section of the PC card. In addition to wiring ROM locations, address latch and decode functions must be provided. A sample 2K ROM system which can be constructed in the User I/O section of the PC card is shown in Fig. 10.

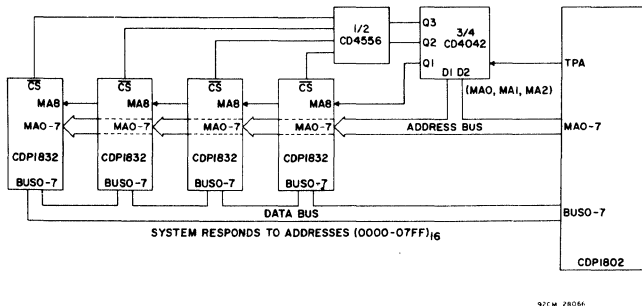


Fig. 10 - Sample 2K ROM system using CDP1832 ROM's.

Use of CMOS RAM CDP1824 with RCA Microprocessor Evaluation Kit CDP18S020*

by J. R. Oberman

The CDP1824, a 32-byte, static, silicon-gate CMOS, random-access memory, is intended for use in microprocessor systems requiring a minimal amount of writable storage. It is directly compatible with the CDP1802 microprocessor, requiring no additional interface components, and operates at maximum microprocessor speeds. In many systems the availability of memory in 32-byte increments will prove to be a cost-effective means of implementing working space or stack storage requirements.

This note describes the CDP1824, its application in the CDP18S020 Evaluation Kit, and presents examples of how the CDP1824 can be combined with the CDP1831 ROM to form efficient ROM-RAM systems.

DESCRIPTION OF CDP1824 FEATURES

The CDP1824 is an 18-pin device consisting of five address inputs, an output disable control (\overline{MRD}), a memory write control (\overline{MWR}), a chip-select control (\overline{CS}), eight common data input-output terminals, and two voltage connections. The CDP1824 requires no precharge or clock signals for proper operation. The CDP1824 electrical characteristics are given in its data sheet.

The five address inputs (MA0-MA4) are buffered and decoded to uniquely select one of 32 bytes. The eight input-output data lines can interface directly with the microprocessor bidirectional data bus. Operation is determined by the state of the \overline{MWR} and \overline{MRD} terminals. These inputs can also be driven directly from the CDP1802 microprocessor. When $\overline{MRD} = 0$, the output drivers are enabled, and the

output data corresponds to the addressed byte. If $\overline{MRD} = 1$ and $\overline{MWR} = 0$, the contents of the data bus are stored at the addressed location. A chip-select input signal \overline{CS} is required to enable the memory. Table I indicates the operational modes of the RAM and Fig. 1 shows the functional block diagram.

The RAM is available in two functionally identical versions. The CDP1824D has a recommended operating voltage range of 3 to 12 volts, the CDP1824CD a recommended operating voltage range of 4 to 6 volts. Both devices operate over the full military temperature range of -55°C to $+125^{\circ}\text{C}$, and both have the same dynamic characteristics.

APPLICATION OF CDP1824 IN EVALUATION KIT

The CDP1824 is used in the CDP18S020 Evaluation Kit by the Utility Program to provide CDP1802 register storage. This application of the RAM is useful in program debugging and illustrates a cost-effective means of implementing minimum storage requirements.

The memory is automatically loaded by the Utility Program upon the initiation of the RUN U control. Memory contents can be examined by use of the Utility Program "?M" command. The CDP1824 is located at the memory addresses 8C00₁₆ to 8C1F₁₆.

In the Evaluation Kit the CDP1824 address, \overline{MRD} , and \overline{MWR} terminals are connected directly to the corresponding terminals of the CDP1802. The RAM input-output terminals are connected directly to the bidirectional data bus. The \overline{CS} input is used to control the selection of the memory.

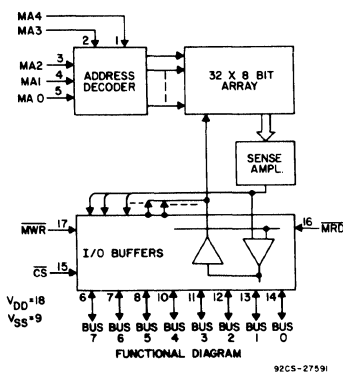
Fig. 2 shows how the CDP1824 chip-select decoding is implemented. The

*Note: The information in this Application Note is useful here; the Evaluation Kit, however, is no longer available.

Table I—CDP1824 RAM operational modes

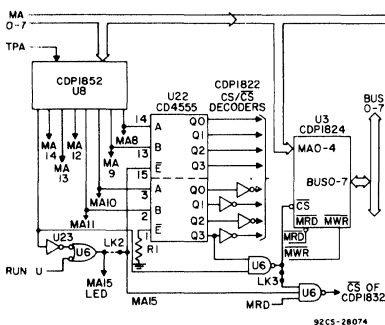
Function	CS	MRD	MWR	Data Pins Status
READ	0	0	X	Output: High/Low Dependent on Data
WRITE	0	1	0	Input, Output Disabled
Not Selected	1	X	X	Output Disabled
Standby	0	1	1	High-Impedance State

Logic 1 = High Logic 0 = Low X = Don't Care



92CS-2759I

Fig. 1 - Functional diagram of CDP1824 32-word x 8-bit static COS/MOS random-access memory.



92CS-28074

Fig. 2 - Implementation of chip-select decoding for the CDP1824 RAM.

eight higher-order address bits are latched in the CDP1852 (Location U8 in the Evaluation Kit) by TPA. The CD4555, a dual binary to 1 of 4 decoder, is used to decode address bit MA8 and MA9 in one section and MA10 and MA11 in the other section. These decodes are used by the CDP1822S RAM system. However, the decode of MA11 and MA10 is also used as part of the CDP1824 chip-select decode together with the MA15 output from the address latch. These two signals are combined in a NAND gate whose output is

connected to the CS terminal of the CDP1824. As a result, the CDP1824 will be selected whenever MA15•MA11•MA10 is true.

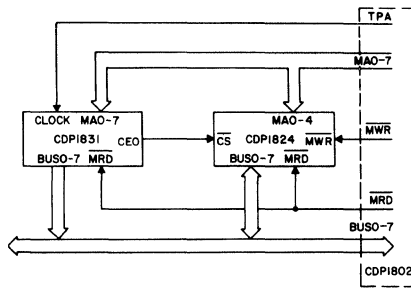
The CDP1824 is spaced 2560 memory locations above the end of the Utility Program. Therefore, if desired, additional memory can be located at 8200₁₆ to 8BFF₁₆ without the need to inhibit the CDP1824.

If it is desired to locate additional memory above location 8C1F₁₆, the CDP1824 must be inhibited. Selection of the CDP1824 can be inhibited by disabling the CD4555 decoder by applying a high voltage level to the enable terminal, pin 1 of the CD4555, whenever the add-on memory at location 8C20₁₆ or above is selected.

For memory expansion above 81FF₁₆, the CDP1832 Utility Program ROM (location U2) must be inhibited. Provisions have been made in the Evaluation Kit to facilitate the necessary logic changes; LINK "LK3" should be removed to disconnect MA15 from the NAND gate which forms the CS input for the Utility Program ROM. Modified CS logic can be externally generated and applied to this point to accommodate additional memory.

DESIGN OF CDP1824-BASED MEMORY SYSTEM

Examples of ROM-RAM systems utilizing the CDP1824 RAM and CDP1831 ROM are illustrated in this section. If desired, these systems can be implemented in the User I/O area provided on the PC card. If necessary, the existing RAM's can easily be inhibited by connecting the enable terminal, pin 15, of the CD4555 decoder to a high voltage level. These examples basically involve techniques for generating the CS command to the memory. Fig. 3 shows a minimum



92CS-28073

Fig. 3 - Minimum memory system utilizing RAM CDP1824 and ROM CDP1831.

RAM-ROM system. The CDP1831 chip-enable output CEO signal is used to enable the CDP1824. If additional ROM's are used, CEO output signals should be

logically OR'ed and connected to the \overline{CS} input of the RAM. RAM/ROM space may also be uniquely defined by the high-order address bit, as illustrated in Fig. 4. This technique is the one used in the Evaluation Kit.

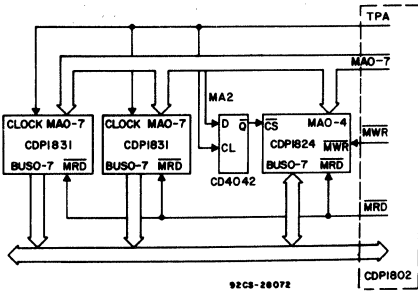


Fig. 4 - Multiple ROM/RAM memory system.

Fig. 5 illustrates a simple technique for interconnecting several CDP1824's and CDP1831 ROM's. The COS/MOS CD4556B, a dual binary to 1 of 4 decoder, is used to perform the necessary decoding. One section is used to decode MA5 and MA6 address bits and to determine which of a maximum of four CDP1824's is selected. The other section can have a maximum of three CDP1831 ROM signals connected to its inputs A, B, and E. The output from the $\overline{Q_0}$ terminal is connected to the enable input E of the RAM decoder section. The 1 of 4 RAM decoder, therefore, will only be enabled if all three CEO signals are in the low state. This condition indicates that no ROM is selected. Thus, by use of a single CD4556 decoder, 1536 bytes of ROM and 128 bytes can be uniquely decoded.

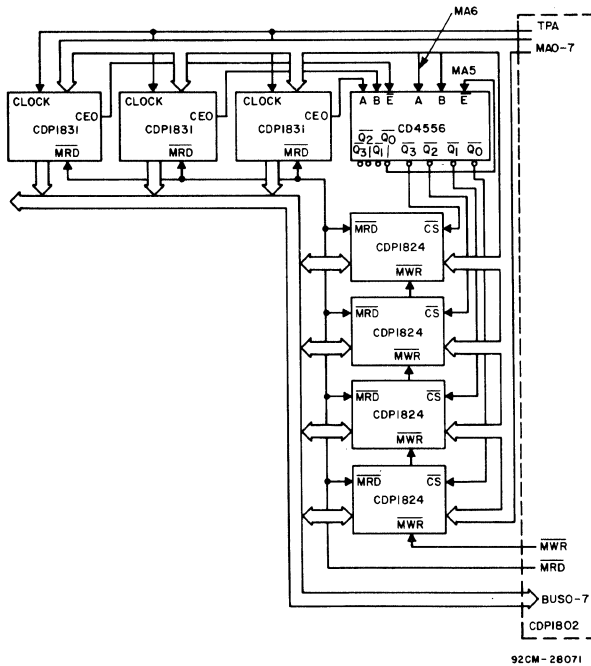


Fig. 5 - Technique for interconnecting several CDP1824 RAM's and CDP1831 ROM's.

Use of the CDP1852 8-Bit I/O Port with RCA Microprocessor Evaluation Kit CDP18S020*

by R.G. Ott

The CDP1852 I/O Port is a versatile circuit permitting fully parallel data transfer between the 8-bit bus of the CDP1802 microprocessor and an external device. The on-chip 8-bit data latch and the service request output signal of the CDP1852 make it an excellent interface circuit for synchronous or asynchronous data transfers. The CDP1852 is mode programmable to function as either an input or an output port and can serve additional functions depending on the use of its control signals. This note describes several applications of the CDP1852 I/O Port and specifically explains its use in the CDP18S020 Evaluation Kit.

DESCRIPTION OF THE CDP1852 FEATURES

The CDP1852, when programmed as an input port (mode=0) or an output port (mode=1), will interface directly with the CDP1802 microprocessor without additional components. When the CDP1852 is used as an input port, data are strobed into the 8-bit register by a high (1) level on the clock line. The high-to-low transition of the clock latches the data in the register and sets the SR service request output to 0. The three-state output drivers are enabled by $CS1 \cdot CS2 = 1$, and the high-to-low transition of $CS1 \cdot CS2$ resets $\overline{SR} = 1$.

When the CDP1852 is used as an output port, data are strobed into the 8-bit register when $CS1 \cdot CS2 \cdot CLOCK = 1$. The data are available at the outputs at all times because the three-state output drivers are always enabled when the mode input = 1. The Service Request (SR) pulse is generated at the termination of $CS1 \cdot CS2 = 1$ and is present (high level) until the next high-to-low transition of the clock.

*Note: The information in this Application Note is useful here; the Evaluation Kit, however, is no longer available.

A CLEAR control is provided for resetting the port's register and \overline{SR} (input mode) or SR (output mode) signal. It is important to note that the polarities of service request (SR/\overline{SR}) and chip select 1 ($CS1/\overline{CS1}$) change when the polarity of the mode input signal is changed.

A functional block diagram of the I/O port is shown in Fig.1. For the electrical characteristics of the I/O Port including loading and timing specifications, refer to the CDP1852 data sheet.

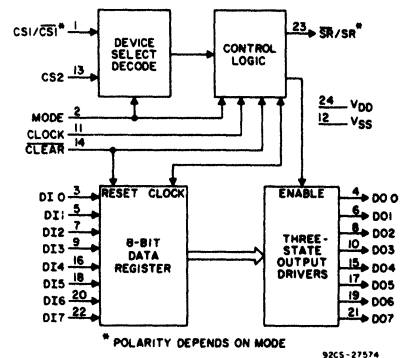


Fig. 1 - Functional diagram of CDP1852 8-bit Input/Output Port.

APPLICATION OF CDP1852 AS AN INPUT PORT, OUTPUT PORT, AND ADDRESS LATCH

The CDP18S020 Evaluation Kit uses the CDP1852 I/O port in three different applications: as an input port, as an output port, and as an address latch. In each of these applications, the CDP1852 is controlled by microprocessor signals.

As an **input port** the CDP1852 (location U5 in the Evaluation Kit) is used to latch 8 bits of data from an external device. The Evaluation Kit connections for this ap-

plication and the associated timing diagram are given in Fig. 2. The mode in-

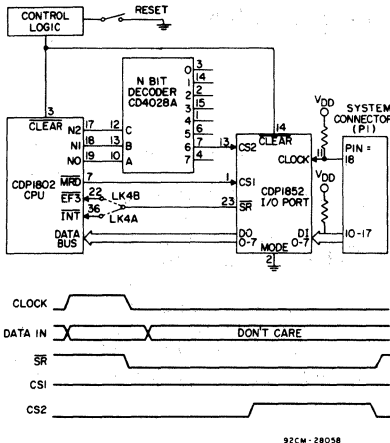


Fig. 2 - Circuit diagram of CDP1852 used as an input port with RCA Microprocessor Evaluation Kit CDP18S020 and associated timing diagram.

put is tied low (mode = 0). When data are latched on the high-to-low transition of the external clock, the SR service request is set low (SR = 0). This signal can be connected either to the microprocessor EF3 or INT inputs depending on the control program being used. An optional connection is made via a jumper on the Evaluation Kit Board. When the microprocessor responds to a service request, the input port is enabled by MRD (connected to the CS1 input) which is normally high and will remain high. The decoded "6" output of the N bit decoder (connected to the CS2 input) selects the I/O port. In a minimal system (up to three ports) it is possible to bypass the N bit decoder and connect an N line directly to CS2. After the data has

been written into memory, the service request is reset (SR = 1) on the trailing edge of the N line. The input port is then ready to accept the next data byte from the external device.

An example of a simple system using the input port follows. Switches can be connected to the data inputs. These lines come from the following connector pins: DI7 to P1-10, DI6 to P1-11, DI5 to P1-12, DI4 to P1-13, DI3 to P1-14, DI2 to P1-15, DI1 to P1-16, DI0 to P1-17. The switches can be used to connect either a logic 1 (VDD voltage) or a logic 0 (ground) to the input port data inputs. A positive pulse for the clock can be generated by connecting a switch to connector P1-18. Once the data switches are set, the clock can be pulsed. Data are transferred to the input port, and the service request signal goes to ground (logic 0).

Table I gives a machine-language routine for acknowledging a service request when the SR line is connected to the EF3 input of the microprocessor. Verification that the correct byte was input into memory can be obtained by using utility program UT4 to read the memory location where the input byte was stored (in this case, location 0F). To read the memory location, press the Reset and then the RUN U button on the Evaluation Kit. Typing a carriage on the terminal (for some terminals, a line feed is required) will then cause an asterisk (*) to be typed. The byte can be read out by typing ?MF 1 and carriage return. Two hexadecimal digits will be typed and can be checked against the input byte set with switches.

As an output port, the CDP1852 (location U4 in the Evaluation Kit) is used to latch 8 bits of data from the CDP1802 data bus for asynchronous transfer to an external device. The Evaluation Kit connections for this application and the associated timing diagram are given in Fig. 3. The mode input is tied high

Table I - Machine-language routine for acknowledging a service request.

!M			
0000 ;	0001	ORG 00	..PROGRAM STARTS AT M(0000)
0000 F80F;	0002	LDI #0F	..SET R5 TO POINT TO
0002 A5;	0003	PLO R5	..M(000F)
0003 F800;	0004	LDI 00	
0005 B5;	0005	PHI R5	
0006 E5;	0006	SEX R5	..SET X TO R5
0007 3E07;	0007 WAIT:	BN3 *	..WAIT FOR EF3 = 1
0009 6E;	0008	INP 6	..INPUT FROM INPUT PORT
000A 3007;	0009	BR WAIT	..LOOP BACK TO WAIT
000C ;	0010	ORG * + 3	
000F ;	0011 STACK:	ORG *	..LOCATION FOR STORING INPUT DATA
000F ;	0012	END	
0000			

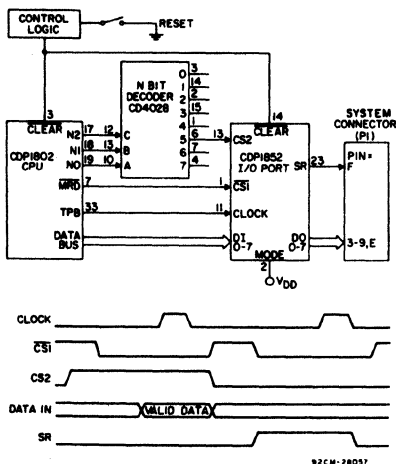


Fig. 3 - Circuit diagram of CDP1852 used as an output port with RCA Microprocessor Evaluation Kit CDP18S020 and associated timing diagram.

(mode = 1). To begin the sequence, the microprocessor executes an output instruction. The output of the N bit decoder (connected to the CS2 input) selects the output port. When MRD (connected to the CS1 input) goes low, the output port is enabled and data is latched on the trailing edge of TPB (connected to the clock input). On the trailing (high-to-low) edge of CS1 • CS2. SR is asserted (SR = 1) and held valid until the trailing edge of the next TPB, which signals the external device that new data is available. If it is necessary to extend the duration of this service request, an external latch must be provided.

The previous example for use of an input port can be extended to use both the input and output ports. Instead of connecting switches to the input port data inputs, the corresponding output port data outputs should be connected to the input port data inputs as follows: P1-9 to P1-17, P1-8 to P1-16, P1-7 to P1-15, P1-6 to P1-14, P1-5 to P1-13, P1-4 to P1-12, P1-3 to P1-11, P1-E to P1-10. The output port service request signal can be used to clock the input port by connecting P1-E to P1-18. The input port service request line is connected to the microprocessor EF3 input.

Table II gives a machine-language routine for outputting a data byte to the output port and then reading it back into memory (location F) from the input port. Verification that data output and input occurred correctly can be obtained by reading the byte which was output and the byte which was input from memory using UT4. To make this verification, press the Reset and then the RUN U button on the Evaluation Kit. Typing a carriage return on the Teletype (for some terminals, a line feed) will then cause an asterisk (*) to be typed. The original byte can then be read out by typing ?M2 1 and carriage return. The byte after data output and input can be read out by typing ?MF 1 and carriage return. Correct connection and functioning of the CDP1852 I/O Ports are verified when both bytes are the same. For example, if location 02 contained 2B, then location 0F will contain 2B when the transfer is complete.

Note that the decode (decode "6") of the N bits used to select the input port is different from the decode (decode "5") used to select the output port. If the same decode is used, the input port SR signal will be reset (SR = 1) at the beginning of every cycle in which the output port is selected. The use of different N decodes

Table II - Machine-language routine for outputting a data byte to the output port and then reading it back into memory from the input port

!M				
0000	;	0001	ORG 00	..PROGRAM STARTS AT M(0000)
0000	E0;	0002	SEX R0	..SET X TO R0
0001	65;	0003	OUT 5	..OUTPUT IMMEDIATE
0002	FF;	0004	,#FF	
0003	F80F;	0005	LDI #F	..SET R5 TO POINT TO
0005	A5;	0006	PLO R5	..M(000F)
0006	F800;	0007	LDI 00	
0008	B5;	0008	PHI R5	
0009	E5;	0009	SEX R5	..SET X TO R5
000A	3E0A;	0010 WAIT:	BN3 *	..WAIT FOR EF3 = 1
000C	6E;	0011	INP 6	..INPUT DATA FROM INPUT PORT
000D	300A;	0012	BR WAIT	..LOOP BACK TO WAIT
000F	;	0013 STACK:	ORG *	..LOCATION FOR STORING DATA
000F	;	0014	END	
0000				

Use of CMOS-SOS RAM CDP1822 With RCA Microprocessor Evaluation Kit CDP18S020*

by J. R. Oberman

The CDP1822 is a 256 x 4, CMOS-SOS, static, random-access memory. It can be used directly in CDP1802 microprocessor systems such as the CDP18S020 Evaluation Kit and requires no additional interface components. The CMOS-SOS technology has all the advantages of CMOS plus the high-speed, low-dynamic-power performance of SOS.

The CDP18S020 Evaluation Kit is designed to accept 32 CDP1822 RAM's in prewired locations forming 4K bytes of RAM storage. The Evaluation Kit is provided with two CDP1822's, 256 bytes of RAM storage, and the necessary decoders to facilitate expansion, if desired, to 4K.

This Application Note describes the CDP1822, its operation, and its application in the CDP18S020 Evaluation Kit.

DESCRIPTION OF CDP1822 FEATURES

The CDP1822 is contained in a 22-lead dual-in-line package consisting of eight address inputs, an output disable control, MRD, a memory write control, MWR, two chip-select controls, CS1, CS2, four data-input and four data-output terminals, and two voltage connections.

The eight address inputs, MA0-MA7, are buffered and decoded to uniquely select 1 of 256 four-bit words. The four data-output terminals are driven from three-state drivers enabled by the MRD

signal when the device is selected. Valid data appears at the output (MRD = 0) of the CDP1822 in one access time following the latest address change to a selected chip. The output data is valid until either the MRD signal goes high or the device is deselected (CS1 = 1 or CS2 = 0). Operation is determined by the state of the MRD and MWR lines. These inputs are driven directly from the CDP1802 microprocessor. The operational modes are listed in Table I and a functional diagram of the memory is shown in Fig. 1. The CDP1822 requires no precharge or clocked signals for proper operation.

The electrical characteristics of the CDP1822 are given in its data sheet. It should be noted that because the CDP1822 has a maximum recommended operation voltage of 10 volts, the Evaluation Kit V_{CC} should be limited to 10 volts when it uses CDP1822 RAM's.

APPLICATION OF CDP1822 IN EVALUATION KIT MEMORY SYSTEM

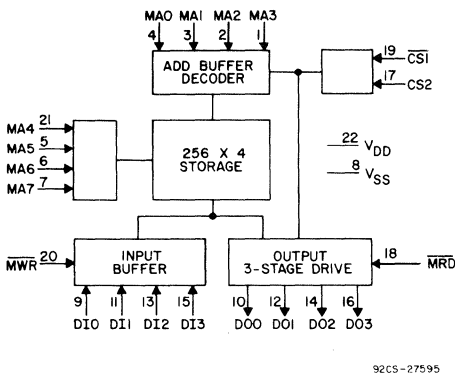
The CDP18S020 Evaluation Kit is prewired for 4K bytes of CDP1822, 256 x 4 RAM storage at locations (0000)₁₆-(0FFF)₁₆. The basic kit is supplied with two CDP1822's forming a 256-byte RAM. At maximum expansion it will contain 32 CDP1822's. The necessary decoding logic for the 4K expansion is also included in the basic kit. Expansion is accomplished by adding additional CDP1822's at the prewired locations in 256-byte multiples (two CDP1822's at a time). To assure contiguous memory when additional devices are added, the CDP1822's should be placed in their designated locations in the Evaluation Kit (U24-U55) in numerical order on the printed circuit card. Fig. 2 shows the memory interconnections. The MRD, MWR, and eight address lines (MA0-MA7) are connected to all 32 RAM locations and directly to the CDP1802

*Note: The information in this Application Note is useful here; the Evaluation Kit, however, is no longer available.

Table I - CDP1822 Operational Modes

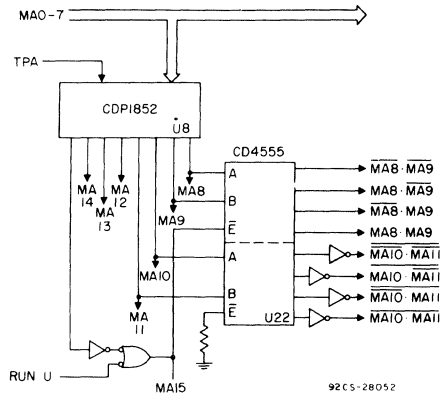
Function	\overline{MWR}	$\overline{CS1}$	CS2	\overline{MRD}	Data Out DO
READ	1	0	1	0	Storage State of Addressed Cell
WRITE (Output Disabled)	0	0	1	1	High-Impedance
WRITE	0	0	1	0	New Data In State
Standby	X	1	X	X	High-Impedance
	X	X	0	X	High-Impedance
	1	0	1	1	High-Impedance

Logic 1 = High Logic 0 = Low X = Don't Care



92CS-27595

Fig. 1—Functional diagram of CDP1822 256-word x 4-bit static COS/MOS silicon-on-sapphire random-access memory.



92CS-28052

Fig. 2—Interconnections of CDP1822 RAM with CDP1802 microprocessor in Evaluation Kit CDP18S020 system.

microprocessor. In the CDP18S020, the four data-in and four data-out terminals of the CDP1822 are connected together and to the CDP1802 bidirectional data bus.

Chip selection is determined by enabling the two chip-select inputs, CS1 and CS2, for each pair of CDP1822's. The chip selects are arranged in a 4 x 4 matrix. This arrangement provides unique selection of 1 of 16 pairs of RAM's and requires only an eight-wire interconnect, as indicated in Fig. 2. Fig. 3 shows the address decode logic for the full RAM system. The I/O port, CDP1852 (U8) is used as an address latch for the higher-order address bits, MA8-MA15. The MA8-MA11 outputs from the latch are decoded by a CD4555B (U22) dual binary to 1 of 4 decoder. MA8-MA9 are decoded in one section and MA10-MA11 in the other section of the decoder. The four outputs from the MA10-MA11 decoder section are inverted and connect to the CS1 terminals of the RAM's; the other four outputs are connected to the CS2 terminals. Together, they form a 4 x 4 selection matrix. Each section of the decoder has a separate enable (E) input and must be enabled in order to select the RAM system. The enable input of the MA8-MA9 decoder is prewired to ground (continuously enabled) and intended for

use for additional memory expansion. The enable input is connected to MA15. As a result, the RAM is inhibited for all address locations of (8000)₁₆ or above. This simple technique is used for separating the Utility ROM and its support RAM from the RAM system. If a finer level of decoding is desired for an expanded RAM system, the second enable input can be used. The CDP1824 RAM Application Note ICAN-6537 discusses one application for this input.

CDP1822 WRITE PROTECTION AND BATTERY HOLD-UP

Memory write protection and provisions for standby power have been included in the design of the Evaluation Kit RAM system to permit ROM simulation. A write-inhibit switch (SW-1, -2, -3, and -4) is placed in the write (MWR) line for each 1K of RAM. The user manually sets the switches in any combination to inhibit a write signal from occurring in the protected sections of memory and destroying the information stored there.

Because of the low current drain, battery power may be substituted for VDD and VCC where non-volatile memory is desired.

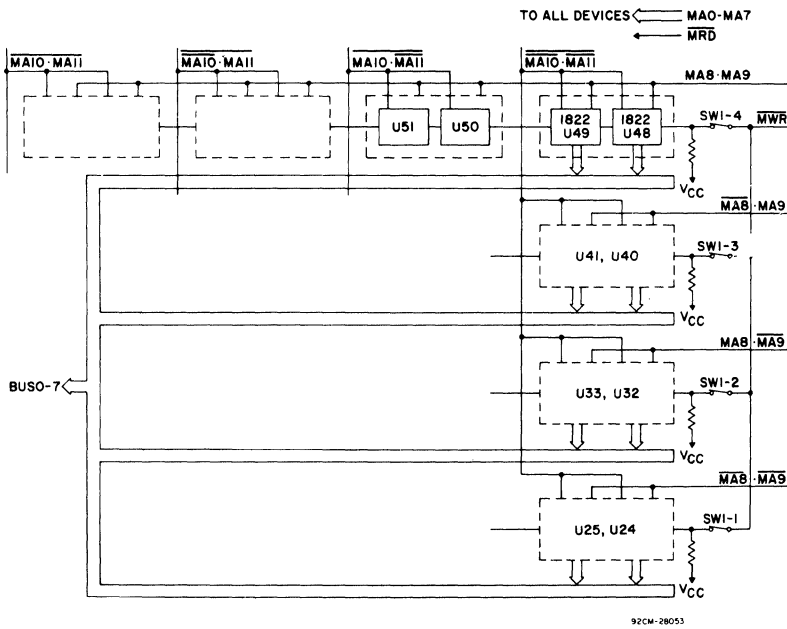


Fig. 3—Address decode logic for full CDP1822 RAM system.

Register-Based Output Function for RCA COSMAC Microprocessors

by N. P. Swales

This Note describes a circuit for use with RCA COSMAC Microprocessors; the circuit provides the capability to output information from any of the 16 general-purpose scratchpad registers contained within the CPU. This circuit is of particular interest to users constrained to operating with memory systems containing only ROM, or to users wishing to transfer 16 bits of data with a single Output instruction. The information given and the terminology used are presented with the assumption that the reader is familiar with the COSMAC architecture and manuals.¹

Although other logic implementations of the functions described in this Note are realizable, the examples given employ the CDP1852 I/O Port. For further information on the CDP1852, the user should refer to the data sheet.²

The circuit in this Note provides a mechanism for transferring data from one of the 16 general-purpose registers contained in the COSMAC CPU to an I/O device. The data is transferred to the I/O device via the Memory Address Bus from the CPU rather than via the Bidirectional Data Bus from the Memory System.

One-Byte Transfer Configuration

Fig. 1 shows a simple configuration de-

signed to transfer the data contained in the low-order byte of one of the 16 CPU registers to a CDP1852 I/O port. The I/O port is configured in a manner similar to that described in the data sheet for the CDP1852; however, instead of connecting the Data In lines of the CDP1852 to the CPU Data Bus, they are connected to the CPU Address Bus. During the execute cycle of the Output instruction which is to transfer the data to the I/O port, the contents of the CPU register specified by the X register (R(X)) are transferred to the memory system via the Memory Address Bus (the most significant byte first followed by the least significant byte). At the time during which TPB occurs, the least significant byte of R(X) is available on the memory address bus lines and is therefore transferred into the I/O port. It should be noted that the output data contained in the CPU register is incremented by one, and that this operation causes a read from the memory system. However, the memory data byte resulting from this operation is ignored by the rest of the system.

Two-Byte Transfer Configuration

Fig. 2 shows a configuration similar to Fig. 1 which may be used to transfer both

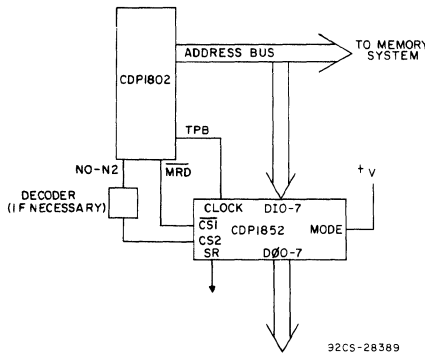


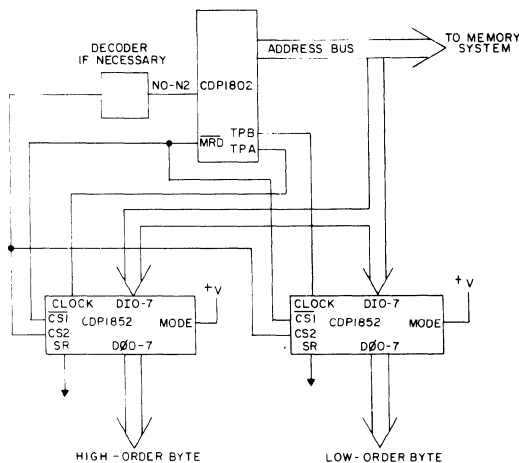
Fig. 1 - Circuit configuration for transferring data from the least significant byte of one of the CDP1802's 16 general-purpose registers to an I/O device.

bytes of one of the CPU's general-purpose registers to an I/O device. TPA is used to transfer the high-order byte of the register into the CDP1852 on the left, and TPB transfers the low-order byte into the CDP1852 on the right. In this configuration, care should be taken to meet the timing requirements of the CDP1852 and the CDP1802 for

the transfer of the most significant byte of data.

References

1. User Manual for the CDP1802 COSMAC Microprocessor, MPM-201A.
2. CDP1852 Series, 8-Bit Input/Output Port; Preliminary Data Sheet.



92CM-28390

Fig. 2 – Circuit configuration for transferring the full contents (two bytes) of one of the CDP1802's 16 general-purpose registers to an I/O device.

Design of Clock Generators for Use with RCA COSMAC Microprocessor CDP1802

by D. Hillman

Clock signal generation for the CDP1802 COSMAC Microprocessor is simple and straightforward. The CDP1802 features of static operation, single-phase clock input, and the on-chip oscillator amplifier make practical the use of a low-cost, highly stable, crystal-controlled oscillator as its clock generator. The design of external oscillators for this purpose, crystal or RC controlled, is equally straightforward and they require only minimal circuitry. In addition to the oscillator amplifier, the CDP1802 incorporates all necessary start/stop logic on-chip. This application note describes clock generator designs suitable for various applications.

Crystal Oscillator Design

The basic oscillator circuit for the CDP1802 consists of the on-chip amplifier and an external feedback network as illustrated in Fig. 1. For oscillation to occur, the gain of

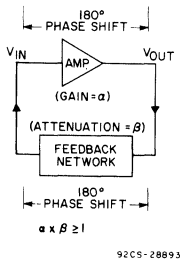


Fig. 1 — Basic oscillator circuit.

the amplifier (α) times the attenuation (β) of the feedback network must be greater than or equal to one. In addition, the total phase shift through the amplifier and feedback network must be equal to N times 360 degrees, where N is an integer. Oscillations occur in any system in which the amplified signal is returned in phase to the amplifier after being attenuated less than it was originally amplified.

The frequency stability of an oscillator is primarily dependent upon the phase-changing properties of the feedback network. Because of their high Q and inherent frequency stability, quartz crystals are commonly used in the feedback network.

A parallel resonant oscillator circuit is shown in Fig. 2. The phase angle for the type

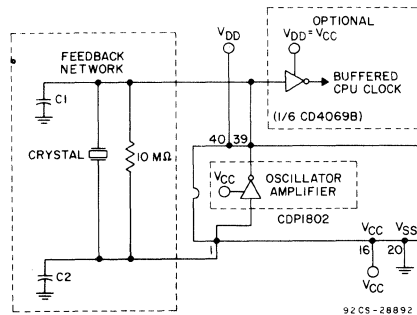


Fig. 2 — Parallel resonant oscillator circuit.

of feedback network shown in this figure is extremely sensitive to a change in frequency, a condition necessary for stable oscillation. If the equivalent resistance of the crystal is in fact zero (infinite Q), a change in phase angle of the feedback circuit would not cause any change in oscillator frequency. Therefore, for an oscillator of highest stability, the Q of the crystal should be as high as possible. In general, Q increases with increasing frequency.

The crystal load capacitance, C_L , is defined as the series sum of C_1 and C_2 . Higher values of crystal load capacitance generally improve frequency stability, but also increase power consumption. The choice of equivalent load capacitance (usually specified to the crystal suppliers) only fixes the series sum of the two capacitors C_1 and C_2 . The value of the amplifier output capacitance C_1 should not be fixed. A trimmer should be connected in parallel with, or used in place of, a fixed output capacitor to permit compensation for variation in stray capacitance and circuit component values.

The required capacitance range for the oscillator trimmer capacitor is determined by the variation in oscillation frequency with load capacitance. The total trimming range is mainly a function of the crystal characteristics. For a more detailed analysis, see Reference 4.

Practical Oscillator Circuits

The amplifier, feedback network, and crystal considerations discussed in the preceding paragraphs can be combined for the design of a crystal-controlled oscillator for the CDP1802. The majority of microprocessor applications do not require the frequency of oscillation to be so exact as to require oscillator trimming. An "untrimmed" crystal oscillator will be within 1% of its specified crystal frequency. For most microprocessor applications the following simple guidelines can be used.

1. The crystal should be connected between terminals 1 and 39 of the CDP1802.
2. For crystal frequencies between 100 kHz and 6.4 MHz, a 10- to 22-megohm feedback resistor should be used in parallel with the crystal.
3. Capacitors C₁ and C₂ are not required but a value of between 20 and 30 pF for each is recommended to improve stability.

It should be noted that the on-chip oscillator and timing generator are capable of operating at frequencies higher than the microprocessor maximum operating frequency. For reliable operation, the crystal frequency must always be less than or equal to the maximum operating frequency specified in the CDP1802 data sheet.

A practical example, the CDP18S020 Evaluation Kit oscillator, consists of a 10-megohm feedback resistor and a 2-MHz AT-cut crystal, both connected in parallel across terminals 1 and 39 of the CDP1802. (Crystal: Part No. X023303; C_L=15 pF; Series M1; holder, series HC330; made by Turotel, Inc., 13402 S. 71 Highway, Grandview, Missouri 68030.) Provisions for oscillator capacitors are made in the Evaluation Kit, but their use is not required. The increase in oscillator stability with respect to supply voltage that can be obtained by adding the capacitors is shown in Fig. 3.

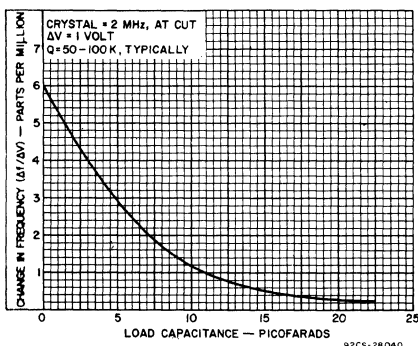


Fig. 3 - Stability of CDP1802 crystal oscillator as a function of load capacitance value.

The amplifier stability also depends upon the value of the resistor in the feedback network. Fig. 4 shows the relationship between the feedback resistor value and oscillator stability. The curve indicates that 10 megohms is an adequate value for the feedback resistor.

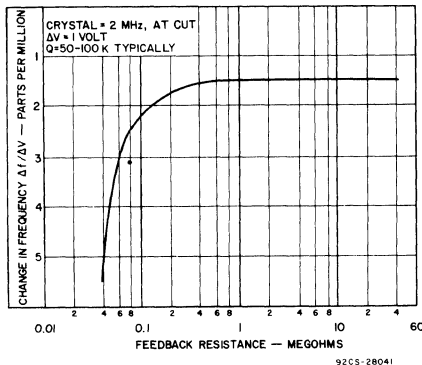


Fig. 4 - Stability of CDP1802 crystal oscillator as a function of feedback resistance value.

External Clock Generators

For low-frequency applications (less than 500 kHz) a cost-effective approach may be to use external RC-controlled oscillators. Three simple RC-controlled oscillators that may be used to clock the CDP1802 are shown in Fig. 5. When an external clock is used in

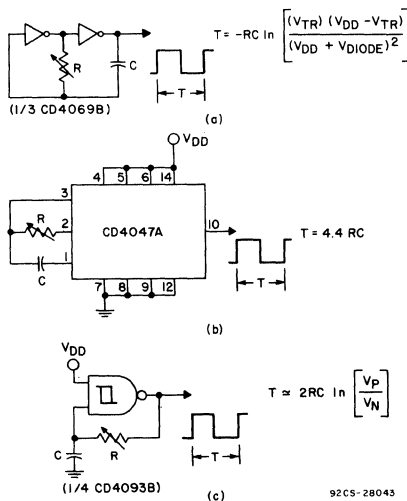


Fig. 5 - Three simple RC-controlled oscillator circuits suitable for use as external clock for CDP1802 microprocessor (output connected to pin 1 through Evaluation Kit P2-W).
 (a) Inverter type oscillator (see References 5 and 6).
 (b) RC oscillator using digital IC CD4047A (see References 5 and 6).
 (c) Schmitt-trigger-type RC oscillator (see CD4093B data sheet).

high-noise environments, a 20- to 30-pF capacitor between terminal 39 (XTAL) of the CDP1802 and ground may be used to increase the microprocessor noise immunity.

The selection of the R and C should be compatible with system requirements. The capacitor should be non-polarized and have low leakage. There is no upper limit for either R or C values to maintain oscillation. However, C should be larger than the inherent stray capacitance. R must be larger than the output impedance of the COS/MOS device, which is typically hundreds of ohms. In addition, with very large values of R, some short-term instability with respect to time may be noted. Based on these considerations recommended values for these components are:

C - greater than 100 pF, up to any practical value

R - greater than 10 kilohms, but less than one megohm

With large values of R and C, the circuit in Fig. 5(c) can be used. This circuit, because of its hysteresis, eliminates multiple output pulses caused by noise on the input RC waveform. For a more detailed analysis, see References 5 and 6.

Clock Buffering

In some applications it may be desirable

to supply the CPU clock signal to other system components. In such cases the loading on the oscillator circuit should be minimized by buffering the clock through a COS/MOS inverter, as shown in Fig. 2. The loading presented by the inverter will be mainly capacitive, about 5 picofarads, and can usually be neglected in non-critical designs. The buffer should be located close to the crystal in order to minimize stray capacitance.

When the crystal oscillator is being trimmed to its desired frequency, the buffered clock technique should also be used to prevent the oscillator from being loaded by the frequency counter.

References

1. CDP1802 data sheet.
2. CD4047A data sheet, File No. 623.
3. CD4093B data sheet, File No. 836.
4. "Timekeeping Advances Through COS/MOS Technology", ICAN-6086.
5. "Using the CD4047A in COS/MOS Timing Applications", ICAN-6230.
6. "Astable and Monostable Oscillators Using RCA COS/MOS Digital Integrated Circuits", ICAN-6267.
7. User Manual for the RCA CDP1802 COSMAC Microprocessor, MPM-201.

Power-On Reset/Run Circuits for the RCA CDP1802 COSMAC Microprocessor

by W. F. Clark

This Note describes several circuits which enable a power-on reset/run capability for COSMAC microprocessor systems. It is assumed that the user is familiar with the hardware considerations for the CDP1802 as explained in the **User Manual for the CDP1802 COSMAC Microprocessor, MPM-201C**.

The power-on reset/run facility is useful in systems which periodically incorporate a power-down phase during system operation. During this power-down phase, care should be taken to prevent signals from reaching any of the microprocessor inputs or outputs because these signals could be coupled through the substrate protection diodes to the V_{DD} line and repower the system.

Clock Input Considerations

The waveforms in Fig.1 show the behavior of the supply lines and CLEAR line for resetting the microprocessor using two different modes of clock inputs, the on-chip crystal oscillator and an external clock source. The required reset phasing is determined by the type of clock used in the system. When the on-chip oscillator is used, the delay in oscillator start-up depends on the loading on the CLOCK input. With

5-picofarad decoupling capacitors on the CLOCK and XTAL lines, a delay of 10 milliseconds is typical for oscillator start up at 25°C. To assure proper operation, the reset pulse should be active until the oscillator has stabilized.

When an external clock is used, it should be gated with the power-on signal. The reset pulse should allow for the V_{DD} , V_{CC} time constant and can be terminated when the clock is in a valid state or, typically, 20 microseconds following power-on. If the reset pulse terminates before the clock is running, internal registers I and N and the Q output will reset. Registers X, P, and R(0) will not reset, however, until the initialization cycle is completed.

Typical Reset Circuits

Some typical reset circuits are shown in Fig.2. The basic one-shot circuit is shown in Fig.2(a). The circuit in Fig.2(b) permits manual reset and allows access to the LOAD mode. The circuits in Figs.2(c) and 2(d) eliminate feedback resistors by using the CD4093 Quad 2-Input NAND Schmitt Triggers. Other variations of these circuits are possible following the above constraints.

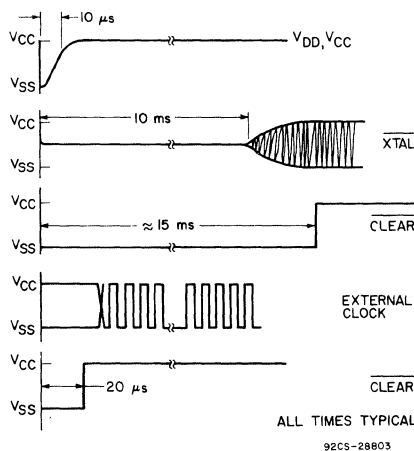
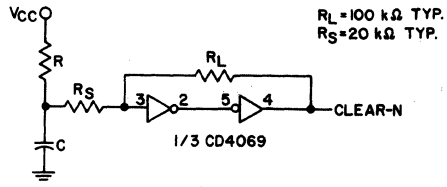
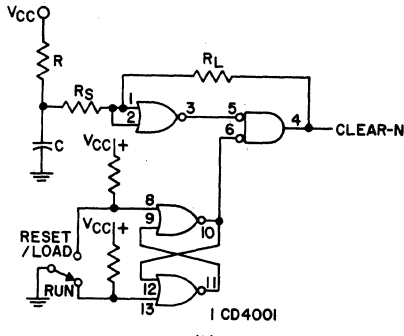


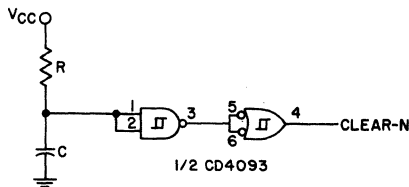
Fig.1 - Typical start-up waveforms.



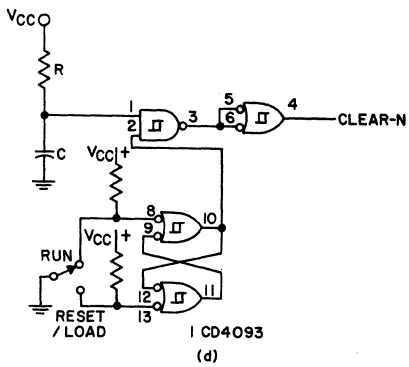
(a)



(b)



(c)



(d)

92CM-28804

Fig.2 - Typical power-on reset/run circuits. IC supply leads are V_{CC} and V_{SS} or ground.

Interfacing Analog and Digital Displays with CMOS Integrated Circuits

by J. E. Gillberg

Many forms of displays are available for interfacing digital and analog information from electronic circuits with the individual end user. The display choice generally takes into consideration not only technical feasibility but also visual impact and often aesthetic appeal. Until recently, the analog display, primarily motors (both synchronous and stepper) with gears, hands, or drums, has been the most widely used. At present, however, new developments are making the digital display the more dominant method.

This Note describes some of the COS/MOS integrated circuits most suitable for interfacing the electronic circuit and the display. In the case of digital displays, it describes basic display operation to help simplify the equipment designer's task in selecting both the most appropriate display and the most suitable interfacing device.

Analog Display Drivers

Analog displays are usually driven from either a synchronous motor or a stepper

motor. The synchronous motor receives an incoming signal at a frequency of approximately 60 Hz and continuously rotates at that frequency. The stepper motor receives an incoming signal at about 0.5 to 2 Hz and rotates only during the active pulse interval. The stepper motor gives the effect of a non-continuous movement of the motor or wheel.

One of the major users of digital circuits with analog displays is the timekeeping market. This market has continued to use analog displays because of the many basic advantages of the familiar clock or watch face with moving hands. These advantages include low cost, high reliability, simplified electronics, familiarity of display mode, and low current drain.

A number of IC's are available for interfacing the electronic clock circuitry and the analog display. An excellent example is the CD4045, a COS/MOS 21-Stage Counter. As shown in Fig. 1, this device can be used in timing applications not only to generate the crystal oscillator output, but also, because of its output current capability, to directly drive a stepper motor. Fig. 2 gives curves illustrating the current capabilities of the CD4045.

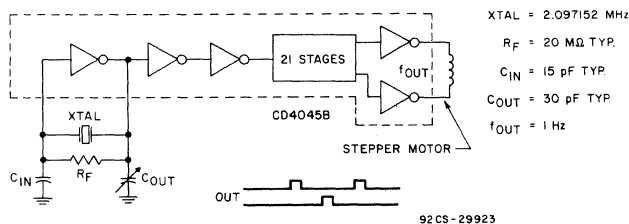


Fig. 1 - CD4045, COS/MOS 21-stage counter, used to generate crystal oscillator output and to drive stepper motor.

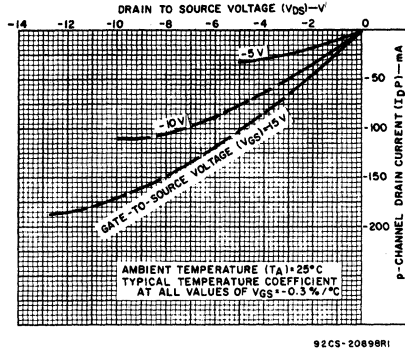
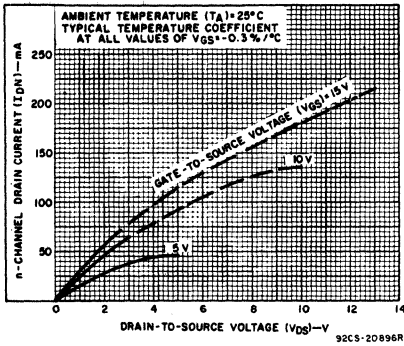


Fig. 2 - Typical output n-channel and p-channel drain characteristics of the CD4045.

One method of reducing the current drain of a stepper motor is to terminate the incoming pulse at the precise moment the armature achieves enough momentum to rotate to the next position without any additional current. The Low-Voltage COS/MOS Analog Timepiece Circuit CD22010E has the capability of detecting, as shown in Fig. 3, when no additional current is required by the motor. It operates as follows. At the beginning of the output pulse because the load is inductive no current will immediately flow ($V = L di/dt$) and the voltage at the output will be at ground, as shown in Fig. 4 at t_0 . After time, the current will begin to flow into the pull-down n-channel transistor of the CD22010E. This current

flow raises the output voltage until the motor begins to rotate and cause a back electro-motive force thereby reducing the voltage at the output. Once the motor has achieved enough momentum to move on its own inertia, however, any added current again raises the output voltage. The time interval from t_0 to t_3 in Fig. 4 is the nominal output pulse. Time t_1 indicates the end of an internal activation period after which any rising edge on the output will trigger internal circuitry to terminate the pulse width, thus saving battery current.

The battery-operated wall clock is one of the major areas for analog displays primarily because of the low-voltage (1.5 to 3.0 V typical) and low-current (60 A typical) operation. A number of display interface circuits are available for this application. The most suitable depends upon the type motor and the voltage being used. Several of these circuits are illustrated in Fig. 5. In Fig. 5(c), the

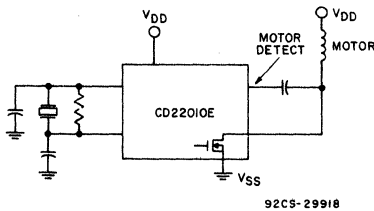


Fig. 3 - CD22010E, a low-voltage COS/MOS analog timepiece circuit, used to detect status of stepper motor current.

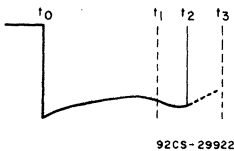


Fig. 4 - Nominal output pulse of stepper motor.

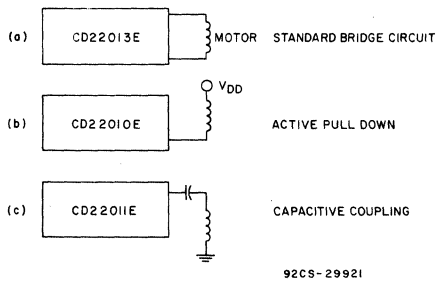


Fig. 5 - Typical motor interface circuits for analog watch or wall clock circuits.

capacitor C_D increases the maximum pulse or spike current supplied to the motor. The value of this added capacitor is typically between 1 and 10 microfarads and is dependent upon the frequency of operation.

Many integrated circuits are available from RCA that can be used to interface an analog high-current-drive display. Table I lists several of these devices. For technical data on these devices see the RCA INTEGRATED CIRCUITS DATABOOK SSD-250 or the technical data sheet for the specific device.

Liquid Crystal Displays

The most important advantage of the liquid crystal display is its very low power consumption, typically 50 microwatts per character. The reason for this low power consumption is that the liquid crystal display does not generate or emit light, but controls reflected or transmitted light generated elsewhere.

The liquid crystal display device consists of a layer of liquid crystal material sealed between two conductive-coated glass plates. The liquid crystals are fluids

Table I – Analog-Display Driver and Counter COS/MOS Integrated Circuits

Type	Family Dev. No.	Package	Volts	Freq.	Description
Clocks					
CD22010E	TA6656	8-DIP	1.5	32 kHz	Portescap stepping-motor drive, with pulse-width control (1 Hz)
CD22011E	TA10294	8-DIP	1.5	4 MHz	SOS stepping-motor drive (2 Hz push-pull)
Auto Clocks					
CD22012E	TA6489	14-DIP	12	4 MHz	Quartz analog auto clock (0.5 Hz push-pull)
CD22013E	TA10176	8-DIP	12	3 MHz	Quartz analog auto clock (64 Hz push-pull)
CD22014E	TA6817	8-DIP	12	4 MHz	Quartz analog clock (60 Hz)
CD22015E	TA10177	8-DIP	12	2 MHz	Quartz analog auto clock (30 Hz push-pull)
Industrial Timers					
CD22017E		16-DIP	10		Universal industrial timer

Digital Display Systems

With the development of MSI and LSI, digital displays emerged as an important method of information transfer and many new display systems appeared. The most popular or promising types of digital displays are listed in Table II along with a brief summary of their major advantages and disadvantages. In the following material each display system is discussed with the emphasis on adaptability to interfacing electronic circuitry.

having molecule alignment characteristics very similar to those of solid crystals. The alignment of the molecules can be changed by the application of an ac signal. This change in alignment can produce image patterns determined by the physical construction of the device. Electrical contact is made to the liquid crystal by means of a transparent conductor. Because the image pattern depends upon molecular alignment, the direction which light strikes the liquid crystal is very critical. As a result, light polarizers are

Table II – Digital Display Technologies

Type	Advantage	Disadvantage
Liquid Crystal	Low power, low voltage	AC signal – difficult to multiplex
Light-emitting diode	Low cost, simple interface	High current, visibility
Gas discharge	Easily read	High voltage
Fluorescent	Low segment current, low cost	High filament current/fragile
Incandescent	Brightness, low cost	High current

attached to the front and back to control whether the display is dark on a light background or light on a dark background. Fig. 6 shows the sandwich-type construction and the arrows illustrate the molecular polarization of a liquid crystal material resulting from an ac field.

There are two basic types of LCD's: dynamic scattering devices and field-effect

ambient light is available. In applications where the ambient light is small, the transmissive display could be used with some form of back lighting.

Liquid-crystal devices require an ac drive signal having no dc component. A dc component can cause an electrolysis plating action which can eventually damage the display. For field-effect

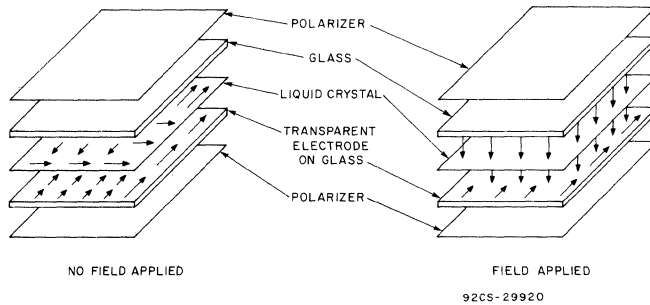


Fig. 6 - Basic operation of field-effect type liquid crystal device showing molecule alignment and major elements.

devices. When an ac field is applied to a dynamic scattering liquid crystal, the molecules which are normally aligned, and hence transparent, are rearranged to scatter any available light, and the display becomes opaque. The field-effect type displays a visual change when the molecule alignment is rotated from one plane to another, as illustrated in Fig. 6. The field-effect liquid-crystal device has become the more popular display.

Two kinds of liquid-crystal devices are available in either the dynamic-scattering or field-effect category: reflective and transmissive. The only difference is that for the former, reflective material is added to the back of the display to reflect the light entering the front. This type is well-suited for applications where substantial

displays, this drive signal may be from 2 to 10 volts at 60 to 10,000 hertz; for dynamic-scattering devices, the signal may be from 7 to 30 volts at 20 to 400 hertz.

When a liquid-crystal segment is activated by a drive signal, the phase relation between it and the transparent electrode applied to the glass backplane is 180° and a visual display results. When no drive signal is applied to the segment, the backplane and segment are in phase and the visual display is off.

The usual method of activating a segment is to apply a square wave which is out of phase with the square wave applied to the common backplane. As shown in Fig. 7, when the segment square wave is in phase with the back-plane square wave,

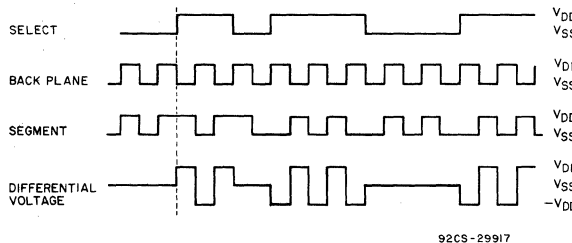


Fig. 7 - Timing diagram of liquid crystal element voltages showing how segments are activated by square waves to avoid damaging effects of a dc component.

the segment is not activated. By the use of a square wave for both the common (backplane) and the selected segment drive signal, the effective dc voltage across the display is always good regardless of whether the display is activated or not.

Liquid crystals offer the important advantages of requiring very little power and low voltage. Their disadvantages are:

1. Because they need an ac signal for operation, multiplexing is difficult.
2. They need good ambient light or back lighting.
3. They have a limited operating temperature range: -20 to 60 or 85°C.
4. Their cost in relation to other displays is high.
5. Their response time is slow: 100 to 300 milliseconds.

RCA offers several display drivers for liquid crystal devices: the CD4054, a 4-segment display driver; the CD4055, a BCD-to-7-segment decoder/driver with "display frequency" output; and the CD4056, a BCD-to-7-segment decoder/driver with strobed-latch function. These devices have level-shifting capability for interfacing low-voltage logic signals to higher-voltage display signals. In addition, a full line of direct drive LCD watch chips is available. For specific technical data, consult the RCA COS/MOS INTEGRATED CIRCUITS DATABOOK SSD-250 or the technical data sheet for the specific device.

Light-Emitting Diodes

The light-emitting diode (LED) has also received wide acceptance as a digital display in the last several years because of its low-voltage operation, long life, ease of multiplexing, high reliability, and fast response time. An LED is a semiconductor diode composed of a p-n junction. In forward-biased operation, because of recombination of holes and electrons, the

diodes radiate a colored light in a narrow spectrum. LED displays are normally constructed of either gallium phosphide (GaP) or gallium arsenide phosphide (GaAsP) semiconductor material. Both types of LED displays have approximately equal advantages and disadvantages. The GaAsP type, however, is more prevalent for red displays. The forward drop is approximately 1.6 volts for GaAsP diodes and 2.1 volts for GaP diodes. Two configurations of LED are available: common anode (requiring sink current) and common cathode (requiring source current). Fig. 8 illustrates both types of device.

Any single LED segment is electrically the same as any conventional solid-state diode although the LED does have a slightly higher forward voltage drop. Once the forward voltage reaches approximately 1.6 volts, the current which up to that point has been very small increases rapidly. A typical GaAsP LED needs approximately 5 to 30 milliamperes for a reasonable amount of brightness. If current continues to increase, the LED will reach a light saturation mode at approximately 100 to 150 milliamperes. At this point any increase in current will not increase the amount of light generated. Because the efficiency is greater for higher currents and the electrical and light output rise times are in nanoseconds, LED's are well suited for multiplexed or pulsed output drive. Pulsed output drive can also decrease the total amount of power required to achieve a given brightness by as much as 30 per cent.

As an example, consider the design of a four-digit multiplexed LED display system to interface with a four-digit information storage device. The LED needs an average of 6 milliamperes of current to achieve the desired brightness. Because there are four digits, the multiplexed signal requires a 25 per cent duty cycle. The peak current, therefore, must be 4 x 6 milliamperes to achieve the 6-milliamper

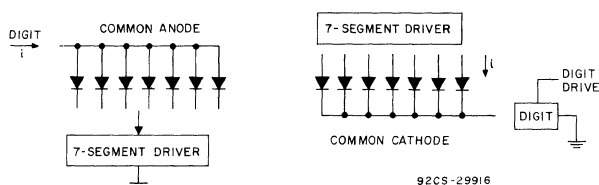


Fig. 8 - Common cathode and common anode light-emitting diode configurations.

average current. The CD4511 BCD-to-seven-segment latch decoder driver is designed with emitter-follower n-p-n bipolar outputs and is therefore able to supply the needed peak current of 24 milliamperes. The digit driver must be able to sink a peak current of 7×24 or 168 milliamperes when all segments are turned on. Many available discrete or integrated bipolar devices can meet this requirement. Fig. 9 illustrates a suitable circuit. This circuit uses a CD4511, a BCD-to-7-segment latch decoder driver; CD4052's, differential four-channel multiplexers; CD4094's, eight-stage shift-and-store bus registers; and CD4011 NAND gates.

The multiplexing digit signal, which can also be used to clock a counter to control the CD4052, can be derived by use of a CD4017 as shown in Fig. 10. The CD4017 is a counter/divider having ten decoded outputs. The number of digits multiplexed can be increased beyond four by taking the digit drive from a higher output on the CD4017. The output should be $N + 1$ where N equals the number of digits to be multiplexed. The CD4017 must be interfaced to a bipolar driver to be able to sink or source the current needed by each digit (168 milliamperes).

Fig. 11 shows a typical digit driving

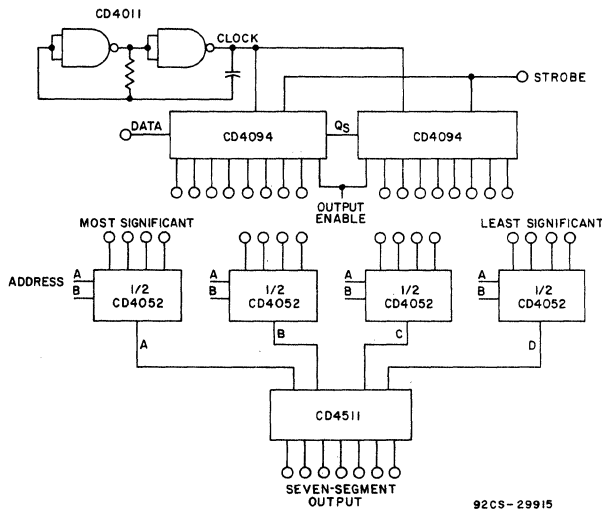


Fig. 9 - Interfacing of four-digit multiplexed LED display system with a four-digit information storage device.

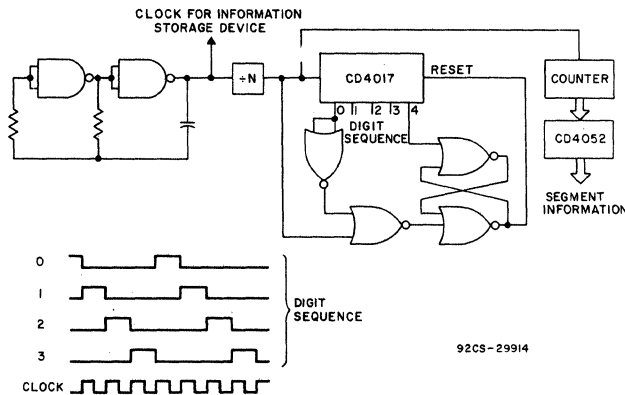


Fig. 10 - Use of CD4017, a counter/divider having ten decoded outputs, to provide the multiplexing digit signal.

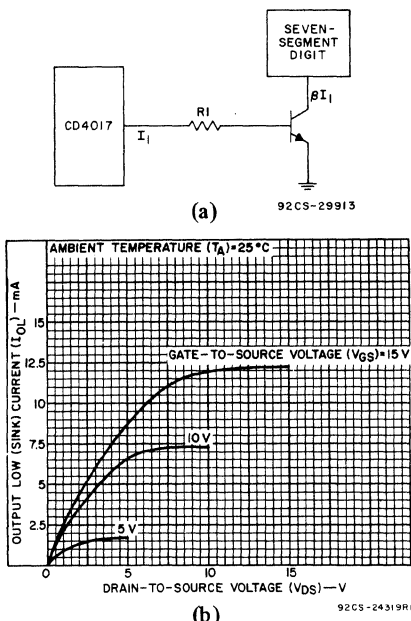


Fig. 11 - Typical digit driving circuit and minimum output n-channel drain characteristic used for calculating value of resistor R1.

circuit. The calculation of the value of resistor R1 can be made as follows:

Let β = the gain of the transistor

$$\text{then } \beta I_1 \geq 168 \text{ mA}$$

$$\text{or } I_1 \geq 168/\beta \text{ mA}$$

Once V_{DD} is established, a given V_{DS} can be taken from Fig. 11b for current I_1 .

Therefore, $R_1 = (V_{DS} - 0.7)/I_1$ kilohms

Fig. 12 shows the segment and digit drive. Resistor R_2 is necessary to avoid current "hogging" in the LED segments. The value of R_2 is calculated from the curves in Fig. 13 showing output current as a function of output voltage for the CD4511B and from the information supplied with the LED.

Let I_S = peak current in segment

V_{OUT} = voltage out of the CD4511B from Fig. 13 at the V_{DD} being used in the system

V_D = voltage across LED segment for required brightness

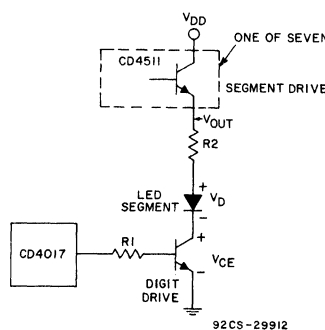


Fig. 12 - Segment and digit drive circuit for LED.

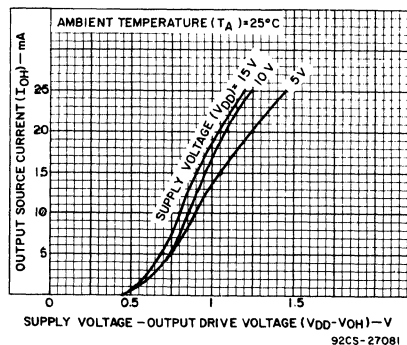


Fig. 13 - Typical voltage drop (V_{DD} to output) vs. output source current as a function of supply for the CD4511.

V_{CE} = voltage across digit driver transistor

Then,

$$R_2 = \frac{V_{DD} - (V_{OUT} + V_D + V_{CE})}{I_S}$$

In this example

$$R_2 = \frac{V_{DD} - (V_{OUT} + V_D + V_{CE})}{24 \text{ mA}} \text{ kilohms}$$

If the value chosen for R_2 is too low, uneven segment lighting can occur. Resistor R_2 , therefore, should be as large as possible.

One major drawback to the use of LED displays is that the contrast ratio of the display is very low in bright light. The easiest means of correction is to place an optical filter in front of the LED. This filter increases the contrast ratio of the LED display and makes it easier to read in any ambient light.

Gas-Discharge Displays

Gas-discharge or cold-cathode displays are available in both seven-segment and one-of-ten decoded displays. The one-of-ten decoded displays operate by energizing one of a series of stacked cathodes each in the shape of the numeral to be displayed. This stacked arrangement causes some viewing problems because the different numbers appear to move in or out within the display. A CD4028 BCD-to-decimal decoder could be used for the one-of-ten decoding necessary for this type of device. The seven-segment decoded gas-discharge displays operate in a very similar manner to the seven-segment LED displays mentioned earlier.

One disadvantage of gas-discharge displays is the high potential needed to activate the display. Typically, a voltage between 80 and 200 volts is necessary to cause ionization of the enclosed gas. Once ionization takes place, the cathode glows a dull red or orange-like color. In multiplexing these devices, care must be taken to make sure that segments energized for one digit are completely deionized before the next digit is activated.

For multiplexing gas-discharge displays, either the shunt or the series method can be used. See Fig. 14. The series method has the advantage of lower power dissipation, but it requires that the switching transistor have higher voltage and lower leakage than the shunt method requires. Fig. 15 illustrates the multiplexing of a one-of-ten gas-discharge display. Because of diode D1, the oscillator using the CD4011 produces a

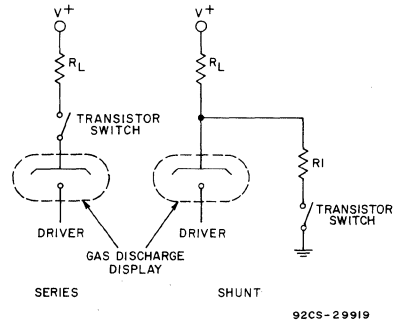


Fig. 14 - Basic series and shunt circuits for multiplexing gas-discharge displays.

non-symmetrical output having an off period long enough to assure that all characters are deionized.

Fluorescent Displays

The fluorescent display, like the LED, is a seven-segment device. Its operation is similar to that of a vacuum tube. The major difference is that the anode of the display has a phosphorescent coating which when struck by an electron beam emits blue-green light. Because this light is of a very wide spectrum, it can be filtered with little loss of display brightness. A positive potential of about 15 to 25 volts from anode to cathode is typically used to accelerate electrons emitted from the cathode. When the cathode is activated, the current flow is approximately 0.5 to 2 milliamperes depending upon the type of display.

The potentials of the anode, grid, and filament are crucial in the operation of the fluorescent display. The potential of the filament in the fluorescent display must be

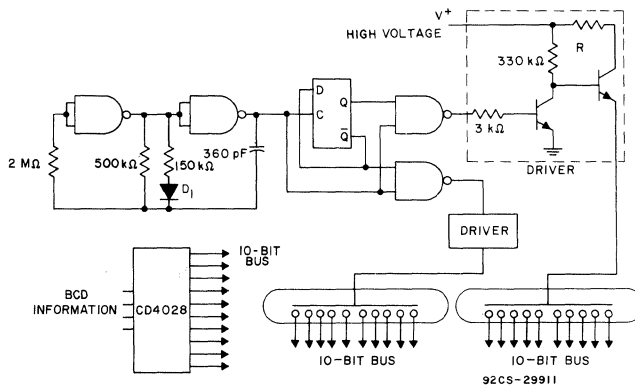


Fig. 15 - Series-type multiplexing of a one-of-ten gas-discharge display.

directly related to both the grid and anode voltages because the filament is acting both as a heater and as the cathode of the display. The potential at which the electrons are emitted from the cathode or filament, therefore, is critical in determining whether or not those electrons are accelerated toward the phosphor-coated anode.

Advantages of fluorescent display systems include low power, low cost, ease of multiplexing, and ease of interfacing to integrated circuits. A disadvantage is that they are more fragile than many other forms of display because they require an evacuated envelope.

A typical circuit for driving a fluorescent display is given in Fig. 16. The display segments are connected to the anodes of the display device and can be driven directly from any COS/MOS High-Voltage B-Series Integrated Circuit at about 20 volts. In many instances, however, the control logic for the in-

formation being displayed is operating at a voltage lower than the 20-volt display supply. In these cases, the CD40109B Quad Low-to-High Voltage Level Shifter can be used to interface the device.

In a multiplexed system, the grid or cathode of the fluorescent display device operates in a manner equivalent to the digit drive on LED devices. A typical grid voltage value necessary to activate the display is 10 volts. If a system is operating below 10 volts, it may be necessary to shift the voltage levels of both the segment and the digit information.

In an unmultiplexed system, the grid voltage should always be enabled to allow the display of the seven-segment information. An example of such a system is given in Fig. 17. Because the grid voltage is constant and not at the control of the system, the only possible level shifting necessary would be for the segment display.

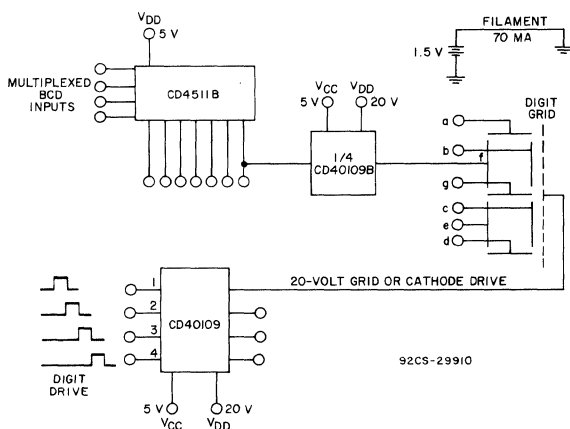


Fig. 16 - Typical circuit for driving a fluorescent digital display.

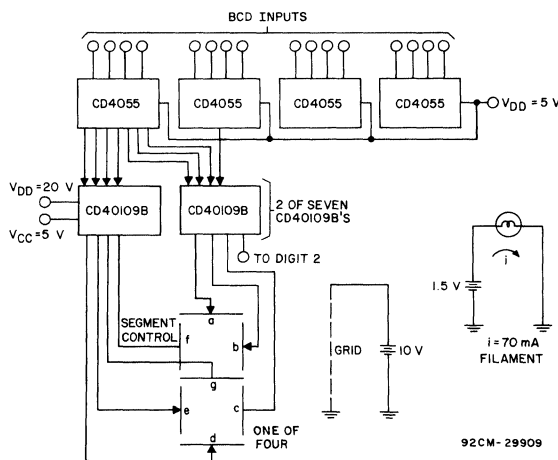
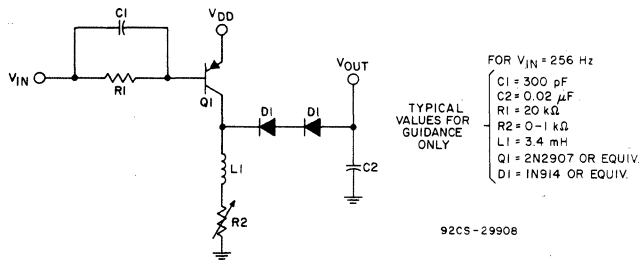


Fig. 17 - Example of unmultiplexed system for driving a fluorescent display.

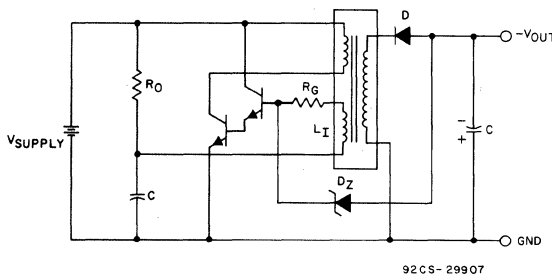
Unlike the LED display, the fluorescent display quite often needs the level-shifting capability of a transistor-inductor flyback circuit to achieve the high potentials necessary for operation. Fig. 18 gives three typical up-converter circuits. The circuit of Fig. 18(a) is pulsed by V_{IN} thus causing a current flow through L . This change in current causes an increase in the voltage across the inductor ($V_L = L \cdot di/dt$). The amount of current ($i_{peak} = V_{DD}/R_2$) is inversely proportional to the value of R_2 . With R_2 adjustable in value, the output voltage can be increased by lowering the value of R_2 or decreased by raising its value. Capacitor C_2 filters the voltage spikes caused by the input frequency, and diode D_1 keeps the capacitor charged while the voltage spike from L di/dt is low.

Fig. 18(b) differs from Fig. 18(a) in that it has a self-contained RCL oscillator and obtains its voltage increase by transformer action. The oscillator formed by R_O , C , and L drives the n-p-n bipolar devices forcing an ac signal across the transformer input windings. Because the turns ratio of the transformer from output to input is greater than one, there is an increase in output voltage. The transformer gives a more precise increase in voltage than the circuit in Fig. 18(a) provides. Capacitor C and diodes D and D_Z clamp the voltage V_{OUT} to the breakdown voltage of D_Z and filter and isolate C from discharging during the period of low output voltage from the transformer.

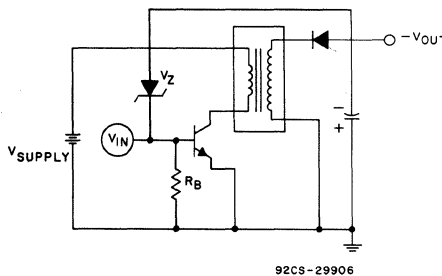
Fig. 18(c) is similar to Fig. 18(b) in the transformer action, but its input is similar



(a) Pulsed, single-transistor-inductor flyback circuit.



(b) Transformer-type circuit with RCL oscillator providing drive.



(c) Transformer-type circuit with external drive.

Fig. 18 - Typical up-converter circuits for fluorescent digital displays.

to that of Fig. 18(a) in that it is driven by an external input.

Circuits similar to those in Fig. 18 can be used to level-shift voltages for the gas-discharge type of display discussed previously. It is necessary, however, that the transformer, capacitors, transistors, and other components be rated to withstand the 200-volt signals which may be necessary to operate the gas-discharge display and be capable of meeting the higher power requirements.

Incandescent Displays

One other display which has had wide acceptance is the incandescent display. Its low cost, high brightness, and ready availability have led to considerable use of this display. Its disadvantages are its high power dissipation and the high amount of heat it generates. Typical power requirements are 1.5 to 5 volts at 8 to 24 milliamperes.

Incandescent displays are available in many sizes and colors. Multiplexing of the digits is easily accomplished by pulsing each segment for a given time period. The wattage for an incandescent lamp at the stated brightness remains constant regardless of duty cycle or waveform shape provided that the multiplexing rate is faster than the thermal time constant of the filament. When incandescent displays are multiplexed, an increase in the forcing voltage by an amount equal to the square root of the number of multiplexed displays will maintain the same brightness on each display that it would have in a static condition.

With incandescent displays, it is recommended that diodes be used in series with each segment to prevent erroneous display indication through stray electrical paths. Fig. 19 illustrates the interfacing of a multiplexed incandescent display. In this circuit, the CD4013 dual "D"-type flip-flop combines with the CD4069 oscillator to generate the four pulse intervals needed to multiplex four digits. For a typical incandescent display requiring 4.5 volts, the voltage necessary for the four-digit display is $4 \times 4.5 = 9$ volts. The CD40107 dual NAND buffer/driver and the p-n-p transistor 40537 assure that sufficient current is generated at this voltage. With a typical filament segment current of 50 milliamperes, the current sourced from transistor 40537 is 50×7 or 350 milliamperes. The minimum beta of the 40537 is 20. Its base current, therefore, is given by

$$I_{B1} = 350/20 = 17.5 \text{ mA.}$$

At V_{DD} of 12 volts and I_{OUT} of 17.5 mA, V_{OUT} from the CD40107B is 0.11 volt. Then,

$$R_1 = (11.3 - 0.11)/17.5 = 640 \text{ ohms.}$$

For 50 milliamperes in each segment and a β of 40

$$I_{B2} = 50/40 = 1.25 \text{ mA}$$

At V_{DD} of 12 volts and I_{OUT} of 1.25 mA, V_{OUT} from the CD4511B is 11.4 volts. Then,

$$R_2 = (11.4 - 0.7)/1.25 = 8.56 \text{ kilohms.}$$

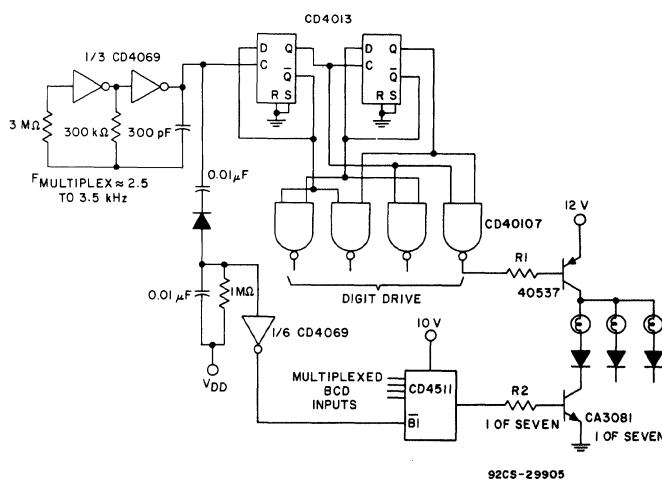


Fig. 19 - Circuit for interfacing a multiplexed incandescent-type digital display.

These calculations depend upon the current gain of each bipolar device and the voltage necessary on the incandescent display. As mentioned previously, the diodes in series with each display segment minimize the possibility of stray leakage

currents. Use of the blanking input of the CD4511 assures that if the oscillator were to cease to function for any reason, the indexed digit and segments would not be destroyed by the static voltage and current applied to the display.

Interfacing COS/MOS With Other Logic Families

By A. Havasy
M. Kutzin

The interface of different logic families requires that the circuits operate at a common supply voltage and have logic level compatibility. In addition the devices must maintain safe power dissipation levels and good noise immunity over the required operating temperature range.

The RCA CD4000A COS/MOS series circuits operate from power-supplies of 3 to 15 V. Thus they can drive and be driven by a number of logic families (within certain conditions and limitations), including all DTL and TTL families. COS/MOS devices have high noise immunity, low quiescent power dissipation and high packing densities. This Note describes the conditions governing the interface of COS/MOS logic circuits with other logic families.

COS/MOS-TTL/DTL Interface

Fig. 1 shows the voltage characteristics required at the output and input terminals of saturated logic devices. Fig. 2 shows the COS/MOS input and output characteristics at $V_{DD} = 5V$. The COS/MOS devices are designed to switch at a voltage level of one-half the power supply voltage. However, TTL/DTL devices are designed to switch at +1.5 V, which is not one-half of the supply voltage.

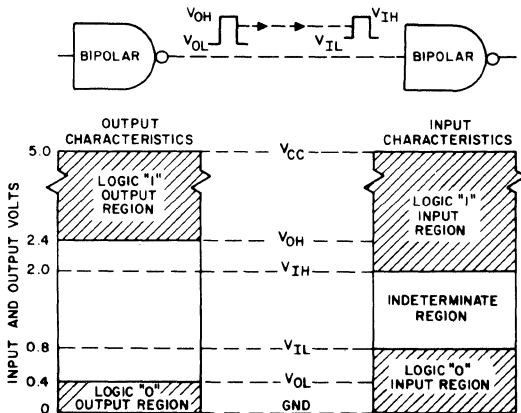


Fig. 1— Interface voltage characteristics required at the output and input terminals of saturated logic devices.

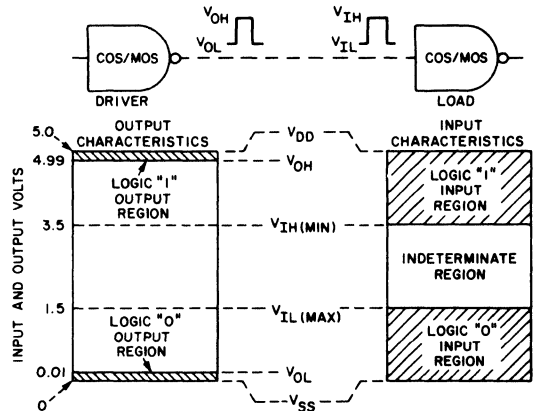


Fig. 2— COS/MOS input and output characteristics at a power-supply voltage of 5 volts.

When interfacing one type of digital integrated circuit with another, attention must be given to the logic swing, output drive capability, dc input current, noise immunity, and speed of each type. Table I shows a comparison of these parameters for COS/MOS, medium power TTL and medium power DTL. The supply voltage column of Table I shows that both saturated bipolar and COS/MOS devices may be operated at a supply voltage of 5 V. Both logic forms are directly compatible at this supply voltage (with certain restrictions).

Bipolar Driving COS/MOS

When a bipolar device is used to drive a COS/MOS device, the output drive capability of the driving device, the switching levels and input currents of the driven device are important considerations. Table I shows that only 10 picoamperes of dc input current are required by a COS/MOS device in either the "1" or "0" state. The input thresholds, (for the driven COS/MOS device) are 1.5 and 3.5V, hence the output of the TTL/DTL driver must be no more than 1.5V ("0" logic voltage) and no less than 3.5V ("1" logic voltage) in order to obtain some noise immunity.

Table I – Comparison of COS/MOS, TTL, DTL Interfacing Parameters

FAMILY	SUPPLY VOLTAGE (VOLTS)	LOGIC SWING/OUTPUT DRIVE CAPABILITY	DC INPUT CURRENT	NOISE IMMUNITY	PROPAGATION DELAYS
COS/MOS	3.5 to 15	V _{SS} to V _{DD} (driving COS/MOS) Output drive is type dependent (see text).	10 pA (typical) 1 and 0 State	1.4 at V _{DD} =5V The switching point occurs from 30% to 70% of V _{DD} which is 1.5V to 3.5V at V _{DD} =5V	35ns (typical) for inverter C _L = 15 pF
DTL and TTL	5	<u>0 State:</u> 0.4V max. at I _{sink} = 16 mA <u>1 State:</u> 2.4V min. at I _{load} = -400 μA	<u>0 State:</u> -1.6 mA max. <u>1 State:</u> 40 μA max.	at V _{CC} = 5V 0.4V guaranteed. The switching point occurs from 0.8V to 2V	20ns (typical) for inverter C _L = 15 pF

Current Sinking

Fig. 3a shows the low state operation of a loaded bipolar driver stage. When the output drive circuit of the bipolar stage is in the low state, (as shown in Fig. 3b) the collector is essentially at ground potential. The "ON" transistor must go into saturation in order to assure a reliable logic "0" level (0 to 0.4V). To attain this voltage level there should be a high impedance path from the output to the power supply. Current sinking capability is not a problem in this configuration because the COS/MOS devices have extremely high input impedances (typically 10¹¹Ω). The voltage level is not a problem either; the COS/MOS devices have high noise immunity (greater than 1 volt).

Current Sourcing

Current flows from the V_{CC} terminal of a saturated logic output device into the input stages of the load, (i.e., the

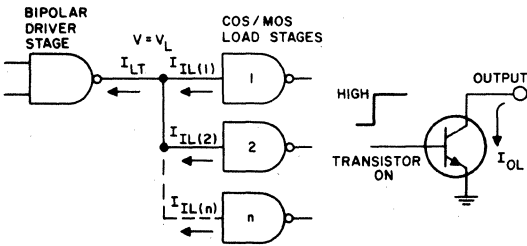
output device acts as the current source for the load). Fig. 4a shows high-state operation of a loaded bipolar driver stage. Whenever a typical bipolar driver circuit is in the high state, a pull up configuration (resistor or transistor) ties V_{CC} to the output pin. The total load configuration should not draw sufficient current to reduce the output voltage level below the V_{IH} required by the COS/MOS devices.

There are three bipolar output configurations to consider:

- a) Resistor pull-up
- b) Open Collector
- c) Active pull-up

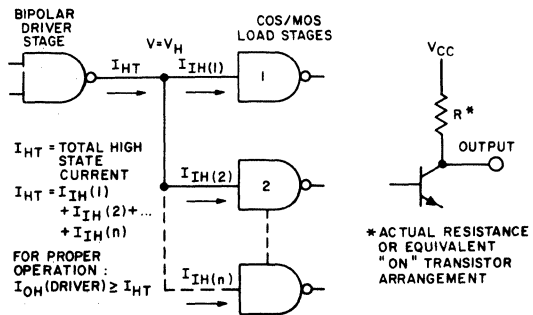
Resistor Pull-Up

Devices with resistor pull-ups, (shown in Fig. 4b) present no problem in the interface with COS/MOS devices.



I_{LT} = TOTAL LOW STATE CURRENT
 I_{LT} = I_{IL(1)} + I_{IL(2)} + ... + I_{IL(n)}
 FOR PROPER OPERATION:
 I_{OL(DRIVER)} ≥ I_{LT}

Fig. 3— (a) Low-state operation of a loaded bipolar driver stage. (b) typical bipolar output-drive circuit in the low state.



I_{OH} = Maximum Permissible output driver current in high state ("1") (Driver Leakage).
 I_{OL} = Maximum output driver sinking current in low state ("0").
 I_{IL} = Low state input current drawn from the load stage (to the driver).
 I_{IH} = High state input current flowing into the load stage from driver.

Fig. 4— (a) High-state operation of a loaded bipolar driver stage. (b) typical bipolar output-drive circuit in the high state.

Open Collector

Devices with open collectors require an external pull-up resistor as shown in Fig. 5. The selection of the external pull-up resistor is discussed below. It is recommended that when driving a COS/MOS device from an arrangement such as the one shown in Fig. 5, (with both V_{DD} and V_{CC} supply voltages at 5V) the driver should not fan out to any TTL/DTL gate, but can be fanned out to other COS/MOS devices.

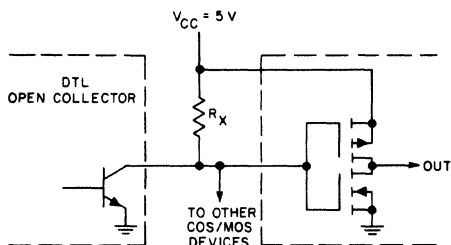


Fig. 5— Example of DTL/TTL circuit with open collectors that require a resistor between the output and V_{CC} .

Active Pull-Ups

When an active pull-up is used such as the transistor plus diode arrangement (shown in Fig. 6), there can be a problem in the "1" state because the minimum output level, (2.4V) cannot assure an acceptable "1" state input for the COS/MOS device. The 2.4-volt minimum TTL/DTL output level is often specified at a load current of $400\mu A$. However, negligible current is being drawn by the COS/MOS device, hence the minimum TTL/DTL high-output level would typically be 3.4 to 3.6V under these conditions. There is no noise immunity in such a configuration. Therefore, it is recommended that a pull-up resistor be added from the output terminal of the bipolar device to V_{CC} . The selection of the external pull-up resistor is discussed below. N in the formulae [(2) and (4)] is equal to 1, because it is not recommended that the outputs of devices that use active pull-ups be tied together.

When driving a COS/MOS device from such an output arrangement (as shown in Fig. 7), the driver should not fan out to TTL/DTL circuits; only to other COS/MOS devices.

PULL-UP RESISTOR SELECTION

Selection of a pull-up resistor for a circuit requires consideration of fan-out, maximum allowable collector current in the low state ($I_{OL(max)}$), collector-emitter leakage current in the high state (I_{CEX}), power consumption, power-supply voltage and propagation delay times. The minimum value of the external pull-up resistor in Fig. 8a is given by:

$$R_{X(min)} = \frac{V_{DD} - V_{OL(max)}}{I_{OL} - (M) I_{IL}} \tag{1}$$

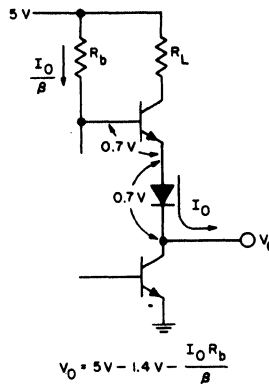


Fig. 6— Transistor-diode pull-up.

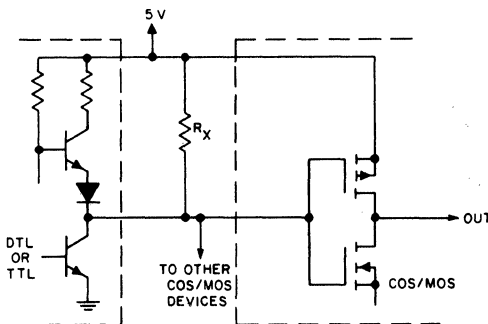


Fig. 7— Modification of the circuit in Fig. 6 to drive COS/MOS devices.

where M is the number of COS/MOS load stages. The maximum value of the external pull-up resistor shown in Fig. 8b is given by:

$$R_{X(max)} = \frac{V_{CC} - V_{IH}}{(N) I_{CEX} + (M) I_{IH}} \tag{2}$$

where N is the number of TTL/DTL driver stages.

As previously discussed, the values of I_{IH} and I_{IL} for the COS/MOS input currents in both high- and low-level states are approximately 10 pA and are neglected because their value is insignificant when compared with the value of the bipolar currents. Therefore, the equations for bipolar-to-COS/MOS interface using a pull-up resistor can be reduced to:

$$R_{X(min)} = \frac{V_{DD} - V_{OL(max)}}{I_{OL}} \tag{3}$$

$$R_{X(max)} = \frac{V_{CC(min)} - V_{IH}^*}{(N) I_{CEX(max)}} \tag{4}$$

* V_{IH} in eq. 4 is the value for the COS/MOS Device.

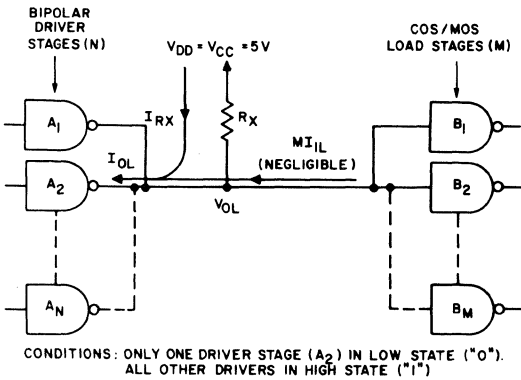


Fig. 8a— Bipolar output (with pull-up resistor) driving COS/MOS in low-state operation.

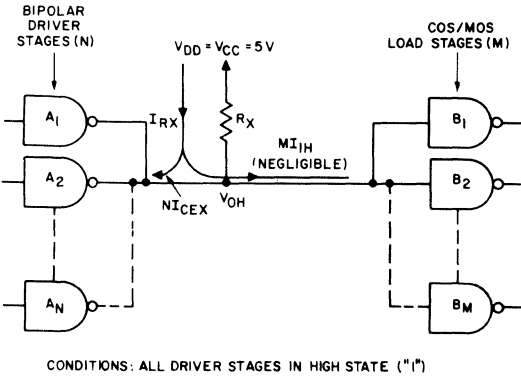


Fig. 8b— Bipolar output (with pull-up resistor) driving COS/MOS in high state operation.

An example in which a typical 9000-series TTL gate is used to drive a CD4000A COS/MOS gate the operating voltage and current specifications might be as follows:

Operating Voltage	Current Specifications
$V_{CC} = 5V \pm 0.5V$	$I_{OL} = 16 \text{ mA (TTL Gate)}$
$V_{DD} = 5V$	$I_{IL} = 10 \text{ pA (COS/MOS Gate)}$
	$I_{IH} = 10 \text{ pA (COS/MOS Gate)}$
	$I_{CEX} = 10\mu\text{A max (TTL Gate)}$

V_{OL} is obtained from Fig. 1 and is 0.40V;

V_{IH} is obtained from Fig. 2 and is 3.5V.

If N & M are both 1, then:

$$R_{X(\min)} = \frac{(5.5 - 0.4) V}{16 \text{ mA}} \approx 330\Omega$$

$$R_{X(\max)} = \frac{(5 - 3.5) V}{100\mu\text{A}} = 15\text{k}\Omega$$

For short propagation delay times, it is best to keep R_X small. However, power consumption increases rapidly at values below 1000 ohms. Some compromise is necessary. The typical power consumption is 14 mW (with the output of the TTL/DTL gate low) if a 2k Ω resistor is used. Final selection of the pull-up resistor, however, will depend on what is most important for the intended application: high speed or low power.

Figs. 8c and 8d show typical speed/power relationships as a function of R_X for two popular bipolar open collector drivers. These figures illustrate the trade-off between speed and power; the power figure shown is the power dissipated in both the bipolar driver and the pull-up resistor.

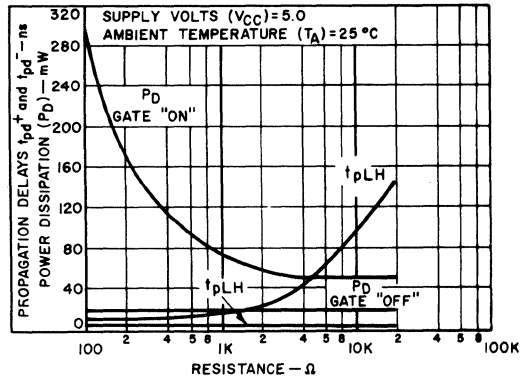


Fig. 8c— Typical speed-power trade-off of open collector TTL buffer and pull-up resistor.

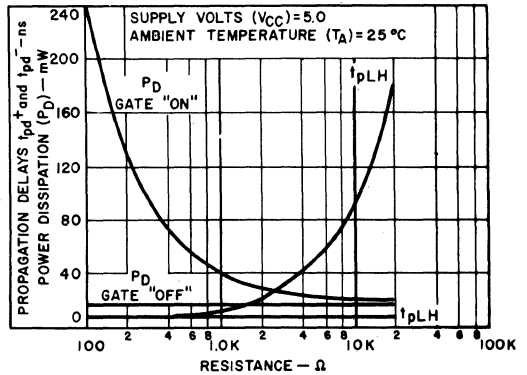


Fig. 8d— Typical speed-power trade-off of open collector TTL gate and pull-up resistor.

COS/MOS DRIVING BIPOLAR

Figs. 9a and 10a show COS/MOS devices driving bipolar devices. The current sinking capability of the COS/MOS device must be considered when the device is driving a medium power DTL and TTL circuit. Table I shows that the TTL/DTL device requires no more than -1.6 mA in the "0" input state and a maximum of 40 μ A in the "1" input state.

The COS/MOS device must be capable of sinking and sourcing these currents, while maintaining voltage output levels required by the TTL/DTL gate. Any given TTL/DTL gate will switch state at a voltage that ranges from 0.8 to 2V. Hence the output drive capability of the COS/MOS driver must be at least $-40\mu\text{A}$ for a given "1"-state output voltage of 2V, and at least 1.6 mA for a given "0"-state output voltage of 0.8V. In order to provide a noise margin of 400 mV, for the driven bipolar device, the COS/MOS device must sink 1.6 mA at a "0" logic state voltage of 0.4V and $-40\mu\text{A}$ at a logic "1" level at 2.4V.

Current Sourcing

In the high-state operation, (Fig. 9b) V_{DD} is normally connected to the driver output through one or more "ON" p-channel devices which must be able to source the total leakage current of the bipolar load stages. The published data for the particular COS/MOS and bipolar devices, must be consulted in order to determine the leakage currents (for the logic "1" state) and drive fan out to be used in the equation shown in Fig. 9a. Saturated-logic devices will not achieve their required switching levels unless this equation has been satisfied. All of the presently available COS/MOS devices are able to source $40\mu\text{A}$ at 2.4V, (supply voltage, $V_{DD} = 5\text{V}$), hence current sourcing is not a problem.

Current Sinking

When the output of a COS/MOS driver is in the low-state an n-channel device is "ON" and the output is approximately at ground potential. The COS/MOS device sinks the current flowing from the bipolar input-load stage. The published data for the COS/MOS device must be consulted to determine the maximum output low-level sinking current, and the published data for the bipolar device must be consulted to determine its input low-level current.

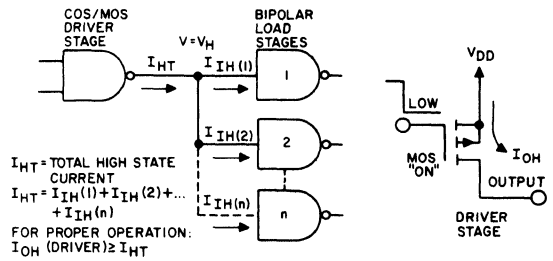


Fig. 9— (a) Logic diagram for a COS/MOS device driving a bipolar device, high-state operation; (b) schematic diagram showing current flow.

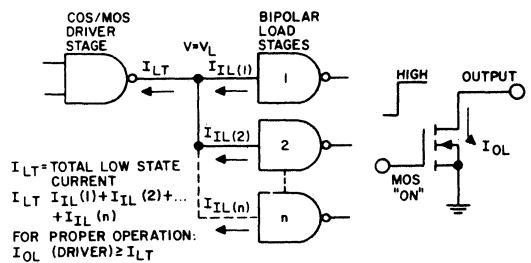


Fig. 10— (a) Logic diagram for a COS/MOS device driving a bipolar device, low-state operation; (b) schematic diagram showing current flow.

Not all COS/MOS devices can sink 1.6 mA required by most bipolar devices at only a 0.8V or 0.4V "0"-state output level. Table II shows eight types of COS/MOS circuits and their n-channel current-sinking capabilities at output voltages of both 0.8 and 0.4V. The CD4009A and CD4010A can each sink 6 mA at 0.8V and thus can drive 6/(1.6) or three

Table II – Current Sinking Limits of COS/MOS Devices

COS/MOS TYPE	DESCRIPTION	SINK CURRENT			
		$V_{OL} = 0.4 \text{ VOLTS}$		$V_{OL} = 0.8 \text{ VOLTS}$	
		CERAMIC	PLASTIC	CERAMIC	PLASTIC
CD4000A	Dual 3-input NOR Gate Plus Inverter	0.4	0.3	0.8	0.6
CD4001A	Quad 2-input NOR Gate	0.4	0.3	0.8	0.6
CD4002A	Dual 4-input NOR Gate	0.4	0.3	0.8	0.6
CD4007A	Dual Complementary Pair plus Inverter	0.6	0.3	1.2	0.6
CD4009A	Inverting Hex Buffer	3.0	3.0	6.0	6.0
CD4010A	Noninverting Hex Buffer	3.0	3.0	6.0	6.0
CD4011A	Quad 2-Input NAND Gate	0.2	0.1	0.4	0.2
CD4012A	Dual 4-Input NAND Gate	0.1	0.05	0.2	0.1

TTL/DTL gates (worst case). None of the other types listed are able to drive one TTL/DTL gate; however, by tying together "N" CD4007A n-channel transistors, "N" inputs of the CD4000A, CD4001A, or CD4002A, or "N" CD4011A or CD4012A gates, "N" times the sink current capability listed for each type may be obtained. Thus, all four inputs of one 4-input NOR gate (CD4002A) could be tied together as shown in Fig. 11 for a maximum sink-current capability of 3.2 mA, which allows a fan out of two TTL/DTL gates at 0.8V, or 1 TTL gate at 0.4V.

Many COS/MOS MSI devices (such as counters and shift registers) have limited drive capability, hence their outputs may require buffering if they are to drive TTL/DTL gates. This buffering (use of added driver stages) can be provided by the CD4009A, CD4010A or any device with the same current sinking capability. The logic and circuit diagrams for the CD4009A and the CD4010A are shown in Figs. 12 and 13, respectively.

The CD4009A and CD4010A are designed primarily for the purpose of shifting voltage levels and may be used to interface TTL with COS/MOS devices operating at supply voltages up to 15V as shown in Fig. 14. With this driving arrangement, designers who use higher voltages for COS/MOS circuits can achieve the maximum COS/MOS speed capability, as well as to be able to fan out to the bipolar devices. Fan out capability versus supply voltage for a bipolar supply level of 5V and COS/MOS levels of 5 and 10V is given in Table III. The drive capability of most COS/MOS devices enables them to drive some low power bipolar circuits (such as the 54L and 74L series) directly, as shown in Fig. 15. Table IV gives dc fan out capability of COS/MOS devices driving low-power TTL devices in unit loads (as defined by the manufacturer) of 0.18 mA.

COS/MOS-BIPOLAR HTL INTERFACE

Bipolar HTL (high-threshold-logic) circuits operate at voltage levels between 14 and 16V. COS/MOS logic circuits

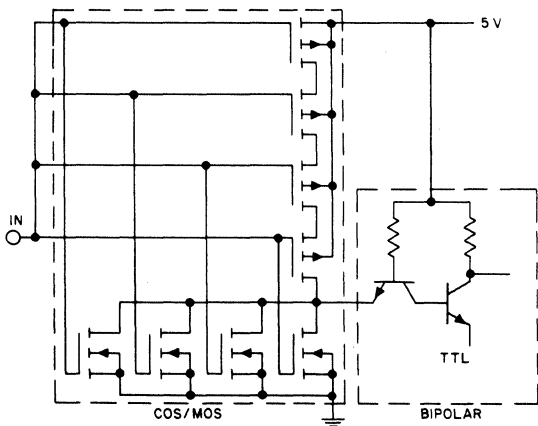


Fig. 11— Method of interconnection of inputs of CD4002A to obtain maximum sink-current capability of 3.2 mA.

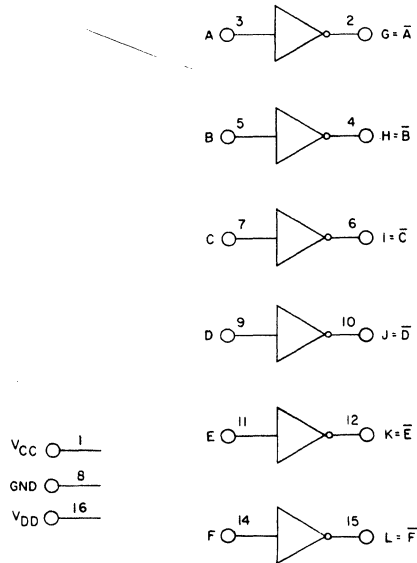
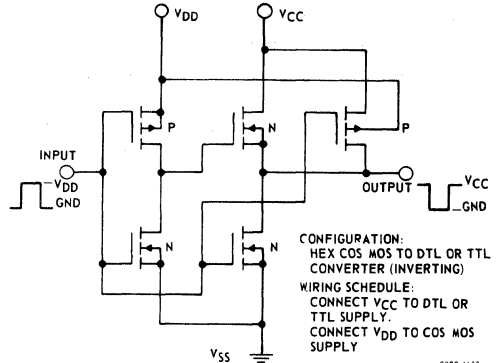


Fig. 12— Logic and circuit diagram for the CD4009A.

can operate at these voltages as well, but generally are limited to voltages no higher than 15V. HTL circuits are, in general, identical in construction to the DTL circuits with some resistance value changes necessary, as a result of the higher voltage levels used and the extremely important distinction of an added base/emitter junction in the reverse direction on the emitter side of the input transistor. This added junction is a zener diode which gives a higher threshold switching voltage. Conduction does not occur until the junction breaks down at a voltage of approximately 6.7V. This provides the HTL with its high noise immunity.

Bipolar high-threshold-logic circuits have a more limited temperature range and dissipate much more power than do COS/MOS circuits. Therefore, care should be exercised when using the combination in extreme temperature environments, as the HTL resistance values vary by about 20 percent from

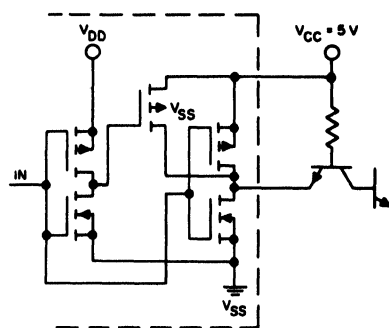
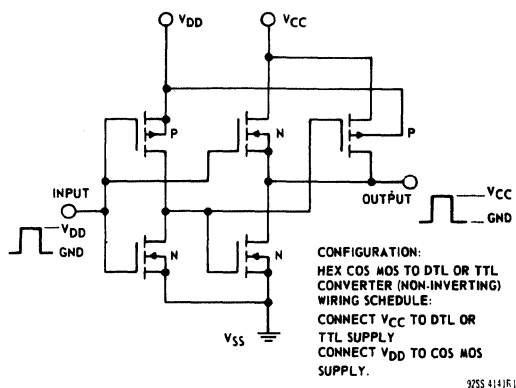


Fig. 14—COS/MOS devices operating at 15 volts interfacing with TTL devices.

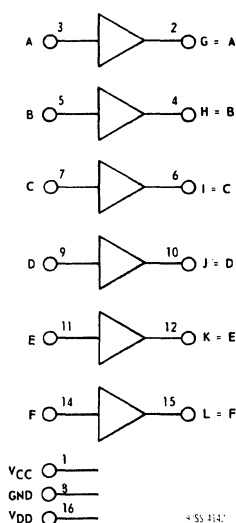


Fig. 13— Logic and circuit diagram for the CD4010A.

one end of their temperature range to the other. In addition, the bipolar transistor's sensitivity to temperature, increases the thermal runaway problems. The V_{OL} level, propagation delay, and noise immunity of HTL circuits vary widely across the temperature range. COS/MOS circuits, however, show almost negligible variation for these same parameters over a temperature range that is approximately 75-percent wider than that of HTL.

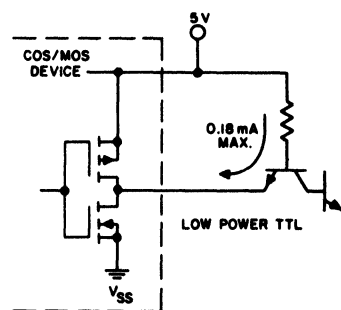


Fig. 15— COS/MOS devices driving low-power bipolar circuits.

The interfacing of COS/MOS devices that are driving ECL devices is a simpler matter. A number of methods are available to reduce the output voltage swing to 0.3 to 0.9V. A precise resistor-divider network arrangement, an emitter follower, or numerous combinations of resistor, diode, and transistor configurations can successfully reduce the COS/MOS output swing (V_{DD} to V_{SS}) to the ECL logic "1" and logic "0" levels. However, care must be taken to meet the necessary current sinking requirements of the ECL device in its logic "0" state. External circuitry is required when the COS/MOS circuit selected does not have the capacity to sink the current from the ECL load. Rise times, fall times, and pulse widths must be restricted to those specified in the published data.

Table III — Fanout Capability and Supply Voltage for the CD4009A and CD4010A

$V_{OL} = 0.4V$	$V_{OL} = 0.8V$	V_{DD}^*
All Package Types	All Package Types	
2	4	5V
4	7	10V

* $V_{CC} = 5V$.

Table IV – DC Fanout Capability of COS/MOS Devices Into Low Power TTL Devices

TYPE	DESCRIPTION	FANOUT (UNIT LOAD = 0.18 mA)			
		-----PLASTIC-----		-----CERAMIC-----	
		VOL = 0.4V	VOL = 0.8V	VOL = 0.4V	VOL = 0.8V
CD4000A	Dual 3-Input NOR gates plus Inverter	1	3	2	4
CD4001A	Quad 2-Input NOR gates	1	3	2	4
CD4002A	Dual 4-Input NOR gates	1	3	2	4
CD4004A	7-Stage binary counter	0	1	1	2
CD4006A	18-Stage Static Shift register	0	0	0	0
CD4007A	Dual Complementary Pair-plus Inverter	1	3	3	6
CD4008A	4-Bit full adder	0	0	0	0
CD4009A	Hex Buffer Inverter	16	33	16	33
CD4010A	Hex Buffer Non-Inverting	16	33	16	33
CD4011A	Quad 2-Input NAND gates	0	1	1	2
CD4012A	Dual 4-Input NAND gates	0	0	0	1
CD4013A	Dual type D Flip-Flops	1	2	2	4
CD4014A	8-Stage static shift register Synchronous parallel-in/serial-out	0	0	0	1
CD4015A	Dual 4-stage static shift register Serial-in/parallel-out	0	0	0	1
CD4017A	Decade counter/divider plus 10 decoded decimal outputs	0	0	0	0
CD4018A	Presettable divide-by-N Counter	0	0	0	0
CD4019A	Quad AND-OR select gate	1	3	3	6
CD4020A	14-Stage binary counter	0	0	0	1
CD4021A	8-Stage static shift register Asynchronous parallel-in/serial-out	0	0	0	1

LEVEL SHIFTERS

The speed consideration is most important when a separate interfacing circuit is used. It is desirable, (unless high ac noise immunity is a prime design consideration) for the speed of the interfacing circuit to be maximized or at least made no slower than either type of logic. No interfacing device other than a pull-up resistor is required, however, between the COS/MOS and TTL logic at a supply voltage of 5 V. Speeds from the COS/MOS to TTL logic (which can be found in the published data for COS/MOS devices) are comparable to the COS/MOS propagation delays. Speeds from TTL to COS/MOS, even with a large external resistor, are no slower than delay times for COS/MOS logic circuits.

Speed, therefore, is not a problem in COS/MOS-TTL logic interfacing, if the clock rates are within the COS/MOS range.

When interfacing DTL or TTL devices with COS/MOS devices which are operated at a higher voltage supply, the same resistor-interface shown in Fig. 5 can be used. The resistor is tied to the higher level (V_{DD}). The maximum supply voltage for the DTL and TTL gates, however, is specified at 8V. Thus, not all units of this type may be used for interface applications that require higher supply voltages (V_{DD}). Guaranteed operation at these higher supply voltages can be accomplished by selection of DTL and TTL units that have breakdown voltages [$V(BR)CER$] which exceed the COS/MOS operating voltage, or by using a level shifting

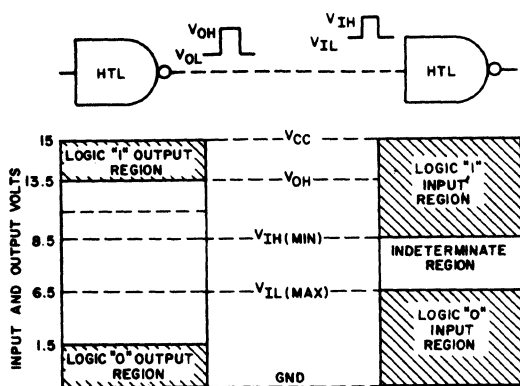


Fig. 16— HTL output and input voltage characteristics at a V_{CC} of 15 volts.

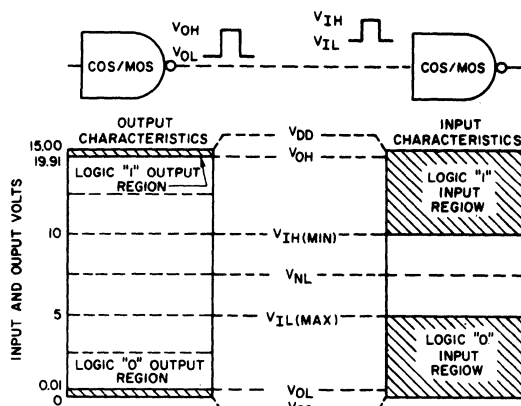


Fig. 17— COS/MOS output and input voltage characteristics at a V_{CC} of 15 volts.

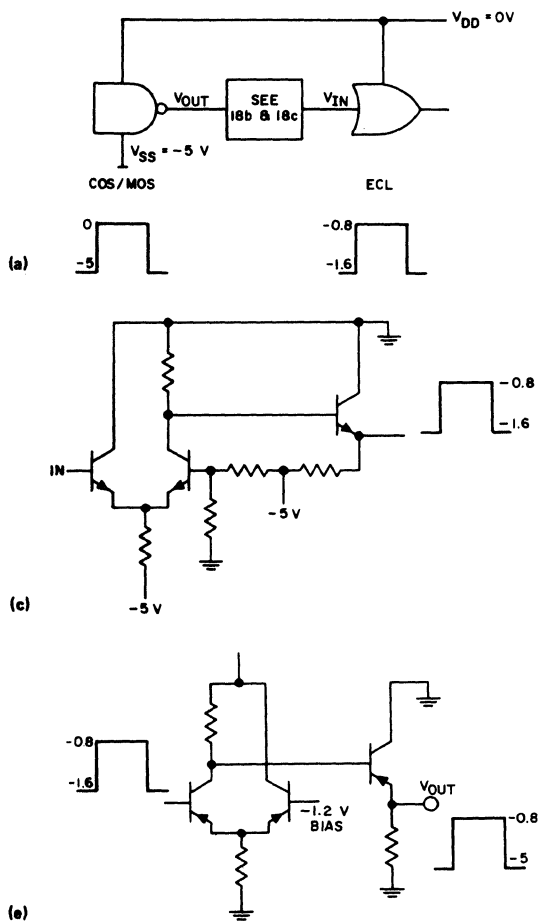


Fig. 18— Interface circuits used with high-speed ECL circuit.

circuit such as the level translator shown in Fig. 19. This circuit converts DTL, TTL, and RTL input logic levels to voltages compatible with COS/MOS circuitry. In interface applications, the supply voltage for the translator should be equal to the supply voltage required for the COS/MOS circuitry. The entire COS/MOS supply-voltage range, that exceeds that of the bipolar device, (up to 15V) may be used.

The COS/MOS CD4009A and CD4010A Inverting Hex Buffer and Non-Inverting Hex Buffer types, respectively, were designed to shift voltage levels from a driven high-voltage supply (V_{DD}) level. This feature enables designers to operate COS/MOS circuits at 15-volt (V_{DD}) levels so that these circuits can achieve their maximum-speed capability, as well as fan out to lower voltage DTL or TTL circuits; or to any other logic forms which operate below the 15-volt level. Further use of these circuits (CD4009A and CD4010A) is described in the COS/MOS-TTL/DTL interface and the COS/MOS-ECL interface sections. The statements in the former section concerning fan out and current sinking are applicable to all other compatible current-sinking logic families as well.

COS/MOS TO P-MOS INTERFACE

A large number of p-MOS devices operate at supply voltages of $V_{DD}=0$ and $V_{SS} = -6$ to -15 volts; voltages compatible to those required by COS/MOS devices. However, if the p-MOS system is using a negative logic convention, a conversion of COS/MOS positive logic functions must be made so that the two systems maintain a single logic criterion. Care must be taken when interfacing COS/MOS with p-MOS to assure that the same logic criterion is being used to describe each device type. An explanation of the simple transposition from positive-logic NAND to negative-logic NOR and similar transpositions of other functions is given in the RCA COS/MOS Manual section "Logic Design" considerations.

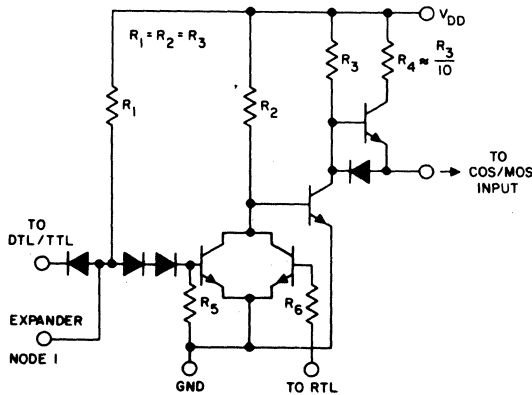


Fig. 19— Level translator used to convert DTL, TTL, and RTL input logic levels and voltages compatible with COS/MOS circuitry.

The same general rules of interfacing described in the section that discusses COS/MOS-TTL/DTL interface also apply to the interface of HTL with COS/MOS circuits. Fig. 16 shows the voltage characteristics required at the output and input of an HTL device for a $V_{CC} = 15V$. Fig. 17 shows the same characteristics for COS/MOS devices at $V_{DD} = 15V$. The HTL types either have a built in pull-up resistor (typically $15k\Omega$) or an active pull-up. An external pull-up resistor is unnecessary when COS/MOS devices are being driven by these HTL circuits. There is a difference in the noise immunity for these circuits. The dc noise immunity in the high state (logic "1") will be 3.5V for an active pull-up circuit and 5V (typically) for a resistor pull-up circuit.

The published data should be consulted to be sure that the rise, fall times and pulse widths of the HTL output are compatible with the required pulse width and input rise and fall time of the COS/MOS circuits. The procedure for the selection of a pull-up resistor for open-collector-output HTL circuits is the same as that of the open collector DTL/TTL. (See section that discusses pull-up resistors.) The procedure and equation for COS/MOS devices driving HTL are the same as those for COS/MOS devices driving DTL/TTL. The input current requirements for the HTL devices are obtained from the published data and substituted in the formulae. (See the section that discusses COS/MOS driving bipolar.) For example, the values, taken from the published data for a COS/MOS CD4009A driving a particular HTL circuit are as follows:

Particular HTC Input Data: $V_{CC} = 15V$

$$V_{IH} = 8.5V \text{ min.} \quad V_{IH} \text{ max.} = 2.0\mu A \text{ (leakage)}$$

$$V_{IL} = 6.5V \text{ max.} \quad I_{IL}(\text{max.}) = 1.2mA$$

CD4009A COS/MOS output Data: $V_{DD} = 1.5V, V_{SS} = \text{gnd}$

$$V_{OH} = 14.99V \quad V_{DD} = 15V, V_{SS} = \text{gnd}$$

$$V_{OL} = 0.01V \quad I_{OH} = 10 \text{ pA (leakage)}$$

$$I_{OL} \text{ max.} = 7.0mA \text{ (sinking)}$$

The COS/MOS CD4009A easily provides the necessary voltage levels and sinking currents required to satisfactorily drive 5 of the HTL types in question.

COS/MOS – ECL/ECCSL INTERFACE

Fig. 18 shows the interface of COS/MOS devices with ECL devices. High-speed ECL (emitter-coupled logic) and ECCSL (emitter-coupled current-steering logic) are non-saturating bipolar logic families. The V_{CC} to V_{EE} voltage range is fixed from a ground level to -5 or $-5.2V$; logic "1" to logic "0" values are separated by only 300 to 500 mV depending upon the particular type of ECL family used. Because each manufacturer shows different logic levels for a number of ECL families, care must be taken to use only the applicable values taken directly from the published data which describe the units chosen.

A logic "1" is the most positive frame of reference and a logic "0" the most negative. For example, for positive logic an RCA type CD2150 OR/NOR gate is at a logic "1" level

when its voltage is -0.8V and at a logic "0" when its voltage is -1.6V (more negative value).

The COS/MOS device requires a greater voltage swing than that of the ECL, hence an amplifier circuit must be connected between the two when ECL drives COS/MOS. Several RCA linear amplifier circuits have been successfully used to provide higher switching level outputs from the high-speed ECL circuits. Among these are the differential amplifier arrays, (CA3026, 3049, 3054, 3050, 3051) and wideband amplifiers (such as CA3035). Use of a separate transistor circuit such as the one shown in Fig. 18e is suggested. Here the transistor must provide a voltage swing at the output of between V_{DD} and V_{SS} in order to drive the

COS/MOS device from an input swing of only 0.8V. Proper biasing of the transistor is essential. It is suggested that the V_{SS} level for the COS/MOS circuit be the same as the V_{EE} level of the ECL circuit, in order to minimize the number of power supplies as well as to provide better interface conditions.

COS/MOS TO N-MOS INTERFACE

A number of n-MOS devices are available which function at the same V_{DD} and V_{SS} ranges of (COS/MOS devices) and with the same positive voltage logic levels. There is relatively no problem in directly interfacing these devices, when operated at the same V_{DD} and V_{SS} .

APPENDIX

The RCA COS/MOS product line includes a standard line of devices designed to operate from voltage supplies of 5 to 15V and a low voltage "A" series designed to operate from voltage supplies of 3 to 15V. These devices are available in any of the package types or temperature ranges shown in Table A.

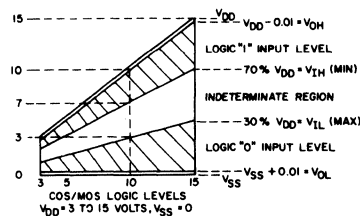
When ordering COS/MOS devices, the appropriate suffix should be affixed to the number of the device required, (i.e., if a low voltage plastic package, four bit full adder is desired, order CD4008AE).

TABLE A

Package		Operating Temperature Range (°C)
Type	Suffix	
Dual-in-line ceramic	5-15V D	-55 to +125
	3-15V ("A") D	
Plastic	E	-40 to +85
Flat Pack	- K	-55 to +125

Appendix Table I - Common Logic Voltages, Supply Voltage and Temperature Range for COS/MOS Devices

LOGIC VOLTAGE SYMBOL	DESCRIPTION	
V_{OH}	Minimum guaranteed noise free output level of device in high-level output state	
V_{IH}	Minimum acceptable input level for device in high-level input state	
V_{IL}	Maximum acceptable input level for device in low-level input state	
V_{OL}	Maximum guaranteed noise free output level of device in low-level output state	
V_{NL}	Maximum (positive) noise level tolerated at low level state	
V_{NH}	Maximum (negative) noise level tolerated at high level state	
$V_{OL} + V_{NL} \geq V_{IL}$		
$V_{OH} + V_{NH} \leq V_{IH}$		
Operating Temperature Range		
V_{DD}	Positive Supply Voltage	-55°C to + 125°C (Full Temperature Prod.)
V_{SS}	Negative Supply Voltage	-40°C to + 85°C (Limited Temperature Prod.)



Appendix Table II – Common Logic Voltages, Supply Voltage, and Operating Temperature Range Required to Interface with DTL/TTL Circuits

LOGIC VOLTAGE	DESCRIPTION	VOLTAGE (VOLTS)
VOL	Maximum output level in low-level output state	0.4
VOH	Minimum output level in high-level output state	2.4
VIL	Maximum input level in low-level input state	0.8
VIH	Minimum input level in high-level input state	2.0
VCC	Positive supply voltage	5.0 ± 1
Operating Temperature Range: -55° to + 125°C—Full temperature range product. 0° to + 85°C—Limited temperature range product.		

Appendix Table III – HTL Common Logic Voltages, Supply Voltage and Operating Temperature Ranges

LOGIC VOLTAGE SYMBOL	DESCRIPTION	VOLTAGE
VOL	Maximum output level in low-level output state	1.5V
VOH	Minimum output level in high-level output state	13.5V
VIL	Maximum input level in low-level input state	6.5V
VIH	Minimum input level in high-level input state	8.5V
VNL	Worst case positive noise level tolerated at low level state	5.0V
VNH	Worst case negative noise level tolerated at high	5.0V
VCC	Positive supply voltage	15.0 ± 1V
Operating Temperature Range -30°C to +75°C		

Appendix Table IV – ECL Common Logic Voltages, Supply Voltages and Operating Temperature Range

LOGIC VOLTAGE SYMBOL	DESCRIPTION	VOLTAGE RANGE**	
		FROM	TO
VOL	Maximum output level low-level state	-1.6	-1.45*
VOH	Minimum output level high-level state	-.8	-.795*
VIL	Minimum input level low-level state	-1.4	-1.7*
VIH	Maximum input level high-level state	-.75	-1.1
VNL	Worst case positive noise level tolerated at low-level state	.20*	.35*
VNH	Worst case negative noise level tolerated at high-level state	-.235	-.305*
VCC	Positive supply voltage	0	0
VEE	Negative supply voltage	-5.5	-5.0
Temperature Range +10 to +60°C *At T = +25°C **These values are representative of the range for several ECL families.			

Keyboard Scan Routine for Use with the RCA COSMAC Microterminal CDP18S021

by D. Block

The RCA COSMAC Microterminal CDP18S021 provides general-purpose subroutines in a utility program UT5 so that the Microterminal can be used as an output display device by the user program. In particular, the REGDIS subroutine sends the contents of registers RA and RB to the display, and the LEDD subroutine provides independent control of all eight display digits and decimal points. Refer to the Instruction Manual for the RCA COSMAC Microterminal MPM-211 for details of their use. UT5, however, does not contain a general-purpose keyboard read routine that a user program can call. This Note contains the code for such a keyboard reading routine which can be added as a subroutine to the user program thereby making the Microterminal useful as a general-purpose input as well as output device.

Microterminal Operation

The following discussion assumes that the Microterminal is being used with the RCA Evaluation Kit CDP18S020. When one of the scanned keys of the Microterminal (0 through F, CA, \$P, INC, or ←) is pressed, the EF3 flag input to the CPU is set low. This line is then the basic signal indicating that the keyboard should be read. In the READ routine given in this Note, the keyboard is read by the 6C input instruction (INP 4) and decoded by comparing the byte against a look-up table occupying locations 8122-8135 in UT5. Register RA is used as a pointer to the table. Because of the special coding scheme used with the keyboard, keybounce produces invalid codes. The keyboard reading program described here will signal an invalid read by setting RA.0

equal to zero. The main program would then normally attempt another read operation until a valid code is found.

Microterminal Keyboard READ Routine

A flow chart for the Microterminal READ routine is given in Fig. 1 and the code listing in Table I. The READ routine assumes that the standard

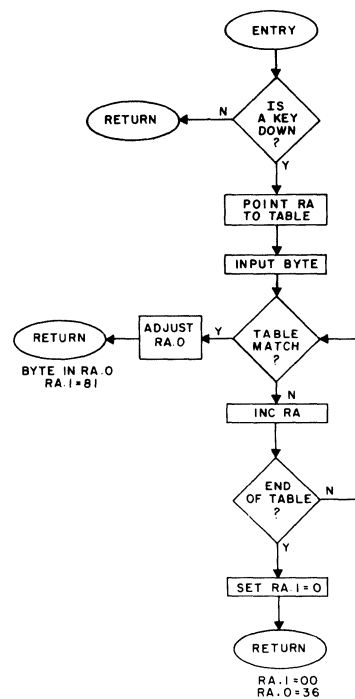


Fig. 1 - Microterminal Keyboard READ Routine Flowchart.

Table I — Code Listing for Microterminal Keyboard Read Routine

READ:	BN3	END	..IF NO KEY DEPRESSED, RETURN
	GHI	R4	..POINT RA AT LOOKUP TABLE
	PHI	RA	
	LDI	#22	..TABLE AT 8122 FOR #13 BYTES
	PLO	RA	
	INP	4	..READ IN KEY TO STACK
FCOM:	LDA	RA	..COMPARE AGAINST TABLE
	XOR		
	BZ	FMAT	..IF MATCH, GO TO FMAT
	GLO	RA	..CHECK FOR END OF TABLE
	SMI	#36	
	BDF	ERROR	..IF SO, GO TO ERROR ROUTINE
	BR	FCOM	..IF NOT, TRY NEXT TABLE ENTRY
FMAT:	GLO	RA	..ADJUST RA.0 TO EQUAL
	SMI	#23	..KEY VALUE
	PLO	RA	
END:	SEP	R5	..RETURN WITH BYTE IN RA.0
			..AND RA.1 = 81
ERROR:	LDI	#00	..SET RA.1 = 0 AND RETURN
	PHI	RA	
	SEP	R5	

COSMAC Call and Return subroutine convention is being used, and it uses RA for byte-transfer operations. Upon a successful read, the key code is entered into RA.0, and RA.1 = 81. An unsuccessful read is denoted by RA.1 = 0 (and RA.0 = 36). Table II gives the keyboard code assignments. If a key is not pressed, the READ routine executes an immediate return without altering RA.

Table II — Microterminal Keyboard Code Assignments

Key Depressed	Table Entry	RA.0 =
0	0A	0
1	8E	1
2	80	2
3	81	3
4	4E	4
5	40	5
6	41	6
7	2E	7
8	20	8
9	21	9
A	1E	A
B	10	B
C	11	C
D	06	D
E	08	E
F	04	F
↔	02	10
\$P	09	11
INC	01	12
CA	0C	13

Note: The keys R, RU, and RP are not scanned inputs.

To distinguish among no key read, key invalid, or key valid, the following calling sequence for the READ routine could be used. First, set RA.1 = 1, call READ, and then examine RA.1. If RA.1 = 1, no key was read; if RA = 00 an invalid read occurred (due to key bounce); if RA = 81, a valid read occurred. These conditions are summarized in Table III. The calling sequence is given in Table IV.

Table III — Conditions on Exit from Read Routine

Condition	RA.1	RA.0
No key depressed	(no change)	
Key read invalid	00	36
Valid key read	81	(key value)

Alternative ways of signaling the main program via the READ routine can be developed. For instance, to distinguish between a read operation in which no key is pressed and one in which an invalid read occurs, the ERROR routine could be modified to set the Q flip-flop or DF. Typically, on an invalid read or a no-key-pressed operation one would simply re-read until a valid key was found. To assure that only one byte is read per key press, the user program should wait for EF3 = 1 before proceeding. In a system with a multiplexed display, the program would loop refreshing the display during this wait period.

Table IV — Code Listing for Calling Sequence

KEYSCN:	LDI	#01	..Set RA.1 = 1.
	PHI	RA	
	SEP	R4	..Call the read routine
	,A	(READ)	
	GHI	RA	..Examine RA.1 for valid
			..Read operation
	BZ	KEYSCN	..Invalid read, try again
	SHR		
	BZ	NOKEY	..No key read, go to no key
	GLO	RA	..Key read: get value

References:

MPM-201A User Manual for the RCA
COSMAC Microprocessor
MPM-203A Evaluation Kit Manual for
the RCA CDP1802 COSMAC
Microprocessor
MPM-211 Instruction Manual for the
RCA COSMAC Microterminal

Use of the CDP1854 UART With RCA Microprocessor Evaluation Kit CDP18S020 or EK/Assembler-Editor Design Kit CDP18S024*

by R. G. Ott

The CDP1854 is a CMOS Universal Asynchronous Receiver/Transmitter (UART) circuit. It is designed to provide the necessary formatting and control for interfacing between serial and parallel data. For example, it can be used to interface between a peripheral or terminal with serial I/O ports and the 8-bit CDP1802 parallel data bus.

The CDP1854 can be used in the CDP18S020 Evaluation Kit or the CDP18S024 EK Design Kit to relieve the user's program of the task of formatting and controlling serial I/O data. It can also be used when high-speed serial data transfer at speeds up to 200K baud at a V_{DD} of 5 volts is needed. Space for a CDP1854 is provided in the user I/O area of the Kit. This application note describes several methods of interfacing the CDP1854 with the CDP1802 microprocessor and specifically explains the use of the CDP1854 in the CDP18S020 and CDP18S024 Kits.

DESCRIPTION OF THE CDP1854 FEATURES

The CDP1854 is capable of half duplex or full duplex operation, i.e., simultaneous conversion of serial input data to parallel output data and parallel input data to serial output data. It is fully programmable with externally selectable word length (5-8 bits), parity inhibit, even/odd parity, and 1, 1½, or 2 stop bits. The transmitter converts parallel data to a serial word containing a start bit, the data bits, a parity bit (optional) and stop bit(s). The receiver converts a serial input word with start, data, parity, and stop bits into parallel data. It verifies proper code by checking parity and the receipt of a valid stop bit. Both the receiver and transmitter are double buffered.

The CDP1854 can be programmed to operate in one of two modes by using the mode control input (pin 2). When the mode input is low (MODE = 0) the device is functionally compatible with industry standard UART's such as the TR1602A and 6402. It is also pin compatible with these types, except that pin 2 is used for the mode control input instead of a $V_{GG} = -12V$ supply connection. A block diagram for mode 0 operation is shown in Fig. 1. When the mode input is high (MODE = 1), the CDP1854 is directly compatible with the CDP1802 microprocessor system without additional interface circuitry. All control and status bits, as well as the data bits, are set up or interrogated through the bus. A block diagram for mode 1 operation is shown in Fig. 2. For additional information refer to the CDP1854 data sheet.

Using The CDP1854 to Interface The CDP1802 Microprocessor in The CDP18S020 or CDP18S024 Kit

When the mode input (pin 2) is high (MODE = V_{DD}), the CDP1854 interfaces directly with the CDP1802 microprocessor. This interfacing can be done either in the user I/O area or on a PC board which plugs into the P2 connector of the Kit. A general CPU-UART connection diagram is shown in Fig. 3. Several options are shown in this figure.

The first option involves the choice of N-bit code which is used to select the CDP1854. One choice is to connect N0 to RSEL input, N1 to the CS1 input, and N2 to the CS2 input. The CS3 input is connected to V_{DD} . This alternative uses N decodes 2 and 3 (decoded by the CDP1854 chip). If the CDP18S021 Microterminal is used, N decode 3 will address the Micro-

*Note: The information in this Application Note is useful here; these Evaluation Kits, however, are no longer available.

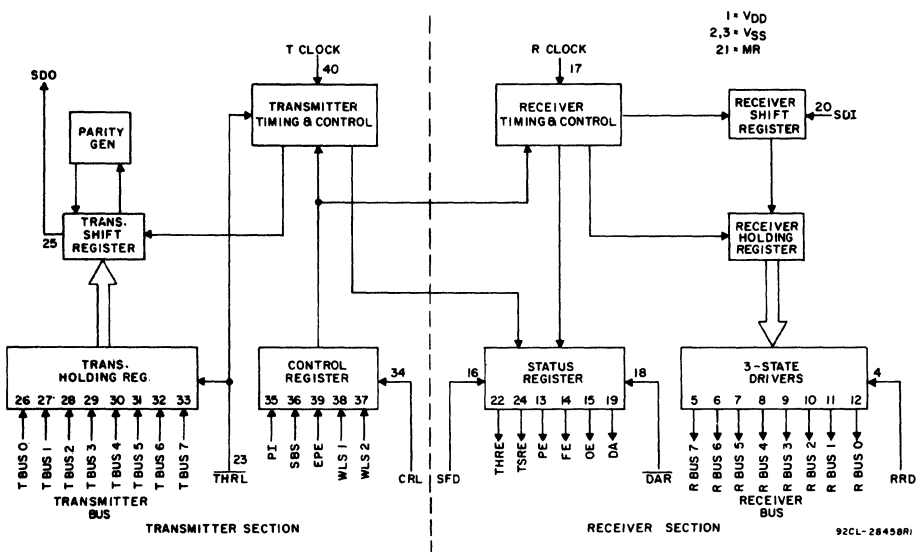


Fig. 1 - Standard Mode 0 block diagram.

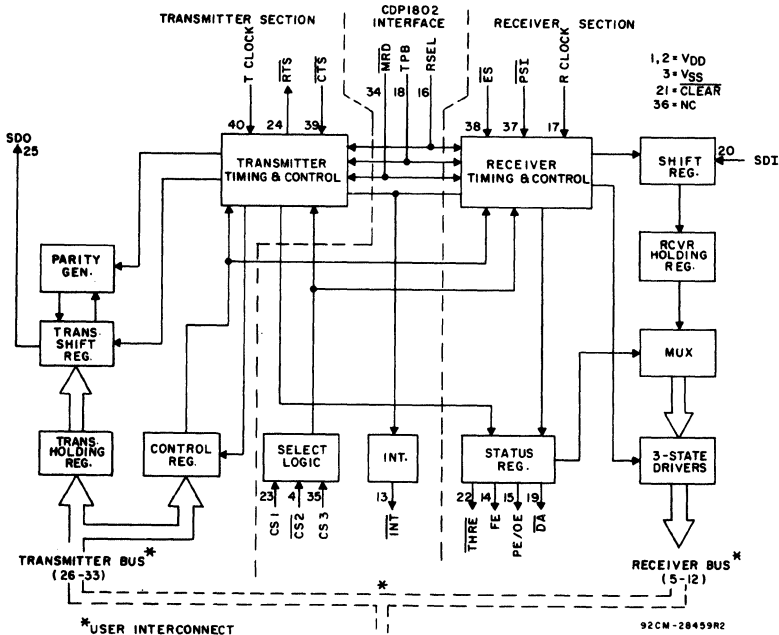


Fig. 2 - Mode 1 block diagram (CDP1802 compatible).

terminal. As long as the user's program does not output to the Microterminal, no conflict or display flickering will occur. If the CDP1852 output port is not used, an alternate connection is the following:

N0 to the CS1 input, N1 to the CS2 input, N3 to the RSEL input, and VDD to the CS3 input.

The first alternative will be discussed in the remainder of this section.

The second option shown in Fig. 3 involves the connection to the microprocessor interrupt and flag lines. For interrupt driven I/O, the INT output from the CDP1854 is connected directly to the INT input of the CDP1802. Determination of

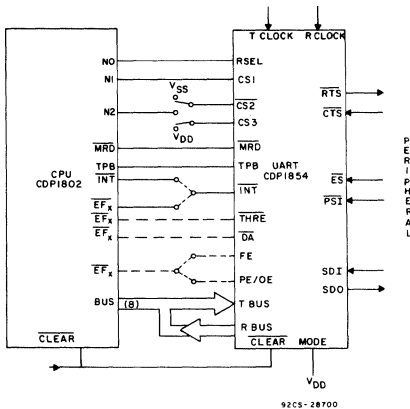


Fig. 3 - CDP1802/CDP1854 connection diagram.

whether an input request or an output request caused the interrupt can be made either by connecting THRE and DA to flag lines (EF1 through EF4) or by reading the UART's status register. A second approach is to test for input or output requests by polling flag lines. A third approach would be to connect the DA output to the microprocessor INT input and THRE to a flag line. For the remainder of this section transfer of data by polling flag lines will be discussed. A system in which DA is connected to the microprocessor EF1 input and THRE is connected to the microprocessor EF2 input will be used as an example.

In addition to the CDP1802 and CDP1854, only peripheral drive and sense circuitry and a clock or baud-rate generator circuit is necessary to complete the interface between the CDP1802 and a terminal.

A schematic diagram of the parallel to serial I/O interface is shown in Fig. 4. Table I lists a machine language routine for setting the UART's Control Register and continuously outputting the data byte in memory location 27 to a terminal until

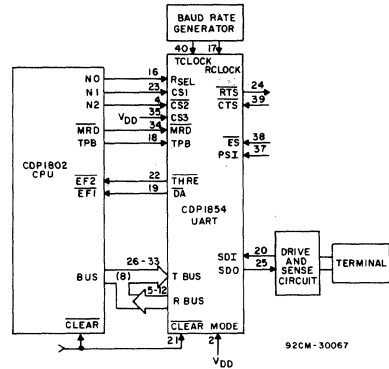


Fig. 4 - Interface of CPU CDP1802 to data terminal using UART CDP1854.

an input character is received and stored in memory location 28. The program then branches to the beginning of the Kit Utility Program (memory location 8000). A flow chart for this program is shown in Fig. 5. Correct program operation can be verified by observing that the terminal continuously prints the character (in this example "J") whose ASCII bit equivalent is listed in the program at address 14 until a key on the terminal is depressed. The program will then transfer control to the Kit Utility Program; the ASCII bit equivalent of the input character can be read from memory location 28. Proper operation will verify both correct hardware and software.

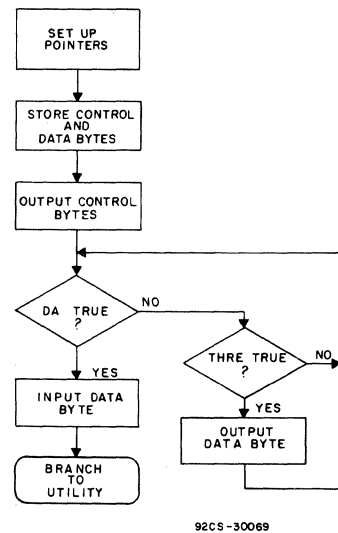


Fig. 5 - Program flow chart.

Table I - CDP1854 Interface Test Program

Memory Address	Byte	Operation	Comments
00	E5	5 → X	X = 5 R5 is Control Byte Pointer R6 is Data Byte Pointer 0025 → R5 0027 → R6
01	F8	00 → D	
02	00		
03	B5	D → R5.1	
04	B6	D → R6.1	
05	F8	25 → D	
06	25		
07	A5	D → R5.0	
08	F8	27 → D	
09	27		
0A	A6	D → R6.0	Store Control Bytes (3D and 80) in memory Output Control bytes to CDP1854 X = 6
0B	F8	80 → D	
0C	80		
0D	73	D → M (25), Dec R5	
0E	F8	3D → D	
0F	3D	Control Byte	
10	55	D → M (24)	
11	63	Output 3D	
12	63	Output 80	
13	E6	6 → X	
14	F8	4A → D	Store data byte (4A = J (ASCII)) in memory location 30 Test for DA
15	4A	Data byte	
16	56	D → M (27)	
17	3C	If DA = 0	
18	1E	Branch to 1E	
19	16	41 → R6.0	Response if DA true Branch to Utility
1A	6A	Input to M (28)	
1B	C0	Long Branch	
1C	80	to 8000	
1D	00		
1E	3D	If THRE = 0	Test for THRE
1F	17	Branch to 17	
20	62	Output data byte	
21	26	Dec R6	
22	30	Branch	Response if THRE true Branch to Test for DA
23	17	to 17	
24		Location for storing Control Byte	
25		Location for storing 80 Control Byte	
26			
27		Location for storing Output Data Byte	
28		Location for storing Input Data Byte	

I/O SYSTEMS USING THE CDP1854 on THE CDP18S024

One or more CDP1854 UART's can easily be interfaced with the CDP18S020 or CDP18S024 Kit. The system described in the previous section of this note is an

example of a single-level I/O system with one CDP1854, a CDP1852 used as an input port, and a CDP1852 used as an output port. The following CDP1802 I/O instructions with the corresponding N-line decodes are used in this system:

I/O	N - Decode	Device (Register) Addressed
62	2	CDP1854 (THR) Transmitter Data
63	3	CDP1854 (Control)
65	5	CDP1852 - Output Port
6A	2	CDP1854 (RHR) Receiver Data
6B	3	CDP1854 (Status)
6E	6	CDP1852 - Input Port

as an N-bit decoder. This system uses the following CDP1802 I/O instructions with the corresponding N-line decodes:

If output port (U4) is not used, it can be used to replace the two level I/O chip select latch. I/O Instruction 65 will then be the two-level I/O select instruction.

I/O	N - Decode	Device (Register) Addressed
62	2	Selected CDP1854 (THR) Transmitter Data
63	3	Selected CDP1854 (Control)
65	5	CDP1852 Output Port
67	7	2 - Level I/O Device Select
6A	2	Selected CDP1854 (RHR) Receiver Data
6B	3	Selected CDP1854 (Status)
6E	6	CDP1852 Input Port

A maximum of three CDP1854's can be used in a single level I/O system. Such a system would leave only one I/O instruction for data transfer to or from a CDP1852 I/O Port. A larger number of I/O devices can be connected to a CDP18S020 or CDP18S024 Kit by using a two level I/O Structure. A system with 2 I/O Ports (included on the CDP18S020 and CDP18S024 Kits) and eight UART's is shown in Fig. 6. A CDP1852 I/O Port is used as a chip select latch, and the CD4028A included in the Kit (U7) is used

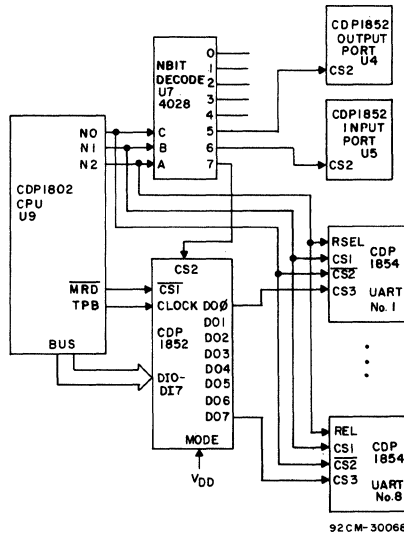


Fig. 6 - Diagram of two-level I/O system.

Use of CMOS ROM's CDP1833 and CDP1834 With the RCA Microprocessor Evaluation Kit CDP18S020 and the EK/Assembler-Editor Design Kit CDP18S024*

The RCA 1800 family of microprocessor products includes two 8192-bit static CMOS mask-programmable read-only memories, the CDP1833 and CDP1834. Each is organized as 1024 8-bit words, but they differ in addressing structure. The CDP18S020 Evaluation Kit and the CDP18S024 EK/Assembler-Editor Design Kit are designed to accept the CDP1833 and CDP1834 as described in this application note.

APPLICATION OF CDP1833 IN EVALUATION KIT AND EK DESIGN KIT ROM SYSTEMS

The CDP1833 read-only memory is functionally identical to the CDP1831 read-only memory except for the storage array size (8192 bits versus 4096 bits) and addition of a mask-selectable chip-enable input (CEI). (Refer to ICAN-6536 "Use of CMOS ROM's CDP1831 and CDP1832 with the RCA Microprocessor Evaluation Kit CDP18S020" for a description of CDP1831 features. All features of the CDP1831 apply to the CDP1833.) For the CDP1833 the 10 least significant bits of the 16-bit address select one of 1024 bytes in the ROM, and the six most significant bits of the 16-bit address select one of 64 1024-byte sectors. (Refer to Figs. 1 and 2 in ICAN-6536 for a comparison with the CDP1831.)

An additional feature of the CDP1833 is the mask-selectable chip-enable input (CEI). This input, as shown in Fig. 1, is

logically "OR'd" with the internal ROM select signal to generate the chip-enable output (CEO). As a result, multiple CDP1833's can be cascaded by appropriate connections of CEO and CEI to generate a combined CEO signal.

The CEI signal appears on pin 22 of the CDP1833. To maintain socket compatibility with the CDP1831, which has no connection on pin 22, the user can either program out the CEI input (when ordering custom patterns) or provide a 22-kilohm pull-down or pull-up resistor on pin 22 when CEI is selected positive or negative, respectively.

The CDP18S020 and CDP18S024 Kits have been prewired to accept the CDP1833 in position U1. Fig. 2 shows the U1 pin connections. The eight CDP1802 memory address lines (MA0-MA7) and TPA are wired to the appropriate ROM pins. The CDP1833 ROM outputs are connected directly to the data bus. The CS1 and CS2 inputs are connected permanently high, and the MRD input is connected to the CDP1802 MRD output. As a result, the ROM puts data on the bus when MRD = 0. The chip-enabled output CEO is connected to the \bar{E} (pin 1 of U22) control input of the RAM system decoder. When the ROM is addressed (enabled), the CEO output goes high and disables the RAM system. The CEO output can be used to disable the CDP1832 Utility Program ROM (U2) as shown in Fig. 3. Link 3 must be broken and the indicated gating inserted. The effect of the extra gating is to "OR" the CEO output with the existing

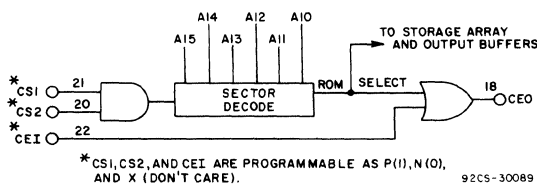


Fig. 1 - Function of CEI for CDP1833.

*Note: The information in this Application Note is useful here; these Evaluation Kits, however, are no longer available.

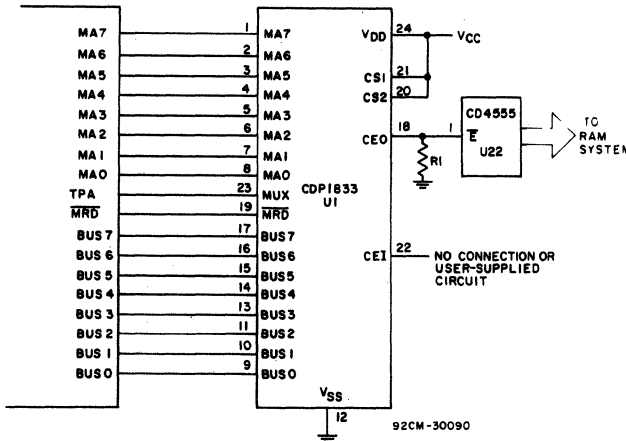


Fig. 2 - Pin connections of CDP1833 ROM in U1 location of CDP18S020 or CDP18S024 Kit.

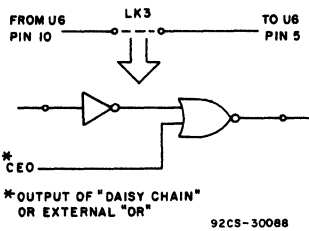


Fig. 3 - Use of CDP1833 CEO output to disable the CDP1832 ROM (U2) Utility Program.

CDP1832 \overline{CS} signal. When using multiple 1833's, the "daisy chained" CEO output is used as shown. These controls assure the ability to place the CDP1833 sector address in any of the possible 64 locations regardless of the RAM or ROM memory space allocation. Note that a pull-down resistor ($R1 = 22k\Omega$) is on the CEO line to assure proper operation of the RAM system when the CDP1833 ROM is removed. When the system operates with a ROM in the U1 location, this resistor may be removed.

The CDP1833 VDD (pin 24) is connected to the Kit VCC. This connection permits maximum memory-system flexibility with or without the CDP1833.

To use the U1 position for the CDP1833, the device is simply inserted in the location as indicated on the PC card. (Optionally, R1 (22 kilohms) can be removed.) Proper operation can be verified by use of the ?M[Starting Address]Δ [400] command of UT4 to list the memory contents. This check assumes that the CDP1833 does not occupy locations 8000₁₆ to 83FF₁₆.

Expanding the ROM System on the Evaluation or EK Design Kit is especially easy. In the User I/O section of the PC card, the CDP1802 memory address bus,

data bus, \overline{MRD} , and TPA signals are provided on the left side. Multiple CDP1833's (up to 12) can be inserted in the pre-drilled locations and interfaced to the address and data buses. A combined CEO output can be generated by appropriate CE1/CEO connections.

It should be noted that memory pattern generation and verification are possible in the Kit RAM system by using the write-protect and battery hold-up features. Once a pattern has been verified, it can be permanently stored by the custom masking of one or more CDP1833 ROM's.

APPLICATION OF CDP1834 IN EVALUATION KIT AND EK DESIGN KIT ROM SYSTEMS

The CDP1834 read-only memory is functionally identical to the CDP1832 read-only memory except for the addition of a tenth address input on pin 22 to select one of 1024 8-bit words and mask-selectable chip selects on pins 18 and 20. (ICAN-6536 describes the operation of the CDP1832 in detail; the description applies equally to the CDP1834 except as noted above.)

The CDP1834 is pin-compatible with the 2708 1024 x 8 bit erasable and electrically reprogrammable ROM and will operate in the same socket when pin 20 is programmed as $\overline{CS}(N)$ and pin 18 as $\overline{CS}(P)$ or don't care (X).

Both the CDP18S020 Evaluation Kit and the CDP18S024 EK Design Kit have been prewired to accept the CDP1832/2704 and CDP1834/2708 combinations at location U2. However, only the Evaluation Kit CDP18S020, when operated at suitable voltages, can accommodate the 2704 and 2708 because these two types are limited to a VCC of 5 volts. Links 1A and 1B per-

sonalize U2 for either combination. The Evaluation Kit PC board is factory programmed for the CDP1832/2704 combination. To change to the CDP1834/2708 combination, remove link 1A (scratch off with a sharp object) and insert link 1B. This change removes ground from U2 pin 22. The ROM which is inserted in location U2 will be selected when either A15 (address 8000_{16}) or RUN U are true. (Refer to ICAN-6536 for operating details when RUN U is activated.) Fig. 4 shows pin con-

nections for the CDP1834 ROM in location U2 after the proper link adjustments have been made.

Expanded CDP1834 ROM-based memory systems are easily constructed in the User I/O section of the Evaluation Kit and EK Design Kit PC card. In addition to wiring ROM locations, address latch and decode functions must be provided. A sample 4-kilobyte system which can be constructed in the User I/O section of the PC card is shown in Fig. 5.

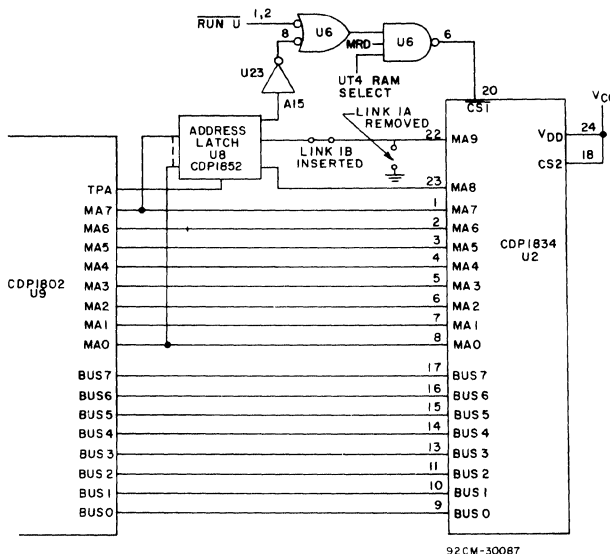


Fig. 4 - Pin connections for CDP1834 ROM in U2 location of CDP18S020 or CDP18S024 Kit.

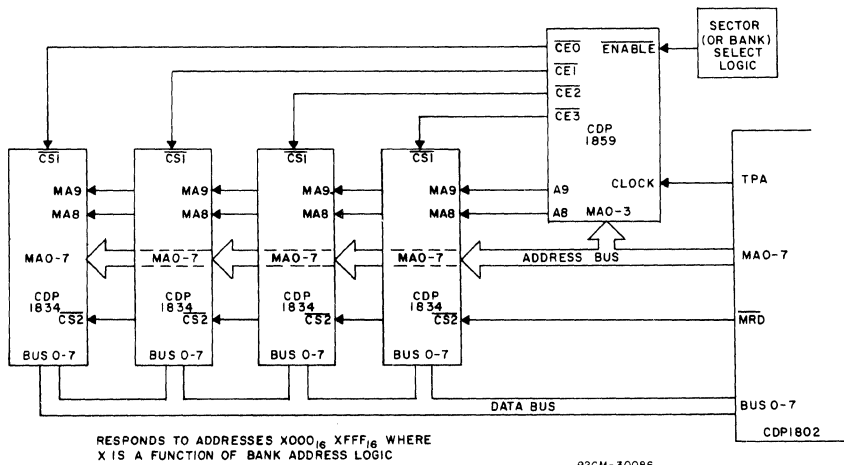


Fig. 5 - Sample 4-kilobyte ROM system using CDP1834 ROM's.

Use of the CDP1856 and CDP1857 Buffer/Separator In CDP1802 Microprocessor Systems

by J. Oberman

The RCA CDP1856 and CDP1857 are four-bit bus buffers or separators intended for those applications that require interface with the CDP1802 bidirectional three-state data bus. They can be used to buffer the bidirectional data bus, for increased driving capability, or to separate the data bus into two unidirectional data buses. This Note describes the uses of the CDP1856 and CDP1857 and, more specifically, how they may be utilized in the RCA Evaluation Kit, CDP18S020, and the E-K/Assembler-Editor Design Kit, CDP18S024.

DESCRIPTION

The functional logic diagram for both the CDP1856 and CDP1857 is shown in Fig. 1. Both parts require a positive chip-select input signal to enable their outputs. They differ only in the polarity of the MRD input signal required to transfer data to the bidirectional data bus. The CDP1856 requires a negative polarity MRD signal, and can be used to buffer or separate data transfers between memory and microprocessor. The CDP1857 requires a positive MRD input signal and

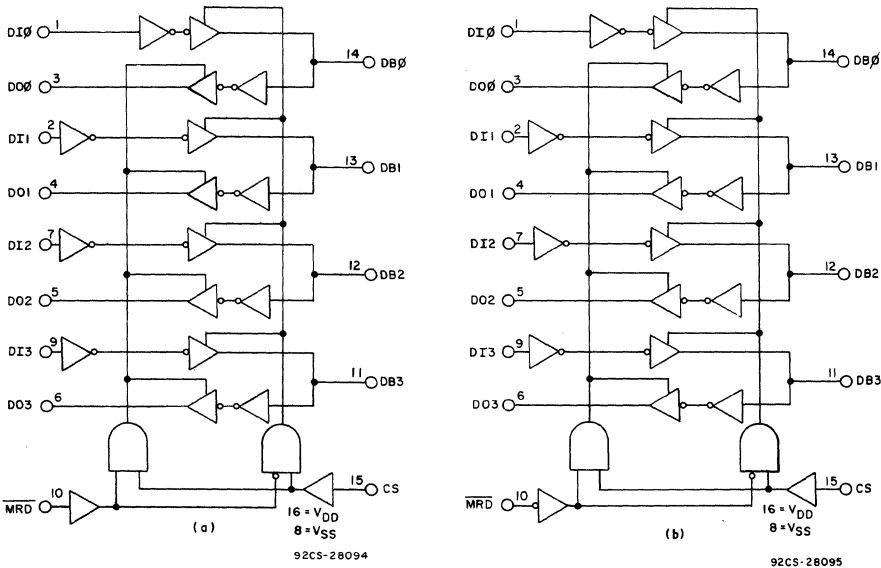


Fig. 1 - Functional diagrams (a) CDP1856, (b) CDP1857.

can be used for data transfers between the bus and various I/O devices.

As shown in Fig. 1, a typical buffer consists of two three-state drivers. On the bus side, the input of one driver is connected to the output of the other driver; they share a common terminal. However, their respective output and input are connected to separate terminals. When the chip is selected, only one output will be enabled, depending upon the polarity of the MRD input signal and the particular part.

APPLICATION INFORMATION

The CDP1856 may be used as a bus separator or bus buffer on the memory side of the bidirectional data bus. Similarly, the CDP1857 can be used for the same functions on the I/O side of the data bus. If an I/O command (N-bits) is present, the data transfer is from I/O to memory and microprocessor, otherwise the transfer is between microprocessor and memory. The MRD command, pin 7 of the CDP1802, determines the direction of the transfer.

If an I/O command is present, and MRD is high, the data transfer is from I/O device to memory and microprocessor, I/O M(R(X)), D. The MWR command will be asserted by the microprocessor. If MRD is low, the transfer is directly from the memory to the I/O device, M(R(X)) I/O. The microprocessor will ignore the data placed on the bus.

For a non-I/O instruction, I/O command lines low, a low level on the CDP1802 MRD output terminal indicates a data transfer from memory to microprocessor. If MRD is high, a data transfer from microprocessor to memory may occur; the output of data from the microprocessor to the bus and the

presence of an MWR command later in the cycle will be the only indication of this occurrence.

The CDP1856 or CDP1857 may be used in conjunction with the RCA COSMAC Kits, CDP18S020 and CDP18S024 when large amounts of logic will be added to the microprocessor's data bus. The components may be mounted in the user area provided on the card with connections made directly to the user I/O connector if the additional logic is to be breadboarded externally. The points for connections to the data bus and MRD command are available along the left-hand side of the user area. Decoded I/O commands, I/O - 1 to I/O - 7, are available at the system connector, P-1.

Fig. 2 illustrates two techniques for using the CDP1856 either as a bus buffer or a bus separator to reduce the loading effects of large memory systems on the data bus. If the memory employs common I/O pins, the buffer configuration should be used. The bus separator configuration is useful if the memory does not have an output disable input or if it is desirable to maintain separate inputs and output connections to the memory devices.

The use of the CDP1857 is illustrated in Fig. 3. It is used on the I/O side of the data bus; its chip enable input may be connected to an I/O command output. Therefore, the CDP1857 can be used to gate information to a particular I/O device under control of the microprocessor. As discussed above, the use of the buffer or bus separator configuration will depend upon the needs of the particular application.

The CDP1856 and CDP1857 provide an efficient low-cost solution to the problem of interfacing with the CDP1802 bidirectional data bus.

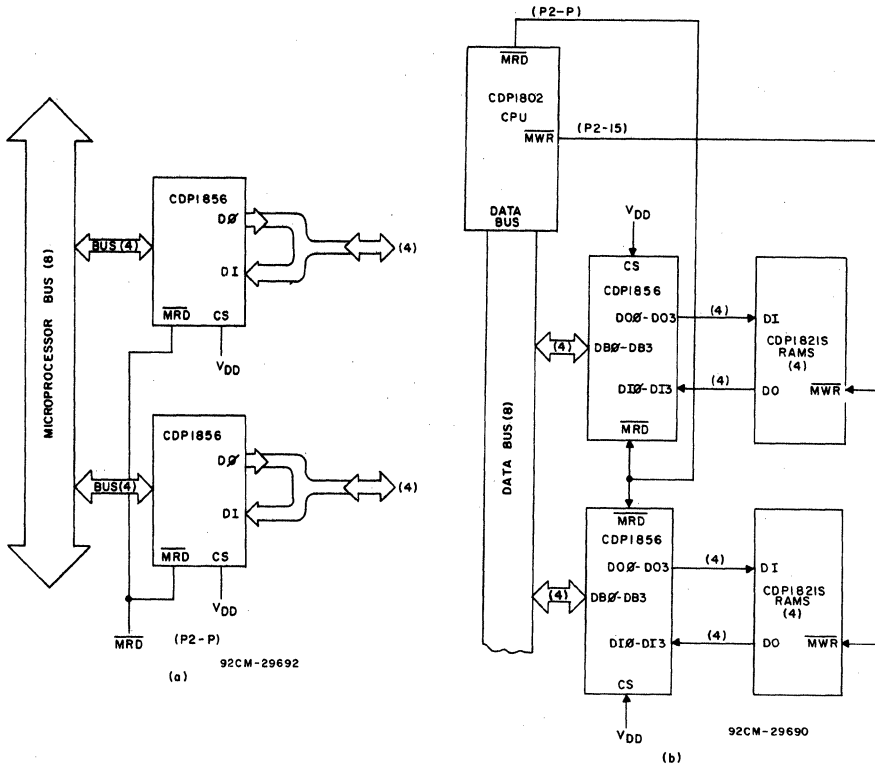


Fig. 2 - (a) Memory bus buffer, (b) memory-data bus separator.

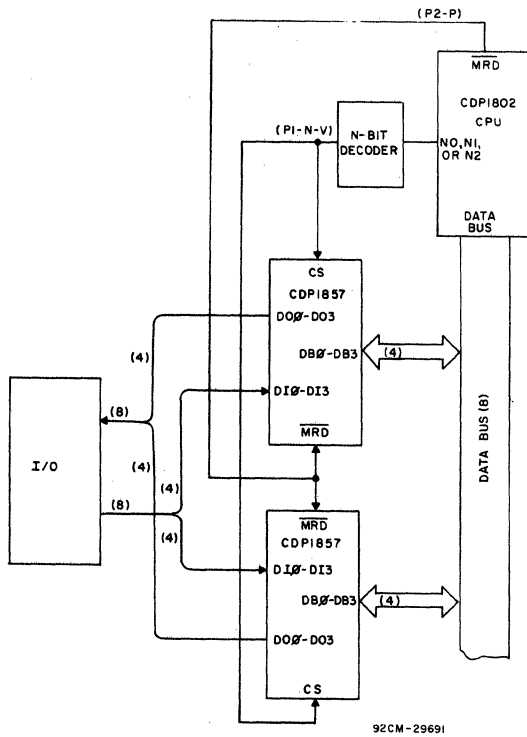


Fig. 3 - I/O bus separator.

Software Control of Microprocessor-Based Realtime Clock

by Kaare Karstad

In many microcomputer applications, a precision time display such as a realtime clock or an accurate time delay must be generated to sequence program flow or to supervise assigned programming tasks. For example, in a data acquisition system, numerous measuring stations producing different physical quantities must be polled for data at predetermined rates to obtain a sufficient number of samples per unit time for each quantity; thus, this system would require a multicycle timer. Also, each sampled value must be associated with its realtime position. Another system example is where multistage processing cycles are programmed and executed sequentially, requiring a clock and a count-down timer.

In microprocessor-based systems, three timing options are usually available. The central processing unit (CPU) may have an on-chip timer or an interface with an external timer, or it may have to generate time bases under software control. Either a hardware or a software design may be implemented for measuring and monitoring real time, and each approach should be evaluated according to the specific application for cost-performance effectiveness.

A microprocessor-based clock system can generate a time base entirely in software. In this design, a 12-h clock, 4-yr calendar, and 12-h elapsed-time indicator are implemented and displayed on a 3½-digit multiplexed readout unit. The system is programmed with priority for the 12-h clock, and the

other two timing functions are displayed for a predetermined time upon demand. The CDP1802 microprocessor¹ is a large-scale integration (LSI), complementary metal-oxide semiconductor (CMOS) 8-bit register-oriented CPU designed for use as a general-purpose computing or control element in a wide range of stored program systems.^{1,2,3,4} It includes a dedicated direct-memory access (DMA) pointer on chip, which is used to generate timing functions with accuracies limited only by the crystal-clock oscillator tolerance. Designers can build a dedicated multifunction time display or simply incorporate the clock functions in specific applications. Advantages of the described system design are low cost, no external parts, high accuracy, and minimal software overhead. Inherent CMOS benefits include minimum power-supply requirements, completely static circuitry (no refreshing needed), wide temperature range, high noise immunity, and CMOS, transistor-transistor logic, and n-channel MOS interface compatibility.

Time Measurements

For a hardware design, timing can be done by external logic, in which a clock signal interrupts the microprocessor and updates a counter at a fixed time interval. For a software design, the microprocessor performs timing by monitoring its instruction flow, since each section of the program executes within a specific time. While the software ap-

proach does not require external hardware, it does require skilled programming. To adjust a time-base interval for a constant value requires careful structuring, particularly with microprocessors having variable length instruction times. Interrupt and subroutine effects must also be thoughtfully evaluated.

For most microprocessors, the instruction time varies considerably within a given instruction repertoire according to which instruction is being executed. If the program also branches to a subroutine, which may call other subroutines to several levels deep, the task of generating a constant time-base period in software becomes complex. However, by utilizing the DMA function integrated into the CDP1802, an accurate software time base can be generated easily, regardless of the instruction sequence.

DMA Technique

Generally, DMA is a mechanism that allows an input/output (I/O) device to take control of the CPU for one or more memory cycles, in order to write into or read from memory. However, a different DMA technique in the CDP1802 generates a steady time base. For example, a register in this microprocessor is preloaded with an address to a specific memory location. When a DMA-IN request is generated by a peripheral, an 8-bit byte on the data bus is written into the addressed location. This write cycle takes place without CPU intervention, as a cycle is stolen from the instruction sequence. CPU execution then continues where it left off. Also, the address pointer is incremented to the next memory location, ready for the succeeding DMA cycle.

The CDP1802 microprocessor has 16 16-bit scratchpad registers—R(0) to R(F)—available to the programmer. For DMA operation, the first register, R(0), is automatically activated as a DMA address pointer, and is incremented once for every byte read from or written to random-access memory (RAM) as follows.

- DMA-IN: BUS→M(R(0)); R(0) + 1 (1)
- DMA-OUT: M(R(0))→BUS; R(0) + 1 (2)

In the DMA-IN mode, a byte on the data bus is written into memory location M(R(0)), addressed by the content of register R(0). Then, register R(0) is incremented to the next address.

If DMA-OUT is asserted, the microprocessor first completes its current instruction cycle and then executes a DMA cycle. When the DMA request is removed, the microprocessor returns to the next instruction; otherwise, it continues to execute DMA cycles as long as DMA-OUT is true. The microprocessor generates its own DMA request by hardwiring state code line SC1 to DMA-OUT (Fig. 1(a)). State code lines SC0 and SC1 are always interpreting one of the four CPU states—fetch, execute, DMA, or interrupt.

A normal CPU instruction sequence consists of two cycles—fetch and execute with SC1 low. When the microprocessor samples a DMA-OUT request (SC1 goes high), it responds with an added—or third—DMA cycle (Fig. 1(a)). When SC1 returns low, it effectively resets the DMA cycle so that the microprocessor reverts to the next fetch-and-execute pair. In this manner the normal 2-cycle sequence of fetch-execute is modified to a 3-cycle sequence: fetch-execute-DMA.

Since every DMA cycle increments register R(0), a mechanism exists for incrementing a counter automatically at a fixed known rate as the program is run, regardless of instruction type and program structure. The only restriction is to avoid CDP1802 3-machine cycle instructions (op codes C(N)), which require a fetch followed by two execute cycles; this involves not using the long branch, long skip, and no operation instructions, 17 of the available 91 basic instructions.

Measuring time is now reduced to monitoring B15—the most significant bit (MSB) of R(0). During the time that R(0) counts to 2¹⁶, bit B15 changes from 0 to 1 half way through the count (Fig. 1(b)). Thus, the designer can choose a convenient time increment to fill the register (e.g., 1 s) and then calculate the required crystal-controlled oscillator frequency.

$$T = \frac{1}{F} \times N \times M \times 2^{16}$$

where
 N=number of clock pulses per machine cycle (eight for CDP1802)
 M=number of machine cycles in modified (fetch-execute-DMA) instruction (three)

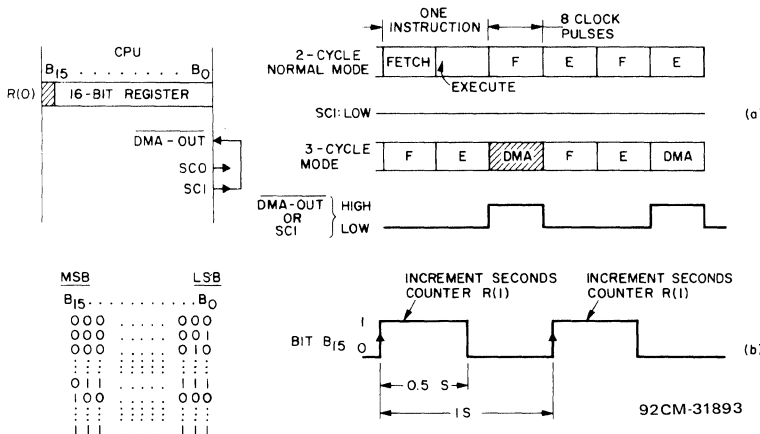


Fig. 1 — DMA technique. Normal Fetch-Execute instruction sequence (a) is modified to include DMA cycle by hardwiring state code line SC1 to DMA-OUT. Each time instruction is executed, 16-bit register is incremented. Monitoring MSB B₁₅ state transitions from 0 to 1 (b) reveals when 1 s has elapsed.

Therefore, for $T = 1 \text{ s}$, $F = 1.572864 \text{ MHz}$.

Four bytes in RAM are assigned to store the four digits (two for hours and two for minutes) for the 12-h clock display. The lower order byte of scratchpad register R(1) in the microprocessor is allocated as a counter for seconds. Consequently, updating the clock buffers in RAM becomes a matter of incrementing the right digit(s) once every 60 s to count minutes.

The required main program (see Fig. 2) is short and consists of calls to various subroutines, such as updating, display, and setting the clock time. However, each pass through the main program varies in length according to branch conditions. For instance, if the test for 60 s is not met, no updating of the four RAM buffers is required, and the updating section of the program is bypassed. The important point is that the microprocessor must check the 0 to 1 transition of B₁₅ in R(0) at least once every 0.5 s, and then increment the R(1) seconds counter if a transition takes place (Fig. 1(b)).

With the microprocessor multiplexing the four display digits at a 100-Hz refresh rate (well above the flicker rate of about 50 Hz), 10 ms is the longest time that the CPU is tied up in timing loops. A complete pass after each update takes only 1 to 2 ms in the full program under discussion. Hence, the critical transition test of R(0), at least once every 0.5 s, can easily be met.

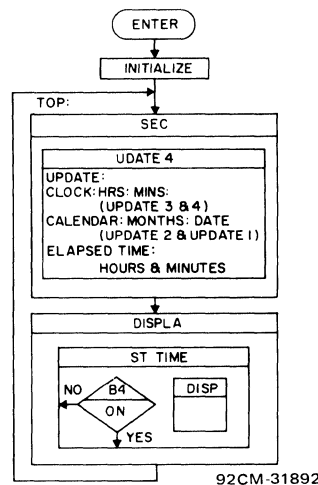


Fig. 2 — Program flowchart. Main program calls only two major subroutines—SEC and DISPLA. SEC calls UPDATE for updating RAM buffers. If clock is being set, DISPLA calls STTIME, which again calls part of DISPLA and debounces pushbutton. SEC counts seconds, generates minutes, updates clock, elapsed time, and calendar. Updating takes place when SEC calls subroutine UPDATE4.

System Description

The microprocessor-based clock system (Fig. 3) can output 8-bit bytes of information from any of its 16 scratchpad registers with only one output instruction, causing the 16 bits to go out over the memory address (MA) bus. Since this bus is only eight bits wide, the higher order byte appears first and is latched in the CD1852 input/output (I/O) port by

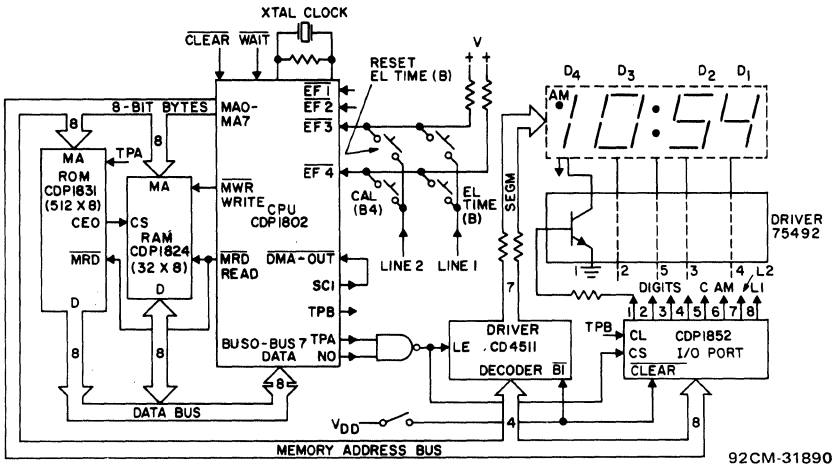


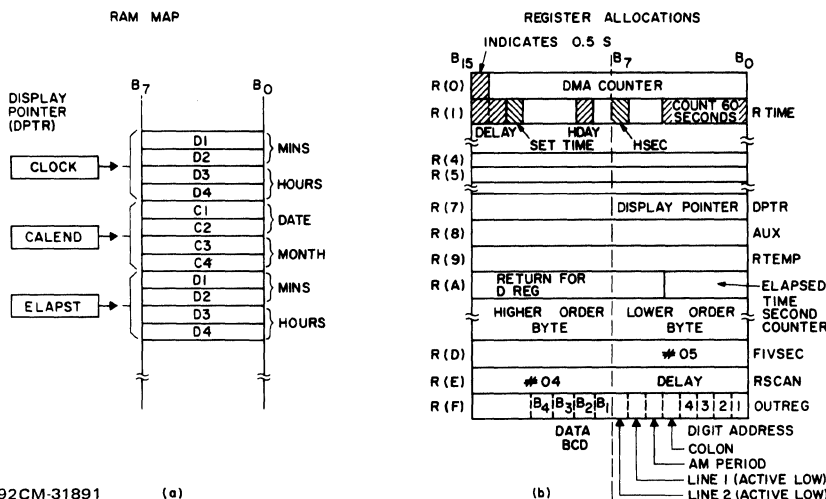
Fig. 3 — Microprocessor-based clock system. Timing functions include multiplexing and outputting to 3½-digit display, accurate timekeeping in software, and monitoring four request push-buttons. Software gives priority to 12-h clock display; upon request, elapsed time or 4-yr calendar display stays on for predetermined time.

timing pulse A (TPA). The lower order byte follows later in the same machine cycle and is latched by timing pulse B (TPB) in another I/O port.

In the microprocessor, a 16-bit scratchpad register, R(F), stores the current content of a memory location for the multiplexed 3½-digit display (Fig. 4). Four binary-coded decimal (BCD) data bits of the higher order byte are decoded by the CD4511 interface device, which drives the seven segments of each light-emitting diode (LED) display digit. Eight bits of the lower order byte, via the CDP1852 I/O port, specify the digit address for multiplexing the four LED displays, as well as additional information for enabling the colon and am time-period displays. When the program executes an output instruction, this 12-bit word travels over the memory address bus in two bytes to the output ports selected by I/O command N0 and provides complete display information for each multiplexed digit. Upon completion of each machine cycle after data are latched, the I/O ports are deselected by N0. Register R(F) bit assignments for the required 12-bit output of the memory address bus are shown in Fig. 4. For example, when the least significant bit (B0) is set to 1, display digit D1 is selected. In the 12-h clock mode, colon bit B4 is also 1, and period bit B5 is 1 during am. Content of the first location in clock buffer D1 (minutes) is next stored as BCD data (bits B8 to B11) in the higher order byte of R(F). An out-

put instruction then outputs the 12 bits, thereby turning on display digit D1. A timing loop in software keeps this digit on for 2.5 ms, after which a test is made to determine whether all four digits have been handled.

In the primary or clock mode of operation, the system displays hours and minutes continuously, and calendar (month and date for up to 4 yr) or elapsed time (hours and minutes) upon pushbutton request. The latter two displays stay on for a pre-programmed time (for example, 5 s), and program control then automatically reverts to the 12-h clock display. The implemented system has four operator-activated pushbuttons—calendar (B4), elapsed time, reset elapsed time, and one spare. Two external flag lines (EF3 and EF4) and two scan lines (L1 and L2) form a 2 by 2 matrix (Fig. 3) from the I/O port to test pushbutton status. In the programmed I/O mode, data transfers are controlled and timed by the program. The flag lines sense external events, i.e., whether the logic levels on the flag lines are high or low. At a certain point in the program sequence, one scan line at a time is made active low. If one of the two pushbuttons (Fig. 3) connected between this scan line and either flag line (EF3 or EF4) is depressed, the activated pushbutton is identified by branch instructions that test the two flag lines. If no pushbutton is depressed, the program proceeds to output hours and minutes data, which is the normal 12-h display mode. If, for instance, the elapsed-



92CM-31891 (a) (b)
 Fig. 4 — CPU registers. Memory map (a) shows RAM buffers for clock, calendar, and elapsed time. Register and bit allocations (b) are contained in CPU for programming. DMA pointer R(0) keeps track of time; register R(F) contains output data for LED displays.

time pushbutton is activated, the RAM pointer moves to the four elapsed-time buffers, and the system displays elapsed time for a predetermined period.

Updating of RAM Buffers for Time Displays

A memory map of the RAM buffers to be updated is shown in Fig. 4(a), and major register allocations and bit assignments are defined in Fig. 4(b). In RAM, three buffer areas, with four bytes in each, are reserved as display buffers for clock (CLOCK), calendar (CALEND), and elapsed time (ELAPST).

A 0.5-s flag bit (HSEC)—B₇ in R(1)—is assigned to monitor bit B₁₅ in the DMA counter, R(0). Assume that the HSEC bit is 0; thus, when B₁₅ is 0, 0.5 s has not elapsed, and software control returns to the main program. However, when B₁₅ goes to 1, the 0.5-s transition has occurred, and the HSEC bit is updated to 1. The lower five bits of register R(1), assigned as a counter for 60 s, are next incremented, and the program checks whether the count is 60 or greater. If the answer is yes, a minute has passed. Therefore, the 60-s counter (five low order bits of R(1)) is reset to 0, and the four RAM locations (D1 to D4) storing the clock hour and minute values are updated. The next step is to check whether the lower order byte of register R(A)—allocated as an elapsed-time seconds computer—has overflowed. If the answer is yes, the four elapsed-time RAM buffers are updated.

Register R(7) is loaded and points to the least significant digit (D1) of MINS (minutes) in the RAM clock buffer. The addressed memory byte is incremented and examined for updating. If the value is less than nine, a 0 is stored in the buffer, and the minutes D2 digit is incremented and examined. If D2 is greater than five, a 0 is stored in the location for D2. For a 12-h clock, it is necessary to examine only D4 for 0 to 1. If D4 is 0, D3 is incremented and tested for the nine limit. If D3 is greater than nine, a 1 is loaded into D4 and a 0 into D3. If D4 is 1, then D3 is incremented and examined. When D3 is greater than 2, a 1 is loaded into D3 and a 0 into D4, since the clock changes from 12:59 to 01:00. Note that while the clock makes this change, the elapsed-time display proceeds from 12:00 to 00:01. Consequently, minute digits D1 and D2 are handled differently by the program according to whether the clock or elapsed-time buffers are being updated. Therefore, a test is made by the program to determine which buffer area is being updated. This information is provided by comparing the current address of the display pointer (DPTR) with the known fixed address for that location in the elapsed-time buffer.

With the clock and calendar displays, am and pm must be tracked for 24-h monitoring. Flag bit (HDAY)—B₉ in R(1)—is defined as 0 for pm. Once every 12 h, when D3 is 2 at noon and midnight, the HDAY flag is complemented and examined. For

pm, the stored HDAY bit reads 0; no action is needed and software control returns to main. When the HDAY flag reads 1 (am), the 24-h transition point has been passed, and the program proceeds to update calendar month and date.

This system implements a 4-yr calendar, in which the correct number of days for each month is indicated automatically by pointing the value of the current month to a table stored in read-only memory (ROM). If the current date is less than the table entry, no action is required; otherwise, 01 goes into the two calendar-date buffers (C1 and C2).

Setting Clock and Keyboard Debouncing

A series of manual B4 (calendar) pushbutton depressions sequentially updates the contents of all 8 RAM buffers to the correct readout by incrementing the four LED display values.

Pushbutton Depression	Result
1. B4 and reset elapsed time (simultaneously)	Enter set clock mode; circulate month display
2. B4	Stop month display; circulate date display
3. B4	Stop date display; circulate hours display
4. B4	Stop hours display; circulate minutes display
5. B4	Stop minutes display; blink colon display
6. B4	Stop colon display; reset seconds display; run clock time

Flag bit B₁₃ (set time) is allocated in R(1) to indicate this mode. It is tested during all update routines, and essentially permits exits at appropriate program points; hence, the calendar-month value, for instance, is updated without affecting the other RAM buffers.

Entering the set clock module requires that both the calendar and reset elapsed-time pushbutton be depressed simultaneously. The program then points to the RAM buffers for calendar month and date and calls the display routine after a 0.5-s delay. As long as calendar pushbutton B4 is not activated, the third and fourth display digits will indicate the current content of the two month locations (C3 and C4) in the calendar buffer for 0.5 s. Next, the update routine for calendar month is called, and the contents in the two buffers

are incremented and updated. Then, the sequence is repeated. The new buffer content is displayed for 0.5 s, the buffers incremented, and the procedure repeated until pushbutton B4 is once more manually depressed. Pushing B4 again freezes the calendar-month value, and the program exits to the next icop, which starts updating the calendar date in a similar manner at a 2-Hz rate.

Loops for date, hours, and minutes are functionally similar to the loop described for month. They all use the update subroutine; only entry and exit points change according to which RAM buffer is being updated. At the end of this sequence, when all 12 RAM buffers have been updated, the colon starts blinking at a rapid rate (10 Hz) to indicate that the clock is set but not yet running. A final manual depression of B4 resets the seconds counter, R(1), and starts the clock.

When pushbutton B4 is depressed to set the clock, a second test is made after a time delay determined by the execution time of routine DISP to debounce the leading edge of the pushbutton depression, the only critical edge for operation. If pushbutton B4 is still held down, the program locks up in a loop for the duration of the depression. During this period, the contents of the selected RAM buffer are being displayed. Upon release of the pushbutton, the program exits from the debounce routine to the set time (STTIME) routine.

Main Program

With the assembly program structured in a number of interrelated subroutines, the main program (see panel of Realtime Clock Software Programs) contains only four instructions after initialization. These call two major subroutines—SEC (seconds) and DISPLA (display)—and one minor display subroutine (DISP). Instruction SEP R4 changes the current program counter to register R(4), and starting address A(SEC) for the SEC subroutine is loaded during the call. The SEC subroutine counts DMA requests (time base) and increments the 60-s counter, R(1). After each minute elapses, the SEC subroutine calls another subroutine (UPDATE 4) for updating the 12 RAM buffers. The 12-h clock is updated first, then elapsed time. Next, the main program

Realtime Clock Software Programs

Main Program

```

TOP:  SEP R4, A(SEC)    Count seconds,
                        generate minutes,
                        update clock,
                        elapsed time, and
                        calendar
                        SEP R4, A(DISPLA) Display content of
                        SEP R4, A(DISPL) relevant buffer
                        BR TOP          Go around again
    
```

SEC Routine

```

SEC:  GLO RTIME        Get HSEC bit
      SHL
      BNF HSEC0        BR if HSEC = 0
      GHI R0           Else, test for 1 to 0
      SHL              Transition
      BNF LABEL1      BR if transition
                        made
                        SEP R5         Else, return
                        Continue
      ---
    
```

```

HSEC0: GHI R0         Test for 0 to 1
       SHL
       BDF LABEL2     BR if made
       SEP R5         Else, return
       Continue
       ---
    
```

```

LABEL2: GLO RTIME     Set
        ORI #80       HSEC = 1
        PLO RTIME     Replace
        GLO DPTR      Hold DPTR
        PLO AUX       During update
        INC RTIME     Increment
                        seconds
        INC ETIME     Increment
                        elapsed
                        seconds
        GLO RTIME     Test for 60
        SMI B'10111100' Seconds
        BNF LABEL3    No, branch
        LDI #80       Zero seconds
        PLO RTIME     Leave HSEC = 1
        LDI A.0(CLOCK) Point to
        PLO DPTR      Clock buffer
        SEP R4,       Update clock
        A(UPDATE4)   and calendar
        Continue
        ---
    
```

UPDATE 4 Routine

```

UPDATE4: LDN DPTR     Get digit one
        ADI #01       Increment it
        STR DPTR      Replace it
        SDI #09       Test if > 9
        BM DIG10      BR if D1 > 9
        SEP R5        Return
DIG10:  LDI #00       Load 0
        STR DPTR      Into D1
        INC DPTR      Point to D2
        LDN DPTR      Get D2
        ADI #01       Increment it
        STR DPTR      Replace D2
        SDI #05       Test second digit
        BM DIG20      BR if D2 > 5
        SEP R5        Else, return
        Continue
        ---
    
```

DISPLA Routine

```

DISPLA: LDI A.0(CLOCK) Display pointer
        PLO DPTR      To clock
        GLO OUTREG    Turn off
        ORI B'10000000' Line 2
        ANI B'10111111' Turn on
        PLO OUTREG    Line 1
        SEX OUTREG    Line 1
        OUR 1         Active low
        DEC OUTREG
        B4 ETBUFF     BR if ETIME
                        Button on
                        BR if function X
                        Buffer on
        B3 FUBUFF
        Continue
        ---
DISP:  GLO OUTREG    Turn on
        ANI #F0       First digit
        ORI #01       Address
        PLO OUTREG
        LDI 04        Assign
        PHI RSCAN     Digit count
        Continue
        ---
    
```

STTIME Routine

```

STTIME: GHI RTIME    Turn on
        ORI B'00100000' Set time bit
        PHI RTIME
        LDI A.0(CALEND) Point to calendar
        PLO DPTR      Buffer
        LDI 50        Load half second
        PLO RTEMP     Delay constant
        SEP R4, A(DISPL) Display calendar
        SEP R4,       Button 4 on?
        A(BUTTN4)
        BNZ (LABEL4)  Yes
        DEC RTEMP     Else, no
        Continue
        ---
    
```

calls the display routine, with the address given by A(DISPLA).

STTIME, used for setting the clock, is called from the DISPLA routine. When the operator sets the clock and watches the display, only the lower part of the DISPLA routine is needed. This part is called from the STTIME routine by loading the program pointer with address A(DISPL).

A Standard Call and Return Technique³ (SCRT) uses two linking subroutines: one when the call operation is to be performed and the other when the return from the subroutine is to be done. Registers R(4) and R(5) must be initialized once in the program to point to the linking call subroutine and the linking return subroutine, respectively. A call to a subroutine is performed by executing a SEP R4 instruction. The two bytes following this SEP instruction must

contain the address of the subroutine to be called. Once a subroutine has been called and has completed its function, control should be returned to the caller by executing a SEP R5 instruction.

Seconds Subroutine

The SEC subroutine contained in the Realtime Clock Software Programs counts seconds, generates minutes, and updates clock, elapsed time, and calendar. Updating takes place when SEC calls subroutine UPDATE 4.

Bit B7 is stored in RTIME register R(1) as an HSEC flag. The first instruction fetches the lower order byte of register RTIME to the D register (CPU accumulator). The shift left instruction moves the bit into a 1-bit data flag (DF) register, where it can be tested. If the HSEC bit is 1, execution continues with the next instruction. The higher order byte of R(0) is fetched in order to test bit B15; B15 is then shifted left into DF and its content tested. If this bit is 0, the routine executes a branch to address LABEL 1 (routine not listed) and resets HSEC bit to 0. If B15 is 1, the program returns to main. Register R(5) is a dedicated program counter for the return routine. If the first test of the HSEC bit reads 0, the program branches to an address given by HSEC 0. Again, the higher order byte of R(0) is read in order to test B15, and the procedure is repeated.

Update 4 Subroutine

This routine updates the 12 RAM buffers pointed at by a display pointer whose address is either clock, calendar, or elapsed time (see Realtime Clock Software Programs). UPDATE 2—part of UPDATE 4—updates calendar month and date; UPDATE 1 is the entry point in the STTIME mode when month values are being updated. The lower order byte of register R(7) is allocated as a display pointer (DPTR). In the display routine, it is initialized to point at the first location in the clock buffer.

Instruction "Load via N (LDN)" places the byte addressed by register DPTR in the D (CPU) register. Instruction ADI adds the constant 01 to the content in the D register, and the next store (STR) instruction stores the ad-

dition result within the D register into the memory location addressed by register DPTR.

Current content of the D register is tested for limit nine by subtracting the constant 09, instruction SDI #09. If the result is negative, the program branches to address DIG10. If the result is less than nine, a return to main is executed by SEP R5, which calls the return routine.

At the address given by DIG10, the load immediate (LDI) instruction loads a constant 00 into the D register. The next store instruction puts the value into the first (D1) location of the clock buffer. Register DPTR is then incremented so it points to the next byte in RAM (D2). This byte is loaded into the D register and a 1 is added to it, after which the result is stored back into D2 and the procedure is repeated for D3 and D4. After setting clock minutes D1 and D2, UPDATE 3 is the entry point within UPDATE 4 for updating the clock hours (D3 and D4) for both clock and elapsed-time buffers.

Display Subroutine

The display routine tests the status of the four pushbuttons and outputs data to the 3½-digit, multiplexed display. If a pushbutton is depressed, a new address is loaded into the display pointer, and data are output from the appropriate buffers for 5 s.

When the routine contained in the Realtime Clock Software Programs is called, the first instruction loads the address for the display pointer. Address A.0 (CLOCK) is the lower order byte for the clock buffer, and LDI puts it into the D register. Instruction PLO (put low) loads the content of D into the lower order byte of DPTR. The lower order byte of OUTREG—R(F)—is fetched to the D register, and its content is ORed with byte 8016 (1000 0000₂). Since scan line 2 is active low, ANDing the current content of D with hexadecimal constant BF (1011 1111₂) turns on scan line 1 and leaves line 2 unchanged, i.e., off. PLO puts this information back into OUTREG, and OUT 1 puts the content of OUTREG on the data bus. Scan line 1 is now active low. Since the output instruction incremented the data pointer, a decrement is done to reset the pointer.

Instruction B4 tests input flag $\overline{EF4}$ on the CPU. If the line is low and the elapsed-time pushbutton is depressed, then the program branches to address ETBUFF and sets the pointer to the elapsed-time buffer. If the test is not met, the program next tests the status of flag line $\overline{EF3}$, and the procedure is repeated.

Set Time Routine

Called only when setting the clock, the STTIME routine (see listing of Software Programs) calls debounce routine BUTTN4 and four update routines at various entry points, according to the values being updated: month, date, hours, or minutes. While the operator watches the LED displays during the STTIME mode, the entry to the display routine is at DISP.

Bit B13 in the RTIME register (R1) is allocated as a set time flag. The first instruction puts the higher order byte into the D register. Its content is ORed with hexadecimal constant 20 (0010 0000₂). When the result is put back into RTIME by PHI, the set time bit is set.

Address A.0 (CALEND) is the lower order address byte for the calendar buffer. The LDI instruction puts this value into the D register, after which PLO loads the address into DPTR. Delay constant 50, equivalent to 0.5 s, is loaded by LDI into the D register and is next stored in the lower order byte of RTEMP—R(9). Instruction SEP R4 calls the display routine starting at address DISP; the procedure is then repeated.

Summary

Hardware and software aspects of a microprocessor-based realtime clock application are covered, with priority given to a 3½-digit, 12-h clock display. Upon request, the same digits can display either a 4-yr calendar in months and days, or elapsed time in hours and minutes for a pre-programmed period. After this, the display automatically returns to the 12-h clock mode.

A timer or realtime clock is frequently needed in many microcomputer systems. Unless hardware is provided on chip, or added externally, timing must be implemented entirely in software. This task often requires carefully structured programs. The described software controlled clock design overcomes these difficulties by taking advantage of the inherent architectural capabilities in the CDP1802 microprocessor, such as DMA. Running any random program sequence automatically provides a known accurate time interval (for example, 1 s). The microcomputer system can be dedicated to various timing functions, and can easily be incorporated into another system where various time displays are required, such as adding a seconds display or an accumulated elapsed time. With a CMOS microprocessor and associated circuitry, this system design can be battery operated and requires less than 2 mA at 5 V, exclusive of the LEDs, for a typical program of 512 bytes.

Acknowledgment

The author wishes to acknowledge the contributions of Larry A. Solomon to this project.

References

1. RCA Corp., Solid State Div., *User Manual for the CDP1802 COSMAC Microprocessor*, MPM-201B, Somerville, N.J., 1977.
2. RCA Corp., Solid State Div., "Register-Based Output Function for RCA COSMAC Microprocessors," Application Note ICAN-6562, Somerville, N.J., 1977.
3. RCA Corp., Solid State Div., "Standard Call and Return Technique," *User Manual for the CDP1802 COSMAC Microprocessor*, MPM-201B, Somerville, N.J., 1977, p. 61.
4. RCA Corp., Solid State Div., "COSMAC Microprocessor Product Guide," MPG-180A, Somerville, N.J., 1977.

CDP1802-Based Designs Using the 8253 Programmable Counter/Timer

by J. Kowalchik and S. Bartosevich

The 8253 programmable interval timer, manufactured by Intel Corp., is an integrated circuit containing three independent 16-bit counters, each programmable in any of six modes.¹ This Note describes methods by which it can be incorporated in RCA COSMAC-based microprocessor systems.

Features of the 8253

1. Three independent 16-bit counters.
2. Six possible modes of operation.
3. Programmable operating modes.
4. Binary or BCD counting.
5. Single 5-volt power supply.
6. Single 24-pin package.

System Advantages of the 8253

1. Frees CPU from menial timing chores.
2. Allows simultaneous control of multiple tasks.
3. Directly interfaced to COSMAC at 5 volts and 2 MHz clock.
4. Permits full use of COSMAC I/O when operated as a memory-mapped I/O device.
5. Consistent with minimum chip-count systems.

Functional Description of the 8253

The 8253 can be described as four 8-bit memory addresses, selected by A0, A1 and CS. Counter 0 is equivalent to address NNN0 where NNN is the user defined chip select. Counter 1 is at address NNN1, counter 2 at NNN2 and the mode control register at NNN3. Data can be written to or read from the three counters by pointing a CPU register at the proper address and using the various store and load instructions. Control words can only be written to the mode-control register; no read operations are allowed at this address.

Use in 5-Volt Systems

The 8253 can be directly interfaced to CDP1802 systems operating at 5 volts when configured as a memory-mapped I/O device. Fig. 1 is a block diagram of a typical system. All data lines (BUS0-BUS7), RD, WR and address lines (A0, A1) are compatible with corresponding COSMAC signals. The CS is a user-defined signal which can be compared to the chip select of a memory device. Each counter section has a clock input line. The maximum clock frequency as specified by the manufacturer is 2 MHz. Therefore, the CDP1802 system clock can often be used directly as the timer clock. If a lower-frequency clock is required by a particular counter section, it can be divided down from the system clock by one of the two remaining counters operating in the programmable squarewave rate-generator mode.

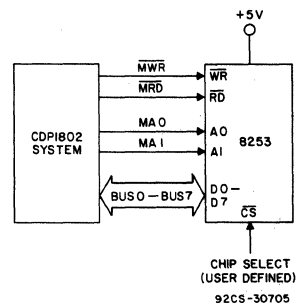


Fig. 1 — A typical CDP1802/8253 system.

Use in Higher-Voltage Systems

Since the 8253 must operate at 5 volts, a level shift is necessary when operating it with COSMAC systems at higher voltages. Fig. 2 shows such an interface circuit employing CD40109 level shifters. This example illustrates a 10- to 12-volt COSMAC system and full bidirectional level shifting.

It may not always be necessary to read data from the counters. When this is the case, a much simpler level shift circuit can be employed. Fig. 3 shows a unidirectional down

shifter using RCA CD4050 IC's. Only data writes to the 8253 can be made with this arrangement.

Communicating with the 8253

As a memory-mapped I/O device, the 8253 can be thought of as a group of four memory addresses. The COSMAC store and load instructions are used to write mode words, write data, and read data from the counter.

Assume that "COUNT" is a CPU register pointing at the 8253 mode-control register

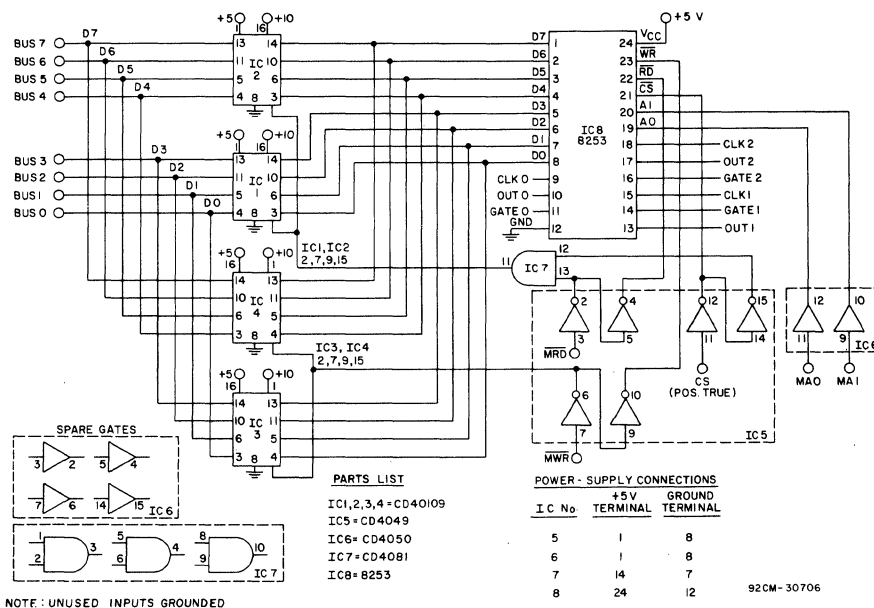


Fig. 2 - An interface circuit employing CD40109 level shifters.

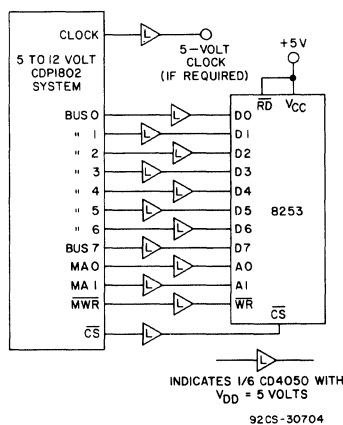


Fig. 3 - A unidirectional down shifter using RCA CD4050 IC's.

in memory. The following sequence of instructions might be used to format and load the three counters:

```

LDI    #31      .. MODE WORD FOR COUNTER 0
STR    COUNT    .. WRITE TO 8253 MODE REGISTER
LDI    #71      .. MODE WORD FOR COUNTER 1
STR    COUNT    .. WRITE TO 8253 MODE REGISTER
LDI    #B3      .. MODE WORD FOR COUNTER 2
STR    COUNT    .. WRITE TO 8253 MODE REGISTER
DEC    COUNT    .. POINT AT COUNTER 2
LDI    #34      .. LSB TO COUNTER 2
STR    COUNT    .. LSB TO COUNTER 2
LDI    #12      .. MSB TO COUNTER 2
STR    COUNT    .. MSB TO COUNTER 2
DEC    COUNT    .. POINT AT COUNTER 1
LDI    #78      .. LSB TO COUNTER 1
STR    COUNT    .. LSB TO COUNTER 1
LDI    #56      .. MSB TO COUNTER 1
STR    COUNT    .. MSB TO COUNTER 1
DEC    COUNT    .. POINT AT COUNTER 0
LDI    #00      .. LSB TO COUNTER 0
STR    COUNT    .. LSB TO COUNTER 0
LDI    #01      .. MSB TO COUNTER 0
STR    COUNT    .. MSB TO COUNTER 0
    
```

a programmable, hardware-triggered one-shot. Pulse width is controlled by means of the count stored in Counter 2.

Read operations are performed similarly. It is suggested that the user refer to the manufacturer's data sheet for exact programming techniques.

Design Example: Programmable Pulse Generator

In this example, Fig. 4, the 8253 is configured to implement a programmable pulse generator. The COSMAC system clock is used as the counter time base. Counter 0 operates as a programmable squarewave rate generator and prescales the system clock according to the count loaded by the operator. Counter 1 is also operated in the programmable square-wave mode. By varying its count, control of period is achieved. Finally, Counter 2, connected to the output of Counter 1, is used as

Operation of the counters in the BCD mode is quite simple: prescale the clock to some convenient increment of time, one millisecond, for example, and program period and pulse width directly in milliseconds.

Additional Designs

One or more 8253 parts can add a new dimension to COSMAC-based systems. Because of their inherent flexibility, these devices can be tailored to the application at hand, and because of programmability, adapted at will to necessary changes. When used as memory-mapped I/O devices, no COSMAC I/O instructions need be sacrificed to support the 8253, and the number of devices that can be employed is limited only by available unused memory space and the imagination of the user.

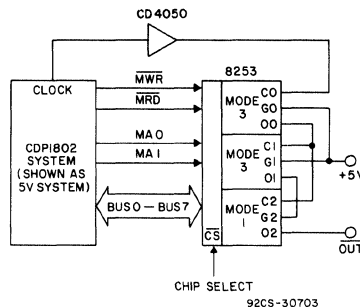


Fig. 4 - Programmable pulse-generator circuit.

Reference

1. 8253 data sheet available from Intel Corp., 3065 Bowers Ave., Santa Clara, California, 95051.

Optimizing Hardware/Software Trade-Offs In RCA CDP1802 Microprocessor Applications

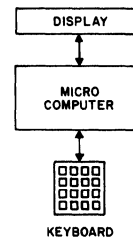
By L.A. Solomon and D. Block

One of the chief reasons for choosing to design with a microprocessor rather than standard IC's is to reduce a system's parts count. To make the best choice requires a careful analysis of hardware/software trade-offs. This analysis usually narrows down to the ratio of ROM to I/O devices in the system. Economics indicates that the more functions handled in software, the less expensive and more flexible the system will be. Thus, a good design practice is to attempt to do everything in software initially and then relegate functions to hardware only as the speed/processing capability of the CPU becomes taxed. This Note will develop some examples of processor interfaces that not only minimize external hardware but, through judicious programming techniques, also minimize speed requirements on the CPU.

The RCA CDP1802 microprocessor is particularly well suited to minimum-cost interfacing because it has a significant number of terminal connections dedicated to I/O operations and an extensive set of I/O instructions. It has three I/O selection lines, called the "N" lines, that are controlled by I/O instructions plus four general-purpose flag input lines testable with branch instructions. There are also DMA-in, DMA-out, and Interrupt Request line inputs as well as two state code and two timing pulse outputs to synchronize I/O devices to the CPU. A single bit output (Q) which can be set or reset under program control is also provided. In all, 15 of the CDP1802 terminal connections are dedicated exclusively to I/O control. In addition, the CDP1802 has other unique architectural features, such as built-in DMA, that can be used to advantage. These features will also be discussed.

A CLASSICAL SYSTEM

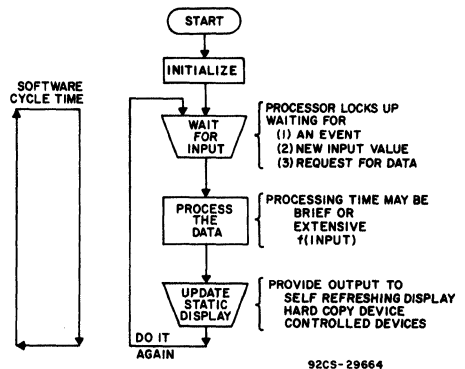
A simple system having a keyboard input and a digital display output is shown in Fig. 1. The specific functions are



92CS-29670

Fig. 1 - Simple microcomputer system.

omitted because the immediate concerns are only the microprocessor and I/O interfaces. These interfaces will be constrained by the programming technique chosen for the system. In the classical software control flowchart for this system, shown in Fig. 2, the standard



92CS-29664

Fig. 2 - Classical software control flowchart for the system of Fig. 1.

initialization block is followed by an input, processing, and output procedure with a final loop back to repeat the action. Even without the details of the hardware or software design or the specific application, certain predictions can be made about this system.

First, consider the software cycle time, that is the time to go completely through one loop of the procedure. The cycle time is the sum of the time spent in each portion of the software, including input, processing, and output. Because the program apparently waits for an input, the time spent in the input block is indeterminate. The system cycle time, therefore, is indeterminate. This parameter has immediate impact on the selection of both input and output devices used in the system. The output device, for instance, must be capable of operating for prolonged periods without processor attention. Therefore, it must be a device that is self-refreshing or contains a latch. It certainly cannot be dynamic because no provision for refreshing is apparent in the simple software structure shown thus far. Because dynamically refreshed displays have the potential for lower cost, the static requirement is a serious drawback.

Next, consider an input device. A keyboard, being human operated, will present data to the processor at an uneven rate. The time between keystrokes may vary from a few milliseconds to several seconds or minutes. With the flowchart given, the processor must complete its processing before the next input can be received. If each keystroke requires some analysis by the microprocessor, a choice between using a very fast (and expensive) processor or lengthening the minimum time between keystrokes must be made. The first alternative would be very wasteful since the processor's very fast speed would only be needed in short bursts; most of the time it would be idling waiting for an input. The second alternative leads to an unresponsive system, one in which the operator will have to adjust to the system rather than the other way around. A third alternative is to design in an "intelligent" keyboard controller or buffering device to smooth out the input rate as depicted in Fig. 3. This alternative, however, is not ideal either because it requires additional hardware expense.

Resorting to additional hardware, however, may not be necessary if the flowchart of Fig. 3 is restructured. By doing the controller functions in the

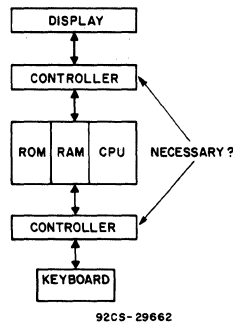
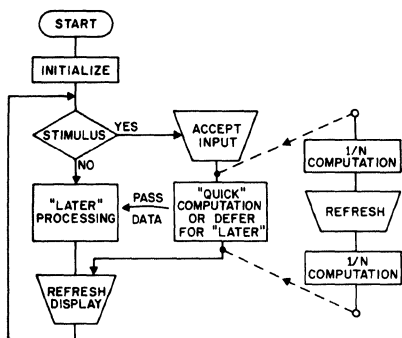


Fig. 3 - Addition of controllers to smooth out input rate.

software the controllers can be eliminated at only the cost of enlarging the system ROM. Moreover, because ROM's come in fixed increments, it may be no more expensive to have a program that is 1024 bytes long than one that is 527 bytes, even though one is nearly twice as long as the other. In fact, if there is unused space in the system ROM, the controller function may be had for "free". Even if an additional ROM is required, it may cost less than the MSI or LSI controller being replaced.

To take advantage of software control, the approach is changed, so that instead of waiting for an input to take place, the system simply looks at the input periodically. If no input is present, it skips the input operation and goes on to something else. That something else could be the refreshing of a dynamic display, for example, or some processing required as the result of the last input. If an input is present, then it is accepted and acted on. There are several options available for handling the processing associated with this input. If the input is small and can be handled immediately, the system will do so. If not, it can be saved for later when there will be time to handle it, or it can be broken up into small computational blocks interspersed among other tasks such as display refresh. These approaches are flowcharted in Fig. 4. The latter approach is the idea behind a powerful technique called interpretive programming in which functions such as display refresh and keyboard scan are written as modular subroutines. Calls to these subroutines, which pass or pick up parameters from the main program, can be interspersed throughout the main program wherever required by the system timing considerations.

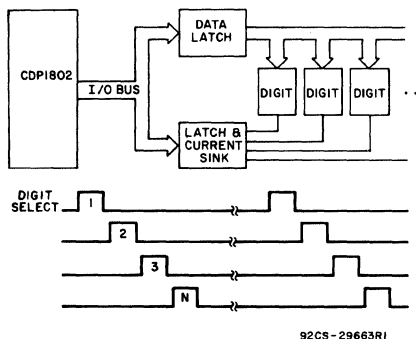


92CS-29661

Fig. 4 - Controller function transferred to software and input loads interspersed.

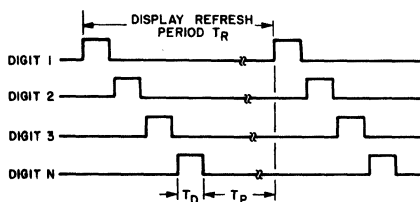
HANDLING A DYNAMIC DISPLAY

Fig. 5 shows a typical multiplexed display system and Fig. 6 gives the details on the display refresh rate. The minimum refresh rate for any digit should be 100 Hz, which is fast enough to prevent flicker under most stationary display conditions.



92CS-29663R1

Fig. 5 - Typical multiplexed display system.



- DISPLAY REFRESH RATE > 100 Hz
- MINIMIZING N · TD WILL MAXIMIZE TP
- TP IS THE AVAILABLE PROCESSING TIME

92CS-29668

Fig. 6 - Details of the display refresh rate.

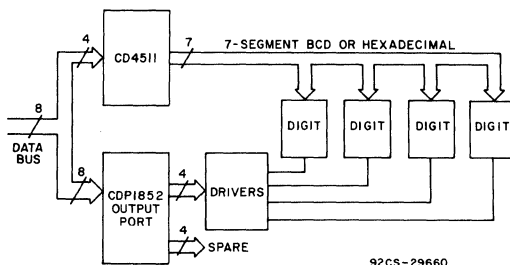
The actual ON time (T_D) of any digit is a trade-off between the intensity of the display and the time remaining within the 100-Hz refresh period for the processor to do some other work. It is desirable to minimize T_D so that a maximum of processing time (T_p) is left for the rest of the processing load.

It is customary to "overdrive" multiplexed LED displays to increase their apparent brightness. The extent to which overdrive is practical is a function of the duty cycle

$$\frac{T_D}{T_R}$$

of the display. This technique, however, is not without risk. Should the program crash or hang up (because of a program bug or noise injected into the system, or component failure, etc.), it is quite probable that a digit driver will be incinerated. Because of this hazard appropriate precautions, particularly when debugging a system, should be taken.

The segment information for a 7-segment display can be handled in either of two ways. If the data is in BCD, a device such as the CD4511 which contains a latch, BCD-to-7-segment decoder, and drivers can be used as shown in Fig. 7. Or,



92CS-29660

Fig. 7 - Handling segment information in hardware by means of a CD4511 BCD-to-7-segment latch decoder driver.

instead of the CD4511 that does code conversion in hardware, a software conversion via a look-up table can be used along with a simple output port as shown in Fig. 8. Hexadecimal or other codes are also easily accommodated in the table look-up method. But, because the output ports may not have sufficient drive to directly handle LED's, an intermediate stage of buffering may be necessary. No clear-cut recommendation can be made because variables such as the number of devices and the type of display chosen are significant.

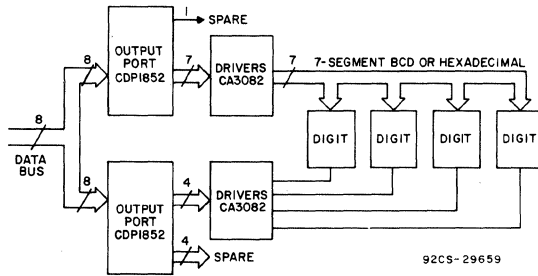


Fig. 8 - Handling segment information in software by means of an output port and lookup table.

SINGLE-SIGNAL INPUTS

The CDP1802 has four flag input lines that can be tested with branch instructions. These inputs are general purpose and can be used for such functions as interrupt vectoring, status indicators, or as single-bit inputs for slowly varying signals such as that of an ASCII terminal having a moderate baud rate. As an example, one of the flag lines is used as an input for a switch in Fig. 9. To signal

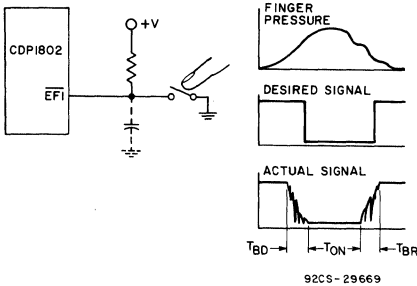


Fig. 9 - Basic switch circuit using microprocessor flag line.

the processor, a change on the flag line from a logic 1 to logic 0 level is used. However, the tendency of mechanical switches to "bounce" prevents this simplistic solution. The actual signal presented to the microprocessor consists of three parts - an initial bounce, a stable ON period, and a release bounce. A program looking only for a simple 1 to 0 to 1 transition may sense many switch closures because of the bounce noise. Although there are hardware solutions to this problem, software techniques may prove more cost-effective. Fig. 10 is a flowchart of a subroutine to debounce a mechanical switch. A test is made on the input signal to test for a switch closure. If none is found, a "switch down" software flag is

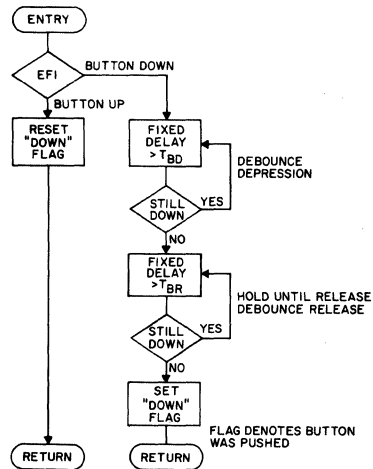


Fig. 10 - Flowchart of subroutine for debouncing a mechanical switch.

reset. This flag may be some convenient bit in one of the CDP1802's sixteen general purpose CPU registers or a bit in a RAM status word. If the switch is down, then the software will loop, waiting for the button to be released. The wait is performed to insure that the switch is not "seen" again for the current depression and to allow for the initial bounce period T_{BD} . Once the switch is released, the switch is again interrogated until it reaches a stable OFF condition. The software flag indicating a "switch down" condition is set, and the program returns to the caller. Although this program is easy to understand, it is, like the earlier simple solutions, not without its problems. For instance, the processor again wastes valuable time. The execution time (see Fig. 9) for this subroutine is at least

$$T_{BD} + T_{BR}$$

and does, in fact, last as long as the button is depressed. Thus, it is obviously not suitable for systems having dynamically refreshed displays. A further drawback, from the human-engineering standpoint, is that a response is made on the release of the switch rather than on its depression, the opposite of what one would normally expect.

Fig. 11 shows a flowchart for an im-

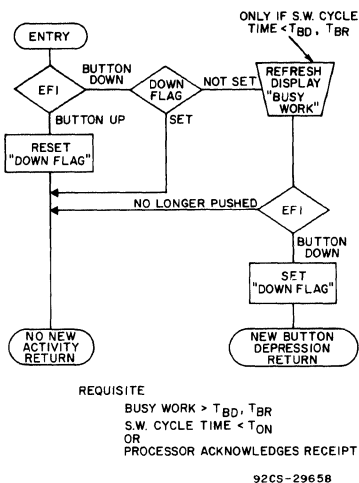


Fig. 11 - Flowchart of improved subroutine for debouncing a mechanical switch.

proved method that overcomes both of these drawbacks. Here, the subroutine that looks at the input signal has the capability of remembering what that signal was the last time it looked. This information is saved in a software flag called the "down flag". The routine operates as follows. If the button is now down and was also down the last time, then it is assumed that the system sees the same button depression seen earlier. A return is made to the caller with an indication of no new activity. If the button is not now down, but was the last time, then the switch has been released. In this case, the "down flag" is reset and a return made to the caller indicating no new activity (it is assumed that the processor is interested only in switch depressions and not their duration). But, if the switch is down now and was not down the last time, then there is a new depression. The switch must be debounced, the "down flag" set, and a message returned to the caller. Notice in the flowchart that a second test was made after the delay generated in the "busy work" block. This delayed second test is a debouncing technique to determine that

the switch has been in the same state for two successive samplings before a decision is made on the true state of the switch. This method is still not optimal because the program is waiting (and therefore wasting time) during the debounce period. If some additional constraints are placed on the software cycle time, however, the program can be further optimized. For example, if the cycle time is greater than the bounce time (T_{BD}) but less than the switch ON time (T_{ON}), then the flowchart can be simplified to Fig. 12. Here there are no timewasting loops because switch bounce, in effect, will not be seen within the given timing restraints.

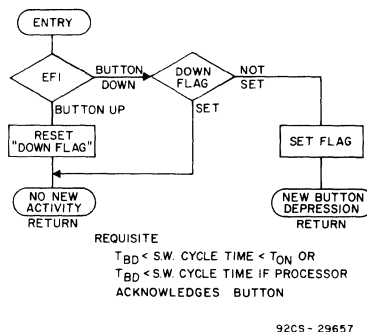


Fig. 12 - Flowchart of simplified debouncing subroutine benefitting from additional constraints.

MULTIPLE INPUTS

Up to four inputs can be handled as described above with each switch connected to a separate flag line of the CDP1802. Another technique is a multiplexing scheme in which the four switches are connected to one flag input, as shown in Fig. 13, and sequentially scanned as described in the flowchart of Fig. 14. This technique is readily expandable to additional scanned functions and, therefore, is discussed in detail. The

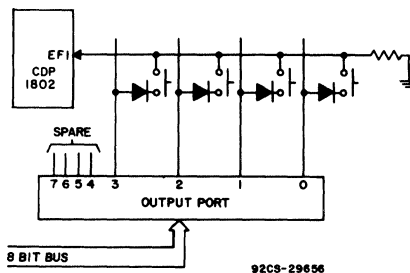


Fig. 13 - Hardware for handling four switch inputs on one flag line by means of a scanning routine.

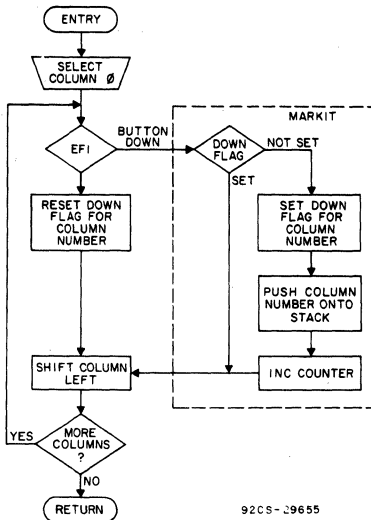


Fig. 14 - Flowchart of scanning routine for handling four switch inputs.

subroutine is designed to look for new switch closures and report them to the main program by "pushing" the switch number of a newly closed switch onto a stack and incrementing a counter. The main program will "pop" switch numbers off the stack and decrement the counter whenever the count is greater than zero. In the CDP1802 any one of the 16 general-purpose registers can be conveniently used as a counter because each has its own increment and decrement instruction.

The auxiliary functions for the subroutine are shown in Fig. 15. It is

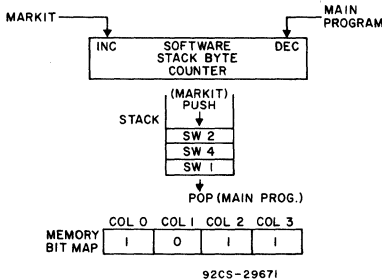


Fig. 15 - Auxiliary functions for the scanning subroutine of Fig. 14.

assumed that the timing constraints of Fig. 12 are met by this routine also, so that Fig. 14 is an extension of the basic flowchart previously developed. Upon entry into the subroutine, the first switch column is selected by outputting a 1 in bit position 0 of the data bus and examining the switch associated with that position. If

a new depression is detected, the "down flag" is set for that switch in the memory bit map, the column number is pushed onto the stack, and the counter incremented. Next, the column is shifted and, if more columns remain to be scanned, the process is repeated. No switch closure or no new switch closure simply results in a column shift and continuation. When all columns have been scanned, a return to the main program is executed. The main program detects if any new switch closures have occurred by seeing if the counter has a value greater than zero. If so, the main program successively "pops" a switch number from the stack and decrements the counter until it reaches zero.

A section of the flowchart in Fig. 14 has been partitioned off and labeled "MARKIT". This routine is a common one that can be used as an expanded keyboard scan routine discussed in the next section. It should be noted that the approach taken above lends itself well to a multi-processor system in which one processor handles the keyboard scanning and puts key numbers in a stack accessible to the other processors as well.

KEYBOARD SCANNING TECHNIQUES

Fig. 16 shows an arrangement for

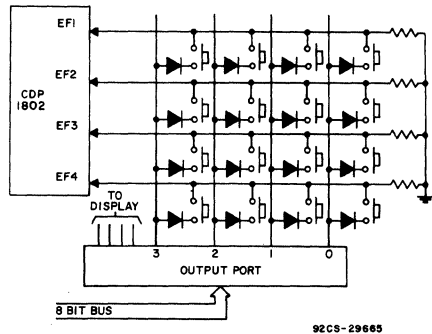


Fig. 16 - Hardware arrangement for handling a 16-key matrix using a scanning routine.

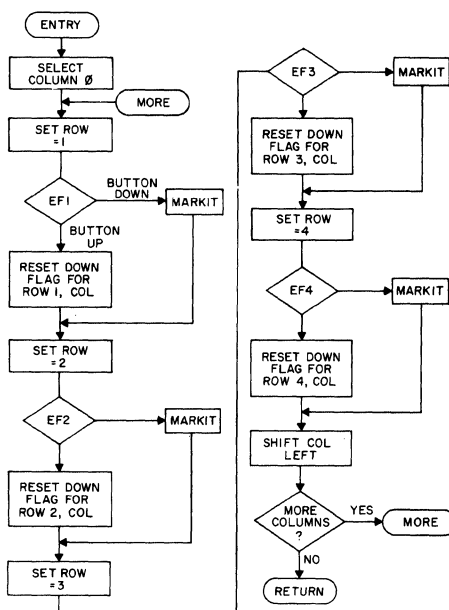
scanning a 16-key matrix. It is a simple extension of the arrangement just discussed. The horizontal lines can go directly into the four flag inputs of the CDP1802 as shown. Fig. 17 gives a flowchart of the software in which "MARKIT" is now responsible for handling row as well as column information. The basic interface between

the main program and the keyboard scan subroutine remains the same; the subroutine place new key depressions on the stack and from there they are passed to the main program. Note that a key number's position on the stack does not necessarily represent when a given key was depressed with respect to the other keys on the stack, but merely indicates the order in which the keys were scanned. Because the stack is emptied on each cycle by the main program and only new key depressions are entered, the presence of two key numbers on the stack tells only that both keys were down when the scan took place. To discriminate in time

between rapid key depressions, a short software cycle time is necessary. But, remember that this time must be kept within the constraints of TBR, TBD, and TON. There is a limitation to the technique discussed in that the software does not indicate to the main program when a key has been released. Thus, it can not be used in a system requiring lockout of other keys when any one key is down.

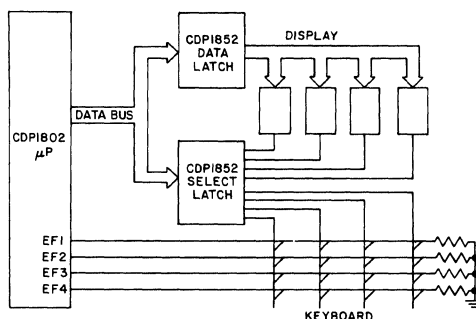
COMBINED DISPLAY AND KEYBOARD

The whole system of Fig. 1 is shown with its component blocks filled in on Fig. 18. The original objective to minimize



92CS-29666

Fig. 17 - Flow chart of software for handling row and column information utilizing "MARKIT" routine.



92CS-29654

Fig. 18 - Simple microcomputer system of Fig. 1 with component blocks expanded.

hardware has been realized in that only two 8-bit output ports are required in this design besides digit drivers (not shown).

A further improvement can be made in the system by combining the keyboard scan and display multiplexing signals as shown in Fig. 19. Here, a single-byte

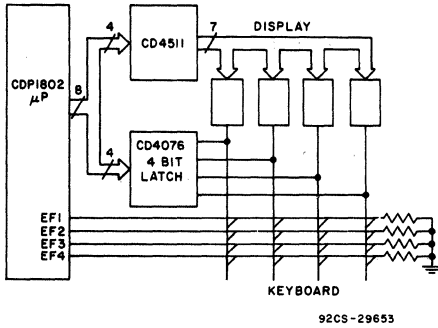


Fig. 19 - System improvement made by combining keyboard scan and display multiplexing signals.

output is used with the upper-order 4-bits being BCD data for the display and the lower-order 4 bits used to simultaneously select a display digit and keyboard column. This arrangement does not reduce the parts count, but does give smaller packages if space is a consideration and cuts down on the number of output operations and output bytes stored. A ready expansion of the system shown in Fig. 20 still uses only two IC's but permits scanning two 16-key keyboards and an 8-digit display.

TIMING GENERATION

In many applications it may be necessary to have some time-keeping ability in the microprocessor system. The requirements may range from having a time-of-day or elapsed-time clock to microsecond timing resolution for generating precision pulse widths. Here again, of the many approaches possible to timekeeping, a cost/performance-optimized one can be found.

Consider an example, shown in Fig. 21, for generating an output pulse of width T_1 each time switch S_1 is closed. The CD-

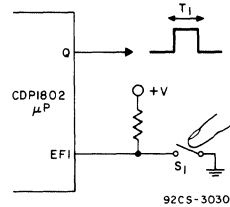


Fig. 21 - System for generating an output pulse for each switch closure.

P1802 has a single-bit output called the Q flip-flop that can be set or reset under program control to perform this function. The simplest technique for generating a fixed delay is by executing a series of "no-ops" in the program as illustrated in Fig. 22. If each "no-op" takes 5 microseconds to execute, for example, and T_1 is 50 microseconds long, then ten "no-ops" would do the job. This technique is obviously not a realistic one for long timing intervals because it is extremely wasteful of memory and fully occupies the processor with a non-productive task.

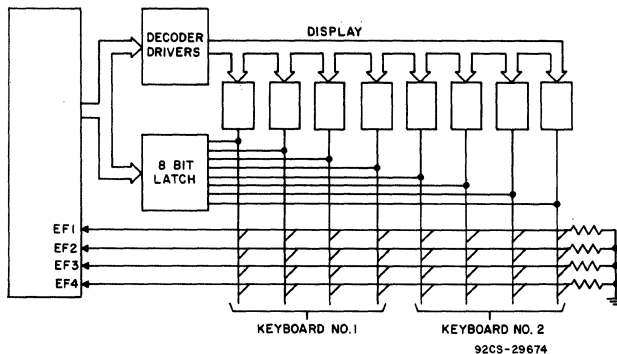


Fig. 20 - An expanded system with 8-digit display and two 16-key keyboards.

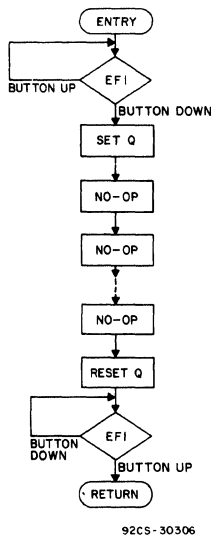


Fig. 22 - Primitive programming technique for pulse generation.

A better technique is shown in Fig. 23(a). Here a counter is preset with a given value and continually decremented until it reaches zero. In the CDP1802 one of the 16-bit general-purpose registers can be used for this function. Fig. 23(b) shows the specific instruction sequence. This technique saves a lot of memory bytes but still ties up the processor. For maximum processor efficiency it would be best to load an external counter that would count at some preset rate and generate an interrupt when it reaches zero. Meanwhile, the processor could be doing some useful work. Such a system is shown in Fig. 24, but it does not minimize system hardware.

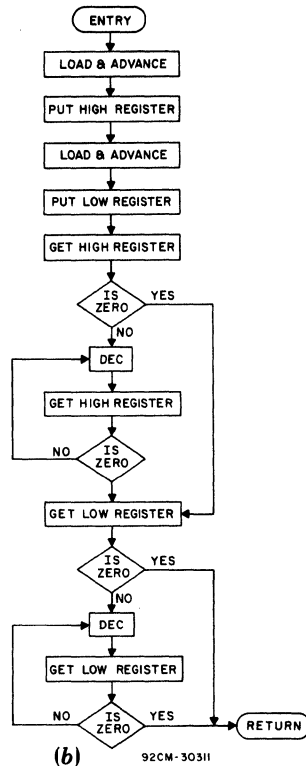
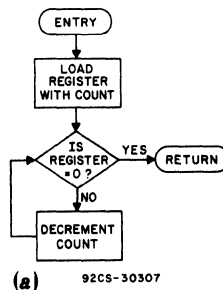


Fig. 23 - (a) Basic flowchart of improved technique for pulse generation. (b) Specific instruction sequence for improved pulse-generation technique.

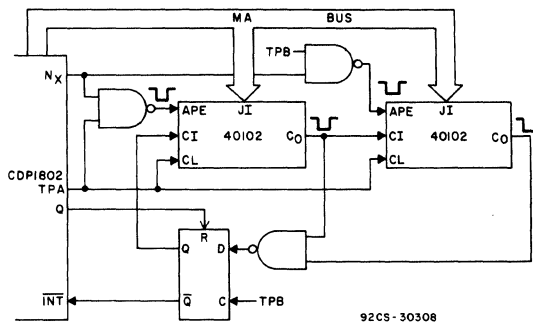


Fig. 24 - System for pulse generation using an external counter.

An interval timer can readily be made for the CDP1802 with no external parts by making use of an internal register that can be automatically incremented. General-purpose register R0 is used as a pointer for DMA operations in the CDP1802 and, as such, is automatically incremented on each DMA (in or out) cycle. By connecting the State Code 1 (SC1) line output back to the DMA Out request line (DMAO), the CDP1802 performs one DMA cycle for each Fetch and Execute cycle, as shown in Fig. 25, thereby providing a built-in timer and instruction counter. With a clock frequency of 1.57 MHz, the most significant bit of R0 will change each 1/2 second, providing a real-time clock. CPU operation is, of course, slowed by 1/3 with this scheme, but with an upper clock frequency of 6.4 MHz the

system can be made fast enough for many applications.

With the circuitry of Fig. 26, a general-purpose interval timer can be realized. This circuit will cause an interrupt when the most significant bit of register R0 goes to a "one". Thus by preloading R0 with a desired count, a timing interval with a range of 2^{15} and a resolution of up to 3.75 microseconds (with a clock frequency of 6.4 MHz) can be obtained with a minimum of external hardware and yet leave the processor free to do useful work.

ACKNOWLEDGMENT

The authors wish to acknowledge the contributions of Messrs. K. Karstad and F. Thorley for developing some of the techniques described.

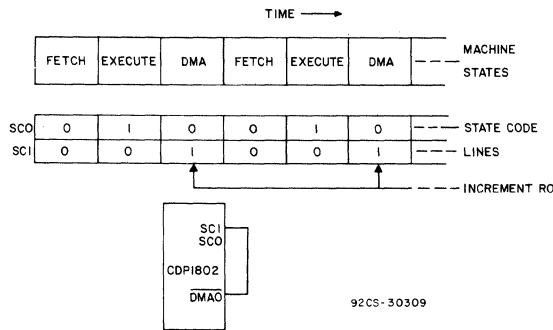


Fig. 25 - Use of DMA-Out line to implement an internal timer.

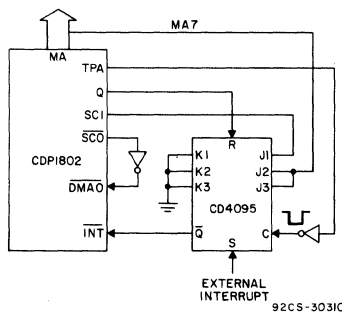


Fig. 26 - Use of Interrupt line to implement a general-purpose timer.

Microprocessor Control for Color-TV Receivers

K. Karstad

A microprocessor control that makes available sophisticated and desirable features not feasible with current standard or custom IC's is described. The TV controller features modular design, which permits essential functions to be implemented on a priority basis while events of secondary importance are held until processing time is available. Physical modularity also permits features to be added in the form of software changes, thus saving manufacturers the expense in time and money of hardware redesign.

The tuning function is implemented by means of a microprocessor controlled phase-locked loop, which constitutes a frequency synthesizer. Scanning and preprogramming of desired channels is possible; for example, a real-time clock allows event programming. The set can be programmed to turn itself on or off or switch stations in a preplanned sequence. User interaction is accomplished through a local keyboard on the set or a remote 35-key unit that transmits pulse-position-modulated infrared pulses. A six-digit LED display provides information and feedback. In this all-CMOS design, a two-level battery back-up system guarantees that stored channels and events will be kept intact for weeks in the event of an ac power failure. An interrupt driven system, with the interrupt load time multiplexed, is combined with a DMA cycle-stealing feature to overcome the difficulties in software structuring.

INTRODUCTION

Until recently the number of user-controllable features on a TV receiver was limited, both for technical and economic reasons, to mechanical channel selection and sound and picture control. The advent of the varactor diode made electronic tuning possible by taking advantage of the voltage-variable capacitance characteristic of the diode. The availability of IC's, in most cases standard MSI circuits or custom MSI/LSI circuits tailored for specific functions, has made feasible the electronic control of channel

selection from the keyboard, either remote or part of the set, with the channel number shown either on or off-screen. This type of IC control represents the present trend in the TV industry; however, the availability of the microprocessor makes possible much more sophisticated control of a much wider range of features.

This Note describes a microprocessor control for a color-TV receiver, a control that supports a large number of features and options.¹ As there is no clear industry trend toward the application of sophisticated electronics to control functions, it can be assumed that many of the features discussed will not find their way into the standard home receiver for some time to come. Nevertheless, this design, based on the CDP1802 microprocessor chip,² illustrates the degree to which the stored program concept can be used to implement receiver control.

The microprocessor control represents a modular approach to TV-control design both in its ability to process functions on a priority basis and in its ability to accept additional or modified functions. Depending on model or market, and without the need for the long-range custom-LSI development phase required in a hardware redesign, manufacturers can quickly implement new features through software changes or additions alone.

GENERAL FUNCTIONAL OPERATIONS

Tuning Control and Preprogramming

The tuning function in this all-electronic system, Fig. 1, is implemented by means of frequency synthesis. A phase-locked loop, PLL, selects the correct local oscillator, L.O., frequency for a specific channel and keeps the receiver tuned, irrespective of changes in the tuner with time and temperature. When any number from 1 to 99 is keyed into the keyboard, the signals are translated by means of data in a ROM look-up table into the 14-bit code, dictated by international channel allocations, required to set the PLL on the constant L.O. frequency for the channel desired.

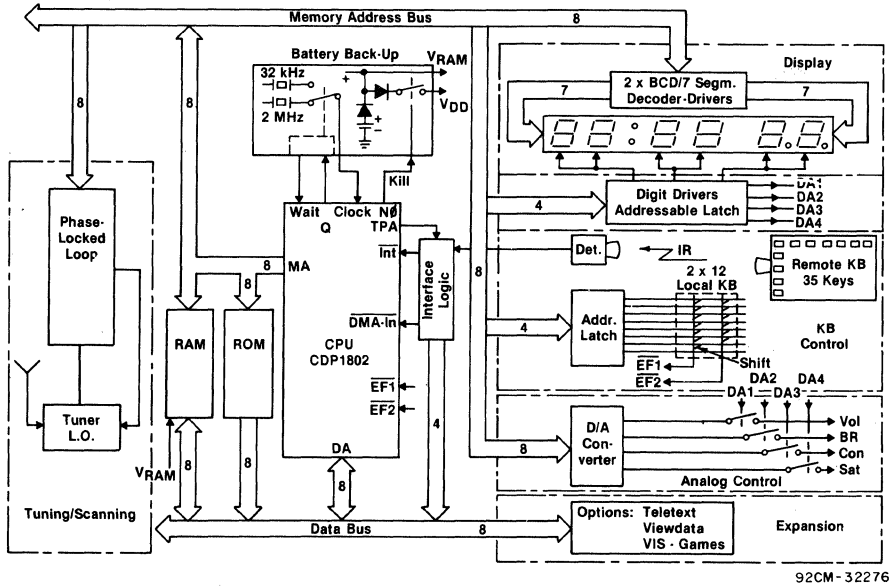


Fig. 1 — The microprocessor-control system. The user interacts with the controller by means of either the local 2 x 12 keyset or the 35-key remote unit. Clock and program information is shown on the six-digit display. Exact tuning is accomplished through the phase-locked loop. The battery back-up system keeps the clock running and stored information intact in the event of a power failure.

92CM-32276

The PLL design also permits direct fine tuning over a whole channel. While the traditional need for fine tuning is eliminated by the exactness of the frequency synthesis approach, some users desire it in fringe areas for improved picture quality. There is also, in some areas, a need to tune in the nonallocated channels used for cable TV and master antenna systems; the fine tuning capability makes this possible. In the VHF band, fine tuning is implemented in 25-kHz steps, and in the UHF band, in 100-kHz steps.

An important feature of the microprocessor tuner is channel preprogramming. In the following discussion, the distinction made between programs and channels is that commonly practiced in Europe, where a given program may be assigned to two different channels in two different geographical areas in much the same way as different networks are assigned different channels in the U.S. The tuner can store 16 programs, which can then be called out simply by depressing a key number between 1 and 16.

This preprogramming technique requires that the channel to which a given program is assigned be known. For the more usual case, where the channel is not known, a scanning mode is available. Depression of the scan key will initiate a search starting at the channel to which the set is currently tuned and terminating at the next higher channel available on the air. If the scan button is not depressed

again, that channel is automatically assigned to the stored-program location currently chosen. With each depression of the scan button, the tuning system searches for the nearest higher available carrier. If no station is found, the scan wraps around and stops on the frequency from which the search began.

Display and Keyboard Control

Channel and program numbers are shown on a six-digit, LED, off-screen display. (Display and keyboards are shown in the Appendix.) For a period of time the on-screen display was popular, but the implementation of the off-screen display is simpler; it can also be on permanently, and does not distract the viewer's attention. The optimum number of digits permitting continuous display of clock and program number, the normal or prioritized display mode, is six. Upon request from the keyboard, both channel and program numbers are shown, the channel number displacing the clock display. In the scanning mode, the channel frequencies assigned to program numbers are automatically displayed.

The availability of a real-time clock permits event programming; i.e., a program can be automatically turned on or off at a specific time. Again, the six-digit display is sufficient for the easy programming of events as well as the listing of events already programmed.

The local keyboard, with 12 keys and shift-key, can handle 24 tasks, including program select scan, analog control (up/down), mute, on/off, and clock set. The

master keyboard is remote, has 35 keys, and communicates with the receiver over an infrared link. Each depressed key generates a unique seven-bit code as a pulse-position-modulated (PPM) pulse train;³ zero and one bits are defined by specific distances between pulses. This pulse train, an asynchronous event with respect to any other task, is processed by the DMA (direct memory access) channel. Hence, a random real-time event is given priority.

The analog functions, volume, brightness, contrast, and saturation, are changed continuously up or down by holding down the appropriate key. A six-bit word in memory, providing 64-step resolution, is output at a predetermined rate and incremented or decremented by one each time. With sample-and-hold multiplexing techniques, one output port is sufficient for serving four different channels.

Event Programming

The six-digit display and 24-hour clock make it possible to preprogram and monitor events for a period beginning at the present and extending 24 hours into the future. An arbitrary number of memory locations, eight in this design, are reserved for event programs. The user, while watching the display, simply keys in the time for the set to turn on (or to switch channels if already on) and the program number. This procedure can be repeated until the list is full, which is indicated by flashing 9's on the display. There is no restriction on the sequence of entering events. The software orders the list in time sequence so that the event nearest in time is always at the top of the list. The software checks every minute to determine whether an event is to be activated; if it is, the event is executed and then erased from the list, and the next event in time is moved up. A lit decimal point on the display indicates that an event is pending. When the event has been executed, the decimal point begins flashing and continues to flash until the viewer acknowledges by depressing any key. If no acknowledgment is received within five minutes, the set turns itself off.

The sequence above describes an event that occurs only once; it is a simple matter to program an event so that it is automatically repeated every 24 hours, for example, an event that turns a program off at a specified time each day. The programmed information (time, program number), along with an indication of whether it is a one-time-only, off, or repeat event is listed on the display by use of a "list" button on the keyboard. An empty or exhausted list is indicated by flashing eights.

Expansion

The modular structure of the microprocessor tuner makes it possible to add options by adding ports to the data

bus and segments of code to the software. Two options of considerable future interest are Teletext⁴ and Viewdata⁵. If the receiver in question is equipped with the Teletext option, for example, the keyboard will contain a key marked Teletext. When this key is depressed, the software will check for the existence of this option. If it is present, certain keys will from that moment be redefined and, when depressed, will lead to the execution of tasks different from those of the normal TV mode. The latter mode is recalled by use of another key, NTV.⁶

SYSTEM SOFTWARE

The requirements on the software portion of this real-time controller are to multiplex the six display digits, multiplex four analog channels, update the clock, monitor the two keyboards, and execute the tasks called for. The clock must, of course, be updated regularly and frequently so that it does not lose time. The display must be refreshed at a rate high enough to avoid flicker, i.e., at 50 to 60 Hz at least. The analog values must also be refreshed steadily at a rate high enough to maintain a dc signal and without excessively large filter constants. Certainly, the keyboards must be checked frequently enough to catch any random key depressions. Finally, whatever command is received, fine tuning, scan, mute, etc., must be processed and executed. Note in this context that some of the tasks, such as volume up/down, require continuous depression of a key for an indeterminate amount of time.

These overlapping and partially conflicting specifications are met by an interrupt-driven system together with the DMA channel used for the remote keyboard.⁷ A pulse train, derived from the CPU clock, interrupts the software program every four milliseconds, as shown in Fig. 2. The interrupt routine, which must be executed with regular frequency, is divided into three sections to assure distribution of the load. Each time the interrupt service routine is called, a test is made to determine which cycle is to be serviced next. Hence, three cycles comprising a complete frame are sequentially serviced. After 12 milliseconds, a full interrupt service frame repeats.

The six digits of the display are grouped in pairs, two per cycle. This arrangement reduces the multiplex rate to 1:3 and provides twice as much time between interrupts as would otherwise be the case. The refresh rate for the display is now 83 Hz, well above the critical flicker rate.

The sequence of events in the interrupt routine is as follows: An interrupt occurs. The last digit pair is turned off and the next pair is turned on. The four analog channels are then multiplexed and refreshed and the clock is updated. Con-

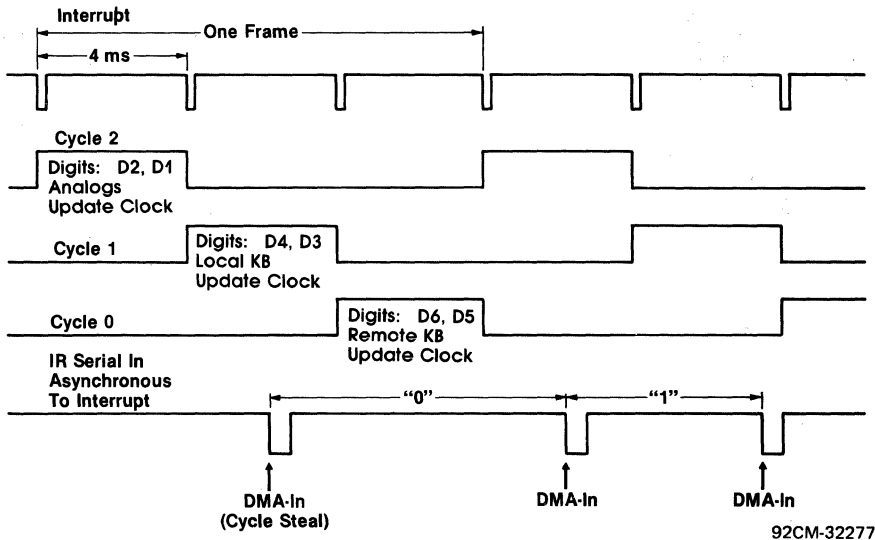


Fig. 2 — The interrupt system. The system is interrupt driven with the interrupt service routine split and time multiplexed. Remote infrared control pulses are received in real time through direct memory access without interfering with ongoing processing.

92CM-32277

control then returns to the main program. At the next interrupt the last digit pair is turned off and the next pair is turned on. The clock is updated again, but in this cycle, the local keyboard is also scanned for key closure. At the third interrupt, the last digit pair is handled, and the service routine checks for reception of a command from the remote keyboard. If a character has been received, the command is processed and executed. Once more the clock is updated and the main program resumes execution. All urgent tasks are thus served regularly and frequently: the clock is updated every four milliseconds, and every twelve milliseconds the display and the D/A channels are refreshed. In addition, both keyboards are monitored, and if a key closure has taken place, the required processing is performed for the main program to act upon later.

A key closure on the remote keyboard generates a bit string of pulses, and for some commands, a long character string. These pulse trains, which occur randomly and sometimes last for long periods, are received without tying up the processor and neglecting other real time events by having them fed into the DMA input. Thus the keyed information is conveyed to the CPU through a cycle-stealing process and, to all practical purposes, without slowing, or interfering with, the CPU's current operation.

The interrupt service routine in each four-millisecond time slot may vary in length according to which cycle is on, and particularly according to how much of the clock update routine is required at a

specific moment. Nevertheless, the processor is idle during the major part of the time slot, and is available for background processing in the main program.

When control returns to the main program in each time slot, the program determines whether a command was received from one of the keyboards. If it has been, that task, for example, change brightness, is executed. If no command has been received, the CPU is essentially idle until the next interrupt occurs. In most cases, the free time in one of the three cycles is ample time for the processing of any task in the main program; if it is not, the next interrupt simply postpones the background processing until one or a few more time slots are available. This manipulation is not apparent to the user.

The flowchart for the main program, Fig. 3, provides additional information on the points mentioned above. An interrupt can occur at any time in the main program; however, most of the time, if no keyboard command has been received, the program loops through the upper portion of the chart. A few other tests are also regularly performed: are there any events pending, was an event executed but not acknowledged, has ac power failed, or is no station on the air.

THE HARDWARE/SOFTWARE INTERFACE

Clock, Display and Keyboards

Four locations in the RAM are assigned to hold the four clock digits. As shown in the flowchart of Fig. 4, at every interrupt, a seconds counter is incremented. Later,

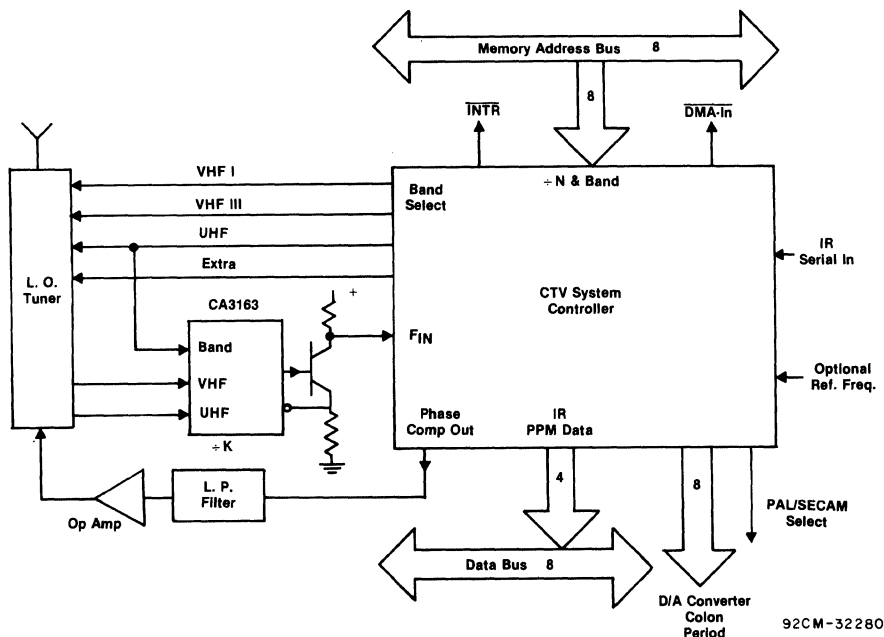


Fig. 7— Precise tuning and station selection is done with a phase-locked loop. The loop is controlled by an LSI chip that contains a divide-by-N counter, band select, phase comparator and reference frequency dividers. Band and channel frequency information is contained in a 16-bit data word sent over the memory address bus.

as unassigned channels used in cable TV and MATV installations. Additional fine tuning is implemented with the same resolution. The 16 bits of data are output over the eight-bit memory address bus with only one instruction, using register-based output. With the exception of the ECL prescaler, the low-pass filter, and operational amplifier, all circuitry is integrated on one CMOS chip. The required interface logic for the remote keyboard, discussed earlier, is also included on the chip.

The PLL system employed in the microprocessor tuner differs fundamentally from the open-loop analog system commonly employed. In the analog system, a predetermined voltage is applied to the tuner with the expectation that the generated frequency will be correct. But because there is no feedback, changes in components with age and temperature influence the frequency. Neither is automatic compensation achieved for the unavoidable differences that occur in tuner characteristics.

Note that with a PLL system, preprogramming of channels and events can be done at any time; no stations have to be on the air. In fact, it is now possible for the dealer to preprogram sets before they are delivered to customers, provided a battery back-up system is implemented.

Battery Back-Up

A two-level battery back-up system takes over in the event of a power failure.

Because of the all-CMOS circuitry of the logic and processor, this system can be implemented with only four rechargeable low-capacity NiCd cells.

In order to conserve power, the CPU, upon detecting a power failure, automatically switches from the 2-MHz crystal in normal use to a low-frequency 32-kHz crystal. The switching takes place during a few milliseconds "wait" state which the CPU also enters automatically with a power failure. If power is not restored within the predetermined number of days, seven, the CPU shuts itself down after shutting down the whole system with the exception of the RAM storing user-programmed information. The remaining battery power is sufficient for approximately three months of storage. If power returns before three months, operation resumes as normal with stored information intact.

The slow-clock mode works satisfactorily because, during the absence of ac power, the only task performed by the software at each interrupt is a clock update. Essentially the same update routine is used, except that, in the slow-clock mode, a branch instruction sets a new limit for the number of interrupt pulses required to measure one second. Again, the software ignores all code except update clock and testing for the presence of ac power.

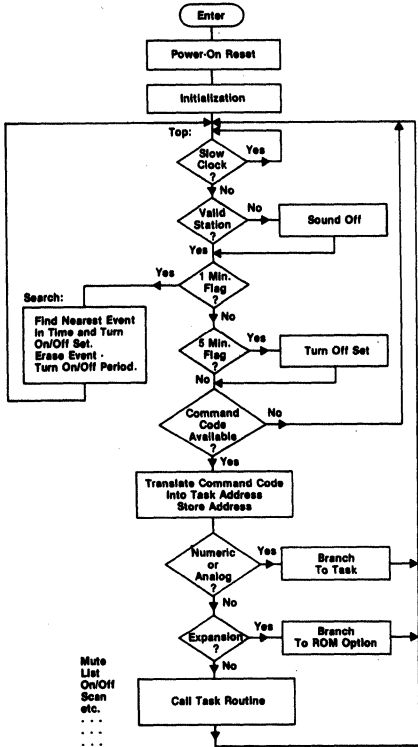


Fig. 3— The main-program flowchart. When a keyboard command is received, the program branches from the upper loop and executes the task called. The main program also checks every minute for a programmed event that calls for action; the user has five minutes to acknowledge an event execution.

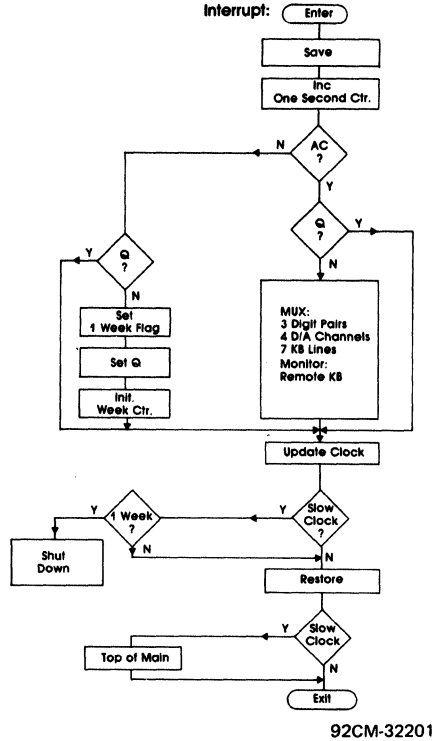


Fig. 4— The interrupt-routine flowchart. The interrupt service routine increments the seconds counter and updates the clock. It also multiplexes the display and the analog values and scans and monitors the keyboards. If ac power fails, the CPU switches to a low-frequency clock and battery power.

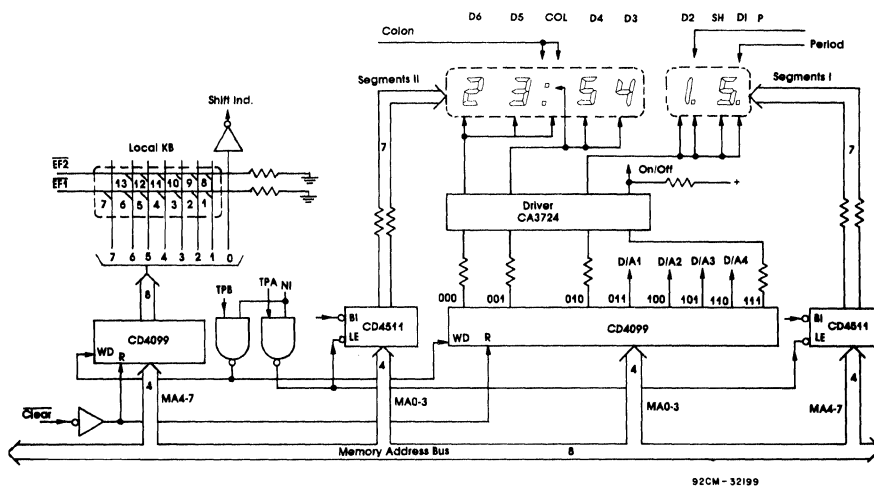
in the update routine, the counter is tested for the value of one second, and an update buffer is loaded.

Another buffer in RAM, the display buffer, contains BCD data for the six display digits. The update routine transfers data for the four clock digits from the update buffer. The display buffer also contains addresses for the digit pairs, the seven scanning lines for the local keyboard, and the four multiplexed D/A lines.

The configuration of the data in the display buffer is closely related to the hardware scheme shown in Fig. 5. Segment data for a digit pair are fed from two separate latch/decoder circuits. An eight-bit addressable latch selects the digit pair or the D/A channel to be sampled. Another addressable latch provides the scan address for testing key closures on the local keyboard. The required 16 bits of data are assembled in one 16-bit CPU register and output over the memory address bus with just one instruction.¹

Four RAM locations store the six-bit D/A data, which is output before the D/A channel is sampled. The latched data is smoothed by a simple R/2R network; no greater precision is required.

The software section of the interrupt routine, which interacts with the hardware of Fig. 5, is detailed in the flowchart of Fig. 6. The program turns off the last select line for a digit pair and tests whether it was cycle 0. If the answer is yes, a new frame of events is about to be repeated and the display pointer is reset to the top of the display buffer. Segment data for the next digit pair are output, and the pair is selected. In this example, the answer to the test of cycle 2 is yes; therefore, the four D/A channels are sampled in a burst mode, after which the routine exits to its update portion. At the next interrupt, since it was not cycle 0, the program turns on the next digit pair, finds itself in cycle 1, and activates the local keyboard routine. Finally, at the third interrupt (cycle 0), the remote keyboard routine is sequenced.



Output Register Data Format

Segments I Data	Segments II Data	Keyboard Scan Data	Address Scan Address	Digit Pairs Data	D/A Channels
BIT: 15 14 15 12	11 10 9 8	7	6 5 4	3	2 1 10

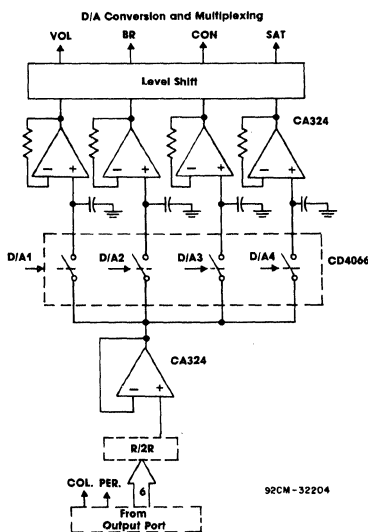
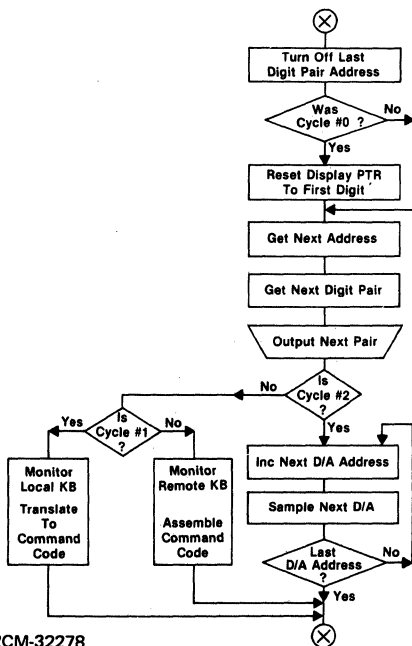


Fig. 5— The multiplexing of the six-digit display and the four analog channels, and the scanning of the local keyboard, is accomplished with a compact 16-bit data format. The eight-bit microprocessor outputs the full data word over the memory address bus with only one instruction.



92CM-32278

Fig. 6— This detail of the interrupt service routine shows how time-division multiplexing is accomplished in three cycles in order to distribute the real-time load.

A key depression on the local keyboard is translated into a command code and stored in memory for the main program to process. The sequence of events is as follows: One of the output lines from the addressable latch is activated; the CPU tests the first flag-line for key closure (Fig. 5). If the output line is line 4 and EF1 is tested (defined as row 0), a possible key closure is identified and marked as key number $4 + 0 = 4$. The CPU next tests line 4 and flag EF2 (defined as row 7). A key closure here is identified as key number $4 + 7 = 11$. The key number is used as an address pointer to a ROM table that contains the actual command code for that particular key (for example, mute). Because the local keyboard has only 12 keys to handle 24 functions, a shift-key depression must also be verified. If the shift key is down, 12 is added to the table address, causing a jump to the second part of the table containing the shift commands. The keyboard scan repeats every 12 milliseconds. If closure is detected, a debounce routine allows 36 milliseconds for verification of a valid key depression.

The received and detected IR pulse train from the remote unit is a PPM pulse stream composed of six bits plus a parity bit for each character. The software monitors the presence of a bit stream, measures the time between pulses, thereby discriminating between ones and

zeros, and assembles the information into a character that represents the actual command code. Finally, the program tags the received command code as to what type it is, a single command (on, off) or a repeat command (volume up), for the duration of the key depression. Ones or zeros are determined by measuring the time between pulses.

A four-bit counter is clocked by a 10-kHz signal. The content of the counter is read at the beginning of each IR pulse, and the counter is restarted. Hence, the number of pulses read represents the time interval between the last two pulses. As the IR pulse itself activates the DMA-in line, the time count is automatically read into a memory location. The DMA pointer advances in readiness for another input byte, which occurs at the next IR pulse. Hence, the seven bits in a command, represented by seven counts, are automatically stored in memory. If the time base is chosen well, it is possible, while maintaining good resolution, to get a count in which the most significant bit reads directly and correctly 1 or 0 for the time interval. It remains only for the CPU to repack the 1's and 0's into a command code word, which is stored for later use. The software code used for servicing the remote unit is in many ways similar to a UART program; it checks for correct number of bits and framing error, and ignores noise pulses and other error conditions.

Tuning and Program selection

A prescaler in the closed loop circuit, Fig. 7, divides the outputs from the L.O. by 64 for VHF and by 256 for UHF; the dividing ratio is defined as K. The output of the prescaler, now within the frequency range of CMOS circuitry, feeds a 14-bit divide-by-N counter, the output of which is compared with a reference frequency. The output from the phase detector is filtered, amplified, and applied to the varactor diode used to tune the L.O. In sum, the L.O. frequency of the tuner is measured, and whatever control voltage is necessary to force the frequency to be correct for a selected channel is generated by the loop. At phase lock, when the inputs to the phase detector are of the same frequency, the L.O. frequency of the tuner is N times K times the reference frequency. Therefore, the divide-by-N counter, which is under software control by the user, selects the channel.

The data format is a 16-bit word; the two most significant bits are decoded to select one of four bands (VHF-I, VHF-III, VHF, extra). The remaining 14 bits are the binary representation of the number N for a specific channel. The 14 bits provide sufficient resolution to place the L.O. frequencies as little as 25 kHz apart for VHF. Thus, it is possible to use the tuner to select any channel in the world, as well

Acknowledgement

The author gratefully acknowledges the contributions of G. Fogarty. The bulk of the material contained in this Note was

published in IEEE Transactions on Consumer Electronics, May 1980 (Vol. CE-26, No. 2, 0098-3063/80/0149-0156\$00.75 © 1980 IEEE).

REFERENCES

1. The few microprocessor-controlled receivers currently on the market are described in: Kleinman, A., "Programmable Color TV," Radio-Electronics (May, 1977).
Baum Wolfgang, "Farbfernsehgerat mit Microprozessor - Steuerung," Funkschau, Heft 17 (1977).
2. The microprocessor used in this design is the CDP1802. If the CDP1804 is substituted for the CDP1802, two kilobytes of ROM and 64 bytes of RAM will be available on the CPU chip.
3. Valvo: Data Sheet SAB 3011.
4. "Teletext - The LSI Solution," Mullard Technical Information TP1606.
5. "Viewdata," Wireless World (Feb. 1977).
6. A video interface system (VIS), which also is Teletext compatible, is another option. This system, built around two LSI CMOS chips (CDP1869 and CDP1870) offers a variety of formats for displaying and modifying data under software control with either NTSC or PAL compatible output signals.
7. User Manual for the CDP1802 Microprocessor, MPM-201B and COSMAC Microprocessor Product Guide, MPM-180B, RCA Solid State Division.
8. "Register-Based Output Function for the RCA COSMAC Microprocessors," RCA Solid State Application Note ICAN-6562.

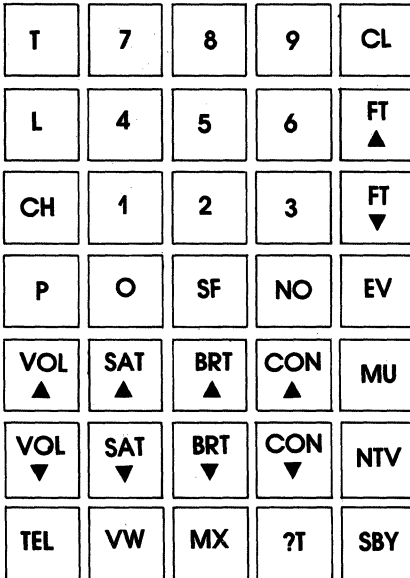
APPENDIX

Summary of single and compound keyboard commands. (t and p refer to numerals 0 through 9). Keyboard diagrams are shown on the following page.

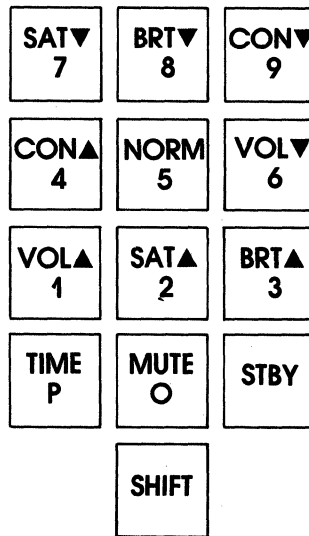
Commands	Actions
ANY KEY	Acknowledge automatic turn-on
BRT	Brightness adjust
CH	Channel scan
ppCH	Preprogram channels
CL	Clear. Returns program to previous mode. Instant escape from error mode. Return from listing, scan.
COL	Color adjust
CON	Contrast adjust
EVppPttttT	Preprogram a one-time event
EV00PttttT	Preprogram off
FT	Fine tuning
L	List preprogrammed events
L NO	Cancel preprogrammed event while listing
MU, MUTE	Mute
MX	Mix, expansion command
NO NTV	Four factory analog values are called
NTV	Return to normal TV mode. Four user's analog values are also restored
P	Program scan
ppP	Program selection. Turn set on with pp = 1-16

SAT	Saturation adjust
SF CH	What channel is currently assigned?
SF EVppPtttT	Preprogram a repetitive event
SF NTV	Store user's analog values
SF P	Toggle between PAL or SECAM decoder
STB, STBY	Standby: toggles set on/off
T	Toggle clock display on/off
ttttT	Set clock
TEL	Tele, expansion command
?T	? Time, expansion command
VW	View, expansion command
VOL	Volume adjust

Remote KB

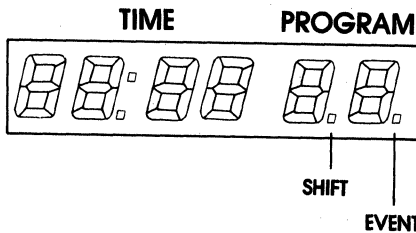


Local KB



92CM-32202

Display



16-Bit Operations in the CDP1802 Microprocessor

by D. Block

Although the CDP1802 microprocessor is an 8-bit machine, it contains mostly 16-bit registers. Its sixteen 16-bit registers are all general-purpose types, giving the CDP1802 a great deal of flexibility and the flavor of a 16-bit microprocessor in many respects. This paper describes various software routines and a few interface circuits that can be used to manipulate full 16-bit values in the CDP1802.

The areas of logical, shift, and arithmetic operations are all considered along with I/O and loop counters. In sophisticated systems where multiplications and/or divisions are required, a hardware approach using the CDP1855 Multiply/Divide unit should be investigated. These 8-bit units can be cascaded, so that up to 32-bit operations can be performed. For simpler systems, the software approach described below is probably the best choice.

General

In many of the following examples, a 16-bit data word will be located in memory. Since the CDP1802 is an 8-bit machine, of necessity the word will have to be stored as two 8-bit bytes. As a convention, assume that the 16-bit word is located in two consecutive locations with the most significant byte occupying the higher address.

Logic Operations

The logic operations provided by the CDP1802 are AND, OR, and EXCLUSIVE OR. These three operations are normally performed between an operand in the D register and a byte from memory. With the addition of a few instructions, register-to-register operations can also be accomplished, as shown in Example 2, below.

Example 1: Register/Memory Operations

In this example, a 16-bit OR is performed between the contents of an R

register called REG1 and two bytes of memory pointed to by another register called POINTR. The results will be in REG1.

```
SEX POINTR    ..POINT X TO FIRST
              (LOWEST)
              MEMORY BYTE

GLO REG1
OR
PLO REG1     ..OR LOW BYTES

INC POINTR   ..POINT TO
              ..HIGHEST
              MEMORY BYTE

GHI REG1
OR
PHI REG1     ..OR HI BYTES
              ..RESTORE
              RESULTS
```

Example 2: Register-to-Register Operations

If the two operands are in two registers called REG1 and REG2, the code below can be used; it is only three bytes longer than that of example 1. Here, again, the results will be returned to REG1.

```
SEX POINTR    ..SET A POINTER
              TO A FREE BYTE

GLO REG1
STR POINTR   ..STORE LOW BYTE
              OF REG1

GLO REG2
OR
PLO REG1     ..OR LOW BYTES
              ..RESULTS
              RETURNED TO
              REG1

GHI REG1
STR POINTR   ..STORE HI BYTE
              OF REG1

GHI REG2
OR
PHI REG1     ..OR HI BYTES
              ..RESULTS TO REG1
```

Shift Operations

In the CDP1802, shifts are performed on the contents of the D-register in either circular shifts, with data bits circulating through DF, or open shifts, in which zeros are shifted into one end of the D-register.

Either type of operation can be performed on the full 16 bits of an R-register, as shown below. Right shifts are performed by substituting the appropriate shift-right commands for the shift-left commands given in the examples.

Example 3: Shift Left (Open)

```
GLO REG1  ..GET LOW BYTE
SHL       ..SHIFT LEFT
PLO REG1  ..RESTORE
GHI REG1  ..GET HI BYTE
SHLC     ..SHIFT LEFT WITH
          CARRY BRINGS IN
          LSB
PHI REG1  ..RESTORE
```

Example 4: Shift Left Circular

```
GLO REG1
SHL       ..SET UP CARRY
          INTO HI BYTE

GHI REG1
SHLC     ..SHIFT IN CARRY
PHI REG1 ..CARRY INTO LOW
          BYTE LEFT IN DF

GLO REG1
SHLC     ..SHIFT CARRY
          INTO LOW BYTE

PLO REG1
```

A 16-bit shift of two consecutive memory bytes is performed similarly by replacing the GET and PUT statement by memory reference commands as follows:

Example 5: Shift Left-Memory Location

```
LDN REG1  ..REG POINTS TO
          LOW BYTE

SHL
STR REG1
INC REG1
LDN REG1
SHLC
STR REG1
```

Example 6: Shift Left Circular — Memory Locations

```
LDA REG1
SHL
LDN REG1
SHLC
STR REG1
DEC REG1
LDN REG1
SHLC
STR REG1
```

Arithmetic Operations

Sixteen-bit arithmetic is straightforward in the CDP1802 since arithmetic operations including the value of the DF flag are included in the instruction set. Arithmetic operations are performed between a byte in the D-register and a byte from memory but, again, the addition of a

few extra instructions makes register-to-register operations possible.

Example 7: Register/Memory Addition

```
SEX POINTR .. POINT TO
          MEMORY
          LOW BYTE
          OPERAND

GLO REG1
ADD
PLO REG1  ..RESULTS
          RESTORED TO
          REG1

IRX       ..POINT TO HIGH
          BYTE

GHI REG1
ADC
PHI REG1
```

Example 8: Register/Register Addition

```
SEX POINTR ..STORE REG2 HI
          BYTE

GHI REG2
STXD
GLO REG2
STR POINTR ..STORE REG2 LOW
          BYTE

GLO REG1  ..SAME AS
          PREVIOUS
          EXAMPLE FROM
          HERE

ADD
PHI REG1
IRX
GHI REG1
ADC
PHI REG1
```

Note that if each byte had been operated on in sequence, as was done in Example 2, rather than storing both halves of REG2 and then operating on them, the code could have been reduced by one instruction. This reduction could be significant in some applications.

Counters

An increment or decrement instruction to an R-register operates on the full 16 bits of the register. Thus, loop counters of up to 65k counts are readily available. The most direct implementation of a loop counter involves the presetting of an R-register with the desired number, decrementing it once each time through the loop, and testing for zero in the counter. The following example sets up a loop counter for 512 counts.

Example 9: Output 512 Bytes

```
LDI #02; PHI COUNTR
          ..SET COUNTER =
          #0200
```

```
LDI #00; PLO COUNTR
LOOP: OUT1    ..DO AN OUTPUT
              OPERATION
DEC COUNTR   ..DECREMENT THE
              COUNT
GLO COUNTR   ..CHECK LOW
              HALF OF
              COUNTER

BNZ LOOP
GHI COUNTR   ..CHECK HI HALF
              OF COUNTER

BNZ LOOP
XX           ..NEXT INSTRUC-
              TION AFTER LOOP
```

Note that an output instruction outputs MR(X) and then increments R(X), so that this example would output 512 consecutive bytes from memory. Of course, for smaller loops of less than 256 counts, only the low half of a register need be examined, and two instructions can be removed from the loop.

Care must be exercised, when designing timing loops, to equalize the various branch path lengths. The method shown in example 9 would not be suitable for real-time loops. Instead, a loop of the form shown in example 10 should be used.

Example 10: Timing Loop

```
LDI #20; PHI TIMER
              ..SET COUNT
LDI #00; PLO TIMER
LOOP: DEC TIMER
              ..DECREMENT
              TIMER

GLO TIMER    ..TEST LOW BYTE
BZ ENDTST
GLO TIMER    ..DUMMY, IN-
              STRUCTION TO
              MATCH DELAYS

BR LOOP
ENDTST: GHI
TIMER        ..TEST HI BYTE
BNZ LOOP
XX           ..NEXT IN-
              STRUCTION
```

Two notes of caution should be mentioned here. First, a short branch instruction takes two machine cycles, whereas a long branch requires three -- avoid mixing them in a loop. In particular beware of the trap caused by editing long branches into a file in response to assembler-generated "branch out of page" errors. Make sure first that these branches are not in a timing loop. Second, beware of inserting NOP instructions to match delays, as these are three-machine-cycle instructions

A Hardware Approach to Timing Generation

A general-purpose time-delay subroutine can be devised which takes a

passed parameter, puts its value into a register, counts it down to zero, and returns to the caller. Significant time delays can be generated by this method when a 16-bit value is passed, as shown below:

Example 11: General-Purpose Time Delay Subroutine

The main program, using the Standard Call and Return Technique, would look like this when #OFFF is the value being passed:

```
.
.
.
SEP R4, A(DELAY), #OFFF
.
```

The Subroutine itself follows:

```
DELAY:LDA R6
PHI TIMER    ..LOAD PASSED
              PARAMETER

LDA R6; PLO TIMER
SKP          ..THIS ENABLES
              ZERO TO BE
              PASSES

LOOP: DEC TIMER
              ..DECREMENT
GLO TIMER    ..TEST LOW BYTE
BZ ENDTST
GLO TIMER    ..DUMMY INST TO
              MATCH DELAYS

BR LOOP
ENDTST: GHI TIMER
              ..TEST HI BYTE

BNZ LOOP
SEP R5       ..RETURN WHEN
              ZERO REACHED
```

A method of creating an on-board time in the CDP1802 by using its built-in DMA facilities is described in Reference 1.

Input/Output

Inputting a 16-bit word to one of the R registers is a trivial matter in both hardware and software. Fig. 1 shows two input ports where N0 and N1 have been used to select the low and high bytes, respectively, of a 16-bit word. The code required to load this word into REG1 appears below:

Example 11: Input a 16-Bit Word to a Register

```
INP1         ..BRING LOW BYTE
              INTO D
PLO REG1     ..TRANSFER IT TO
              LOW HALF OF
              REG1
INP2         ..BRING IN HI BYTE
PHI REG1     ..STORE IT
```

A shorter software sequence can be developed to input the word into two con-

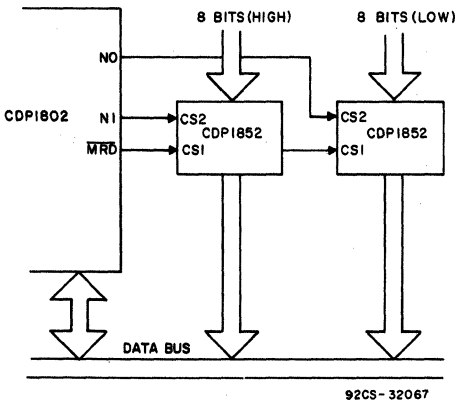


Fig. 1—Basic 16-bit Input circuit.

secutive memory locations since an input byte is automatically written into MR(X) as well as the D register. However, hardware can be devised that will automatically input both bytes when a single input instruction is executed. This circuitry, shown in Fig. 2, operates as follows: Execution of input instruction INP 1 will not directly read in a byte. However, it sets the Service Request (SR) of PORT 1, causing the next machine cycle to be given over to a DMA-IN operation which will read in PORT 1. The timing is such that two DMA cycles will actually occur, inputting PORT 1 and PORT 2 to sequential memory locations pointed to by R(0). The

software for this sequence would consist only of:

```
SEX R0 ..
INP 1 ..
```

Besides saving code, this approach is faster in the input operation (since DMA operations take only one machine cycle) than a sequence involving two inputs and a register increment.

A full 16-bit output can be obtained from the CDP1802 with one instruction, as shown in Fig. 3. Here, the contents of R(X) will be latched into PORTS 1 and 2 during the execute cycle of the output instruction.² Since any one of the 16-bit registers can be output to the address lines, this technique is a very powerful one.

Reference

1. "Optimizing Hardware/Software Trade-Offs in RCA CDP1802 Microprocessor Applications," L.A. Soloman, D. Block, RCA Solid State Application Note ICAN-6704.
2. A more detailed discussion of this "register output" operation can be found in "Register Based Output Function for RCA COSMAC Microprocessors," N. Swales, RCA Solid State Application Note ICAN-6562.

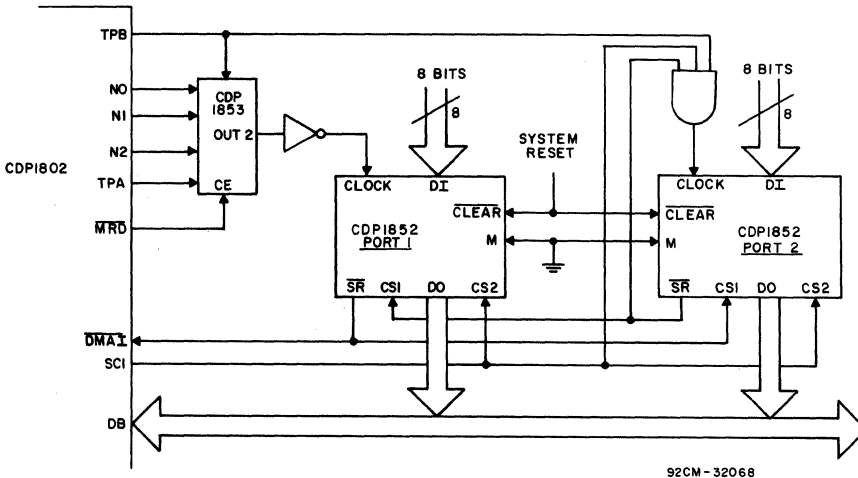


Fig. 2—Circuit for automatic input of a 16-bit value to memory.

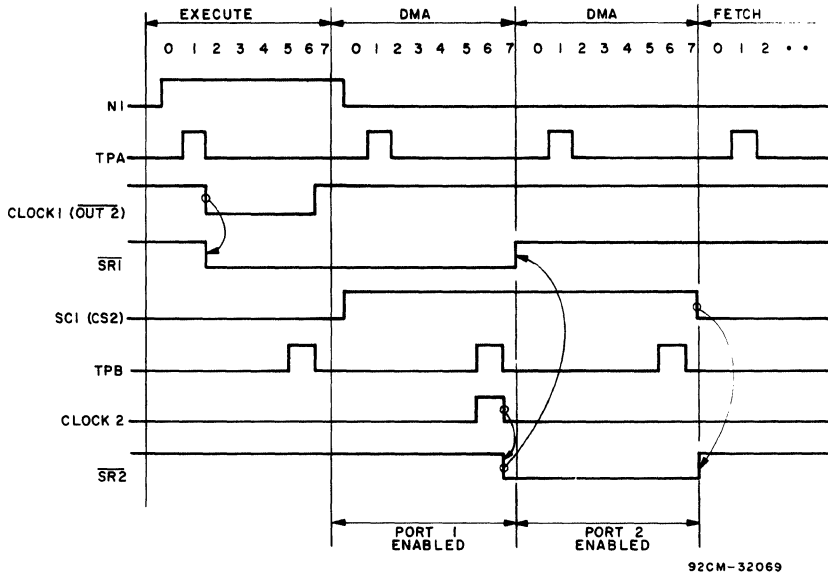


Fig. 3—Timing diagram for the circuit of Fig. 2.

Programming 2732 PROM's with the CDP18S480 PROM Programmer

by D. Block

The CDP18S480 PROM Programmer was designed to program a variety of industry-standard PROM's, including the Intel 2704, 2708, 2758, and 2716's and equivalent products from other suppliers. With a simple hardware addition to the PROM Programmer, and without any software changes, the CDP18S480 can also be used to program Intel 2732 PROM's. This addition to the PROM family is organized as 4K x 8 (4 kilobytes x 8 bits) and operates from a single +5-volt supply. All inputs are T²L compatible except for pin 20 (\overline{OE}/V_{pp}), which must be pulsed to 25 volts during programming.

Many electrical and mechanical characteristics have been maintained between the 2716 (organized as 2K x 8) and the 2732 for upwards compatibility. For example, only the definitions of pins 20 and 21 have changed between the two versions, as shown in Fig. 1. In the 2732, pin 20 continues to represent an output enable function, but with V_{pp} (the programming voltage) also impressed on it; pin 21 becomes the A₁₁ address pin.

Electrically, the programming voltage (+25 volts) and programming pulse width (50 milliseconds nominal) are the same

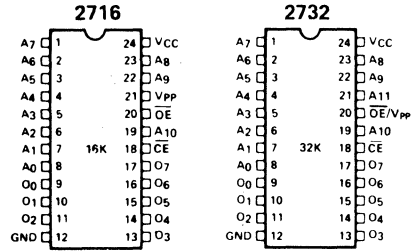


Fig. 1—Pin configurations. 92CS-32591

for both the 2716 and 2732. The technique for 2732 programming described here takes advantage of this fact. The 2732 will be treated as two 2716's for all operations, and the logic circuitry described below will handle necessary logic-level conversions and voltage switching. The hardware is designed to plug into the 2716 socket of the PROM programmer.

Theory of Operation

Fig. 2 shows the programming waveforms for the 2716 and 2732. Note that the voltages on pin 21 of the 2716 are almost those required for pin 20 of the

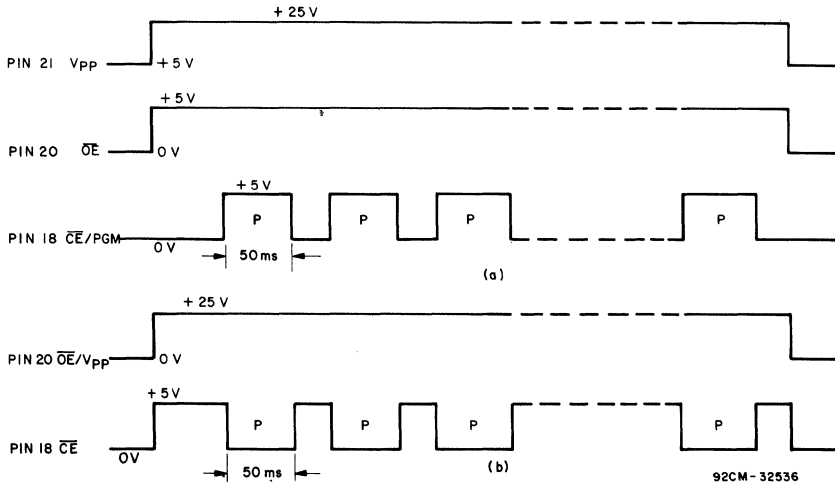


Fig. 2—Programming waveforms (not to scale): (a) 2716, (b) 2732. 92CM-32536

2732. However, pin 20 must go to ground in the Read mode. The transistor circuitry in Fig. 3 is designed so that when pin 20 of the 2716 is low, the input to pin 20 of the 2732 will also be low. When pin 20 of the 2716 is high, pin 20 of the 2732 is connected to the programming voltage (if a programming operation is in effect) or to +5 volts, disabling the chip.

The programming pulses applied to pin 18 of the two devices differ in polarity. Inversion of the pin 18 signal is accomplished through pin 20 of the 2716 by means of an Exclusive-OR gate. The conditions for reading and programming are met by this arrangement.

As mentioned above, pin 21 on the 2732 is the A₁₁ address pin. Fig. 3 shows this pin connected to a toggle switch. The 2732 will be exercised twice for each operation: once with A₁₁ low, and then with A₁₁ high.

Hardware Construction

The circuitry of Fig. 3 can be constructed on a small vectorboard that plugs directly into the 2716 socket of the

PROM Programmer. A ZIF, zero-insertion-force, socket is required for the 2732. It is recommended that solder connections not be made directly to this socket, since the solder may wick up the leads and interfere with operation of the pin clamps. Instead, the ZIF socket should be plugged into a second socket, and connections made to that socket.

Only a CD4049 inverter should be used to drive pin 18 of the 2732. Other CMOS inverters may not have sufficient drive to handle the T²L input. The use of capacitor C1 is recommended by Intel to prevent transients from exceeding the +26-volt maximum rating of the device.

Operation

The 2732 will be programmed as two 2716's.¹ When programming from a disk file, sequential sections of the file must be loaded in, since the disk-file reader loads only 2 kilobytes at a time into the buffer. Similarly, when copying one 2732 into another, several steps are necessary. The examples below should help clarify the requirements. CDS output is underlined. The CDOS software version is assumed.

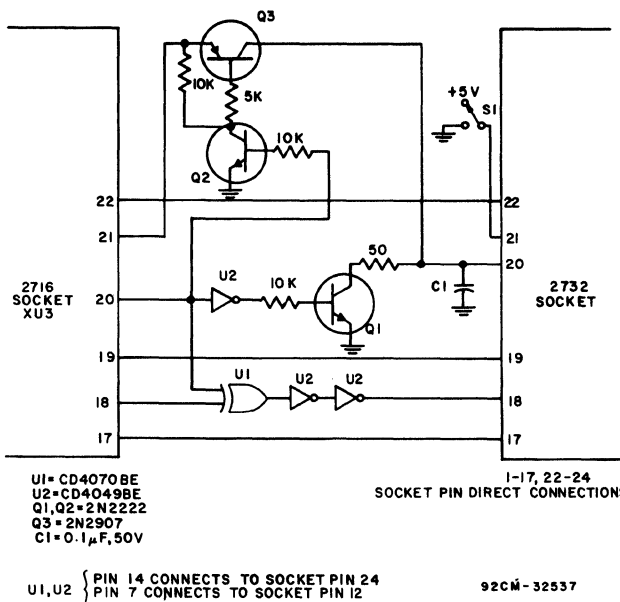


Fig. 3—Circuit diagram.

Example 1:

Program a 2732 from a disk file called FILE.OBJ for the address range 0000-0FFF. First, set S1 so that pin 21 is low.

```

PROM PROGRAMMER, VERSION XX
ENTER V,C,P,F,S,U: P
TYPE#: 2716 (CR)
LOGIC = P,N: P
(D)ISK OR (R)AM: D
INPUT FILENAME: FILE.OBJ
ENTER LOWEST PROM ADDRESS (XXOO): 0000(CR)
LOADING COMPLETED
DONE! .. The first half of the PROM is done
REPROGRAM 2716?: N .. Set S1 in the opposite position
PROM PROGRAMMER, VERSION XX
ENTER V,C,P,F,S,U: P
TYPE#: 2716(CR)
LOGIC = P,N?: P
(D)ISK OR (R)AM?: D
INPUT FILENAME: FILE.OBJ
ENTER LOWEST PROM ADDRESS (XXOO): 0800(CR) .. Second half of
address range.
LOADING COMPLETED
DONE! .. Entire 2732 is now programmed
REPROGRAM 2716? N
    
```

Example 2:

Copy two 2716's into one 2732. Insert the lower address 2716 first.

```

PROM PROGRAMMER, VERSION XX
ENTER V,C,P,F,S,U: C
TYPE#: 2716(CR)
LOGIC = P,N?: P PAGE# = 08(CR) .. First half = pg 8-F
DONE!
PROM PROGRAMMER, VERSION XX .. Insert higher address 2716
ENTER V,C,P,F,S,U: C
TYPE#: 2716(CR)
LOGIC = P,N?: P PAGE# = 10(CR) .. Second half = pg 10-17
DONE!
PROM PROGRAMMER, VERSION XX .. Remove 2716, insert 2732, set S1 low
ENTER V,C,P,F,S,U: P
TYPE#: 2716(CR)
LOGIC = P,N?: P
(D)ISK OR (R)AM? R PAGE?: 08(CR)
DONE!
REPROGRAM 2716: N .. Set S1 high
PROM PROGRAMMER, VERSION XX
ENTER V,C,P,F,S,U: P
TYPE#: 2716(CR)
LOGIC = P,N?: P
(D)ISK OR (R)AM? R PAGE# = 10(CR) .. Second half of data
DONE!
REPROGRAM 2716? N .. Entire 2732 complete
    
```

Reference:

1. The operating procedure for the 2716 is given in: "Operator's Manual for PROM

Programmer CDP18S480," RCA Solid State publication MPM-222A.

Simplified Design of Astable RC Oscillators Using the CD4060B or Two CMOS Inverters

D. Rodman

Application Notes are available that deal with theoretical approaches to oscillator design; this Note stresses practical aspects of design and provides easy-to-use algebraic equations that permit values of R and C for a given oscillator frequency to be quickly determined.

Astable Design Approach

The most basic RC oscillator circuit is that shown in Fig. 1. The time period T for one cycle of this oscillator is given by the equation:¹

$$T = -RC \left[\ln \frac{V_{DD} - V_{TR}}{V_{DD}} + \ln \frac{V_{TR}}{V_{DD}} \right] \quad (1)$$

where:

V_{DD} = supply voltage
 V_{TR} = transfer voltage

By letting $V_{TR} = 0.5 V_{DD}$, equation 1 can be simplified to:

$$\begin{aligned} T &= -RC (\ln 0.5 + \ln 0.5) \\ T &= 1.39 RC \end{aligned} \quad (2)$$

The problem with this circuit is that transfer voltage can vary from 33 to 67 percent of V_{DD} . Therefore, the maximum variation in the time period, T, can be as high as 9 percent, with a ± 33 percent variation in transfer voltage from unit to unit.

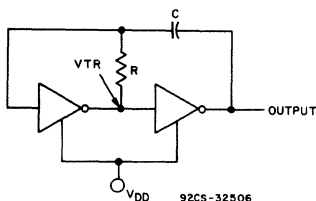


Fig. 1 - The most basic RC oscillator circuit.

An improvement to this basic circuit can be made by adding resistor R_S , as shown in Fig. 2. The resistor makes the frequency independent of supply-voltage

variations and reduces the time-period variations to less than 5 percent with variations in transfer voltage.

R_S should be 10 times the value of R_X . If R_S is made less than $10 R_X$, the variation in period T increases to about 10 percent as the value of R_S approaches zero.¹ If R_S is made too large, a time constant and phase shift is produced by R_S and stray wiring and breadboard capacitance. This shift creates a switching delay in the circuit which changes the time period.

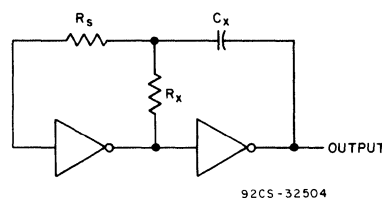


Fig. 2 - An improved oscillator circuit made by adding resistor R_S to the circuit of Fig. 1.

The time period T for the circuit in Fig. 2 is:¹

$$T = -R_X C_X \left[\ln \frac{V_{TR}}{V_{DD} + V_{TR}} + \ln \frac{V_{DD} - V_{TR}}{2 V_{DD} - V_{TR}} \right] \quad (3)$$

If $V_{TR} = 0.5 V_{DD}$, equation 3 can be simplified to:

$$\begin{aligned} T &= -R_X C_X (\ln 1/3 + \ln 1/3) \\ T &= 2.2 R_X C_X \end{aligned} \quad (4)$$

Equation 4 will only be true in the CD4060B for values of R greater than 50 kilohms and for values of C greater than 1000 picofarads. At values of C less than 1000 picofarads, stray capacitance will have a much greater effect on the entire system.

It is advised that a buffer circuit, Fig. 3, be added to the circuit of Fig. 2 to prevent the jitter that would otherwise be introduced into the circuit by noise picked up by connecting cables and by stray wiring and breadboard capacitance. The buffer circuit is not needed with the CD4060B since it has an internal buffer and is internally connected to a counter.

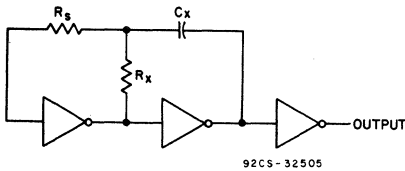


Fig. 3 - A buffer circuit used to improve the performance of the circuit of Fig. 2.

Compensation for 50-Percent Duty Cycle

A true square-wave pulse is obtained only when the transfer voltage occurs at the 50-percent point. If the transfer voltage is at either 33 or 67 percent, the duty cycle will not be 50 percent. The duty cycle can be controlled, however, if part of the resistance of the RC time constant is shunted out with a diode, as shown in Fig. 4.

Because adjustment of this diode shunt to obtain a specific pulse factor causes the frequency of the circuit to

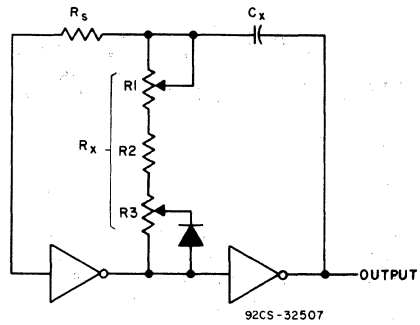


Fig. 4 - Method of controlling the duty cycle of the RC oscillator.

stray, a frequency control, R1, is added. This circuit is not needed when using the CD4060B since it is used in conjunction with a counter. A 50-percent duty cycle will be derived from the divider/counter outputs.

References and Bibliography

1. "Astable and Monostable Oscillators Using RCA COS/MOS Digital Integrated Circuits," RCA Solid State Application Note ICAN-6466.
2. "COS/MOS 14-Stage Ripple-Carry Binary Counter/Divider and Oscillator," RCA Solid State Data Bulletin File Number 1120.

USING SLOWER MEMORIES WITH THE VIS DISPLAY SYSTEM

G. T. Fogarty

The VIS (Video Interface System) Display System (CDP1869 and CDP1870)¹, a minimal-device-count approach to color-character generation, is essentially a CRT controller designed to interface to the CDP1800 series of microprocessors (CDP1802, CDP1804). The system relieves the CPU of the chore of generating screen refresh timing or data. Other capabilities include programmable background and character colors, white-noise and tone generator, and hardware scrolling. The scheme described in this Note, while requiring a few more parts, very nearly doubles the memory access-time requirement of the system, and permits the use of memories approximately half as fast as those normally required with the VIS System.

VIS System Operation

The VIS System was designed with minimal chip count as a goal. A minimum I/O system requires only the CDP1869, CDP1870, page memory, character memory, and two bus separator chips, Fig. 1. The bus separators are required to allow the CPU to access the page memory. The character memory bus multiplexing is internal to the CDP1870.

The character generating scheme is as follows:

The page memory is a sequential list of character positions on the CRT screen. Its data is a pointer to the character to be displayed at that screen position. The character memory contains the actual dot pat-

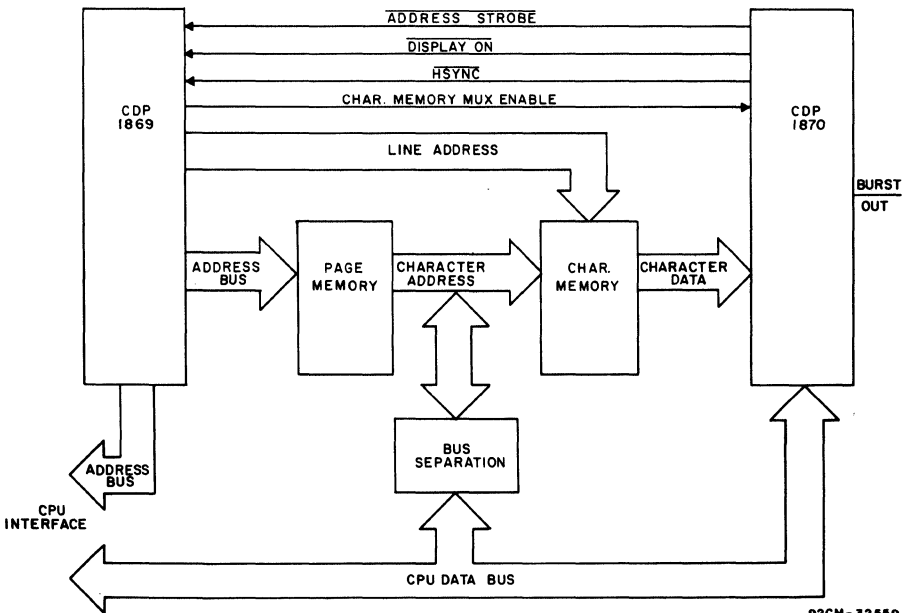


Fig. 1—A minimum I/O system.

92CM-32559

tern of all the possible characters that can be displayed. During screen refresh, the CDP1869 generates addresses to the page memory. The page memory output data, in turn, addresses the character memory, whose data is then latched into the CDP1870 for serial output to the CRT screen.

Note that one character cycle involves two memory access times: page address to page memory, and page data (character address) to character memory. In a 40-character-per-line system, the total of these two times is about one microsecond (six dot clocks) minus the page memory address delay from the CDP1869 and the character data setup time into the CDP1870.

The timing diagram in Fig. 2 shows that the signal, ADDRESS STROBE, from the CDP1870 advances the page-memory address in the CDP1869 at the trailing edge,

signal. The multivibrator signal is used to advance the address counter in the CDP1869 and to latch page-memory data. The burst signal, occurring once per horizontal sync., is used to generate the "extra" address count. During nondisplay time, the one-shot is disabled to prevent the address counter from advancing, and the latch control is held true. The latter makes the latch feed through, enabling normal character memory access. Gates A and B inhibit the last strobe in half and full resolution, which allows for proper hardware rolling or scrolling. PMA9 through PMA9 are addresses 4 through 9 of the CDP1869 and are used for the page memory.

With this "staggered access" circuit, page memory access starts at the trailing edge of the new strobe pulse (pulse address out delay) and terminates at the trailing edge of the next strobe; the total

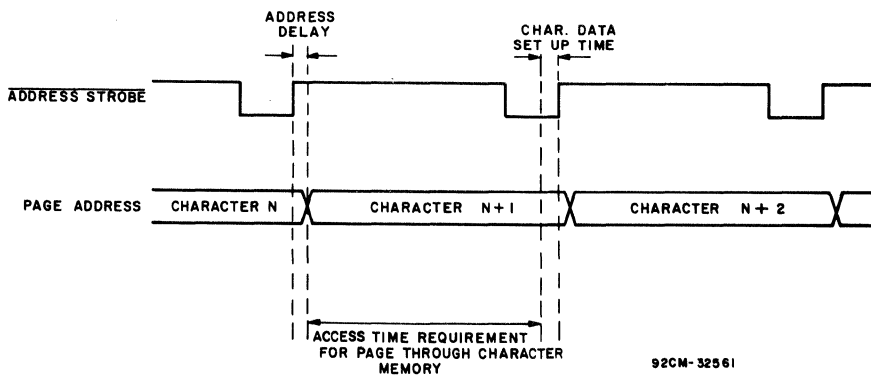


Fig. 2—Timing diagram for Fig. 1.

initiating an address cycle. This signal is analogous in time to the shift-register load signal in the CDP1870. Therefore, the next ADDRESS STROBE terminates the current access, and initiates the next character access.

Adaptation for Slower Memories

If page and character memory were accessed in parallel, the memory speed requirement would be eased substantially. This parallel accessing can be accomplished by latching the page-memory data and keeping the page memory one character ahead of the character memory; the circuit shown in Fig. 3 accomplishes this task. A TTL one-shot multivibrator was chosen for minimal delay. The multivibrator generates a pulse of approximately 200 nanoseconds starting at the trailing edge of the ADDRESS STROBE

time required is approximately that to access one full character. Character memory access starts at the leading edge of the strobe signal (plus latch delay) and terminates at the data setup time requirement of the CDP1870, also approximately the time to access one full character. Since the delay through the latch is similar to the address out delay of the CDP1870, this scheme, while it requires a few more parts, essentially doubles the memory access-time requirement of the VIS System, and permits the use with it of memories approximately half as fast as those normally required.

References

1. "COS/MOS Video Interface System," types CDP1869 and CDP1870, RCA Solid State Data Sheet No. 1197.

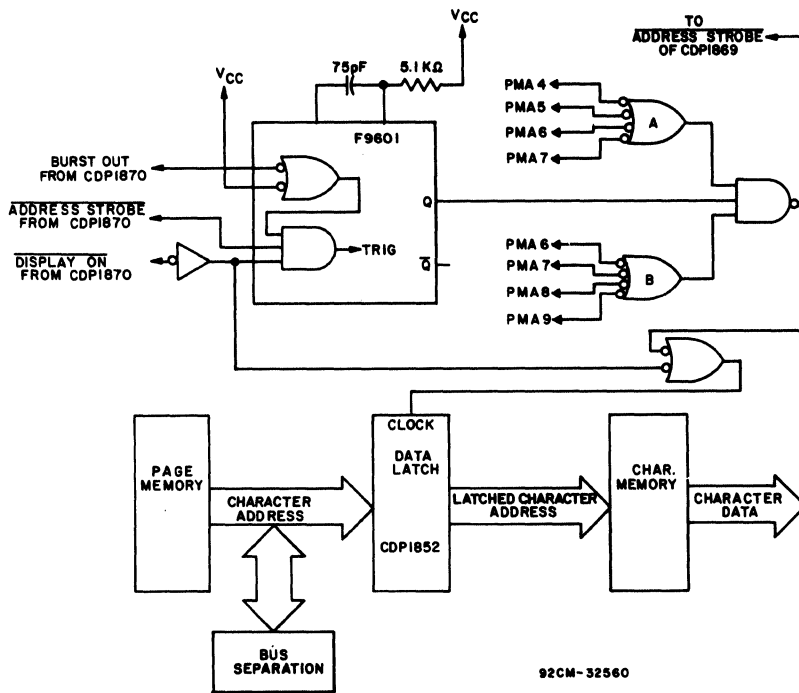


Fig. 3—Circuit used to adapt VIS System to slower memories.

CDP1802 Microprocessor-Based Setback Thermostat

by R. Vaccarella
K. Krolikoski

This Note describes an inexpensive, programmable, setback thermostat circuit, based on the RCA CDP1802 microprocessor,¹ that can be used to control both heating and air-conditioning systems; a block diagram of the thermostat circuit is shown in Fig. 1. The description below applies to a home thermostat with four programmed times and temperatures; however, the system can be adapted to industrial and institutional use by expanding it to accommodate a greater number of preset times and temperatures. The basic thermostat consists of a CDP-1802 eight-bit microprocessor, a CDP1833 1K x 8 ROM, a CDP1823 128 x 8 RAM or two CDP1824 32 x 8 RAM's, a temperature

sensor, type AD537,² a keyboard and interface, and a display and interface. The system was designed to keep the number of components and cost to a minimum, but to provide temperature and time measurements sufficiently accurate for its intended application.

GENERAL SYSTEM OPERATION

The programmable-thermostat system has four modes: set time, set time and temperature set points, normal operation, set current-temperature set point. These modes allow the user to set the clock to the correct time and to program the times at which he wishes the temperature to be

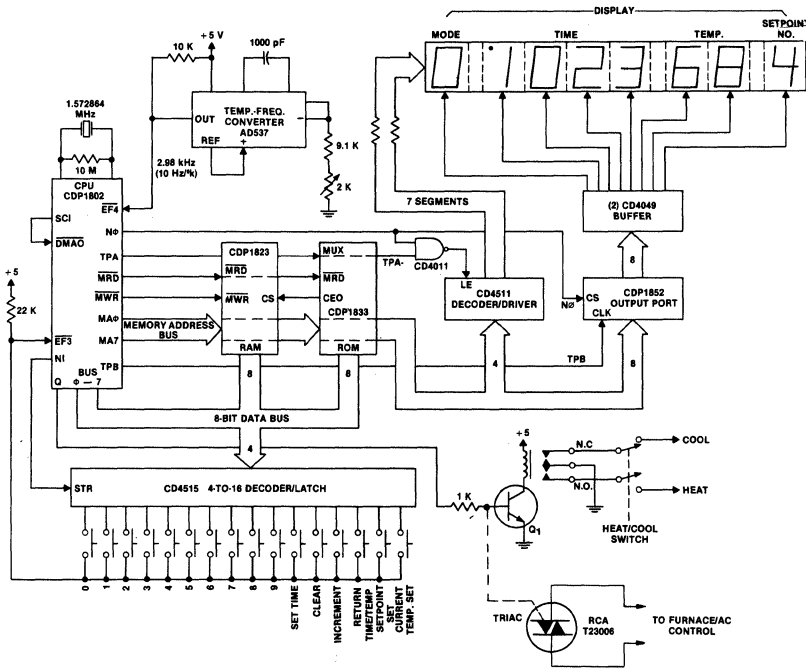


Fig. 1 - Block diagram of CDP1802-based thermostat system.

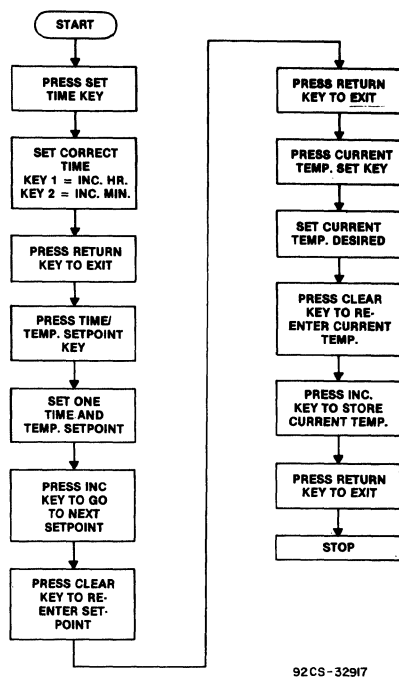


Fig. 2 - Thermostat use flowchart.

maintained at a given level. He can at any time override the current temperature setting by inserting a new temperature into the system. The flowchart of Fig. 2 will aid in following the description of system setup and modes given in the succeeding paragraphs.

System Setup - During initialization of the system, or any time upon power-up, the time will default to 12:00 PM and the current temperature set point will default to 68°F. The system is now in the normal operation mode, which is described more fully below.

Set Time - The set-time mode, entered from the normal-operation mode by pressing the Time Set key, allows the user to set the clock to the correct time. To increment hours, the user presses the 1 key; to increment minutes, he presses the 2 key. The clock used in the system is a twelve-hour clock with an AM/PM indicator (a 24-hour clock for foreign markets requires a slight program change). The AM/PM indicator, important when setting time and temperature set points, is a decimal point in the top left-hand corner of the time display; it is on for AM, and off for PM. If, when setting time, the user advances the clock up to or past 12:00, the AM/PM indicator will change to the state opposite its present one. To exit the set-time mode and re-enter the regular-operation mode, the user presses the Return key. All modes can be entered from the regular-operation mode only, and the user must return to this mode after setting either time or temperature.

Set Time and Temperature - The set-time-and-temperature mode, entered by pressing the Time/Temp Set key, is used to set the times of desired temperature change and the corresponding temperatures. Once this mode has been entered, the set-point number indicator will show a one, indicating that the user is looking at the memory space for the first set point. The system provides four set points, although many more could be made available to the user through software expansion.

To change or program new set points, the user presses the Clear key, which blanks out both the time and temperature display, an indication that the memory space is ready to be programmed. The user then enters the time starting from the left-most digit. If an error is encountered, such as an effort to enter the time 13:00, the incorrect digit will not be accepted. (In this case, with a one in the left-most position, the only valid entries in the next position are 0, 1, and 2.) The minutes are entered in the same way with valid minutes being those less than sixty. The AM/PM indicator is entered next with the 1 key indicating AM and the 0 key indicating PM. The temperature is then entered; valid temperatures range from 50°F to 80°F. Once entered, the time-and-temperature set point is stored by pressing the Inc key. Simultaneously, the system steps to the next memory-space set point. If, when entering a set point, the user finds that a previously entered digit is incorrect, he can clear the memory space by pressing the Clear key. To review the four set points, the user presses the Inc key. Each time this key is pressed, the next memory space can be viewed, at which time the user can elect to exit from the program, change the set point, or continue to the next set point. To exit the set-time-and-temperature mode, the user presses the Return key, which will return him to the normal-operation mode.

Normal Operation - Upon return to the normal-operation mode from the set-time-and-temperature mode, the times now programmed are checked immediately against the present time. If the present time matches any of those just set, the current-temperature set point is set equal to the temperature for that time. If the ambient temperature is 2°F below the current-temperature set point, the thermostat indicates through the Q line, by means of a single output bit from the CDP1802, that the heater should be turned on. (The Q-line signal activates a relay or SCR that replaces the normal contact switch in the typical household thermostat. If the thermostat is to be used to activate an air-conditioning unit, software changes will be required to control the Q-line output.) If the room temperature is 2°F above the current temperature set point, the heater shuts off. Current time and temperature are then checked once per minute against the stored information.

Set-Current-Temperature-Set-Point - The set-current-temperature-set-point mode, entered by pressing the Current Temp key, is used to enter a temperature set point, which will become effective immediately. The new current-temperature set point entered will override the existing temperature set point and will be used by the system to regulate temperature until a stored time is found in the memory to match the then current time.

When this mode is entered, the current-temperature set point is displayed. To change this set point, the user presses the Clear key, which blanks the display. The temperature is then entered, left digit first; again, only temperatures between 50°F and 80°F can be entered. As in the set-time-and-temperature set-point mode, if the user makes a mistake in entering the new temperature, he can clear the display by pressing the Clear key and then enter the correct data. The Inc key is pressed to store this new current-temperature set point. To exit this mode, the user presses the Return key, which returns him to the normal-operation mode.

SYSTEM HARDWARE

Keyboard and Interface

The CD4515 (4- to 16-line latch decoder) is used to perform the keyboard interface. Its outputs are low when selected, and only one output, determined by the decoder inputs, can be low at any given time. The keyboard-scan software routine, Fig. 3, outputs four bits to the CD4515. The scan routine then checks to see if EF3 has gone low. If it has, the line identified as low by the

decoded four bits has been connected to EF3 by the corresponding line-key switch. If EF3 remains high, the line-key switch corresponding to the decoded four bits has not been depressed, and the four bits are incremented and sent to the CD4515 where a test is made for a different key-switch depression. The actual key depressed is determined by examining the four bits that were decoded by the CD4515 when EF3 went low.

Display and Interface

The display used in this system is an eight-digit, multiplexed, LED display. Four digits are used for timekeeping, two for temperature, one for the mode indicator, and one for the memory set-point indicator. The AM/PM indicator is a decimal point in the upper left-hand corner of the time display. A CD4511, a BCD-to-7-segment decoder driver, is used to drive the LED segments. A CDP1852 I/O port is used to select the individual digit when multiplexing the display.

The display-refresh flowchart is shown in Fig. 4. The memory-address bus is used

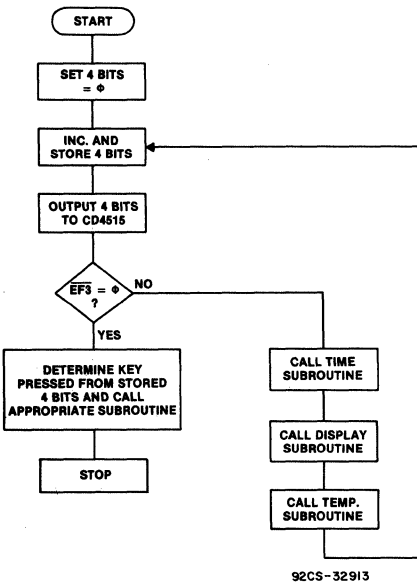


Fig. 3 - Keyboard scan-routine flowchart.

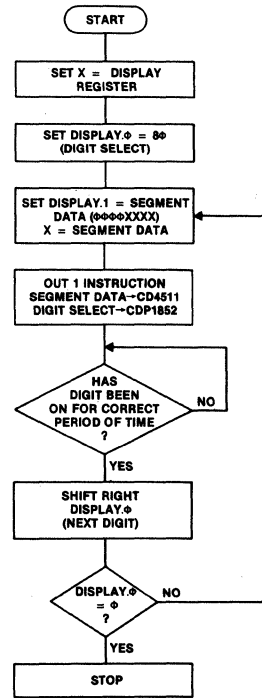


Fig. 4 - Display refresh flowchart.

instead of the data bus to load data for the latch driver and digit selector. This method uses only one output instruction for every multiplexed LED digit instead of two separate output instructions. When an I/O instruction is executed by the CDP1802, the higher order address of the register,

pointed to by the internal four-bit register X, R(X), is loaded into the memory-address bus; this loading is done during TPA. The lower half of R(X) is then put onto the memory-address bus during the latter half of the I/O instruction-execution cycle. Advantage is taken of this procedure to load the four-bit BCD digit information for the display into the upper half of R(X) and the digit-selector information into the lower half. When a 61 I/O instruction is executed (N0=1), and during TPA, the BCD digit information is latched into the CD4511; the digit selector is latched into the CDP1852 during TPB. To multiplex the entire eight digits, the upper half of R(X) is loaded with new BCD information while the lower half is shifted right by one to accommodate the next digit.

Should the thermostat system be adapted for battery operation, the LED display would have to be off a high percentage of the time to minimize drain. One way to accomplish this would be to make the display visible only when setting time or temperature, in much the same way that an LED watch display is called up by depressing a display key. An alternative approach might involve the use of a low-power liquid-crystal display.

Temperature Sensor

Two important qualities of the temperature-detection circuit are low cost and relatively good accuracy ($\pm 1^\circ\text{C}$) within the normal home temperature range of 50 to 80°F (the sensor may be less accurate at the extremes of this range).

The temperature sensor used in this system is a voltage-to-frequency converter, type AD537;² a block diagram of the converter is shown in Fig. 5. The AD537 has a temperature-proportional output that enables it to be used as a temperature-to-frequency converter; its output frequency changes 10 Hz per °K. At 25°C (298°K) with one external capacitor, resistor, and resistor trimmer, the output frequency is 2.98 kHz.

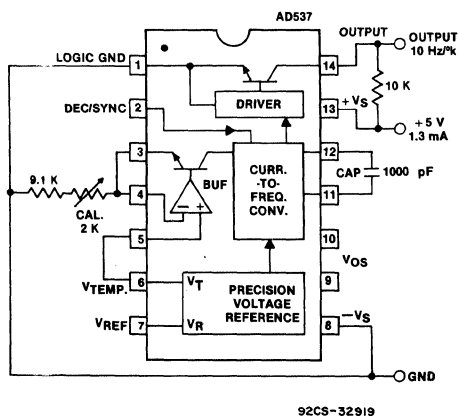


Fig. 5 - Absolute temperature-to-frequency converter.

In detecting temperature, the output of the AD537 is fed into a flag line ($\overline{\text{EF4}}$) of the processor which, by executing a software branch on flag-high/low instructions, counts six hundred periods or pulses while keeping track of the elapsed time; a flowchart of temperature-sensor operation is shown in Fig. 6. When six hundred pulses have been detected and the elapsed time stored, the temperature can be determined through the use of a look-up table. This table is based on temperatures stored at addresses calculated by using elapsed time as variables in the address equation.

Other temperature sensors have been investigated. One possible circuit, Fig. 7, uses a CA555, an RC controlled timer, in conjunction with a thermistor, a device whose resistance varies with temperature. The thermistor is used as the R in the RC network that controls the output frequency; as the temperature changes, the resistance of the thermistor changes and the output frequency changes. The circuit of Fig. 7 has a 1 kHz/°C frequency output. A plot of frequency versus temperature change is shown in Fig. 8.

The main requirements for a circuit employing the temperature-to-frequency method of temperature measurement are that the $\Delta\text{Hz}/^\circ\text{C}$ ratio is large enough to be accurately measured by the CDP1802 software, and that the circuit can be duplicated using identical components from the same manufacturer; i.e., the $\Delta\text{Hz}/^\circ\text{C}$ ratio is identical for all circuits manufactured.

Another possible temperature-sensor circuit is one that converts temperature to a voltage. The voltage is then converted to a numerical form by an analog-to-digital converter. Analog Devices Corporation manufactures a temperature-to-current converter, the type AD590,² that passes 1 $\mu\text{A}/^\circ\text{K}$ (298.2 μA at 298.2°K (25° C)).

Software Program Flow

The main-program software flowchart is shown in Fig. 9. The system program first initializes all constants and memory-space set points, and sets the default values for time and temperature. It then calls a keyboard scan routine (Fig. 3), a software loop that will be exited only when a key has been pressed. While in this loop, display and timekeeping subroutines are called, so that a display is visible and accurate time is kept. In addition, once every minute, a temperature sensing subroutine is called and the heater is turned on or off as necessary.

If a key is pressed, the program determines if it is a valid key and, if so, calls the appropriate set-time, set-time-and-temperature set point, or set-current-temperature-set-point subroutine. Table I shows the functions performed in each subroutine.

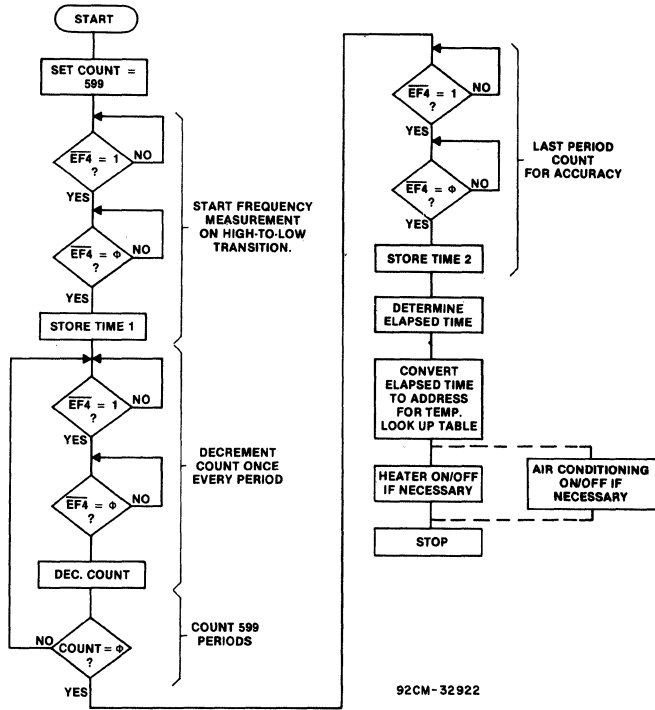


Fig. 6 - Temperature sensing flowchart.

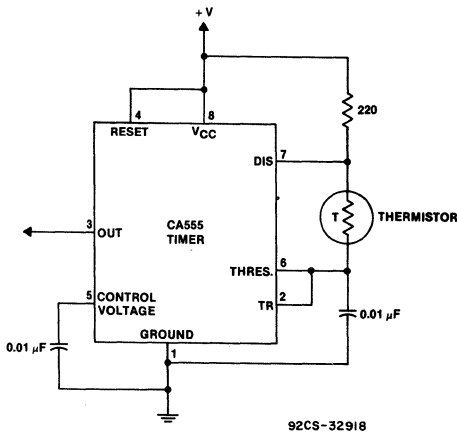


Fig. 7 - Alternative to the temperature-to-frequency converter temperature sensor.

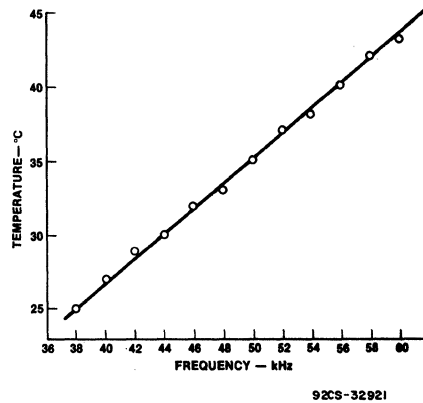


Fig. 8 - Temperature versus frequency chart used with the circuit of Fig. 7.

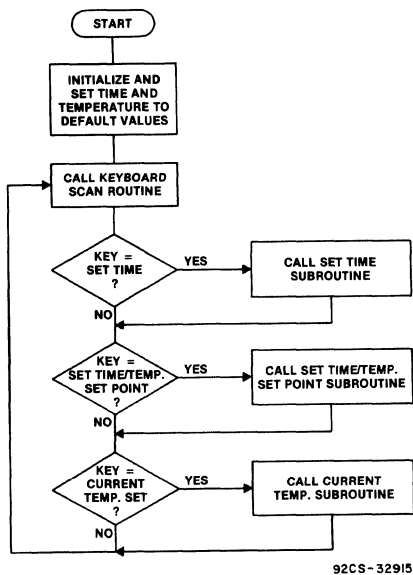


Fig. 9 - Main program flowchart.

Scratchpad Memory

The scratchpad memory used in this system is a 128 x 8 RAM. This memory is used almost exclusively for time and temperature set points entered by the user, and for present time and temperature data. The size of the memory is sufficient to accommodate most programming options that may be added to the basic system. A smaller memory could be used, but system expansion would be hindered.

TIMEKEEPING

Timekeeping for the thermostat is performed in software;³ the software-clock flowchart is shown in Fig. 10. The only restrictions on using this software clock design are that only two-cycle instructions be used in the program and that the timekeeping subroutine be called at least every half second. Use of this software-clock design eliminates some of the external logic normally required for timekeeping operations. The technique takes

Table I - Functions Performed in Each Subroutine

<u>Subroutines</u>	<u>Keep Time</u>	<u>Sense Temp. and Turn Heater On/Off If Necessary</u>	<u>Show Display</u>
Set Time	O	O	X
Set-Time-and-Temperature Set Point	X	O	X
Set-Current-Temperature Set Point	X	O	X
Keyboard Scan	X	X	X

X=function performed

O=function not performed

Program Storage

The software program may be stored in a CDP1833, a mask-programmable 1K x 8 ROM. The advantages of using a microprocessor instead of discrete logic in the system design are that options, such as day-of-the-week programming, can be easily implemented by using a different ROM. The remaining hardware will remain intact with all the major changes being in software. One can envision a product line of microprocessor-controlled thermostats with such options as day-of-the-week programming for heating only and for heating and cooling, a centigrade temperature base, etc., with the only major changes being the software in ROM.

Another interesting feature that could be implemented entirely through software is "self-learning". A microprocessor thermostat system could "learn" how long beforehand the heater would have to be activated to have the house reach a desired temperature at a desired programmed time, or how long the heater would have to be off (or air conditioner on) to cool the house to a desired temperature by a given, programmed time.

advantage of the characteristics of the CDP1802 DMA-OUT feature and the state-code status outputs, and is independent of program execution. If the DMA-OUT line on the CDP1802 goes low, the microprocessor completes the fetch and execute cycles of the current instruction, and then executes DMA cycles as long as the DMA-OUT line is low. During a DMA-OUT cycle, the processor reads data from the memory location pointed to by R0, one of the sixteen internal scratch registers. (The data is normally latched off the bus by the peripheral device that requested the DMA-OUT.) R0 is then automatically increased by one.

SC0 and SC1 are state code outputs of the CDP1802 that indicate the type of cycle that the processor is in; Table II identifies these states. If the SC1 output from the processor is wired to the DMA-OUT input, the following sequence of events will take place. During the fetch and execute cycles, the processor detects a DMA-OUT request. On completion of the fetch-execute cycle pair, a DMA-OUT cycle is performed. Only one DMA-OUT cycle is executed because SC1 is now high, and the next cycle will

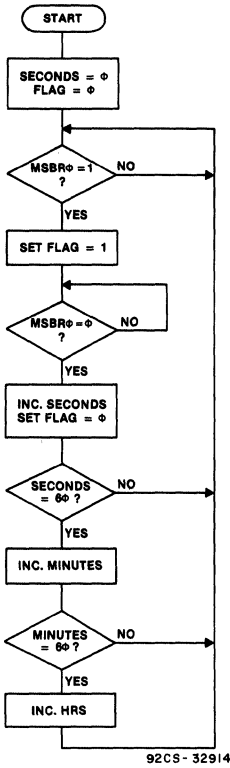


Fig. 10 - Software-clock flowchart.

Table II - Processor States

	SC0	SC1
Fetch	0	0
Execute	1	0
DMA	0	1
Interrupt	1	1

default to a fetch. If SC1 is left wired to the DMA-OUT line and only two-cycle instructions are used, the three-cycle sequence, fetch, execute, DMA-OUT, with R0 being incremented after every DMA-OUT, will continue to be executed.

Because every CDP1802 machine cycle is executed in the same time (eight clock periods), R0 is incremented every 24 clock cycles (fetch, execute, DMA). The frequency at which the 16-bit register R0 would require exactly one second to overflow, that is, the frequency needed to perform 2¹⁶ (R0 overflow) increments every 24 clock cycles (three machine cycles) can be calculated from the following equation:

$$24 \text{ cycles} \times \frac{1 \text{ sec.}}{f \text{ cycles}} \times 2^{16} = 1 \text{ second}$$

where f is calculated to be 1.572864 MHz. A CDP1802 operating at this frequency and wired in the manner described above keeps accurate time by monitoring the MSB (most significant bit, bit 15) of R0 at least every half second. Each time the MSB changes state, one half second has passed. The flowchart for the software clock is shown in Fig. 2. The accuracy of the clock depends on the frequency source used. The need to check MSB is not a hindrance in programming because 2¹⁵ (32,768) instructions can be executed within each half-second interval. Another advantage of this technique is that although the time is kept in software, no intricate timing loops are required.

CONCLUSIONS

The advantages of using this energy saving, CDP1802-based, thermostat design are many:

1. Ease of Operation - programming of the thermostat is easily accomplished once the programming instructions have been reviewed by the user.
2. Low Cost - the minimal amount of components used should keep the retail price low enough so that the actual cost of the unit will not outweigh the fuel savings.
3. Redesign Capability - through the use of ROM-based software, a variety of models could be marketed with different options. Each model could differ in as little as a plug-in ROM module.
4. Low Power - the low-power CMOS technology of the CDP1802 makes battery operation of this system feasible. Power drain could be minimized through the use of a demand LED display or a low-power liquid-crystal display.
5. Versatility - the home-thermostat design is readily adaptable to industrial and institutional applications.
6. Reliability - through the use of CMOS IC's, moving parts, such as those found in mechanical timers, are eliminated, thereby increasing the system's reliability.

REFERENCES

1. RCA COS/MOS Memories, Microprocessors, and Support Systems," Solid State Databook SSD-260. "RCA COS/MOS Integrated Circuits" Solid State Databook SSD-250.
2. Analog Devices Data Acquisition Products Catalog, 1978.
3. "Software Control of Microprocessor Based Realtime Clock," K. Karstad, RCA Application Note ICAN-6677.

A Methodology for Programming COSMAC 1802 Applications Using Higher-Level Languages

by W. Fritchie

This Note defines a method of optimizing the time-critical portions of programs written in higher-level languages for COSMAC 1802 applications by recoding those portions in assembly language.

The Dilemma

The three main reasons for using higher-level languages in the development of microprocessor applications are to assure the development of highly reliable software, to reduce overall program development time (including program design, coding, and debugging) and thus the cost of software development, and to aid in program maintenance. Once a program is written in a higher-level language, its maintenance becomes much simpler because its logic is described in a much more readable and understandable form.

Of special importance in the maintenance area is program documentation. The earliest step in program development involves its design, and some form of design notes or documentation, such as flowcharts, are always formulated before the coding begins. However, unless a great deal of time is spent on updating the design information while an assembly language program is being developed, the program, when finally debugged and running, will probably not match the initial set of design notes. To a degree, the higher-level language eliminates the need to go back and update the original design notes because the syntax of the language not only accurately defines the logic of the program but also becomes the design information. Therefore, the use of higher-level languages assures more reliable software, speeds up program development, and aids in program maintenance.

It would seem obvious, then, that microprocessor applications should be designed in a structured higher-level language, and there are a large number of non-time-critical applications that can be written totally in higher-level languages

like PLM. Moreover, if time-critical applications are programmed for hardware that is not finalized, it may be possible to redesign the hardware so that its overall performance will match the application requirements when it is used with a program written entirely in the higher-level language. However, because it is not always possible to redesign the hardware, or for other reasons described below, it may not be possible to code the entire application in a higher-level language and still achieve the desired performance. Therefore, the only alternative is to rewrite portions of the program, or the entire program, in assembly language.

This performance dilemma was verbalized in a U.S. Government report¹ as follows:

“Unfortunately, machine language insertions are necessary for interfacing special-purpose devices, for accessing special-purpose hardware capabilities, and for certain code optimizations on time-critical paths. Here we have an example of Dijkstra's dilemma [see below], in which the mismatch between higher-level language programming and the underlying hardware is unacceptable and there is no feasible way to reject the hardware. The only remaining alternative is to “continue bit pushing in the old way, with all the know ill effects.” Those ill effects can, however, be constrained to the smallest possible perimeter, in practice, if not in theory.”

Dijkstra's dilemma is stated as follows:

“In the past, when we used ‘low-level language’ (assembly language) it was considered to be the purpose of our programs to instruct our machines; now,

when using 'high-order language', we would like to regard it as the purpose of our machines to execute our programs. Run-time efficiency can be viewed as a mismatch between the program as stated and the machinery executing it. The difference between past and present is that, in the past, the programmer was always blamed for such a mismatch: he should have written a more efficient, more 'cunning' program! With the programming discipline acquiring some maturity, with a better understanding of what it means to write a program so that the belief in its correctness can be justified, we tend to accept such a program as 'a good program' if matching hardware is thinkable, and if, with respect to a given machine, the aforementioned mismatch then occurs, we now tend to blame that computer as ill-designed, inadequate, and unsuitable for proper usage. In such a situation, there are only a few ways out of the dilemma: (1) accept the mismatch, (2) continue bit pushing in the old way, with all the known ill-effects, and (3) reject the hardware, because it has been identified as inadequate."

Thus, the problem of having to mix assembly-language code with higher-level language code is well known. For example, an application cannot be programmed in a higher-level language if its characters must be sent to a terminal at 19.2 kilobaud; the loop required to process these characters would not be fast enough to handle the baud rate. The solution to this problem is to recode the data-communications driver in assembly language and leave the non-time-critical paths in the higher-level language.

An example of a case in which none of the final application can be programmed in a higher-level language is one in which the program must reside in 500 bytes; most programs require a library of routines larger than 1000 bytes. Therefore, the application in question can be designed in the higher-level language, but the final program must be recoded entirely in assembly language.

Method of Solution

This Note defines a methodology for programming COSMAC 1802 applications using PLM, or for programming any microprocessor application using a structured language like PLM. (It should be noted that PLM itself is not developed to the extent that it is usable in all applications of the COSMAC 1802.) The first step in the method is to design, code, and

debug the program in PLM (using structured programming techniques) with the goal of highlighting time-critical paths. In the second step, those time-critical paths that cannot perform fast enough must be recoded in assembly language using the PLM code as an overall design guide. At this point, the application program is complete. The final result of this approach is that the maximum amount of code will be written in PLM. This should be a major goal in any application in order to reduce design time, generate a maintainable program, produce good documentation, and develop the program in a structured way that will assure the highest degree of reliability.

A Practical Example

Higher-Level Language Program

The following example, Fig. 1, shows how the methodology described above can be applied to the writing of a microprocessor application. This example has a time-critical path that was first written in PLM, found to be too slow, and then rewritten in assembly language. The time-critical path is the one involving the transfer of characters from the read file to the write file; the code is found in the innermost DO WHILE loop. The purpose of the program is to combine one or more files into a single file. The multiple files that are to be combined are referred to as the READ\$FILE; the single file created is referred to as the WRITE\$FILE.

The code in Fig. 1 shows only the program's main section. The code for the associated declarations, procedures, and macros is not shown, but is straightforward and not difficult. The variable CONTINUE\$PROCESSING is a global variable that is set to false when an abort condition or a read or write error occurs, or when all programs have been combined. Assembly language statements can be added to PLM source programs by starting the statements with the characters \$A. Thus, the first two statements of the program are assembly language statements, where EJECT instructs the assembler to issue a TOP OF FORM character to the line printer and ENTRY is a macro definition that sets up the X-register to register 2 and the program counter to register 3. The macro capability in the assembler can aid considerably in writing applications in PLM for the COSMAC 1802 microprocessors. DC3 is the character that ends the source file. The constant ASCII is equal to the value assigned to a file when the file is ascii. INPUT\$FILENAME is a global variable that keeps track of whether a file is opened for reading; this variable is used mainly in the OPEN\$READ\$FILE procedure. The array FNAME contains the name of the file that is currently opened for reading. Each time a file has been

```

DO;
/* PLACE ASSOCIATED DECLARATIONS AND PROCEDURES HERE */
$AEJECT
$AENTRY
/* BEGIN MAIN LOOP */
CALL SETUP$WRITE$FILE;
DO WHILE CONTINUE$PROCESSING = TRUE;
  CALL OPEN$READ$FILE;
  CALL OPEN$WRITE$FILE;
  IF CONTINUE$PROCESSING = TRUE
  THEN
    DO;
      DO WHILE CHAR NE DC3;
        PARAMETER = ADDR (READ$IOCB);
        CHAR = GET$CHAR (PARAMETER);
        PARAMETER = ADDR (WRITE$IOCB);
        CALL PUT$CHAR (PARAMETER);
      END;
      PARAMETER = ADDR (READ$IOCB);
      CALL CLOSE (PARAMETER);
      INPUT$FILENAME = NOT$OPENED;
      PARAMETER = ADDR (FNAME);
      CALL TYPE (PARAMETER);
    END;
  END;
  IF OUTPUT$FILENAME = OPENED
  THEN
    DO;
      IF MTYPE = ASCII
      THEN CALL PLACE$DC3;
      PARAMETER = ADDR (WRITE$IOCB);
      CALL CLOSE (PARAMETER);
    END;
  END;
$ACDOS
END;
/* END MAIN LOOP */
EOF

```

Fig. 1 - The sample program coded in PLM.

combined, the name of the file, found in FNAME, is typed. OUTPUT\$FILENAME is a global variable that keeps track of whether the output file has been opened. The arrays READ\$IOCB and WRITE\$IOCB contain the input/output control block for the files being read and written, respectively. MFILE is a variable that contains the output file type.

A description of the procedures used in the example follows:

SETUP\$WRITE\$FILE: This procedure does some initialization for the setting up of the write file.

OPEN\$READ\$FILE: Before any READ\$FILE can be used, the file must be opened. This procedure takes a specific source name representing one of the files to be combined and tries to open that file on a diskette. If the open is successful, control returns to the main loop. If there is a problem, the user is informed of the specific problem and has the option of retying the file name or aborting the program and returning to CDOS. If the user types another file name, control returns to the beginning of the OPEN\$READ\$FILE procedure and checking resumes. This loop continues to be executed until a successful open or abort condition occurs.

OPEN\$WRITE\$FILE: Before WRITE\$FILE can be written to, it must also be opened. This procedure works in the same way as OPEN\$READ\$FILE except that an attempt is made to open the specified file for writing. Control remains with this procedure until a successful open or abort condition occurs.

GET\$CHAR: This procedure takes one character from the opened read file and returns it in such a way that it is stored in the BYTE variable CHAR. If any fatal errors occur while the character is being retrieved, an error message is printed and control returned to CDOS.

PUT\$CHAR: This procedure outputs the current character stored in CHAR to the WRITE\$FILE. In the program under discussion, CHAR is a global variable that can be referenced by the PUT\$CHAR procedure. This type of global definition makes it unnecessary to pass the character to output through a parameter. However, it is possible to rewrite the procedure call and the procedure so that the character can be accepted by output only through a parameter. Once again, if any fatal errors occur while the character is being retrieved, an error message is printed and control returned to CDOS.

CLOSE: The close procedure will close the file referenced by the accompanying parameter. If any errors occur during the closing function, an error message is printed and control returned to CDOS.

Two more points should be made concerning the program. First, the statement \$ACDOS, found at the end of the program, is another macro definition. PLM transforms this statement into the macro call CDOS. This macro definition is a member of the macro definition file that contains the return mechanism to CDOS. Second, the argument passed to some of the procedures is the ADDRESS variable PARAMETER. This arrangement is necessary because PLM does not allow an address to be generated in a parameter list. Therefore, the address has to be placed in a variable, and that variable used as the parameter.

Mixed Language Program

When the coding and debugging of the application under consideration were complete it was discovered, as mentioned above, that the innermost DO WHILE loop performed too slowly; the following steps have been taken to increase its speed. The DO WHILE loop has been replaced by one PLM statement \$ATRANSF, Fig. 2. From this statement the PLM compiler generates one assembly language TRANSF. TRANSF is a macro definition that contains an assembly language routine that transfers data on a sector rather than a character basis. When the output of the PLM compiler is to be assembled, the macro option of the COSMAC 1802 assembler is selected to read the macro definition file that contains the macro definition TRANSF. When the assembler processes the statement TRANSF, it substitutes the code from the previously loaded macro definition. Thus, the final application program, Fig. 2, becomes a blend of PLM and assembly language statements.

Summary

Skill in making these tradeoffs is the key to writing effective applications in PLM. The end result of the methodology advanced in this Note can be compared with the original motivations for using higher-level languages as discussed above. First, design time was reduced because a large portion of the code was written in PLM. Second, the final program is a rather easy program to maintain. Third, the PLM code can double for a portion of the design documentation because it is much easier to read and understand than a program composed only of assembly language, and it accurately reflects the logic of the program. Finally, the flow of the program is in a structured form. Note that the use of the DO WHILE statement eliminates GOTO's

```

DO;
/* PLACE ASSOCIATED DECLARATIONS
   AND PROCEDURES HERE */
$AEJECT
$AENTRY
/* BEGIN MAIN LOOP */
CALL SETUP$WRITE$FILE;
DO WHILE CONTINUE$PROCESSING = TRUE;
  CALL OPEN$READ$FILE;
  CALL OPEN$WRITE$FILE;
  IF CONTINUE$PROCESSING = TRUE
  THEN
    DO;
      $ATRANSF
      PARAMETER = ADDR (READ$IOCB);
      CALL CLOSE (PARAMETER);
      INPUT$FILENAME = NOT$OPENED;
      PARAMETER = ADDR (FNAME);
      CALL TYPE (PARAMETER);
    END;
  END;
IF OUTPUT$FILENAME = OPENED
  THEN
    DO;
      IF MTYPE = ASCII
      THEN CALL PLACES$DC3;
      PARAMETER = ADDR (WRITE$IOCB);
      CALL CLOSE (PARAMETER);
    END;
$ACDOS
END MAIN LOOP;
/* END MAIN LOOP */
EOF

```

Fig. 2 - The sample program of Fig. 1 recoded in assembly language to optimize the speed-critical loop DO WHILE of Fig. 1.

from the code; one of the main objectives in the production of structured programs is the elimination of GOTO's.

In summary, all applications can be *designed* in PLM, but not all applications, if they are to perform optimally, can be completely *written* in PLM. However, the time spent in coding the application in PLM is not wasted because most of the PLM code will be used (assembly language will be required in only small segments of the program), and the PLM code will represent the overall logic and documentation of the program, both of which contribute to its reliability.

Reference

1. "A Common Programming Language for The Department of Defense—Background and Technical Requirements," D.A. Fisher, U.S. Government Report P-1191, U.S. Government Printing Office, Washington, D.C.

Understanding and Using the CDP18U42 EPROM

by R. M. Vaccarella

As the range of microprocessor applications continues to grow, system designs not previously concerned with data storage are becoming very memory-intensive. Low power, nonvolatile data storage, especially in battery-powered systems, is becoming an important design consideration. Until recently, mask-programmable read-only memories (ROM's) provided most of the semiconductor nonvolatile data storage. However, when ROM's are used, a large production volume is required to make the system cost-effective, and program changes require a new masking operation, the latter resulting in long turn-around times. Erasable, reprogrammable ROM's, called EPROM's and designed with NMOS technology, overcame most of these problems, but the power dissipation involved still prohibited battery operation. Now, the RCA CDP18U42 EPROM, which employs the CMOS technology, provides the advantages of an NMOS EPROM while meeting the low power requirements of today's system designs.

This Note describes the design and programming characteristics of the RCA CDP18U42 nonvolatile ultraviolet-erasable/programmable read-only memory. The CDP18U42 EPROM is organized as 256 words by 8 bits (2048 bits), and features silicon-gate SOS COS/MOS circuitry that is CD4000-series and CDP1800-series compatible. Static operation and a single 4 to 6.5-volt read power supply make this device suitable for applications in which simplicity of design and low power are desirable. The CDP18U42 was designed primarily for systems in which rapid design changes and experimentation are necessary. The eight-bit data bus has tristate capability and is TTL-output compatible, allowing it to interface directly with eight-bit microprocessors. Typical applications include system breadboarding, where program instructions stored in a CDP18U42 EPROM are likely to change as design development progresses; end products in which program instructions

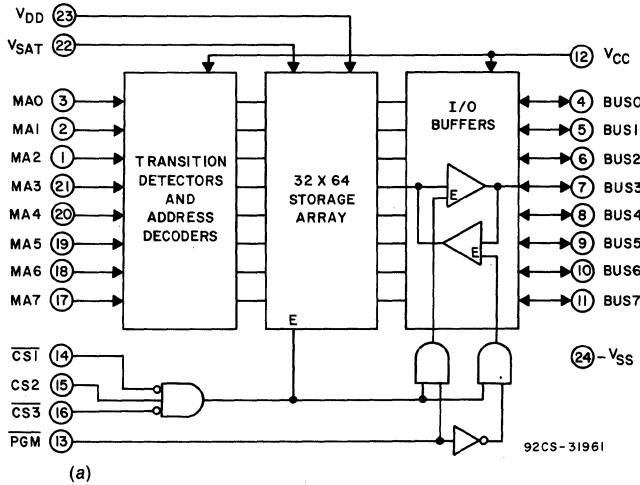
change depending on customer requirements; field-programmable products, such as point-of-sale terminals; specialized code-conversion ROM's; and field-programmable logic control.

The information presented in this Note gives the user some design insight by reviewing the technology and design parameters used to fabricate the CDP18U42 EPROM. Also included are programming circuits and system-design examples.

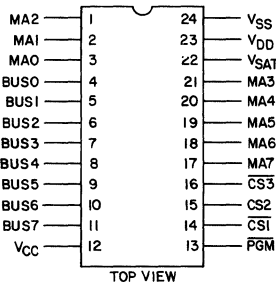
DEVICE DESCRIPTION

Fig. 1(a) is a block diagram of the CDP18U42 EPROM; Fig. 1(b) is a terminal-assignment diagram. Each of the eight address inputs contains a single-stage transition-detector circuit that provides buffering to the address decoder circuits and a trigger to an internal precharge generator. The purpose of the precharge generator is to momentarily ground all memory-cell bit lines after any address change. The storage array consists of 2048 storage cells, which are accessed by the row and column decoders. Data from the storage array is detected on the bit lines by sense amplifiers and fed to the input/output buffers. These eight I/O buffers control the flow of data to and from bus lines 0 through 7. Data direction is determined by the $\overline{\text{PGM}}$ input, which is gated with the three chip-select inputs. When the $\overline{\text{PGM}}$ input is at a high logic level, with the chip selected, data from the storage array is directed out to the bus lines (READ operation). When the $\overline{\text{PGM}}$ input is at a low logic level, with the chip selected, data from the bus lines is directed into the storage array (PROGRAM operation). When either of the three chip-select inputs ($\overline{\text{CS1}}$, $\overline{\text{CS2}}$, $\overline{\text{CS3}}$) are not true (chip de-selected), the I/O buffers are turned off and the bus lines go to a high-impedance state (tristate).

Three power supply inputs are used, although only two different voltage levels are required. During a PROGRAM operation, V_{DD} and V_{SAT} are connected to the program voltage supply (20 volts, typical) and V_{CC} is connected to 5 volts.



(a)



(b)

Fig. 1 - The CDP18U42CD Erasable/Programmable Read-Only Memory: (a) block diagram, (b) terminal assignment.

Since V_{CC} supplies all input and output devices, these logic levels (standard COS/MOS levels) remain the same for either a READ or a PROGRAM operation. During a READ operation, all supplies (V_{DD} , V_{SAT} , V_{CC}) are connected to 5 volts.

The V_{DD} supply is connected to the storage array and provides the memory-cell source voltage during READ operations and the high-voltage source bias during PROGRAM (WRITE) operations. The V_{SAT} supply is connected only to the gate inputs of the storage devices of each memory cell, thereby providing a high-impedance, low-current, capacitive-type input. During either a READ or PROGRAM operation, V_{SAT} is connected to the V_{DD} supply voltage.

CELL DESCRIPTION

The 2048 storage cells in the storage array, shown in Fig. 2, are organized as eight groups of 64 rows by four columns. Address inputs A0 through A5 are decoded to provide the 64 row-select inputs (word lines). Address inputs A6 and A7 are

decoded to provide the four column-select inputs (bit lines). Eight cells are simultaneously selected, one from each 256 cell group, to provide program data to the bus lines.

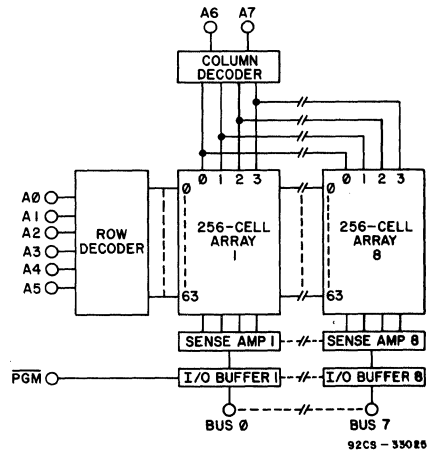


Fig. 2 - CDP18U42 storage array.

The storage cell, shown in Figs. 3(a) and (b), consists of two p-type SOS transistors. One transistor, P1, is used for word-line switching and the other, P2, for data storage. P2 is a floating-gate, avalanche-injection (FAMOS) transistor, and provides the basic storage mechanism of the CDP18U42. The primary requirement of the storage mechanism is that it provide a charge storage area for electrons such that no current path is available to remove that charge under normal read conditions or when device power is interrupted.

Writing of the FAMOS storage cell is accomplished with hot-electron injection using localized avalanche effect. The device is first selected with the word line (row address); this action causes transistor P1 to turn on and provide V_{DD} to the drain of transistor P2. Transistor N2 is also turned on by the column address, and supplies program data to the bit line from the write circuitry. If the cell is to be programmed on (logic 1), the bit line will be held low (V_{SS}). With the drain of P2 at the program voltage (V_{DD} is typically 20 volts) and the source of P2 (bit line) at V_{SS} , hot-electron injection occurs; i.e., electrons pass through the oxide to the floating polysilicon gate. Because the floating gate is isolated from everything else by the field oxide, the electrons remain at the floating gate. If the cell selected is not to be programmed on, the bit line is held high (V_{CC}) and hot-electron injection does not occur. The floating gate will then remain in its erased state (no electron storage).

Similarly, reading of the FAMOS storage cell is accomplished by selecting a particular cell with the row and column decoders, except that the write circuitry is disabled, and data from the bit line is fed to the sense amplifier. If the selected cell is programmed on (logic 1), the V_{DD} voltage (V_{DD} is 5 volts during read) is transferred from the drain of P1 to the bit line because

transistor P2 is held on by the stored electron charge on the floating gate. If the selected cell is not programmed on, P2 is off and the bit line is discharged low by the precharge cycle that occurs after every address transition, with the result that transistor N1 is turned on momentarily.

ERASING PROCEDURE

The CDP18U42 is erased by removing the stored electron charges from all floating-gate transistors in the storage array. Removal is accomplished by exposing the device, which is supplied in a package with a transparent lid, to high-intensity short-wave ultraviolet light at a wavelength of 2537 angstroms. The ultraviolet light causes the electrons stored at the floating gates to pass through the field oxide to the source and drain diffusions, with the result that the FAMOS transistors are turned off. The erase operation is a bulk erase in which all 2048 bits assume an off state (outputs are logic 0), after which the user may program a new bit pattern. (To assure reliable erasing, V_{SAT} should be held at V_{SS} during the erase-verify procedure.)

PROGRAMMING PROCEDURE

Programming (writing) is accomplished by introducing a logic 1 state in the desired bit locations, as stated in the cell description. Programming is done on a byte basis, and any previously erased bits may be selectively changed to a logic 1 state. This feature allows the user to change any logic 0 bits to logic 1 bits within a particular byte, and in a CDP18U42 already programmed, without having to do a bulk program or bulk erase, and saves considerable system-development time.

The program procedure consists of selecting a desired location with the row and column address inputs and asserting the chip-select inputs. The data to be

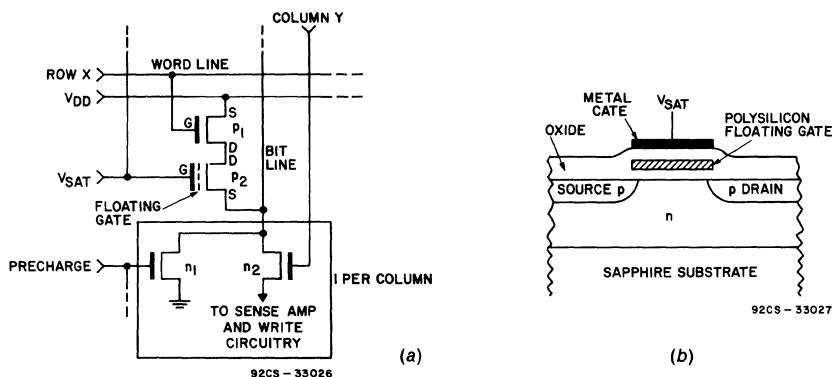


Fig. 3 - CDP18U42 storage cell: (a) schematic diagram, (b) FAMOS cell cross section.



programmed is presented at the I/O bus lines (0 through 7). With the write (PGM) input at logic 0, the VDD and VSAT inputs are pulsed to the proper program voltage (20 volts, 10 milliseconds, typical). Because of the low power required to program the CDP18U42, multiple program loops, in which each location is programmed several times, are unnecessary. This feature simplifies programmer circuitry considerably.

Table I shows the operating modes of the CDP18U42. Notice that three chip-select inputs, CS1, CS2, CS3, are available for easy system-memory expansion. The three-state output capability simplifies microprocessor interfacing, as data-bus buffer/separators are not required to solve bus contention problems.¹

at the address indicated by the address LED's, in binary, using eight toggle switches. When the advance switch is released, a timing cycle is initiated in which the CDP18U42 is put in the program mode, and the program voltage (20 volts) is applied to the VDD and VSAT terminals. At the end of this timing cycle, the address counter is incremented by one, and the CDP18U42 is ready to accept the next data byte.

When programming is complete, the CDP18U42 may be checked by placing the PGM/verify switch in the verify position. In this mode, the CDP18U42 is selected in the read mode and the program-voltage circuit is disabled. If the reset switch is depressed, each data byte may be read on the data-out LED's, starting at address zero, using the

TABLE I - OPERATING MODES FOR CDP18U42CD

OPERATING MODE	VSS (V)	VCC (V)	VDD (V)	VSAT (V)	CS1 (V)	CS2 (V)	CS3 (V)	PGM (V)	BUS 0-BUS 7
READ	0	5	VCC	VCC	VSS	VCC	VSS	VCC	OUTPUT
PROGRAM	0	5			VSS	VCC	VSS	VSS	INPUT
DESELECT	0	5	VCC	VCC	X	X	VCC	VCC	3-STATE
DESELECT	0	5	VCC	VCC	X	VSS	X	VCC	3-STATE
DESELECT	0	5	VCC	VCC	VCC	X	X	VCC	3-STATE

X=DON'T CARE

PROGRAMMER CIRCUIT OPTIONS

PROM programmer hardware and software for programming the CDP18U42 is available to RCA CDP1800-family microprocessor users. The CDP18S480 PROM programmer package can be used with the CDP18S005 Development System or the CDP18S691 Microboard Prototyping System. This programmer package performs sophisticated program and verify operations in addition to saving data on files and printing program listings. It is anticipated that commercial PROM programmer manufacturers will provide support to the CDP18U42, in the form of plug-in modules compatible with their existing equipment, for users lacking the Development or Microboard Systems facilities. However, since the CDP18U42 is relatively easy to program, the user may design and build a low-cost PROM programmer circuit to perform the basic programming functions. Two such circuits are suggested in Figs. 4 and 5.

PROM Programmer Circuit No. 1

PROM programmer circuit No. 1, Fig. 4, employs discrete COS/MOS logic devices and is intentionally limited to the basic requirements of the CDP18U42 to reduce design and construction costs. Two modes of operation are available, program and verify. In the program mode, data is entered

advance switch to sequence through the bytes.

Although the basic timing requirements of the CDP18U42 are provided by the circuit of Fig. 4, other features, such as a hex keypad circuit, may be easily added to enhance programming efficiency. This circuit can provide an effective low-cost alternative to other programming options in low-use CDP18U42 EPROM service.

PROM Programmer Circuit No. 2

PROM programmer circuit No. 2 is a more sophisticated design, based on the RCA CDP1802 microprocessor, that provides more user options. The circuit can be easily constructed in the user area of the RCA Evaluation Kit (EK) or as a stand-alone design, as shown in Fig. 5. Operation is simplified by using the standard UT4 (CPR512) or UT5 (CPR522) utility-program software. The CDP18U42 EPROM verify and program functions are implemented by utilizing the ?M and !M utility-program functions. The CDP18U42 is treated as RAM by UT4 and UT5.

The verify function (?M) is performed by locating the CDP18U42 below address space 8000₁₆. Since the utility program is defined as starting at address 8000₁₆, any read (?M) operation done below this address will read from the CDP18U42

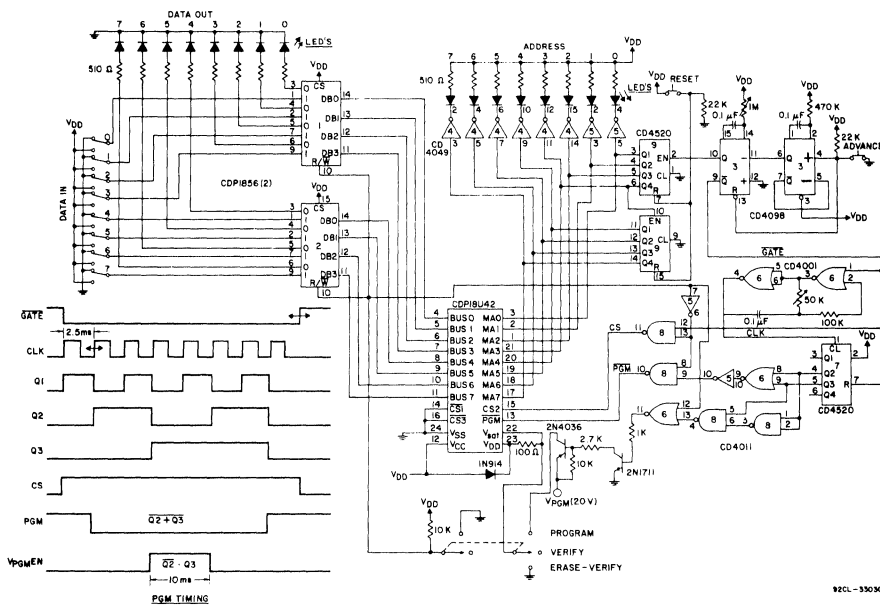


Fig. 4 - PROM programmer circuit no. 1.

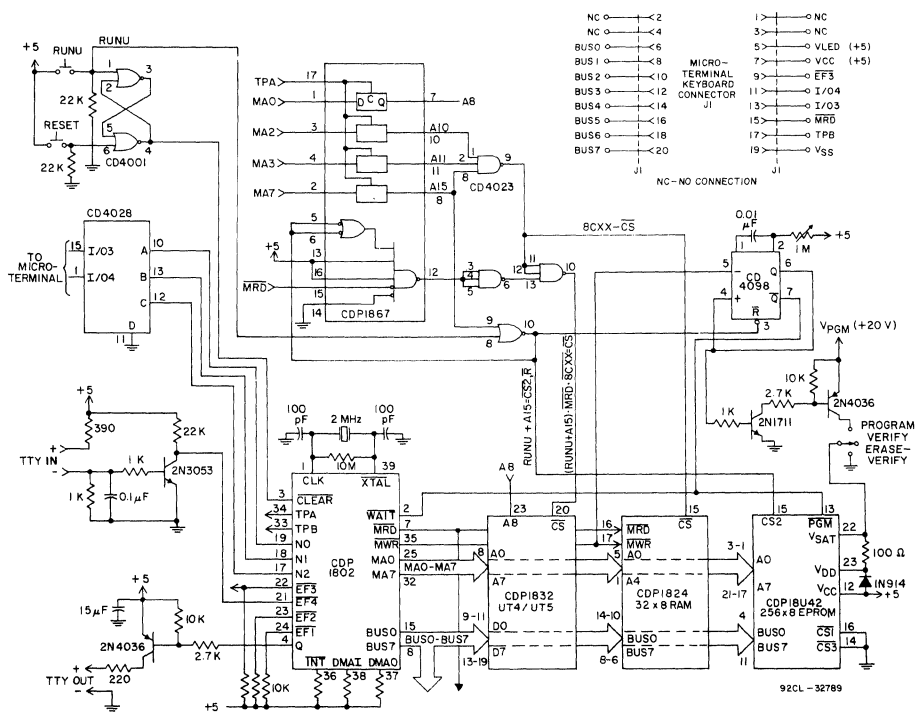


Fig. 5(a) - PROM programmer circuit no. 2, schematic diagram.

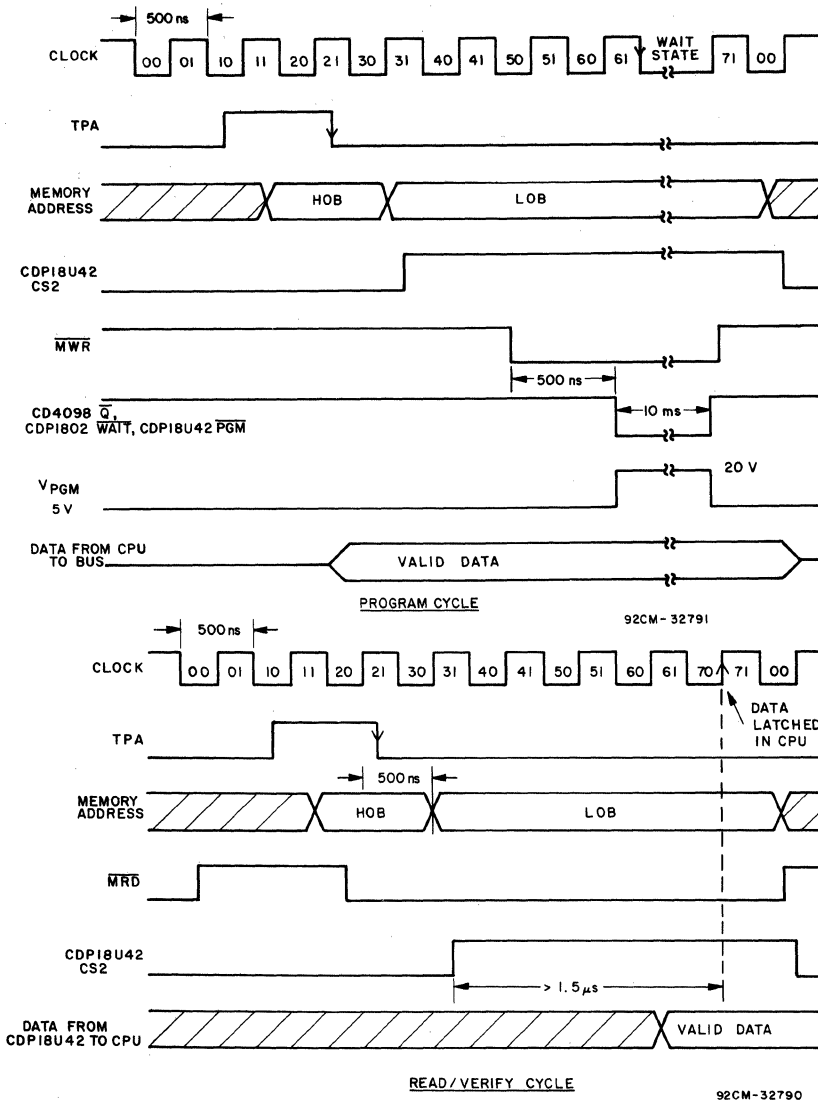


Fig. 5(b) - PROM programmer circuit no. 2, timing diagram.

EPROM. The necessary address decoding is already in place on the RCA EK and is handled by the CDP1867, CD4023, and CD4001 in the stand-alone design. When using the RCA EK, it is necessary to disable the first page of RAM.

The program function (IM) is performed by taking advantage of the wait-state feature of the CDP1802; this feature extends the CPU machine cycle to provide the 10 millisecond program time required by the CDP18U42 EPROM. When a write cycle (IM) is initiated below address 8000₁₆, the MWR output from the CDP1802 triggers the CD4098 one-shot. The one-shot Q output puts the CPU in the wait-state and the CDP18U42 in the program mode. Since the wait-state occurs late in the machine cycle, the address and data bits are held valid to

provide the required CDP18U42 programming information. The one-shot Q output is used to enable the program voltage (20 volts) when the PGM/verify switch is in the PGM position.

When this circuit is used with the RCA Microterminal, CDP18S021, data is entered from the hex keypad and displayed on the LED display. The UT5 utility program, used with the Microterminal, also contains a count subroutine that increments the address once per second; this feature is useful as an automatic verify of PROM data. The circuit also provides a serial TTY (20 milliamperes current loop) interface for use with a standard ASCII terminal (UT4 is required for this option). This mode of operation provides keyboard data entry and hard-copy printout of EPROM data.

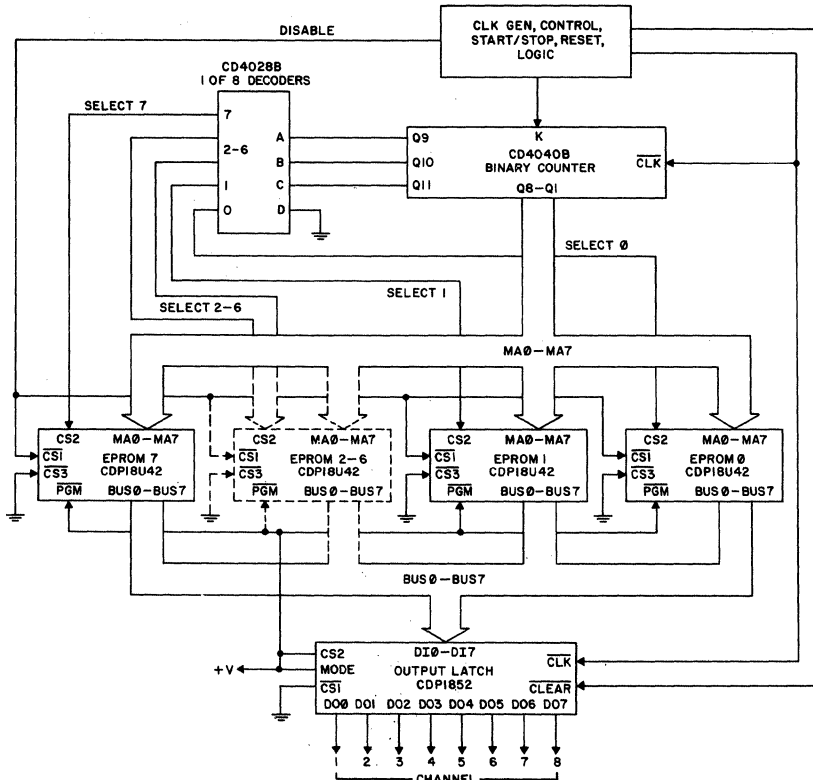


Fig. 7 - Eight-channel programmable sequence generator.

92CL - 33029

Code Converter

The CDP18U42 EPROM may be used to convert from one type of digital code to another. Since the EPROM has eight address inputs and eight data outputs, 2⁸ (256) combinations are possible. Typical code conversions include: ASCII-to-Hollerith/Hollerith-to-ASCII, ASCII-to-Baudot/Baudot-to-ASCII, ASCII-to-EBCDIC/EBCDIC-to-ASCII, ASCII-to-Selectric/Selectric-to-ASCII, and Sine/Cosine/Arctangent look-up tables (refer to the appropriate code source for the conversion tables). The conversion technique involves the programming of the memory location corresponding to the input-code address with the data that is required at the output.

REFERENCES AND BIBLIOGRAPHY

1. For specific dynamic electrical characteristics and timing waveforms: RCA CDP18U42 Objective Data Sheet, File No. 1219

Also see:

"A CMOS/SOS Electrically Alterable Read Only Memory", *Journal of Solid State Circuits*, Stewart, R., Oct. 1979

- RCA COSMAC Microprocessor Product Guide, **MPG-180**
- RCA Operator's Manual for PROM Programmer CDP18S480, **MPM-222**
- RCA CDP1802-Based PROM Programmer Circuit for the CDP18U42 EPROM, R. Vaccarella, F. Thorley, J. Stahler
- RCA COS/MOS Integrated Circuits Databook, **SSD-205**
- RCA COS/MOS Memories, Microprocessors, and Support Systems Databook, **SSD-260**

All RCA publications are available through the RCA Solid State Division.

Interfacing PLM Code to CDOS System Functions

by W. Fritchie

This Application Note defines a method for interfacing PLM programs to CDOS' system functions without the need for assembly language; the interface is an array of PLM procedures (which can be included in a PLM library) and supportive macro definitions, all of which are described in detail and used in a sample program.

Background

The system functions in CDOS are described fully in the CDOS manual.² The description shows the programmer how to interface his applications to the system functions in assembly language. Since PLM allows assembly language statements to be mixed with PLM statements, by using the \$A characters, the programmer can interface his PLM applications with the CDOS system functions in this way. However, it is more desirable to have the entire application written in PLM, as this approach leads to shorter development time and prevents the user from having to learn the mechanics of applying the system functions in assembly language. For example, if the user wanted to type a message to the console, the PLM statement would be:

```
CALL TYPE, (MESSAGE);
```

where MESSAGE is an address variable containing the pointer to the message to be typed. The equivalent code in assembly language presents some problems. As an example, it would be very difficult to determine the address of the variable message that would have to be passed to the CDOS system function that types the characters associated with the variable MESSAGE. A solution would be to declare the variable MESSAGE in assembly language and not in a PLM statement; however, this solution would have to be accomplished with care so that the message would not be positioned in the mainstream of instructions generated by the PLM compiler.

Another drawback to mixing assembly language with PLM statements is that the resulting overall program is less readable and, therefore, more difficult to maintain.

PLM is a block-oriented language in which the block is emphasized on the listing by indentation. Example 1, Fig. 1, the block format of a program written in PLM, shows how indentation is used to group the statements of a block. This example is very simple; in more complex examples, the block structure becomes much more important in understanding and maintaining the program. In Example 1, it is assumed that a procedure exists called TYPE which is contained in a library of commonly used procedures. It is easy to see where the TYPE statement appears in the overall flow of the program. If the TYPE statement were not in the correct logical sequence, it could easily be moved into the correct block.

Example 2, Fig. 2, shows assembly-language instructions mixed with PLM instructions to accomplish the same results as the program in Example 1. Example 2 also illustrates some of the problems incurred when linking CDOS system functions and PLM programs with assembly language. First, the programmer

```
DO;
DECLARE DATA (5) BYTE INITIAL ('PRINT');
DECLARE (X,Y,Z) BYTE;
DECLARE MESSAGE ADDRESS;
MESSAGE = ADDR(DATA);
X=Y;
IF X=Z
  THEN
    DO;
      Y=3;
      X=1;
    END;
  ELSE
    DO;
      X=1;
      CALL TYPE (MESSAGE);
    END;
END EXAMPLE 1;
```

Fig. 1 - Example 1.

ICAN-6928

must understand how to set up the values of UCALL and TYPE. Second, he must learn where to position the assembly-language statement for MESSAGE so as not to cause any run-time problem. Example 2, like Example 1, is straightforward; it is relatively easy to read and maintain the code. However, again, when the application becomes more complex, the readability and maintainability of the program, when assembly language code is mixed with PLM code, becomes much more of a problem. The PLM procedures and the macros described below will permit the programmer to interface his PLM program with CDOS system functions without having to use assembly language.

```
DO;
DECLARE (X, Y, Z) BYTE;
  X=Y;
  IF X=Z
    THEN
      DO;
        Y=3;
        X=1;
      END;
    ELSE
      DO;
        X=1;
      END;
$AUCALL=#B453
$ATYPE=14
$ACALL UCALL, TYPE, A (MESSAG)
  END;
$AMESSAG: 'T'PRINT'
END EXAMPLE 2;
```

Fig. 2 - Example 2.

Supportive Macros

The following interfaces between PLM statements and the CDOS system functions are discussed in detail:

- Typing a message to the console - TYPE
- Accepting input from the keyboard - CREAD
- Typing out a CDOS error message - CDERR
- Parsing a file name from an input stream into an IOCB - SRNAM
- Returning to CDOS - CDENT
- Setting up the X and P registers
- Setting up the entire PLM library as a macro

In order to perform these interfaces, the macro definitions described below and in Fig. 3 are required.

ARG and PARM1: Macro definitions ARG and PARM1 deal with passing the addresses of parameters to the assembly language calls for CDOS. For example, Example 1 declared the variable MESSAGE in a declaration statement, but CDOS must know the memory address of the variable MESSAGE. The macro definitions ARG and PARM1 assist in this linkage.

CDOS: The macro definition CDOS returns control to the operating system when the application program has finished its task.

ENTRY: The macro definition ENTRY is used to set up the X and P registers. This code generates a label called START. When the output from PLM has been assembled, the address of START must be ascertained from the listing; this address is used as the starting address when making a binary file from the ASCII-HEX file (the conversion of ASCII-HEX files to binary is done by the CDOS command CDSBIN).

EQUATE: The macro definition EQUATE contains values for the constants used in the assembly-language statements in the Code of Procedure sections described under the heading "PLM Procedures," below.

PLMMAC: The macro definition PLMMAC contains the standard PLM library in a macro-definition form. The advantage of keeping the PLM library as a macro definition rather than as a CDOS file is that the former is easier and faster to use, primarily because the MERGE command does not have to be used to combine the output of the PLM compiler with the PLM library. When the PLMMAC macro definition is used, the library is automatically combined with the output of the PLM compiler during the assembly process to produce the desired object code. This method is faster because the step involving the combination of the PLM output with the PLM library is eliminated. In addition, the problems associated with merging, such as disk-full conditions, are also eliminated.

PLM Procedures

The PLM procedures associated with the supportive macros described above are specified in the following format:

- Name
- Function
- Assumptions
- Call to Procedure
- Code of Procedure

Name: TYPE

Function: Types out a message to the console.

Assumptions: MSG\$ADR is an address variable.

MESSAGE contains the message to type.

Call to Procedure:

MSG\$ADR=ADDR (MESSAGE);

CALL TYPE (MSG\$ADR);

Code of Procedure:

TYPE: PROCEDURE (TYPE\$PARM);

DECLARE (TYPE\$PARM,X) ADDRESS;

X=TYPE\$PARM;

\$AARG 'TYPEAD'..SETTING UP CDOS

TYPE PARAMETER

\$ACALL UCALL,ZTYPE..CALL TO

CDOS

\$ATYPEAD: ORG *+2..STORAGE FOR MESSAGE ADR.

END TYPE;

Name: KEYBD

Function: Reads a record from the console into a buffer.

Assumptions: BUFFER is declared as the input buffer. BUF\$LENGTH is declared as a byte variable. BUF\$ADR is an address variable.

Call to Procedure:

```
BUF$ADR=ADDR (BUFFER);
CALL KEYBD (BUF$ADR,
  BUF$LENGTH);
```

Code of Procedure:

```
KEYBD: PROCEDURE
  KEYBUF,LGT);
  DECLARE (KEYBUF,X) ADDRESS;
  DECLARE (Y,LGT) BYTE;
  X=KEYBUF;
  $AARG 'KEYADR'
  Y=LGT;
  $APARM1 'LENGTH'
  $ACALL UCALL,KEYIN
  $AKEYADR: ORG *+2
  $ALENGTH: ORG *+1
  END KEYBD;
```

Name: PRINT\$ERROR

Function: Print out a CDOS error message.

Assumptions: BUFFER is an array with a minimum of 2 bytes; the error message number is in the second byte. MSG\$ADR is an address variable.

Call to Procedure:

```
MSG$ADR=ADDR (BUFFER);
CALL PRINT$ERROR (MSG$ADR);
```

Code of Procedure:

```
PRINT$ERROR: PROCEDURE
  (IOCB1);
  DECLARE (IOCB1,X) ADDRESS;
  X=IOCB1;
  $AARG 'ERADR'
  $ACALL UCALL,CDERR
  $AERADR: ORG *+2
  END PRINT$ERROR;
```

Name: PARSE

Function: Searches a specified input buffer for a file name and reformats the information into the appropriate area of an IOCB. PARSE is designed to help in setting up an IOCB by taking file name information from a line buffer (put there by KEYBD) and relocating it into an IOCB.

Assumptions: SRBLK is an address array with two entries. The first entry points to the address of the input buffer. The second entry points to the logical unit number byte in an IOCB.³ SRNAM-\$STATUS is a byte variable that will hold the status of the PARSE operation.⁴ MSG\$ADR is an address variable.

Call to Procedure:

```
MSG$ADR=ADDR (SRBLK);
SRNAM$STATUS=SEARCH
$FILENAME (MSG$ADR);
```

Code of Procedure:

```
PARSE: PROCEDURE (BLOCK)
  RETURNS (NAME);
  DECLARE NAME BASED (NAME+PTR)
  BYTE
  DECLARE (BLOCK,X) ADDRESS
  X=BLOCK
  $AARG (SRADR)
  $ACALL UCALL,BRNAME
  $ASRADR ORG*+2
  RETURN (NAME);
  END PARSE;
```

Name: CDOS

Function: Returns control to the CDOS operating system.

Assumptions: None

Call to Procedure:

```
CALL CDOS;
```

Code of Procedure:

```
CDOS: PROCEDURE;
  $ACDOS
  END CDOS;
```

Name: ENTRY

Function: Set up X and P registers.

Assumptions: None

Call to Procedure:

```
$AENTRY
```

Code of Procedure: None

Name: PLMMAC

Function: To automatically load the PLM library using macros.

Assumptions: None

Call to Procedure:

```
$APLMMAC
```

Code of Procedure: None

Application

The following program, Example 3, Fig. 4, called EXAMPL, shows how to use all of the above-mentioned macros and PLM procedures. The program reads in a file name from the keyboard of the console. If no file name was typed, an error message is printed and control is returned to CDOS. If a file name was typed, the input data is retyped on the console's printer. Control stays with the program and is signaled by the issuance of another prompt. The prompt character is a percent sign (%). Note that the procedures TYPE, KEYBOARD, PRINT\$ERROR, PARSE and CDOS, which comprise about 50 percent of the program, do not have to be written by the programmer. The description of the macro definition file, MACRO, is shown in Fig. 5. The CDOS commands required to compile, assemble and execute the sample program are shown in Fig. 6.

References

1. **Operator Manual for the RCA COSMAC CDP18S007**, RCA Solid State publication MPM-232.
2. See Reference 1, Chapter 6.
3. For further information on the IOCB, see Reference 1, Chapter 6, Fig. 18, pp 66.
4. See Reference 1, Chapter 6, pp 71, "IOCB Setup Aid Routine," for a discussion of status values.


```

MACRO
ARG %LABEL
ORG *-5
A.1(%LABEL)->R7.1
A.0(%LABEL)->R7.0
RF.1->@R7
INC R7
RF.0->@R7
MEND

MACRO
ENTRY
A.1(START)->R3.1
A.0(START)->R3.0
A.1(STACK)->R2.1
A.0(STACK)->R2.0
SEP R3
START: ORG *
MEND

MACRO
PARAM1 %LABEL1
ORG *-5
A.1(%LABEL1)->R7.1
A.0(%LABEL1)->R7.0
RF.0->@R7
MEND

MACRO
EQUATE
CDERR=#28
CDENT=#1E
ZTYPE=#14
SRNAME=#24
UCALL=#B453
KEYIN=#12
MEND

MACRO
CDOS
CALL UCALL,CDENT
MENT

MACRO
PLMMAC
[The Standard PLM Library]

MEND

```

Fig. 3 - Macro definitions required in PLM/ CDOS interfaces.

```

DO;
/*
DATA DECLARATIONS FOR EXAMPLE */
$AEQUATE
DECLARE PROMPT ADDRESS INITIAL (2500H);
DECLARE BUFFER (20) BYTE ;
DECLARE END$BUF BYTE INITIAL (0DH);
DECLARE BUF$LENGTH BYTE INITIAL (20);
DECLARE CONTINUE$PROCESSING BYTE INITIAL (0);
DECLARE NAME$PTR ADDRESS INITIAL (0B452H);
DECLARE TRUE BYTE INITIAL (0);
DECLARE FALSE BYTE INITIAL (1);
DECLARE (MSG$ADR,BUF$ADR) ADDRESS ;
DECLARE SRBLK(2) ADDRESS ;
DECLARE IOCB(36) BYTE ;
DECLARE (SRNAM$STATUS,I) BYTE ;
DECLARE NULL BYTE INITIAL (80H);
$AEJECT

/*
/* THE PROCEDURES TYPE, KEYBD, PRINT$ERROR, PARSE AND CDOS */
/* ARE FROM A LIBRARY OF PROCEDURES AND DO NOT HAVE TO BE */
/* WRITTEN BY THE USER */
/*
TYPE: PROCEDURE (TYPE$PARAM) ;
DECLARE (TYPE$PARAM,X) ADDRESS;
X=TYPE$PARAM;
$AARG 'TYPEAD' ..SETTING UP CDOS TYPE PARAMETER
$ACALL UCALL,ZTYPE ..CALL TO CDOS
$ATYPEAD: ORG *+2 ..STORAGE FOR MESSAGE ADR.

END TYPE;

```

Fig. 4 - Example 3, program EXAMPL.

```

KEYBD: PROCEDURE (KEYBUF,LGT);
  DECLARE (KEYBUF,X) ADDRESS;
  DECLARE (Y,LGT) BYTE;
  X=KEYBUF;
  $AARG 'KEYADR'
  Y=LGT;
  $AFARM1 'LENGTH'
  Y=LENGTH(X);
  $ACALL UCALL,KEYIN
  $AKEYADR: ORG **2
  $ALENGTH: ORG **1
END KEYBD;
PRINT$ERROR: PROCEDURE (IOCB1);
  DECLARE (IOCB1,X) ADDRESS ;
  X=IOCB1;
  $AARG 'ERADR'
  $ACALL UCALL,CDERR
  $AERADR: ORG **2
END PRINT$ERROR;
PARSE: PROCEDURE (BLOCK) RETURNS (NAME);
  DECLARE NAME BASED (NAME$PTR) BYTE ;
  DECLARE (BLOCK,X) ADDRESS ;
  X=BLOCK;
  $AARG 'SRADR'
  $ACALL UCALL,SRNAME
  $ASRADR: ORG **2
  RETURN (NAME);
END PARSE;
CDOS: PROCEDURE;
  $ACDOS
END CDOS;
$AEJECT
/*
*/

/* THIS IS THE BEGINNING OF THE MAIN PROGRAM */
/*
$ENTRY
DO WHILE CONTINUE$PROCESSING=TRUE;
  DO I=1 TO 20; /*INITIALIZE INPUT BUFFER*/
    BUFFER(I)=0;
  END;
  MSG$ADR=ADDR(PROMPT); /*OUTPUT PROMPT*/
  CALL TYPE(MSG$ADR);
  BUF$ADR=ADDR(BUFFER); /*INPUT FILENAME*/
  CALL KEYBD (BUF$ADR,BUF$LENGTH);
  SRBLK(1)=ADDR(BUFFER); /*SETUP FOR PARSING FILENAME*/
  SRBLK(2)=ADDR(IOCB)+11;
  MSG$ADR=ADDR(SRBLK);
  SRNAM$STATUS=PARSE(MSG$ADR); /*PARSE FILENAME INTO IOCB*/
  IF SRNAM$STATUS=NULL /*NULL IMPLIES NO FILENAME INPUT*/
  THEN
    DO;
      IOCB(2)=11; /*SETUP ERROR NUMBER FOR SYNTAX ERROR*/
      MSG$ADR=ADDR(IOCB);
      CALL PRINT$ERROR(MSG$ADR); /*PRINT SYNTAX ERROR MESSAGE*/
      CONTINUE$PROCESSING=FALSE; /*SET FLAG TO STOP PROGRAM*/
    END;
  ELSE
    DO;
      MSG$ADR=ADDR(BUFFER);
      CALL TYPE(MSG$ADR); /*RETYPE INPUT FILENAME*/
    END;
  END;
  CALL CDOS;
  $AFLMMAC
END PROGRAM;
EOF

```

Fig. 4 - Example 3, program EXAMPL [cont'd].

MACRO ARG
MACRO PARM1
MACRO CDOS
MACRO ENTRY
MACRO PLMMAC
MACRO EQUATES
EOM STATEMENT

Fig. 5 - Description of macro definition file.

```

PLM EXAMPL (CR)
[PLM output file is called EXAMPL.ASM]
ASM4 (CR)
TYPE SOURCE FILENAME EXAMPL.ASM (CR)
WRITE TO DSK OR PRINTER (D/P)? D
WRITE? OUTPUT (CR)
?R,B,M,L,H,U=M
MACRO READ?MACRO (CR)
?R,B,M,L,H,U=L
?R,B,M,L,H,U=U

[Now get a listing and ascertain address of label start. Assume value is
F9. Using CDSBIN, get a binary executable file.]

CDSBIN OUTPUT; F9 (CR)
(EXECUTE FILE).
OUTPUT (CR)
%
```

Fig. 6 - CDOS commands required to compile, assemble, and execute the program of Fig. 5. Underlined sections are printed by the computer.

Cassette Tape I/O For COSMAC Microprocessor Systems

C.D. Smith

This Note describes a circuit and the software needed to add a low-cost cassette-tape input and output to the COSMAC Evaluation Kit (CDP18S020, CDP18S024, and CDP 18S025), the COSMAC Development System (CDP-18S005 and CDP18S007), or the Microboard Prototyping Kit (CDP18S691).

Cassette-Tape Format

The RCA COSMAC VIP cassette-tape format was chosen because it is well documented¹ and requires only a simple software and hardware interface. Bits on the tape consist of one cycle of 2 kHz for a

logical 0 and one cycle of 800 Hz for a logical 1. Data is preceded by approximately four seconds of continuous logical 0's for sync followed by a specified number of data bytes. Each byte begins with a logical 1 start bit followed by eight data bits (least significant bit first) and ends with a parity bit; odd parity is used in this code. Fig. 1 shows typical waveforms for recorded data.

Electrical Interface

The electrical interface circuit shown in Fig. 2 is identical to that used with the COSMAC VIP.² Bit-serial data is output to U14A-Pin 2

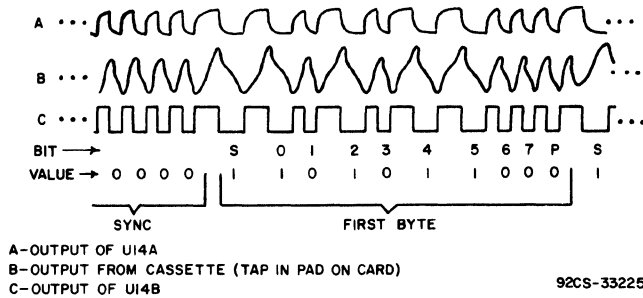


Fig.1 - Typical waveforms of recorded data.

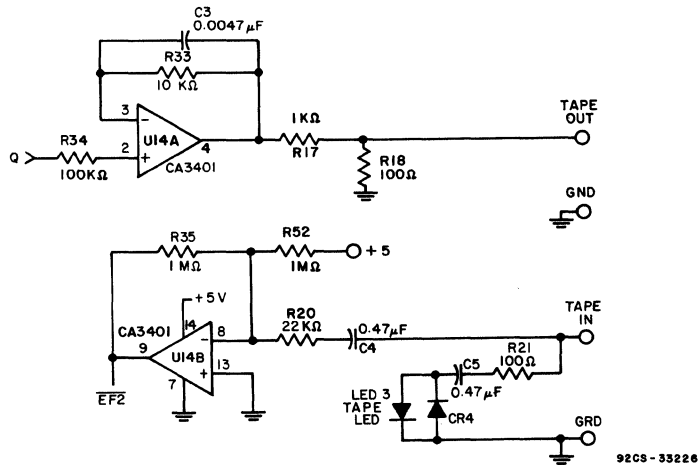


Fig.2 - Schematic diagram for COSMAC cassette tape I/O interface.

through the CDP1802 Q bit. Bit-serial data is input from a cassette recorder to U14B-Pin 8 and then to the EF2 input of the CDP1802. Thus, by providing software to control the Q output and sample the EF2 input, data is recorded and recovered from the tape.

Software Interface

Three software routines are described below. The first is an input/output routine for COSMAC systems that use the COSMAC Microterminal as an operator interface. The second is an input/output routine for COSMAC systems that use a standard RS-232 or current loop remote terminal. The third is a boot-loader routine that can be used to load the appropriate input/output routine after power up. This latter routine can be either ROM resident or entered by hand when required.

Using the Microterminal

The use of the cassette input/output routine for the Microterminal-based system requires an understanding of the Microterminal itself.³ The Microterminal system is divided into three functional sections: control, display, and keyboard.

The control section contains the requisite hardware for controlling the operation of the microprocessor system. The function keys are as follows:

R Reset: Resets the logic of the Microterminal and microprocessor system. Puts the CDP1802 in the reset state.

RU Run Utility: Starts execution of the utility program (UT5), which is at location 8000.

RP Run Program: Starts program execution at location 0000 with R0 as program counter.

CONT/STEP Slide switch to enable continuous or single-step operation of the microprocessor system.

--- Entry Mode Control: Toggles between the address-entry and data-entry modes.

INC Increment Address: Each depression increments the address shown in the display. In the data-entry mode, it also causes the data byte shown to be written to the address shown before the address is incremented.

\$P Start Address Program: Starts program execution at the location shown in the address display.

CA Clear Address: Clears (resets) the address display to 0000.

The keyboard section of the Microterminal contains 16 digit keys (0 through F) that are used to enter hexadecimal numbers into the address or data field. The destination of entered data is controlled by the Entry Mode Control toggle switch (---). The functions of the hexadecimal digit keys and some of the control function keys are encoded and sent to the microprocessor system on the bidirectional data bus. Logic is provided for scanning and signalling keyboard activity to the microprocessor. The actual scanning, debounce, and decoding algorithms are performed by software routines in the utility program.

The display section consists of an eight-digit, seven-segment, LED display, display drivers, and refresh logic. In either the data-entry or address-entry mode of operation, the display shows a four-digit address field on the left side and a two-digit data field on the right; the fields are separated by blank positions. In other subroutine-oriented display modes, all eight digits are available in two groups of four separated by blank positions. Illuminated decimal points in either the address or data field indicate the current operating mode. Software routines in the utility program perform digit selection, multiplexing, and hexadecimal-to-seven-segment code conversion.

The Microterminal cassette I/O software routine uses the displayed address as the first location to be output or input and the displayed data byte as the number of 256-byte pages to be input or output. If the Microterminal is in the data entry mode when the routine is executed, data is transferred from memory to tape; if it is in the address mode, data is transferred from tape to memory. Initially, the Microterminal is used to enter the cassette I/O routine, Fig. 3, beginning at location #0000. After entry and verification, execute the routine by depressing RESET-RUN P. Then change to the data entry mode by depressing --- and enter #01 as data. The address display should be #0000. Start the cassette in the record mode, let the tape advance beyond the nonmagnetic leader, and then depress \$P. When output is complete, the routine returns to the monitor program.

Verification of the recorded data is a two-step process:

1. Play the data back into the tape input circuit and adjust the volume to a point where the LED flickers. Then execute the

```

!M
0000 90B3B8F809A3C0810ED4801A3609D4801A3E0ED4801A94BAF822AA6C4AF33227;
0020 8AFF343309301C8AFF23AAFAF03A458CF63338D480E530098E73D4809E60F0AE;
0040 D480D630098AFA0FFC4FAA93BA4AA3535609091C30091EBEBEF800AEBFC63BAA;
0060 F898A8F80ABDD833632D9D3A66D83B6DF809ADAB9D76BD2BD88B3A741D8DF633;
0080 8A9D5F1F2E9E3A6D30EA9FBA8FAA9EBB8EABD481A63092D3F8103D9A3DA5FF01;
00A0 3A9C1DF880FE35A63097F8D6A8F840BDFC00B82D9D3AB0F810ADF808AB4FBBFF;
00C0 00D89BF6BBD82BBB3AC21D8DF6D82E9E3AB7D830EAD3F80A3BDDFB201D7B73FF;
00E0 0133DF39D57A60F030DFF8B0B0E0F800A0D0
    
```

Fig.3 - Object code for Microterminal cassette I/O.

cassette tape I/O routine and load the data into memory. Loading is accomplished by first depressing RESET, RUN, P, then →. Enter #01 as data and then depress → and enter #0100 as address. Rewind and start the cassette in the play mode; let the tape advance beyond the non-magnetic leader and then depress \$P. After loading, the routine returns to the monitor.

2. Use the Microterminal to verify the content of memory starting at location #0100. If loading is unsuccessful, repeat the loading process while making slight adjustments to the tape recorder volume. After successfully loading the data, make a note of the volume-control position for future reference.

Using a Standard Data Terminal

Using the cassette I/O routine for systems having a ROM based utility program that operates through a standard data terminal (RS-232 or 20 milliamper loop) requires an understanding of the ROM utility program.⁴ This program allows the user to examine memory, alter memory, or begin execution at any specified address.

?M (ADDRESS) (SP) (NUMBER OF BYTES) (CR)

— The specified number of bytes in memory, beginning at the specified address, will be transmitted to the data terminal.

!M (ADDRESS) (SP) (DATA) (CR)

— Data bytes consisting of two hexadecimal digits are stored in sequential memory address beginning at the specified address.

\$P (ADDRESS)

— Program execution begins at the specified address with P=0 and X=0.

Initially, the !M command is used to enter the cassette I/O routine, shown in Fig. 4, beginning at location #0000: After entry and verification, using the ?M command, execute the routine using the \$P command. The system will respond by outputting "VIP

CASSETTE I/O FOR RCA EK", followed by "?U, L, S≡". Enter U to return to the utility program, L to load memory from tape, or S to save memory on tape, all followed by CR, and the system will respond by outputting "FIRST?". Enter the starting address of the data to be output followed by a CR. The system will respond by outputting "LAST?". Enter the address of the last data byte to be output, but before entering a CR, start the tape in the record mode and let the tape advance beyond the nonmagnetic leader. Unlike the Microterminal I/O routine, which inputs and outputs data in 256-byte blocks, this program allows the user to record and load any number of bytes. However, it is recommended that data be recorded beginning at a page boundary, and that blocks of 256 bytes be output.

To record the cassette I/O program, depress RESET, RUN U, and enter \$P0000. Enter S-CR, then #0000 for the "first" address and #01FF for the "last" address. Start the tape in the record mode before entering the last CR. "XFERING" is output, and when the recording of data is complete, the program returns to its beginning.

Verification of the recorded data is accomplished in a manner similar to that used with the Microterminal: Play back the tape, adjust the volume, and execute the program using the \$P command. Enter #0200 for the "first" address and #03FF for the "last" address. Start the tape in the play mode, and when the tape advances beyond the leader, enter the last CR. Use the ?M command to verify a successful load.

Boot Strapping After Power Off

To reload either the Microterminal or the data terminal cassette I/O routine after power off, it is necessary to load only the cassette boot routine. After entering and verifying the boot routine in Fig. 5, position the tape just before the saved cassette I/O routine, and start execution by depressing RESET-RUN P, after starting the tape in the

```

!M
0000 710090FC01B8F88CB2F81FA2E290B3B4B5F81BA3F892A4F8BB6A5D314D401B10D;
0020 0A56495020434153534554544520492F4F20464F522052434120454B0D0A3F20;
0040 552C4C2C533D203F2000D4813B9FFB5332669FFB4C32729FFB553A1BF880B0F8;
0060 00A0DC12E0D014D40154331B14D400C1301B14D40154331B14D401003B1B14D4;
0080 01B10D0A4C4F4144204552524F5200301BD3C830A39673867393B683A646B346;
00A0 A330919673867393B683A646736046A393F4B33091D396B386A36072A6F0B630;
00C0 B5F82EABF840BDFC00D82D9D3AC7F810ADF808AB84AB8FF00D89BF6BBD82B8B3A;
00E0 D91D8DF6D829893ACE993ACE8D5000000000000000000000000000000000000;
0100 F842ABF80ABDD833032D9D3A06D83B0DF809ADAB9D76BD2B88B3A141D8DF633;
0120 2C9D5A1A29893A0D993A0DFED5D3F80A3B35F8201D7B8FF013337392D7A9F30;
0140 37D3F8103D443D4FF013A461DF880FE3550304114D400B10D0A46495253543F;
0160 2000F800BDADD4813B33669FFB0D3AAE9DBA8DA14D400B10D0A4C4153543F20;
0180 00F800BDADD4813B33859FFB0D3AAE9D738D7314D400B10D0A58464552494E47;
01A0 210060BAF5A9609A75B919FC00C8FF00D5DC1246BF32BCD481A430B3D5
    
```

Fig. 4 - Object code for data-terminal cassette I/O.

```

!M
0000 7100F801BAF802B990AAA9BFFB3BAFF80ABDDF330F2D9D3A12DF3B19F809ADAB;
0020 9D76BD2BDFB3A201D8DF633379D5A1A29893A19993A19C0800D0F8103D3D3D;
0040 48FF013A3F1DF880FE3549303A
    
```

Fig. 5 - Object code for VIP cassette-tape-format boot loader.

ICAN-6934

play mode. The most significant byte of the first address is in M(0003), and the number of pages to be input is in M(0006). The contents of these addresses can be changed to suit the individual user's needs.

References

1. **RCA COSMAC VIP CDP18S711 Instruction Manual**, RCA Solid State publication VIP-311.
2. See ref. 1, pp. 70, Fig. E-3-Keyboard, decoding, audio oscillator, and cassette interface circuits.
3. **Instruction Manual for RCA COSMAC Microterminal**, RCA Solid State publication MPM-212.
4. For ROM Utility Program information refer to the following RCA Solid State publications:

For Evaluation Kits:

- **Evaluation Kit Manual for the RCA CDP1802 COSMAC Microprocessor**, MPM-203.
- **Instruction Manual for the RCA COSMAC Evaluation Kit CDP18S020 and the EK/Assembler-Editor Design Kit CDP 18S024**, MPM-224.

For COSMAC Development Systems:

- **Operator Manual for the RCA COSMAC DOS Development System (CDS III) CDP18S007**, MPM-232.
- **Operator Manual for the RCA COSMAC Development System II CDP 18S005**, MPM-216.

For Microboard Prototyping Kits:

- **RCA COSMAC Microboard Prototyping System CDP18S691, CDP18S691V3, MB-691**.

DESIGNING MINIMUM/NONVOLATILE MEMORY SYSTEMS WITH CMOS STATIC RAM'S

R.M. Vaccarella

The design of memory systems to operate with the RCA CMOS CDP1802 microprocessor is very straightforward because the CDP1802 is designed to interface easily with the RAM's available in the CDP1800-series product line. With some care, a power-down system design for battery-backup applications, in which stored data is retained in RAM memory, is also easily implemented. This Note details the system considerations and circuit requirements for CDP1800-series RAM operation and data retention in CDP1802-based systems. Included are details relating to interfacing complexity as a function of memory-array size, power-distribution considerations, power-down/power-up control, and battery selections.

WHY CHOOSE CMOS RAM'S

The CMOS technology is the logical choice for the design of memory systems for power-sensitive or minimum-complexity systems. A power-sensitive system is defined as one in which a power interruption could result in the permanent loss of vital data. A minimum-complexity system is one in which a minimum number of RAM and interface devices are used to create a byte-wide microprocessor system that successfully performs the desired functions. The statement that devices fabricated by means of the CMOS technology are best suited to the design of such systems stems from the fact that CMOS RAM's have operating and standby currents considerably lower than those of other technologies, specifically, an order of magnitude lower than NMOS. This capability makes feasible battery back-up as well as total battery operation. In addition, all RCA CMOS RAM's are static devices, and have no need for the special refresh circuitry and cycle-stealing operations required with dynamic RAM systems. This static feature also enhances the attractiveness of battery-backup operation, since input lines need not be toggled to retain data.

Battery backup is becoming increasingly popular, especially in applications in which power failure in a volatile memory-based

system could result in the unrecoverable loss of vital information. Data collection instruments, point-of-sale terminals, energy management systems, and other portable equipment invariably require the assurance of nonvolatility provided by primary-battery/battery-backup operation. Backup power is effectively utilized in long-term data storage requirements as well.

The data presented in this Note indicates that a typical 180-milliampere-hour, 5-volt battery source can maintain data in a total CMOS system, including CPU, ROM, and RAM, under quiescent conditions for as long as 68 days. With a 2.5-volt battery source, data retention time is increased to 4.5 years.

MINIMUM CDP1802 SYSTEM CONFIGURATIONS

The interface requirements for CDP1800-series RAM's of various array sizes used in a minimum CDP1802-based system composed of CPU, ROM, and RAM (I/O circuitry is omitted for clarity) are illustrated in the following paragraphs. The techniques for interfacing include those with and without the need for external decoding circuitry, depending on array size, for RAM arrays of from 32 bytes to 4K bytes.

In each configuration shown, a 1Kx8 CDP1833 ROM was used for program memory. The CDP1833 has two features especially useful for minimum-system design: on-chip address latch/decode and a chip-enable output. The address latch and decode feature is used to capture the higher-order byte of the multiplexed CDP1802 address bus and to decode this byte to locate the ROM in any user-defined 1K block within the 64K addressable space of the CPU, without the use of external decoders. The other feature is a daisy-chain output that passes a CEO chip-select signal to other system devices (RAM's in this case) when the ROM is deselected. In systems in which only one RAM device is used, the CEO output is connected directly to the CS input of the RAM, enabling it whenever the CEO output is low (ROM deselected).

Figs. A-1, A-3, A-5, A-7, and A-9 (see Appendix) show minimum-system configurations for each of the RAM devices in the RCA CDP1800-series product line. Expansions of each of these configurations showing the additional RAM and interface components required are found in Figs. A-2, A-4, A-6, A-8, and A-10 (see Appendix). In each of the systems shown, the program ROM is arbitrarily mapped in the first 1K of available memory space; however, the selection of higher-order address bits to be latched and decoded for RAM chip-selects is dependent upon where the program ROM is located.

The RCA CDP1800-series family includes a variety of RAM and interface devices for memory-system design. Table I provides a selection chart for minimum-system configuration. Specific dynamic read and writing timing information can be found in the individual data sheets for each device type listed.

DESIGNING NONVOLATILE MEMORY SYSTEMS ($4V \leq V_{DD} \leq 6.5V$)

Although the simplicity of integrating large memory arrays into microprocessor systems has been clearly illustrated above, additional parameters must be considered when designing volatile systems. Specifically, in the areas of hardware and software, such factors as power isolation, voltage-loss detection, battery selection, interfacing problems, and power-down/power-up sequencing must be identified and understood if a reliable battery-backup design capable of retaining important system data is to be provided.

Since the development of appropriate software subroutines is beyond the scope of this Note, no specific examples are presented. (However, the use of such subroutines to transfer unrecoverable data to and from the RAM memory should not be minimized.) Instead, the remainder of this section concerning the design of the nonvola-

Table I — RAM Selection Chart

RAM SYSTEM SIZE (WORDS x BITS)	FIG. NO.	NO. OF DEVICES PER SYSTEM							TOTAL (QTY)	
		RAM(S) (QTY)					INTERFACE (QTY)			INCLUDES 1 CPU AND 1 ROM
		CDP1824	CDP1823	CDP1822/MWS101	CDP1821	CDP1825/MWS114	CD4556B	CDP1858		
32 x 8	A-1	1							3	
64 x 8	A-2	2					1		5	
128 x 8	A-2	4					1		7	
128 x 8	A-3		1						3	
256 x 8	A-4		2						4	
256 x 8	A-5			2					4	
512 x 8	A-6			4				1	7	
1K x 8	A-7				8				11	
1K x 8	A-9					2		1	5	
2K x 8	A-8				16			1	19	
2K x 8	A-10					4		1	7	
4K x 8	A-6			32				1	35	
4K x 8	A-8				32			1	35	
4K x 8	A-10					8		1	11	
		32 x 8	128 x 8	256 x 4	1K x 1	1K x 4	1 of 4 Decoder	4-Bit Latch/Decoder	4-Bit Latch/Decoder	

tile memory systems will deal with the important hardware considerations, and will present detailed circuits and explanations necessary for efficient system operation. The related data-sheet parameters required for successful hardware implementation are shown in Table II.

POWER ISOLATION CONSIDERATIONS

When using RAM arrays for data storage under battery-backup power, the main ac power supply (V_{DD}) must be isolated from the battery source (V_{RAM}). One simple technique involves the use of diodes, as shown in Figs. 1 and 2. The circuit in Fig. 1 is the more effective; it provides longer data stor-

age time with a lower battery-power requirement because only the memory array draws power during battery-backup operation. In operation, diode D1 prevents current flow to the battery when V_{DD} is active, and diode D2 prevents current flow to the system when V_{DD} is not present. Diode D2 should have a forward voltage drop of at least 0.5 volt so that the RAM data-sheet maximum input-voltage range ($-0.5V$ to $V_{DD} + 0.5V$) is not exceeded. The circuit of Fig. 2 overcomes this limitation at the expense of shorter storage time by providing power to the entire system during battery-backup operation.

Table II — Device Data-Sheet Specifications

DEVICE ¹ TYPE	$I_{QUIESCENT}$		$V_{IL} (MAX)$		$V_{IH} (MIN)$	
	(μA)		$V_{DD}^{(2)}$	V_{DD}	$V_{DD}^{(2)}$	V_{DD}
	TYP.	MAX.	4V	5V	4V	5V
<u>CPU</u>						
CDP1802	0.02	200	1.0	1.5	3.0	3.5
<u>ROM</u>						
CDP1833	0.02	200	1.0	1.5	3.0	3.5
<u>INTERFACE</u>						
CD4556B	0.04	150	1.0	1.5	3.0	3.5
CDP1858	5	50	1.0	1.5	3.0	3.5
CDP1866	5	50	1.0	1.5	3.0	3.5
<u>RAMS</u>						
CDP1824	250	500	1.0	1.5	3.0	3.5
CDP1823	50	500	1.0	1.5	3.0 <td 3.5	
CDP1822	50	500	1.0	1.5	3.0	3.5
CDP1821	50	500	1.0	1.5	3.0	3.5
MWS5101	100	200	1.0	1.5	3.0	3.5
MWS5101A	100	200	0.65	0.65	2.2	2.2
CDP1825	50	100	0.80	0.80	2.4	2.4
MWS5114	50	100	0.80	0.80	2.4	2.4

- 1. $T_A = -40^\circ$ to $+85^\circ C$
- 2. Extrapolated Values

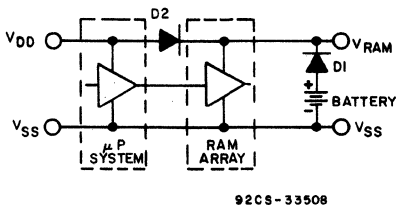


Fig. 1 - Power distribution diagram: only the memory array draws power during battery-backup operation.

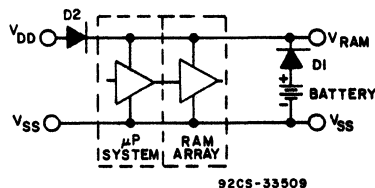


Fig. 2 - Power distribution diagram: power is provided to the entire system during battery-backup operation.

CMOS INTERFACING FOR BATTERY-BACKUP OPERATION

In the RAM-only battery-backup mode of operation, some special techniques must be utilized to terminate the CMOS RAM input lines for the following reasons. When RAM devices are used that do not have the address inputs internally gated with the chip selects (all RCA CDP1800-series RAM's except CDP1822/MWS5101), the potential for an input problem called *multiple-cell selection* exists during normal operation and, in particular, during the power-down sequence. A second input problem arises when power is removed from the rest of the system and the driver circuits connected to the RAM inputs become inactive or essentially left unterminated. The existence of this *floating-input* condition on any CMOS device can result in excessive current (I_{DD}) flow, and significantly reduce battery life.

Multiple-cell selection occurs when the input rise and fall times to the memory-device row decoders change too slowly, that is, at times greater than 5 microseconds. This slowness results in the previously selected cell remaining valid while a new cell is being selected. If the two cells selected are in the same row and contain opposite data, the cell with the weaker data will be changed to the data state of the stronger cell. Although this condition is peculiar only to the row addresses, since cells in different columns are isolated by the column sense amplifiers, it is recommended

that all address inputs be buffered to prevent timing skewing. The use of buffer devices, such as the CD4049B and CD4050B, effectively isolates any slow input transitions from the RAM array, Figs. 3, 4. The buffers are powered by V_{RAM} and provide a constant output-transition time, typically 100 nanoseconds, during normal and battery-backup operation. However, as input rise and fall times become increasingly longer, the buffers will become more susceptible to system noise because the device will remain in the transfer region for a longer period. For this reason, and depending upon the amount of system noise present, other techniques for reducing slow input transitions may be necessary.

During power-down operations, the address input rise and fall times can be especially slow because of the long RC discharge time associated with the system V_{DD} . One suggested solution to this problem is the use of a "crowbar" circuit, in which a power transistor is used to discharge the V_{DD} line to ground when a power-fail signal is detected. Another solution can be implemented in software by placing all zeros on the address bus when a power-fail signal is detected, thereby holding all address lines at ground.

The floating-input condition, a solution to which is shown in Fig. 3 for the standard B-series devices (CD4000), and Fig. 4 for the CDP1800-series interface devices, involves the use of pull-down resistors on the RAM-array input buffers. (The use of pull-

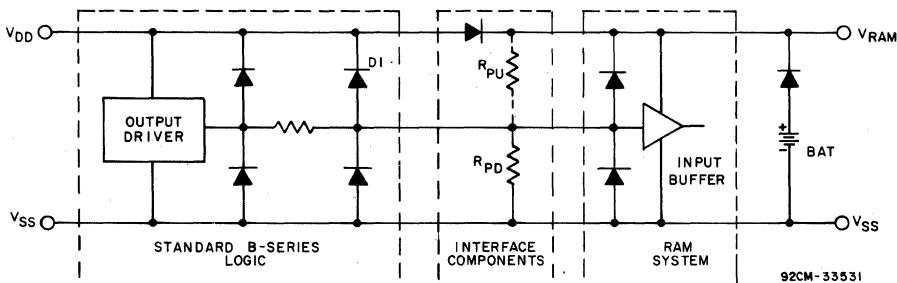


Fig. 3 - Standard B-series (CD4000) interface.

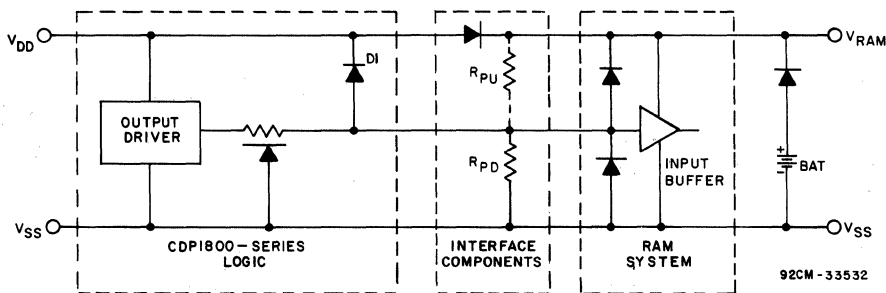


Fig. 4 - CDP1800-series interface.

up resistors is not recommended because of the current path they would create through the output protection diodes built into all CMOS devices. As the figures show, in the battery-backup mode, with the V_{DD} supply removed, the use of a pull-up resistor would result in current flow through R_{PU} and the output-protection diode D1 into the V_{DD} supply line. The results, in addition to the current draw from the battery supply, could be erratically powered-up logic circuitry and unpredictable activity on the output driver circuits.) The method used to select the pull-down resistors (R_{PD}) is illustrated in Fig. 5.

The R_{PD} (min) value is selected to guarantee that the logic 1 output level of the driver does not fall below the V_{OH} (min) for the I_{OH} (min) being sourced. The R_P resistor represents the on resistance of the p-channel transistor of the driver:

$$R_P = \frac{V_{DD} - V_{OH}}{I_{OH}}$$

The value of R_{PD} (min) is found by:

$$R_{PD} \text{ (min)} = \frac{V_{OH} \text{ (min)}}{I_{OH} \text{ (min)}}$$

Examples:

$$V_{OH} \text{ (min)} = 4.6V, I_{OH} \text{ (min)} = 1.0 \text{ mA}$$

$$R_{PD} \text{ (min)} = \frac{4.6}{1.0 \times 10^{-3}} = 4.6 \text{ kilohms}$$

The R_{PD} (max) value is selected to guarantee that the logic 0 output level of the driver does not exceed the V_{IL} (max) level of the input buffer. The R_{LK} resistor represents the equivalent resistance limiting the input leakage current (I_{IN}):

$$R_{LK} \text{ (min)} = \frac{V_{RAM} \text{ (max)}}{I_{IN} \text{ (max)}}$$

The value of R_{PD} (min) is found through the ratio of the resistance and the voltage drops:

$$R_{PD} = R_{LK} \left(\frac{V_{RAM}}{V_{LK}} - 1 \right)$$

Example:

$$V_{RAM} = 5.0V, V_{IL} \text{ (max)} = 1.0V, V_{LK} = 4.0V, I_{IN} \text{ (max)} = 1.0 \mu A$$

$$R_{LK} = \frac{V_{RAM}}{I_{IN}} = \frac{5}{1.0 \times 10^{-6}} = 5 \text{ megohms}$$

$$R_{PD} \text{ (max)} = 5 \times 10^6 \left(\frac{5}{4} - 1 \right) = 1.25 \text{ megohms}$$

The range of values within which R_{PD} should be selected, therefore, is 4.6 kilohms to 1.15 megohms. Total system requirements will ultimately determine the exact value required.

POWER-DOWN/POWER-UP CONTROL

Loss of power in a RAM-based microprocessor system can occur for two reasons: scheduled and unscheduled power interrupts. Both interrupts can result in the loss of valuable data unless certain design safeguards and power-down/power-up procedures are observed. The scheduled power interrupt is relatively easy to deal with because the system operator is aware of a pending power shut-down and can save important data or transfer operations to the battery-backup mode in an orderly fashion. The unscheduled power interrupt, however, requires the system to react automatically in response to power-supply control, software management, and memory control.

Power Supply Control

The power supply control function must:

- Detect the loss of ac power, or that V_{DD} has decreased below an acceptable level,
- Generate a power-fail signal to interrupt CPU operation,
- Hold dc voltage after an ac power loss (i.e., with hold-up capacitors) long enough for the CPU to capture and store all data and register and status information needed to effect a restart operation after power is restored.

Software Management

The software management function must:

- Recognize the power-fail signal as the highest priority, unmaskable interrupt,
- Store all necessary information and shut-down peripheral I/O devices.

Memory Control

The memory control function must:

- Deselect the memory array in a manner that assures data integrity and prevents invalid write operations,

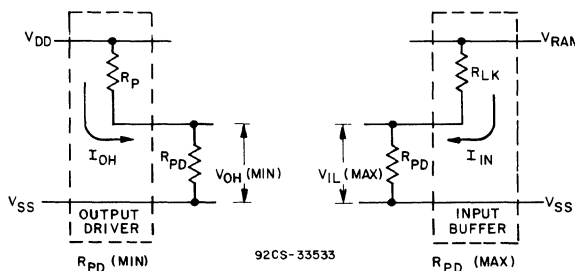


Fig. 5 - Pull-down-resistor selection methods.

- Isolate data outputs such that undesirable current paths are eliminated.

As an option, it may be important to store the fact that a power failure caused the interrupt, so that the power-up subroutine required to re-establish the last CPU machine state can be selected upon restoration of power.

POWER-LOSS DETECTION

Some advanced warning in the event of an unscheduled ac power loss is imperative. Enough time must be provided to allow shutdown of normal system operations at a convenient breakpoint in the program. A circuit capable of such early-warning is shown in Fig. 6.

To prevent interference with the regulation of the V_{DD} supply, the power-loss detection circuit is powered from a separate winding of the main power transformer. The bridge rectifier (BR2), current-limiting resistor (R_S), and zener diode (D_Z) form the basic power-loss detection circuit; their outputs provide a reference input level to the schmitt-trigger gate. C_T and R_T are selected to provide an RC time constant long enough to prevent system shutdown if the loss of ac power occurs for a few cycles only. With a more lengthy power loss than this RC time constant, the input of the schmitt-trigger gate will decay below the trigger threshold and cause the gate output to issue a power-loss signal. Since the out-

put of the V_{DD} supply bridge rectifier (BR1) is followed by a large filter capacitor (C_F), and since the V_{DD} line itself usually has some distributed capacitance (C_D), the V_{DD} supply will remain stable for a long enough time after the power-loss signal is issued to effect an orderly system shutdown.

VOLTAGE-SAG DETECTION

In addition to preparing for a total power-loss condition, some precautions must be taken to assure that V_{DD} does not fall below an acceptable system operating voltage (4.0V (min) in this case). These precautions take the form of the voltage-sag-detector circuit shown in Fig. 7, which constantly monitors the V_{RAM} voltage and compares it to the acceptable minimum-system voltage. The circuit uses an RCA CA3078 Micropower Operational Amplifier or an RCA CA3130 BiMOS Operational Amplifier configured as a voltage comparator and a minimum of external components; the CA3130 has the advantage of FET input and output buffers, which allow the output to swing to either supply rail. Although the CA3078 output voltage is offset by 0.5 volt for both high and low output levels, it presents no problem to the input buffers on RCA RAM's. Moreover, both of these types have been found to operate successfully at voltages as low as 2 volts, the specified data-retention voltage for the RCA RAM's. R_4 in Fig. 7 is adjusted to set V_{REF} to a value equal to or

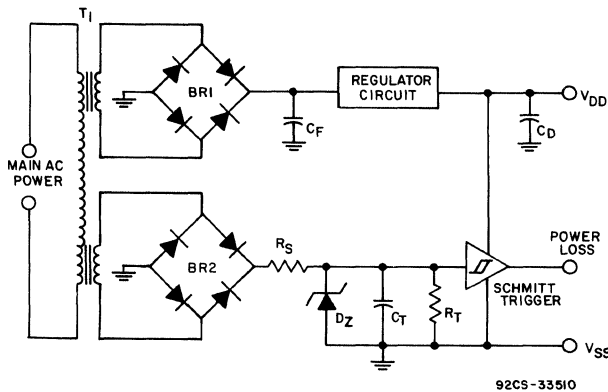


Fig. 6 - Power-loss detection circuit.

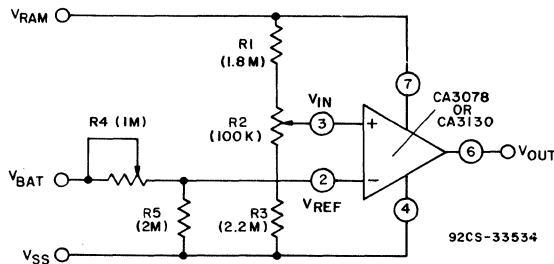


Fig. 7 - Voltage-sag detector circuit.

less than V_{RAM} (min). With V_{RAM} set to the value that the comparator is to sense (at least 4 volts), R2 is adjusted to set V_{IN} equal to or slightly greater than V_{REF} . Any further reduction in the V_{RAM} voltage as a result of system voltage sag will cause V_{OUT} to be changed from high to low. (If a change in V_{OUT} from low to high is desired, terminals 2 and 3 are reversed). This output is used to deselect the RAM array.

The combination of the power-loss detection circuit used to give advanced warning of an ac power failure to provide time to store vital information, and the voltage-sag detector circuit used to place the RAM array on battery-backup operation, provide a virtually failsafe power-interrupt warning system.

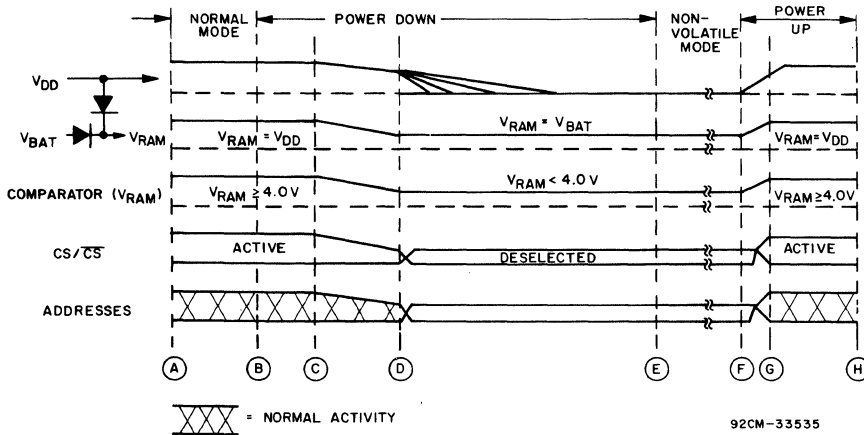
POWER-DOWN/POWER-UP SEQUENCES

The power-loss and voltage-sag detector circuits are utilized to provide time for the system to respond to supply-voltage problems. During this time, the RAM array must be properly deselected to assure reliable data storage. The timing diagram illustrates the flow of a typical power-down/power-up sequence, Fig. 8.

INTERFACING FOR POWER-DOWN DESELECTION

The RAM's in the RCA product line are designed with a variety of control inputs and from one to four chip selects, with active high and active low input levels, along with read, write, and output disable inputs. The majority of RAM devices offered interface easily in complex systems, but a few present a challenge to the system designer. Some single and multiple chip-select interface-circuit designs are illustrated and described below.

RAM's with more than one chip-select simplify system design because one of the chip-select inputs can be dedicated to the voltage-sag detector, as shown in Fig. 9. The remaining chip selects are available for other system functions, such as memory mapping. RAM's with only one chip select require that this control input be shared with the other system functions. A circuit using this technique is shown in Fig. 10. The RAM-array chip-select input (\overline{CS}) is controlled by either the main system, through a buffer/driver, or by the voltage-sag detector, which has unconditional con-



POWER-DOWN:

- A → B Normal operation, system V_{DD} held up by ac line voltage.
- B ac power failure detected.
- B → C System V_{DD} held up by filter and distributed capacitance. Start of POWER-DOWN subroutines (store data, CPU register and status information, etc.).
- C → D System V_{DD} begins to sag.
- D Voltage Sag Detector senses V_{RAM} falling below 4.0 V and forces appropriate RAM chip-select buffers to the inactive (deselected) state.
- D → E RAM array is held up by batteries. V_{DD} decays to 0 V. ($V_{RAM}=V_{BAT}$). RAM array is unconditionally deselected; address inputs are held stable by buffers with pull-down input resistors.
- E → F RAM array in non-volatile battery back-up mode. All vital data stored.

POWER-UP:

- F → G ac power restored. V_{DD} begins to increase to normal operating voltage. Voltage Sag Detector senses V_{RAM} rising above 4.0 V and enables the RAM chip-select buffers (active state).
- G → H System V_{DD} held up by ac line voltage. $V_{RAM}=V_{DD}$. Start of POWER-UP subroutines (load data, CPU register and status information etc.). Normal operation resumes.

Fig. 8 - Power-down/power-up sequences.

trol during battery-backup operation. The diode in series with the voltage-sag detector output should have a forward voltage drop of no more than 0.5 volt to guarantee a V_{IH} (min) value of at least 3 volts at the chip-select input with a V_{RAM} of 4 volts. The 3-kilohm series limiting resistor is necessary to guarantee that the V_{IH} (min) value will be maintained by the voltage-sag detector in the event that the buffer driver produces a random-logic 0 (0V) as the V_{DD} supply voltage decays during the power-down sequence. Although this resistor, R_s , creates an RC time constant with the distributed chip-select line capacitance, C_D , system performance is not seriously affected. With a 50-picofarad simulated load capacitor, the RAM chip-select rise and fall times for the circuit shown were found to be 200 nanoseconds, well within the maximum speed capability of the RCA CDP1802 microprocessor: 2.5 MHz at a V_{DD} of 5 volts.

With RAM arrays that require the use of address decoders (CD4556B) or address latch and decoders (CDP1858, CDP1866), it is convenient to use the enable input, \bar{E} , \overline{CE} , for power-down control. As shown in Fig. 11, the decoder/buffer is powered from V_{RAM} to provide active deselection during battery-backup operation; the extra battery drain, however, is minimal. This configuration is also a good alternative for single

chip-select interfaces, as the decoder outputs are unconditionally deselected by the enable input.

In ROM/RAM-based systems that employ the RCA CDP1831 or CDP1833 ROM's, the \overline{CEO} line is normally connected to the RAM-array chip select or the decoder/driver enable input to deselect the RAM's whenever the ROM is enabled. This arrangement does not present a problem during power-down, however, since the ROM's also have a chip-enable input, CEI, that operates in a feedthrough daisy-chain fashion. The voltage-sag detector is connected to the CEI input, as shown in Fig. 12, to provide unconditional control of the RAM array for power-down deselection. This circuit is especially suitable for use with the CD4556B and CDP1858 decoder/buffer devices, which have only one enable input, \bar{E} .

Whichever system configuration is used, the requirement that all RAM array inputs be at stable dc levels, that is, address and data inputs at V_{DD} or V_{SS} , read and write inputs in the inactive state, and all inputs (OD, CS, etc.) at V_{DD} or V_{SS} , prior to deselection and power-down, cannot be over-emphasized. Only through conscientious observation of the precautions described can users be assured, with maximum confidence, that system data will be retained.

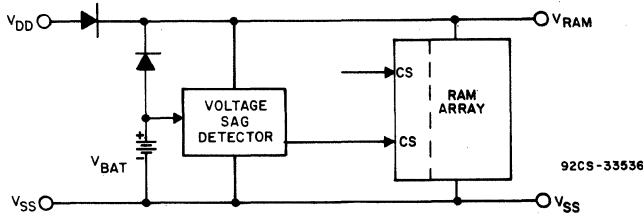


Fig. 9 - Multiple chip-select RAM interface.

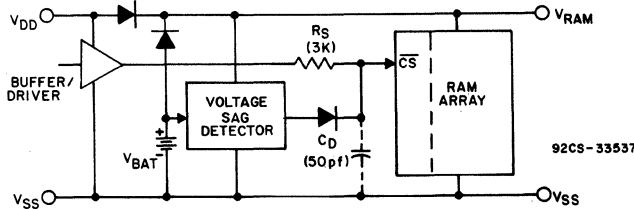


Fig. 10 - Single chip-select RAM interface.

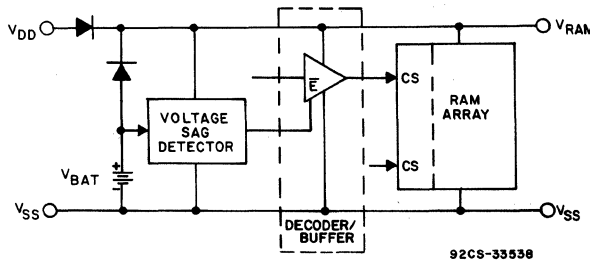


Fig. 11 - Single or multiple chip-select RAM interface.

BATTERY SELECTION

The choice of a battery for a nonvolatile battery-backup system application involves the examination of a variety of factors established by system requirements and available battery types. Among some of the more important considerations are:

- System load-current requirements
- Full-charge battery voltage
- End-of-life battery voltage
- Physical characteristics of battery (size, weight, shape)
- Battery performance in the system environment (operating and nonoperating)
- Battery life under worst-case load conditions
- System operating schedule
- Battery performance over system temperature range (operating and nonoperating)
- Primary type (nonrechargeable) or secondary type (rechargeable)

The factor most often overlooked in the choice of battery is the effect of temperature on battery performance. Generally, batteries do not perform as well at their rated temperature extremes as they do at 25° C.

The system operating schedule should play a major role in battery selection. If the battery is only expected to retain data during ac power losses, which are assumed to occur rarely, a primary-type (nonrechargeable and inexpensive) battery, which should be replaced at regular intervals (i.e., once a year) to guarantee full charge when needed, should satisfy the system require-

ments. On the other hand, if frequent power interrupts are anticipated (either scheduled or unscheduled), with long-term data retention required, a secondary-type (rechargeable) battery is recommended. Use of the rechargeable battery will require a slight modification of the power-isolation circuit to accommodate a trickle-charge resistor, Fig. 13.

Because of changing battery technology and the large assortment of battery types offered, it is recommended that the system designer obtain detailed specifications from the battery manufacturer so that he can fully evaluate the product offered with respect to his particular system. Battery cost, size, discharge characteristics, efficiency, and capacity vary widely and should be considered at an early stage in the design.

BATTERY LIFE IN RCA RAM SYSTEMS

Because of the low quiescent power dissipation of an RCA RAM, the life expectancy of a battery used with such a device can range from days to months depending on the battery type and system complexity. Table III shows a comparison of typical and worst-case battery-life expectancy for various RAM systems. The table is based on the use of a typical nickel-cadmium, 180 milliampere-hour, AAA battery available from a number of suppliers. The quiescent current (I_0) values used to determine the total current and battery life can be found in Table II. As an example, the typical battery life expectancy for a CPU, ROM, and 1K-byte RAM system using RCA MWS5114 1Kx4 RAM's is 71 days. With only the RAM array on battery backup, the battery life increases to 75 days.

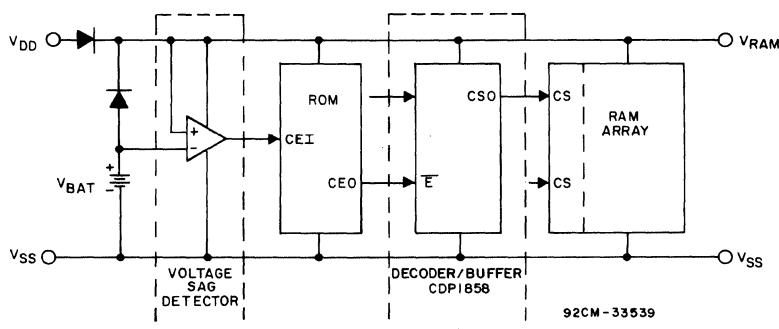


Fig. 12 - ROM/RAM chip-select interface.

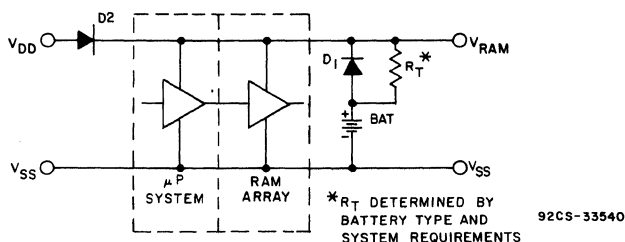


Fig. 13 - Trickle-charge power-distribution modification.

Table III — Battery-Life Expectancy Chart

RAM SYSTEM SIZE (WORDS x BITS)	FIG. NO.	NO. OF DEVICES PER SYSTEM						TYPICAL*				WORST CASE*							
		RAM(S) (QTY)			INTERFACE (QTY)		TOTAL (QTY)	TOTAL I _b (MA)		BATTERY LIFE (DAYS)		TOTAL I _b (MA)		BATTERY LIFE (DAYS)					
		CDP1824	CDP1823	CDP1822/ MW55101/A	CDP1821	CDP1825/ MW55114	CD4556B	CDP1858	CDP1866	INCLUDES 1 CPU AND 1 ROM	INTER- FACE CPU ROM	— — —	INTER- FACE CPU ROM	— — —	INTER- FACE CPU ROM	— — —	INTER- FACE CPU ROM	— — —	
32 x 8	A-1	1							3	0.25	0.25	30	30	0.90	0.50	8.3	15		
64 x 8	A-2	2					1		5	0.50	0.50	15	15	1.40	1.00	5.4	7.5		
128 x 8	A-2	4					1		7	1.00	1.00	7.5	7.5	2.55	2.00	2.9	3.8		
128 x 8	A-3		1						3	0.05	0.05	150	150	0.90	0.50	8.3	15		
256 x 8	A-4		2						4	0.10	0.10	75	75	1.40	1.00	5.4	7.5		
256 x 8	A-5			2					4	0.10	0.10	75	75	1.40	1.00	5.4	7.5		
512 x 8	A-6			4				1	7	0.205	0.20	37	38	2.45	2.00	3.1	3.8		
1K x 8	A-7				8			1	11	0.405	0.40	18.5	18.8	4.45	4.00	1.7	1.9		
1K x 8	A-9					2			5	0.105	0.10	71	75	0.65	0.20	12	38		
2K x 8	A-8				16			1	19	0.805	0.800	9.3	9.4	8.45	8.00	0.89	0.94		
2K x 8	A-10					4		1	7	0.205	0.200	37	38	0.95	0.40	7.9	19		
4K x 8	A-6			32				1	35	1.605	1.600	4.67	4.68	16.45	16.0	0.45	0.47		
4K x 8	A-8				32			1	35	1.605	1.600	4.67	4.68	16.45	16.0	0.45	0.47		
4K x 8	A-10					8			11	0.405	0.400	18.5	18.8	1.25	0.80	6.0	9.4		
				32 x 8															
				128 x 8															
				256 x 4															
				1K x 1															
				1K x 4															
				1 of 4 Decoder															
				4-Bit Latch/Decoder															
				4-Bit Latch/Decoder															
										T _A = 25°C V _{DD} = 5.0V				T _A = + 85°C V _{DD} = 5.0V					

*Battery Type=Nickel-Cadmium, 180 mAh, AAA

DESIGNING LOW-VOLTAGE NONVOLATILE MEMORY SYSTEMS (2.5V ≤ V_{DD} ≤ 4.0V)

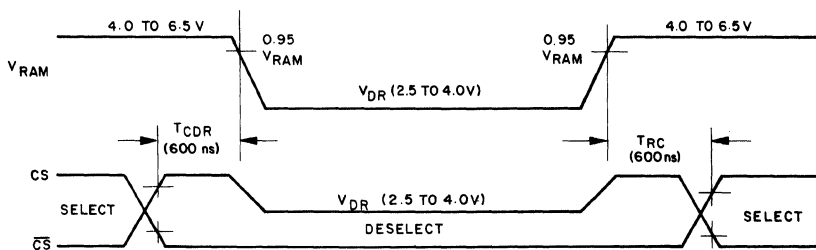
The RAM's in the CDP1800-series product line, which are designed to operate at voltages as low as 4 volts, also feature a low-voltage data-retention mode. Although normal read and write operations cannot be performed in this mode, its usefulness under battery-backup operation is evident.

All of the advantages and hardware/software considerations discussed above for 4 to 6.5-volt operation also apply in the 2.5 to 4-volt range. In addition, at 2.5 to 4 volts, battery life is extended, battery size is reduced, and system cost is lowered. In both higher and lower-voltage systems, battery cells are cascaded to create the required voltage; in a 2.5 to 4-volt system, however, adequate power can be maintained with at least one less cell.

The device specifications that are important for a successful low-voltage design are listed in Table IV. Although all of the values shown are not currently on the device data sheets, the devices have been found to work successfully to the levels indicated. Table V shows a comparison of typical and worst-case battery-life expectancy for various low-voltage RAM systems.

LOW-VOLTAGE DESELECTION

A data-sheet specification that warrants further examination in low-voltage system design is the deselection sequence. It is critical that the RAM array be fully deselected before the V_{RAM} level is lowered below the operating level (4.0V to 6.5V). The low-voltage data-retention timing diagram found in the data sheets is used in this process; Fig. 14 is a typical example of this diagram.



92CM-33541

Fig. 14 - Low-voltage data-retention timing diagram.

Table IV — Low-Voltage Data-Retention Specifications

DEVICE* TYPE	I _{DR} (μ A)		V _{DR} (VOLTS)	V _{IH} (VOLTS)	V _{IL} (VOLTS)
	TYP	MAX	MIN	MIN	MAX
CPU					
CDP1802	0.5	5.0	2.4	(2.0)	(0.5)
ROM					
CDP1833	(0.5)	(5.0)	(2.5)	(2.0)	(0.5)
INTERFACE					
CD4556B	(0.5)	(5.0)	(2.5)	(2.0)	(0.5)
CDP1858	(0.5)	(5.0)	(2.5)	(2.0)	(0.5)
CDP1866	(0.5)	(5.0)	(2.5)	(2.0)	(0.5)
RAMS					
CDP1824	(15)	50	2.5	(2.0)	(0.5)
CDP1823	(30)	50	2.0	(2.0)	(0.5)
CDP1822	30	100	2.0	(2.0)	(0.5)
CDP1821	(30)	(100)	(2.0)	(2.0)	(0.5)
MWS5101	2	15	2.0	(2.0)	(0.5)
MWS5101A	2	15	2.0	(2.0)	(0.5)
CDP1825	1	15	2.0	(2.0)	(0.5)
MWS5114	1	15	2.0	(2.0)	(0.5)
X	T _A = 25° C	T _A = 70° C	X	V _{DR} = 2.5V	

*Values in () not presently published in Device Data Sheets.

Table V — Low-Voltage Data-Retention Battery-Life Expectancy Chart

RAM SYSTEM SIZE (WORDS x BITS)	FIG. NO.	NO. OF DEVICES PER SYSTEM						TYPICAL*				WORST CASE*					
		RAM(S) (QTY)		INTERFACE (QTY)		TOTAL (QTY)	TOTAL I _o (MA)		BATTERY LIFE (DAYS)		TOTAL I _o (MA)		BATTERY LIFE (DAYS)				
		CDP1824	CDP1823/ MWS5101/A	CDP1822/ MWS5101/A	CDP1821	CDP1825/ MWS5114	CD4556B	CDP1858	CDP1866	INCLUDES	INTER-FACE CPU ROM RAM	INTER-FACE CPU ROM RAM	INTER-FACE CPU ROM RAM	INTER-FACE CPU ROM RAM			
32 x 8	A-1	1							3	0.016	0.015	469	500	0.060	0.050	125	150
64 x 8	A-2	2				1			5	0.032	0.030	234	250	0.115	0.100	65.2	75
128 x 8	A-2	4				1			7	0.062	0.060	121	125	0.215	0.200	34.9	37.5
128 x 8	A-3		1						3	0.016	0.015	469	500	0.060	0.050	125	150
256 x 8	A-4		2						4	0.031	0.030	242	250	0.110	0.100	68.2	75
256 x 8	A-5			2					4	0.061	0.060	123	125	0.210	0.200	35.7	37.5
512 x 8	A-6			4			1		7	0.122	0.120	61.5	62.5	0.415	0.400	18.1	18.8
1K x 8	A-7				8			1	11	0.242	0.240	31.0	31.3	0.815	0.800	9.20	9.38
1K x 8	A-9					2		1	5	0.004	0.002	1875	3750	0.045	0.030	167	250
2K x 8	A-8				16			1	19	0.482	0.480	15.6	15.6	1.615	1.60	4.64	4.69
2K x 8	A-10					4		1	7	0.006	0.004	1250	1875	0.075	0.060	100	125
4K x 8	A-6			32			1		35	0.962	0.960	7.79	7.8	3.215	3.20	2.33	2.34
4K x 8	A-8				32			1	35	0.962	0.960	7.79	7.8	3.215	3.20	2.33	2.34
4K x 8	A-10					8		1	11	0.010	0.008	750	938	0.135	0.120	55.6	62.5
X		32 x 8	128 x 8	256 x 4	1K x 1	1K x 4	1 of 4 Decoder	4-Bit Latch/Decoder	8-Bit Latch/Decoder	X				T _A = 25°C V _{DR} = 2.5V		T _A = 70°C V _{DR} = 2.5V	

*Battery Type — NiCad, 180 mA/H, AAA

CONCLUSION

As has been demonstrated in this Note, the use of RCA CMOS static RAM's is a practical design alternative to other types of non-volatile data storage. In applications in which the storage or transfer of data in remote systems is not possible by conventional means, magnetic tape or disk, the design techniques shown will prove extremely useful.

Figs. 15, 16, and 17 show the logic and parts used in the RCA CDP18S622 8-Kilobyte Battery-Backup RAM Card, a unit that can be used to effectively summarize and tie together the design techniques discussed above. This 4.5-by-7.5 inch PC board is part of the RCA COSMAC Microboard Microcomputer product line; it contains 8192 bytes of RAM (16 MWS5114, 1024x4) and all buffering and decoding needed for inter-

facing with the RCA Microboard and RCA CDS system backplanes. Data retention, using three 180 milliampere-hour, nickel-cadmium, AAA batteries, is 96 hours minimum. The system is normally powered from the Microboard backplane through V_{DD}, but it is also provided with an optional on-board rectifier/regulator circuit (VR₁) that can be used to power the RAM board and/or the entire system through V_{REG}. Power distribution and isolation are handled by CR1, CR2, CR3, and CR4, with B1, B2, and B3 providing battery-backup power to V_B, and R₉ providing a trickle-charge to the batteries.

The power-loss detection circuit, consisting of operational amplifier U25, provides an unconditional deselect signal through the bank-select decoding circuitry (U20, U21, U18) to the address and chip-select

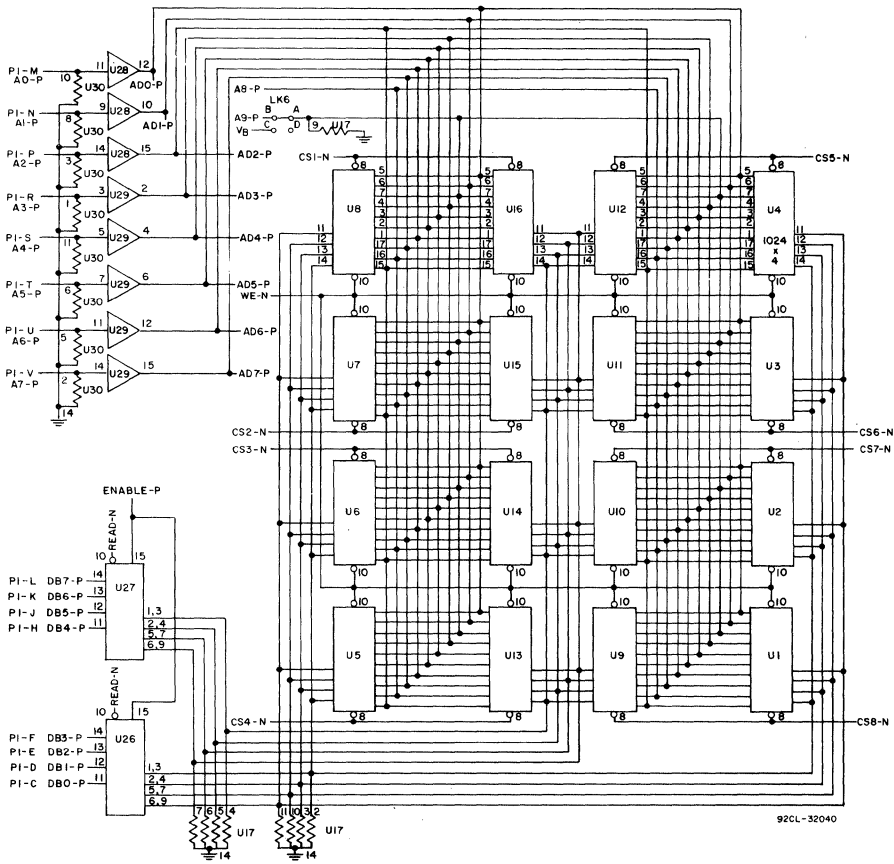
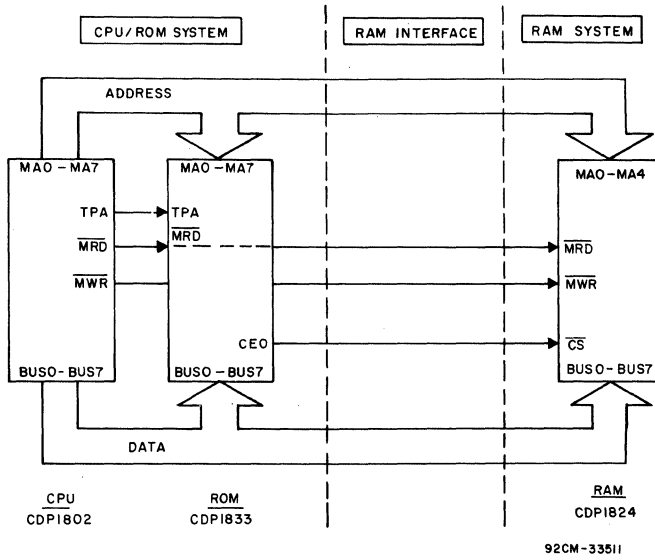
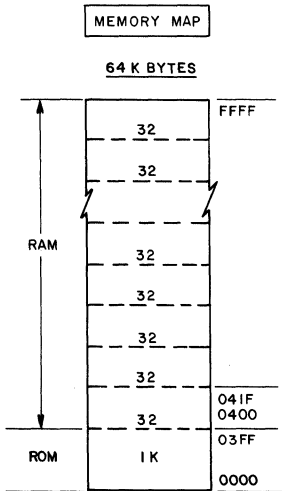


Fig. 16 - Logic diagram for RCA COSMAC Microboard Battery-Backup RAM CDP18S622 — memory and buffer portion.

APPENDIX



92CM-33511



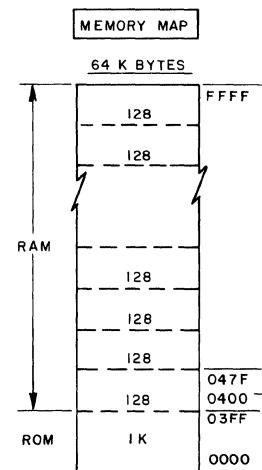
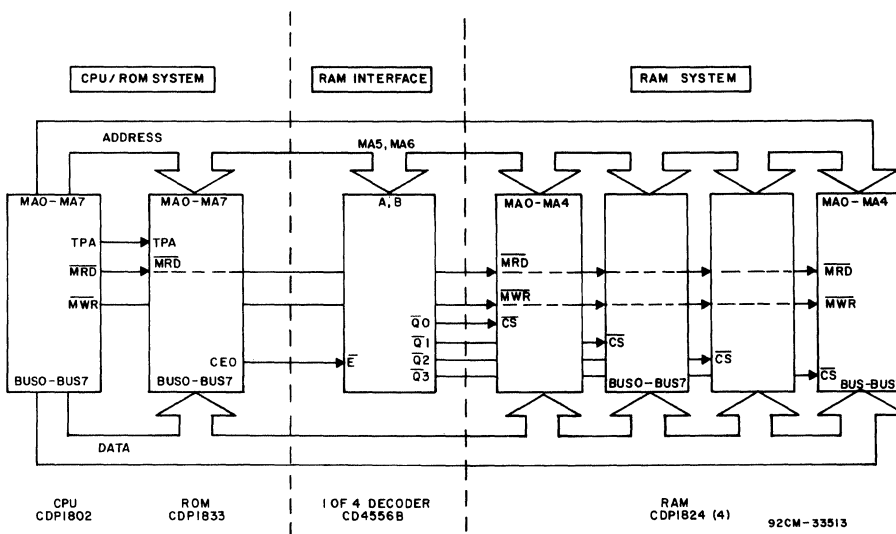
NOTE:
THIRTY-TWO BYTES OF RAM MAY
RESIDE IN ANY 32-BYTE BLOCK
BETWEEN 0400 AND FFFF.

RAM INTERFACE (PURPOSE)

- NONE REQUIRED.

92CS-33512

Fig. A-1 - CDP1824 (32 x 8) minimum system (32 x 8).



NOTE:
128 BYTES OF RAM MAY RESIDE IN ANY 128-BYTE BLOCK BETWEEN 0400 AND FFFF.

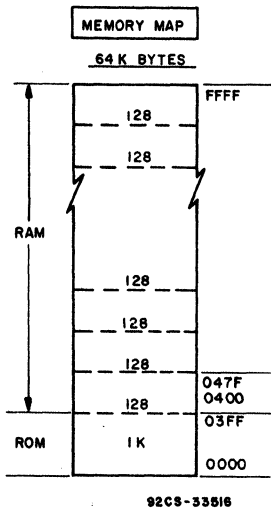
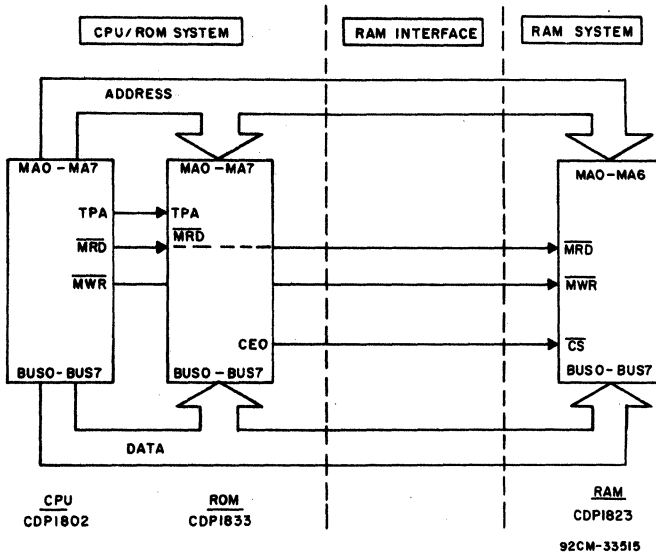
MA 5	MA 6	
1	1	32 BYTES
1	0	32 BYTES
0	1	32 BYTES
0	0	32 BYTES

92CS-33514

RAM INTERFACE (PURPOSE)

- TO DECODE MA5 AND MA6 FOR USE AS CHIP SELECTS.

Fig. A-2 - CDP1824 (32 x 8) expanded system (126 x 8).

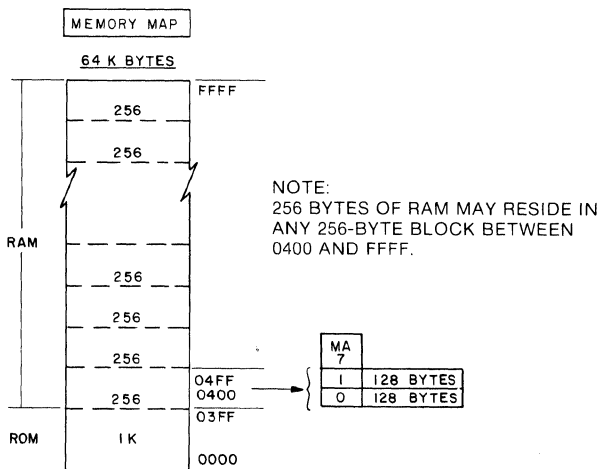
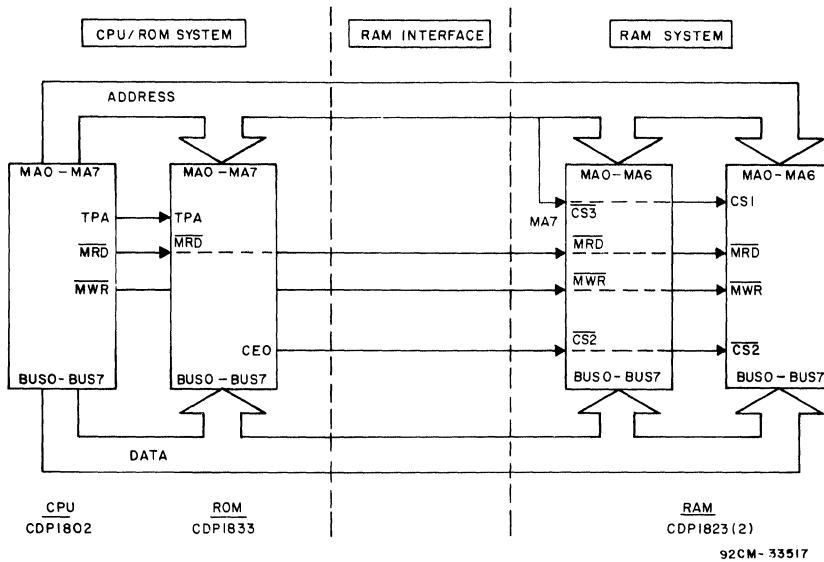


NOTE:
128 BYTES OF RAM MAY RESIDE IN ANY 128-BYTE BLOCK BETWEEN 0400 AND FFFF.

RAM INTERFACE (PURPOSE)

- NONE REQUIRED.

Fig. A-3 - CDP1823 (128 x 8) minimum system (128 x 8).

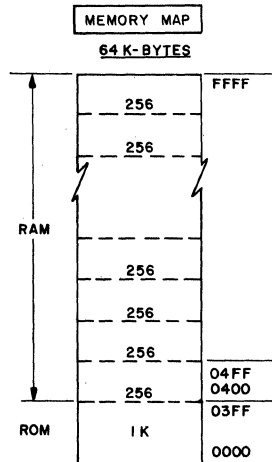
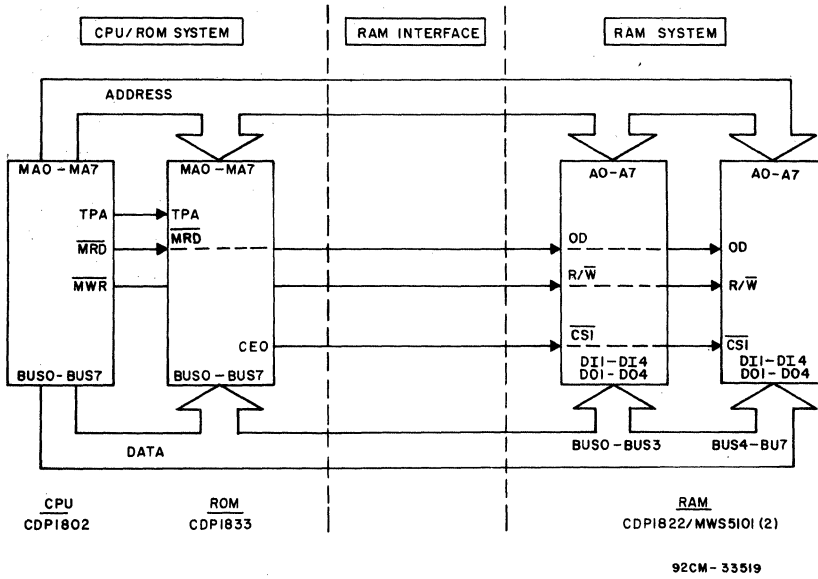


92CS-33518

RAM INTERFACE (PURPOSE)

- NONE REQUIRED.

Fig. A-4 - CDP1823 (128 x 8) expanded system (256 x 8).

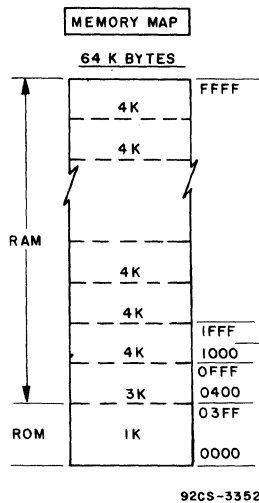
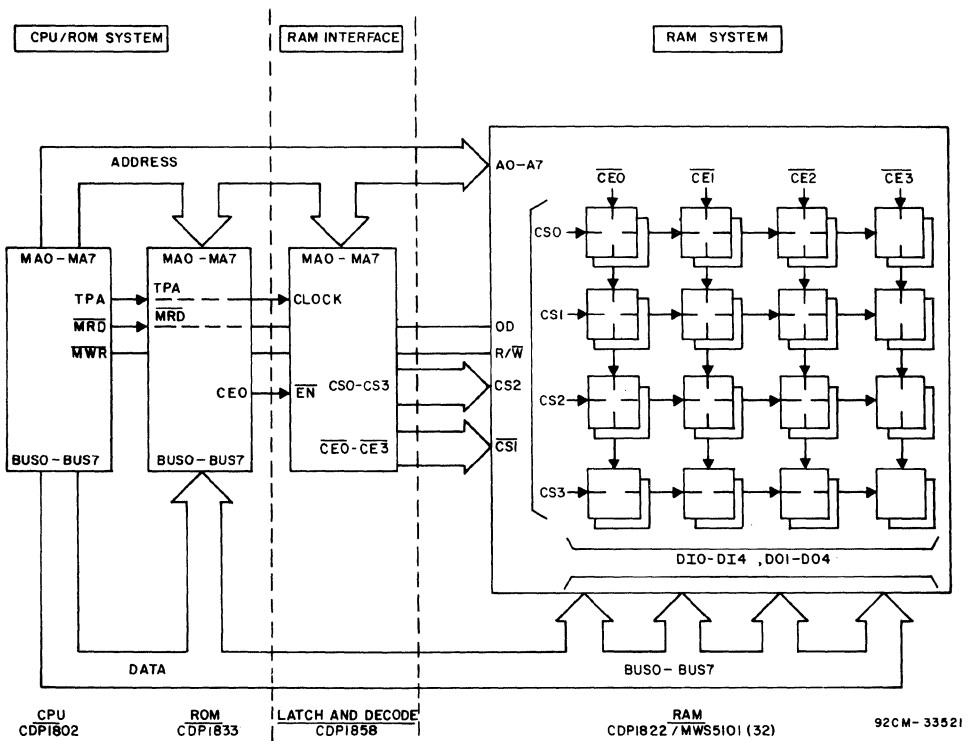


NOTE:
256 BYTES OF RAM MAY RESIDE IN
ANY 256-BYTE BLOCK BETWEEN
0400 AND FFFF.

RAM INTERFACE (PURPOSE)

- NONE REQUIRED.

Fig. A-5 - CDP1822/MWS5101 (256 x 4)
minimum system (256 x 8).



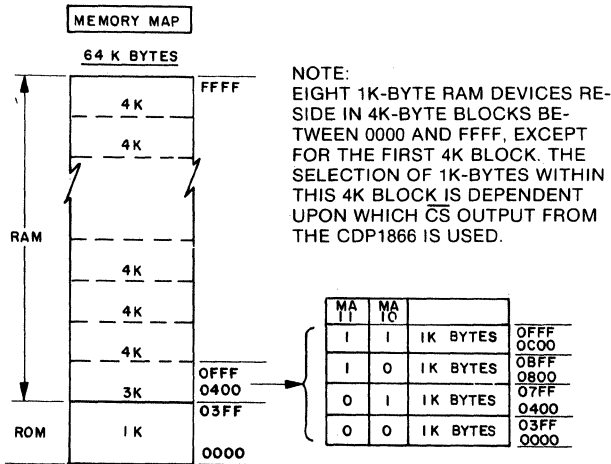
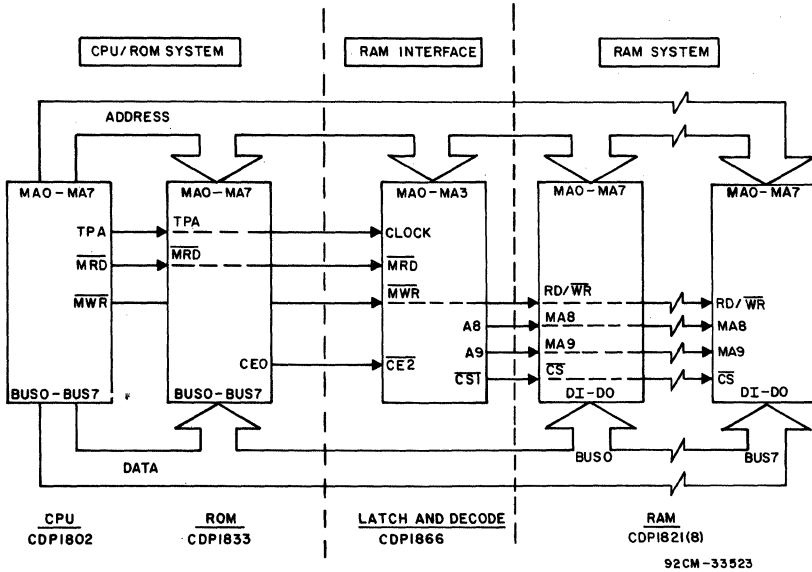
NOTE:
THIRTY-TWO 256-BYTE RAM DEVICES WILL RESIDE IN 4K-BYTE BLOCKS BETWEEN 0000 AND FFFF, EXCEPT FOR THE FIRST 4K-BLOCK WHICH CONTAINS 3K-BYTES OF RAM AND 1K-BYTES OF ROM.

MA 11	MA 10	MA 9	MA 8	
1	1	1	1	256 BYTES
1	1	1	0	256 BYTES
1	1	0	1	256 BYTES
1	1	0	0	256 BYTES
0	0	1	1	256 BYTES
0	0	1	0	256 BYTES
0	0	0	1	256 BYTES
0	0	0	0	256 BYTES

RAM INTERFACE (PURPOSE)

- TO LATCH AND DECODE HIGH-ORDER ADDRESS BITS FOR USE AS CHIP SELECTS.
- PERMITS AN ARRAY OF UP TO 32 MEMORY DEVICES TO BE SELECTED FOR A 4K X 8 SYSTEM.

Fig. A-6 - CDP1822/MWS5101 (256 x 4) expanded system (4K x 8).

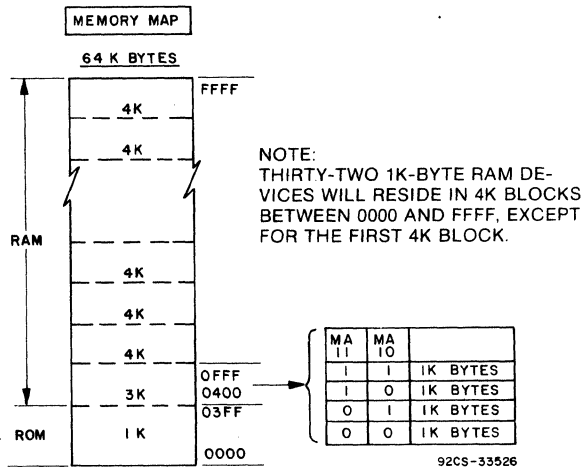
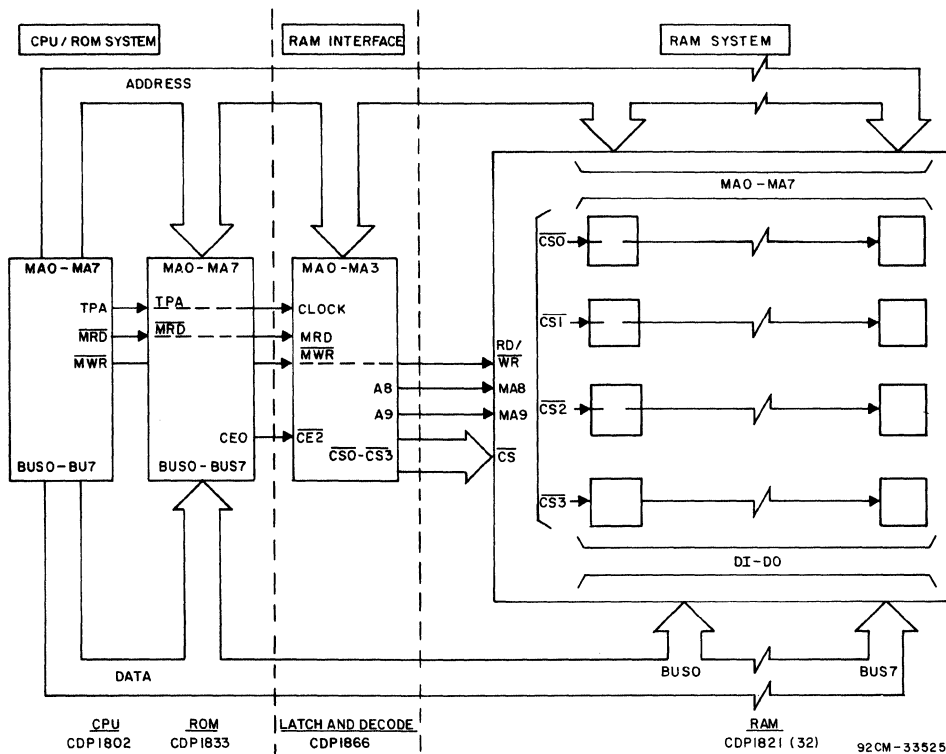


92CS-33524

RAM INTERFACE (PURPOSE)

- TO LATCH AND DECODE HIGH-ORDER ADDRESS BITS FOR USE AS CHIP SELECTS.
- TO GATE THE CHIP SELECTS WITH MRD AND MWR TO PREVENT BUS CONTENTION WITH CPU.
- TO LATCH HIGH-ORDER ADDRESS BITS TO PROVIDE MA8 AND MA9.

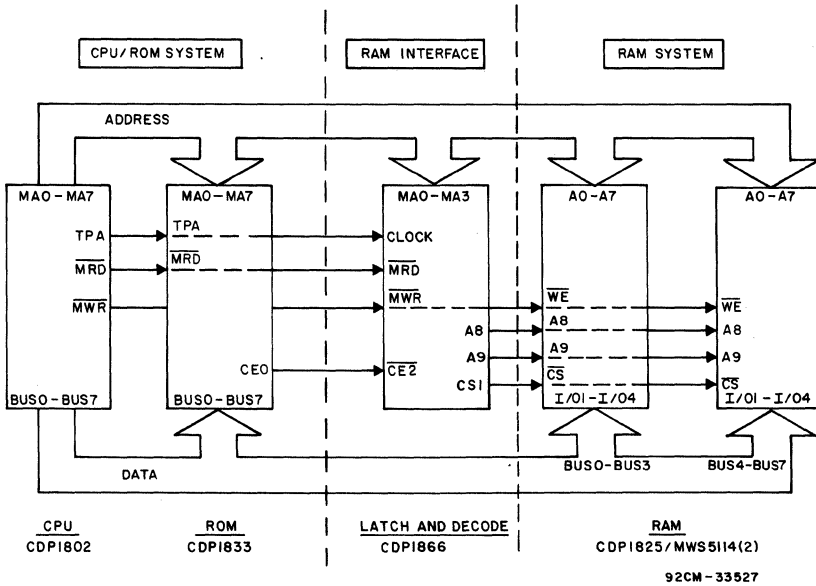
Fig. A-7 - CDP1821 (1K x 1) minimum system (1K x 8).



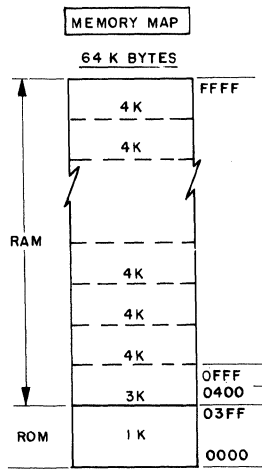
RAM INTERFACE (PURPOSE)

- TO LATCH AND DECODE HIGH-ORDER ADDRESS BITS FOR USE AS CHIP SELECTS.
- TO GATE THE CHIP SELECTS WITH MRD AND MWR TO PREVENT BUS CONTENTION WITH CPU.
- TO LATCH HIGH-ORDER ADDRESS BITS TO PROVIDE MA8 TO MA9.

Fig. A-8 - CDP1821 (1K x 1) expanded system (4K x 8).



92CM-33527



NOTE:
 TWO 1K-BYTE RAM DEVICES WILL RESIDE IN 4K-BYTE BLOCKS BETWEEN 0000 AND FFFF, EXCEPT FOR THE FIRST 4-K BLOCK. THE SELECTION OF 1K-BYTES WITHIN THIS 4K BLOCK IS DEPENDENT UPON WHICH CS OUTPUT FROM THE CDP1866 IS USED.

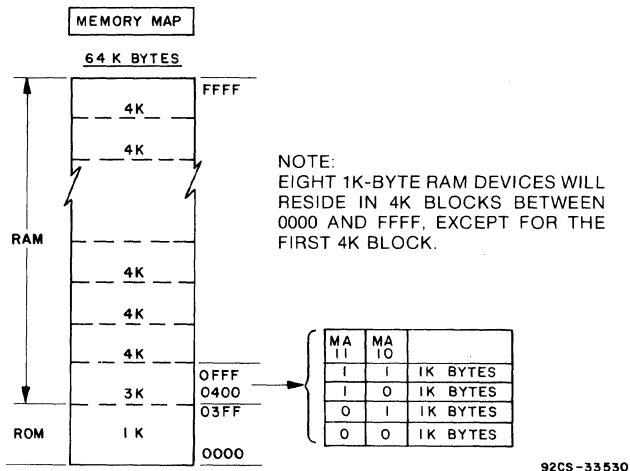
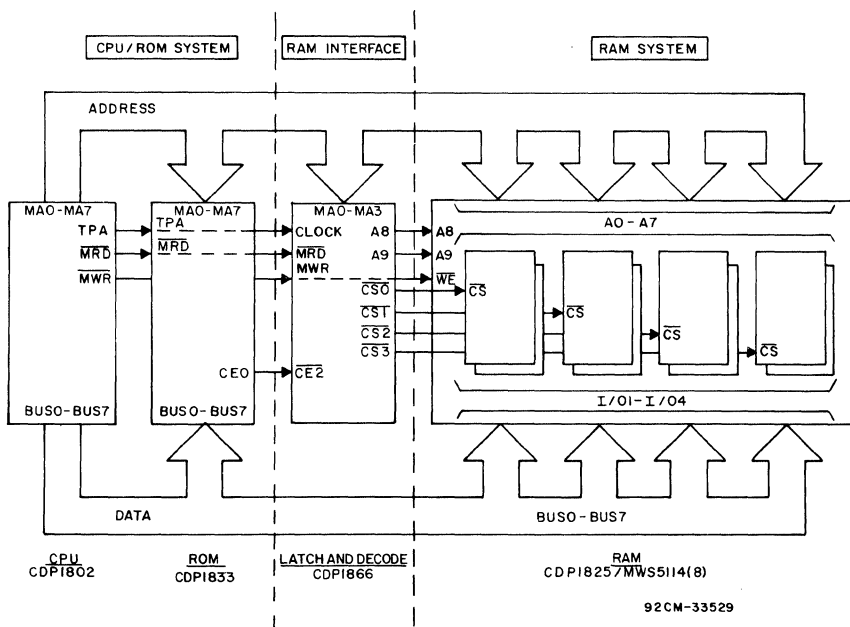
MA 11	MA 10		
1	1	1K BYTES	0000 - 00FF
1	0	1K BYTES	0800 - 08FF
0	1	1K BYTES	0400 - 04FF
0	0	1K BYTES	03FF - 0000

92CS-33528

RAM INTERFACE (PURPOSE)

- TO LATCH AND DECODE HIGH-ORDER ADDRESS BITS FOR USE AS CHIP SELECTS.
- TO GATE THE CHIP SELECTS WITH MRD AND MWR TO PREVENT BUS CONTENTION WITH CPU.
- TO LATCH HIGH-ORDER ADDRESS BITS TO PROVIDE A8 AND A9.

Fig. A-9 - CD1825/MWS5114) (1K x 4) minimum system (1K x 8).



RAM INTERFACE (PURPOSE)

- TO LATCH AND DECODE HIGH-ORDER ADDRESS BITS FOR USE AS CHIP SELECTS.
- TO GATE THE CHIP SELECTS WITH MRD AND MWR TO PREVENT BUS CONTENTION WITH CPU.
- TO LATCH HIGH-ORDER ADDRESS BITS TO PROVIDE A8 AND A9.

Fig. A-10 - CDP1825/MWS5114) (1K x 4) expanded system (4K x 8).

Parallel Clocking of Sequential CMOS Devices

T. Chesney
R. Funk

It is a well-established fact that process variations lead to different input MOS-transistor thresholds, and that these differences directly affect the clock input trigger voltage of sequential CMOS logic circuits. Fig. 1(a) illustrates a cascade of two different sequential CMOS devices (D-type flip-flops, type CD4013) that causes a logic error when data is transferred from IC_A to IC_B. In this example, the same clock transition triggers IC_A at its trigger voltage of V_{TA} and IC_B at a voltage of V_{TB}. As shown in Fig. 1(b), the propagation of data from the output stage of IC_A to the input of IC_B is faster than the clock transition time from V_{TA} to V_{TB}. Hence, IC_B responds to the wrong logic state, and its output goes low when it should stay high.

The solution to this logic-error condition is a clock transition time that is fast enough compared to the propagation delay for a worst-case V_{TA} and V_{TB} combination to assure that logic-state errors will not occur.

Parallel Clocking Limits

A study of the parallel clocking condition for any combination of two different sequential devices has resulted in the development of equations for modeling the maximum permitted clock input rise time, t_{RCL}.

For A-series devices:

$$t_{RCL}(\max) = \frac{0.8V_{DD}(V)}{1.25(V)} \times t_p(\text{ns})$$

For B-series devices:

$$t_{RCL}(\max) = \frac{0.8V_{DD}(V)}{1.15(V)} \times t_p(\text{ns})$$

The factor t_p is equivalent to t_{PHL} or t_{PLH}, whichever is smaller, for IC_A, Fig. 1. The typical value at a specified value of V_{DD} is selected at the loading condition shown on the device data sheet. The factor 0.8 V_{DD} specifies t_{RCL} for a rise or fall time of from 10 to 90 percent. The voltages in the denominator (1.25V for A-series types and

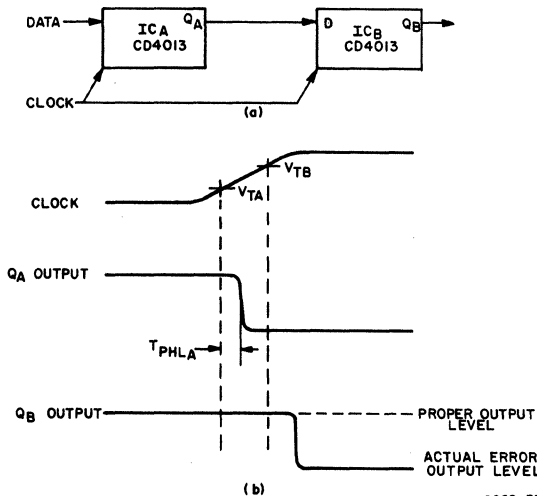


Fig.1-Parallel clocking of sequential CMOS IC's.

92CS-33223

1.15V for B-series types), are the expected deviations in clock input transfer voltages.

Tables I and II list the maximum clock rise times permitted when cascading CD4000A and CD4000B-series types, respectively. The maximum values of t_{RCL} are applicable when sequentially cascading identical or dissimilar IC types when IC_A (see Fig. 1) is the type listed in the "Type" column in the table. However, some restrictions apply; namely both IC_A and IC_B must accept positive or negative (CD4006 type) clock pulses, and the logic combination of IC_A and IC_B must be meaningful. The connections assumed are primarily of the parallel-clocked shift register or counter type.

The t_{RCL} limits shown in Tables I and II are less than those shown on individual

CD4000-series data sheets; the data sheet limits are for individual IC operation, not cascaded operation. Recommended operating-temperature ranges remain as shown in the data sheets for cascaded as well as individual device operation.

TABLE I — Maximum Clock Rise Time When Cascading CD4000A Types¹

Type	Rise Time (μs)	V_{DD} (V)
CD4006A ²	0.80	5,10
CD4013A	0.48	5,10
CD4014A	0.96	5
	0.64	10
CD4015A	0.96	5
	0.64	10
CD4018A	1.12	5
	0.80	10
CD4021A	0.96	5
	0.64	10
CD4027A	0.48	5,10
CD4029A	1.04	5
	0.74	10
CD4031A	1.28	5,10
CD4034A	1.92	5
	1.54	10
CD4035A	0.80	5
	0.64	10

Notes:

- $C_L = 15$ picofarads.
- Negative-edge-triggered device, cascades only with itself.

TABLE II — Maximum Clock Rise Time When Cascading CD4000B Types¹

Type	Rise Time (μs)
CD4006B ²	0.70
CD4013B	0.45
CD4014B	0.40
CD4015B	0.55
CD4021B	0.40
CD4027B	0.45
CD4029B	0.84
CD4031B	0.80
CD4034B	0.85
CD4035B	0.70
CD4076B	0.90
CD4089B	0.40
CD4094B	0.90
CD4095B	0.70
CD4096B	0.70
CD4510B	0.70
CD4516B	0.70
CD4517B	0.84
CD40100B	1.20
CD40102B	0.91
CD40103B	0.91
CD40104B	0.70
CD40160B	0.66
CD40161B	0.66
CD40162B	0.66
CD40163B	0.66
CD40174B	0.50
CD40192B	0.56
CD40193B	0.56
CD40194B	0.70

Notes:

- $V_{DD} = 5, 15V$;
 $C_L = 50$ picofarads.
Data does not apply to units with Schmitt triggers in the clock input.
- Negative-edge-triggered device, cascades only with itself.

An Introduction to the Video Interface System (VIS) Devices - CDP1869 and CDP1870

By W. H. Schilp

This Note describes a circuit and the software required to mate the RCA-CDP1869 and CDP1870 VIS (Video Interface System) chip set¹ to the Evaluation Kit, CDP18S020. The capabilities of the VIS chip set are demonstrated by employing the set in the video portion of an intelligent terminal. Also included in the Note is the circuitry for a CPU controller which, combined with the video board, permits implementation of a stand-alone video output from serial ASCII input. The program shown can also be employed with a Microboard system using the VIS Microboard CDP18S661A.² The VIS circuitry can be constructed on a 44-pin, 4.5 by 7.5-inch PC board, which will mate to the P-2 connector on the Evaluation Kit.

The sample software program displays ASCII encoded alphanumeric characters and symbols, responds to line feed and carriage return, and uses control characters to move the cursor up, down, right, left and home, and to clear the screen. The program will operate from a ROM or PROM with a 6-byte RAM stack outside the program area.

THE VIDEO INTERFACE SYSTEM

The RCA VIS chip set, CDP1869 and CDP1870, is a two-chip system that contains all of the circuitry necessary to generate the video signals needed to drive a monitor in color or black and white. Sound output is also available.

THE VIS CIRCUIT

The video-output circuit, of which the CDP1869 and CDP1870 are a part, is shown in Fig. 1. It is recommended that the video driver transistor and the crystals be placed close to the CDP1870. The CMOS parts should be handled in accordance with recommended procedures.³ The numbers, P2-1, etc., in Fig. 1, refer to the Evaluation Kit connector P2. The MWS5114's and the

2758 must have access times less than 300 nanoseconds for the memory to access properly.

THE CONTROLLER CIRCUIT

The controller, Fig. 2, consists of a CDP1802 Microprocessor, a 2758 EPROM for program storage, a CDP1824 RAM stack, and a CDP1852 for high-order address latching. The data is entered through the CDP1854 UART. The CDP1863 counter generates the clock frequency for the BAUD rate. As shown, the counter is set for 300 BAUD, but it can be rewired to generate other rates. It is recommended that the rate be limited to 1200 BAUD or that suitable hardware be added between UART and I/O device to signal the input device that a new character can be entered. The comments on wiring and handling mentioned above apply here also. The connections P-1, etc., mate with those of the VIS circuit (Fig. 1).

MODIFICATIONS TO EVALUATION KIT

The VIS chip set is memory mapped into memory locations F800 through FFFF. To separate the FXXX memory locations for the VIS chip set, the following changes must be made to the Evaluation Kit, Fig. 3:

1. On the top side of the board:
 - Under U6, cut the connection between pins 11 and 12.
 - Cut the line from U23/15 to U6/8.
2. On the bottom side of the board:
 - Cut line from U6/9 and U22/15 at U22/15.
3. Add a CD4011 in the user area and make the following circuit connections:
 - U23/15 to CD4011/1.
 - CD4011/3 to U22/15.
 - U6/1 to CD4011/2.
 - U21/12 to CD4011/5.
 - U8/21 to CD4011/6.
 - CD4011/4 to U6/8.
 - Don't forget V_{cc} and ground connections.

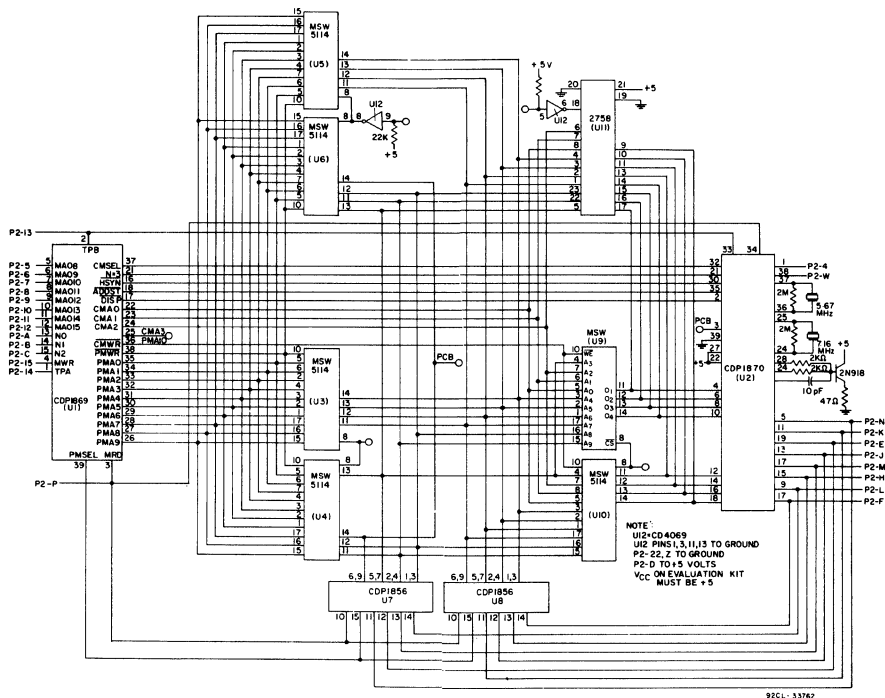


Fig. 1 - Video output circuit employing the CDP1869 and the CDP1870.

OPTIONS FOR THE VIS CIRCUIT

The following represent options for the VIS circuit of Fig. 1.

Page Memory

For two pages of page memory:

- Connect U1/25 to U12/9 and U3/8 and U4/8.

For one page of page memory:

- Do not use U5 or U6. Connect U3/8 and U4/8 to ground.

NOTE: In the two-page mode, the double-page bit must be set high.

Character Memory

The VIS circuit of Fig. 1 is designed to be used in any of three character-memory configurations: up to 64 characters in RAM, up to 64 characters in ROM, or the combination of the two, the 128-character mode. If the RAM-only configuration is used, U11 is not used and U4/14 and U6/14 are connected to U2/3; U9/8 and U10/8 are connected to ground. In the ROM-only version, U9 and U10 are not used, and U4/14 and U6/14 are connected to U2/3. In these configurations, the system can be programmed for all eight color combinations.

In the 128-character memory version, U4/14 and U6/14 are connected to U12/5, U9/8 is connected to U10/8, and U2/3 is connected to ground. In this mode, the PCB

(page color bit) is used to select the RAM (PCB at ground) or ROM (PCB at V_{cc}). In the 128-character mode, only the color combinations available with the character color bits can be used. In black-and-white operation only, the 7.16-MHz crystal and the capacitor on U2/26 are not needed.

NOTE: In this system, EF1 is used during screen refresh to determine whether the display is active.

SOFTWARE PROGRAM

The program example given in the Appendix is one that initializes the system, loads the bit patterns into memory, clears the screen, homes the cursor, and waits for a character input. Each time a displayable character is input, it is displayed in the position of the cursor, which is then incremented one location to the right. If the cursor was on the right margin, it moves down one row and to the left margin. Automatic scrolling is accomplished each time the cursor is indexed past the lower right corner of the screen or above the top displayed line.

The program shown in the Appendix is written for the Evaluation Kit. To utilize the program with the controller board of Fig. 2, the routine that inputs the ASCII characters must be changed to that shown in Fig. 4. In addition, for a Microboard system, bytes

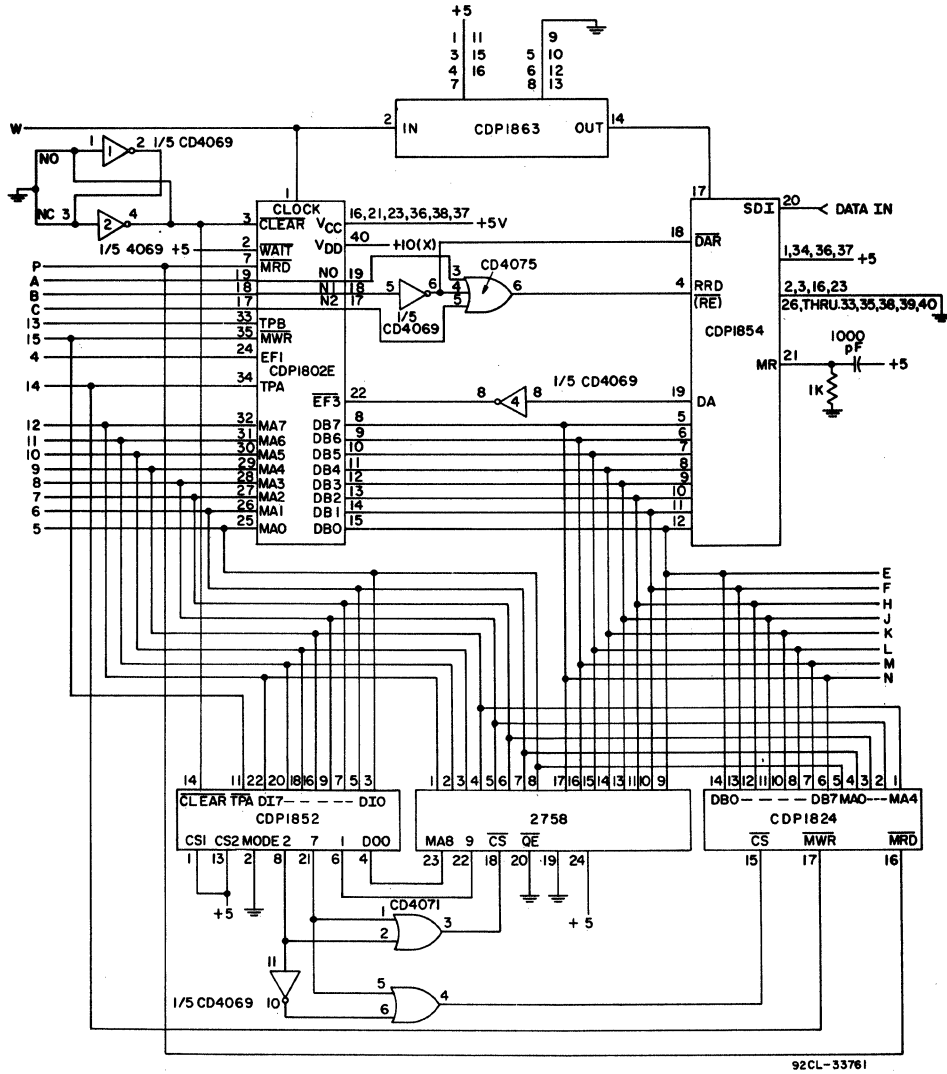


Fig. 2 - A CPU controller for the VIS that permits implementation of a stand-alone video output from serial ASCII input.

0002 and 0003 must be changed to 6180 to enable the two-level I/O, and the program will run as described. A CPU board such as the CDP18S601 is also required; either its on-board memory or a separate memory card must be available to the program at locations 0000 through 03FF. This memory area can be supplied by a RAM, ROM or PROM. A stack is required at locations 0600 through 0606; a CDP18S640 control and display card equipped with the appropriate utility ROM must also be used. The Prototyping Kit, CDP18S691, contains all that is required with the exception of the VIS board.

Program Information

Register Usage

R(2) Stack pointer

- R(3) Utility program counter
- R(4) Stores home address
- R(5) Main program counter
- R(7.0) Character counter for storing bit patterns
- R(8.0) Character row counter for storing bit patterns, then ASCII value storage for previously stored character
- R(9) CMEM location pointer for bit pattern storage, then column counter
- R(A) Page memory location pointer
- R(B) Bit pattern pointer
- R(F.1) Storage for ASCII input from utility program

RUNNING THE PROGRAM

To run the program with either the Evaluation Kit or Microboard System, depress RESET, RUN U, and then the carriage return to set up the timing and registers; then depress either RESET, RUNP or enter \$P0. Each time an ASCII printable character is input, it is displayed in the position of the cursor, which, again, is incremented one position. Line feed, carriage return, cursor up (control O), cursor left (control N), and cursor right (control L), all move the cursor as indicated and restore the previously displayed character to the position from which the cursor came. Clear screen (control K) clears all characters and homes the cursor. Control H causes the screen to scroll down, and Control I causes it to scroll up.

To run the program with the stand-alone system, depress the momentary contact switch; the program will come up running and waiting for an input.

This Note is meant to describe the circuitry and software relating to the CDP1869/70

VIS chip set and to introduce the reader to its potential. Therefore, the program example given in the Appendix is written to be instructive, and as such is not memory or speed efficient. However, an understanding of the program and the VIS data sheet will allow one to utilize more of the many features of the set, such as color, sound, graphics, and motion.

REFERENCES

1. "COS/MOS Video Interface System," RCA Solid State File No. 1197 on the CDP1869C/CDP1870C or "COS/MOS Memories, Microprocessors, and Support Systems," RCA Solid State Databook SSD-260.
2. "RCA COSMAC Microboard Video-Audio-Keyboard Interface CDP-18S661V1 and CDP18S661V3," RCA Solid State publication MB-661.
3. "Guide to Better Handling and Operation of CMOS Integrated Circuits," RCA Solid State Application Note ICAN-6525.

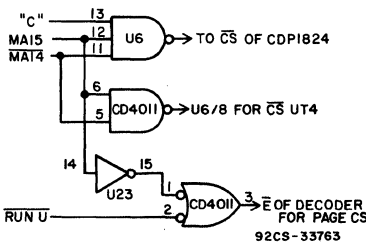


Fig. 3 - Schematic diagram of circuit used to modify the Evaluation Kit to separate the FXXX memory locations for the VIS chip set.

```

0068 EE;          0120 REEN: SEX E          ..GET RID OF X
0069 3E69;        0121 BN3 *           ..WAIT FOR DATA AVAILABLE FROM UART
006B 6ABF;        0122 INP 2; PHI AIN     ..STORE IT IN REG F
006D C4C4;        0123 NOP; NOP         ..WASTED SPACE
    
```

Fig. 4 - Modification of the program shown in the Appendix to make it usable with the Microboard system of Fig. 2.

Appendix - Program Used with the Evaluation Kit

```

IM
0000 ;
0000 ; 0001
0000 ; 0002 ..*****
0000 ; 0003 .. PROGRAM FOR THE VIS TO EVALUATION KIT BOARD ..
0000 ; 0004 .. ASSUMES THE CHARACTER BIT PATTERNS ARE AT 0200 TO 03FF ..
0000 ; 0005 ..*****
0000 ; 0006
0000 ; 0007 ..*****
0000 ; 0008 DEFINE CONSTANTS
0000 ; 0009 ..*****
0000 ; 0010
0000 ; 0011 READ=#B13R ..ADDRESS OF READ ROUTINE IN UTILITY
0000 ; 0012 STK=2 ..REG 2 IS THE STACK POINTER
0000 ; 0013 UTCT=3 ..REG 3 IS THE PROGRAM COUNTER FOR UT4
0000 ; 0014 HA=6 ..STORE THE HOME ADDRESS IN REG 6
0000 ; 0015 PPTR=#0A ..PAGE POINTER IS REG A
0000 ; 0016 CTR=5 ..COUNTER REG FOR THIS PROGRAM IS REG 5
0000 ; 0017
0000 7100; 0018 DIS=#00 ..DISABLE THE INTERRUPTS
0002 C4C4; 0019 NOP/NOP ..LEAVE SPACE FOR 2 LEVEL I/O FOR THE MICROBOARD
0004 ; 0020
0004 FB00B5; 0021 LDI A.1(INIT); PHI CTR ..SET UP REG 5
0007 FB0BA5; 0022 LDI A.0(INIT); PLO CTR
000A D5; 0023 SEP CTR ..START COUNTING IN REG 5
000B ; 0024
000B E5; 0025 INIT: SEX CTR ..SET X FOR OUT IMMED
000C 63B0; 0026 OUT 3; #80 ..TURN ON DISPLAY, SET UP 40 CHARACTERS
000E FB00BA; 0027 LDI #80; PHI PPTR ..SET CMEM HIGH, NOISE OFF, FULL RES VERT, NTSC
0011 FB09AA; 0028 LDI #C9; PLO PPTR ..DBL PG BIT FOR 1ST PG OF MEM
0014 EA65; 0029 SEX PPTR; OUT 5 ..SET X TO PG PTR, OUT 5 SETS THE BITS
0016 ; 0030
0016 ; 0031 CHST=7 ..REG 7 HAS NUMBER OF CHARACTERS STORED
0016 ; 0032 ROCT=8 ..REG 8 IS THE ROW COUNTER WITHIN EACH
0016 ; 0033 ..CHARACTER BIT PATTERN
0016 ; 0034 CMEM=9 ..REG 9 POINTS TO CHARACTER MEMORY
0016 ; 0035 CPAT=#0B ..REG B POINTS TO CHARACTER PATTERN
0016 ; 0036
0016 FBFBBA; 0037 LDI #F8; PHI PPTR ..POINT THE PAGE POINTER AND HOME ADDRESS TO
0019 345; 0038 PHI HA ..F800
001A FBFAE9; 0039 LDI #F4; PHI CMEM ..POINT THE CHARACTER MEM AT F400
001D FB00AB; 0040 LDI #00; PLO CPAT ..PUT 00 IN LOW BYTE OF ALL NEEDED REG'S
0020 AAA9; 0041 PLO PPTR; PLO CMEM
0022 ABA7; 0042 PLO ROCT; PLO CHST
0024 A6; 0043 PLO HA
0025 ; 0044
0025 FB02BB; 0045 LDI #02; PHI CPAT ..CHARACTER PATTERN IS AT 0200
0028 ; 0046
0028 FB06B2; 0047 LDI #06; PHI STK ..STACK IS AT 0606
002B A2; 0048 PLO STK
002C ; 0049
002C 672A; 0050 OUT 7; DEC PPTR ..SET THE HOME ADDRESS
002E 662A; 0051 OUT 6; DEC PPTR ..SET THE PAGE MEMORY POINTER
0030 ; 0052
0030 ; 0053
0030 ; 0054 ..*****
0030 ; 0055 .. CHARACTER LOAD ROUTINE
0030 ; 0056 ..*****
0030 ; 0057
0030 87; 0058 CHLD: GLO CHST ..GET THE ADDRESS OF THE 1ST CHARACTER PATTERN
0031 3431; 0059 B1 * ..WAIT FOR NON-DISPLAY
0033 5A; 0060 STR PPTR ..STORE IT IN PAGE MEMORY
0034 4B; 0061 BTLD: LDA CPAT ..GET THE FIRST ROW OF THE 1ST CHARACTER BIT PATTERN
0035 3435; 0062 B1 * ..WAIT FOR NON-DISPLAY

0037 59; 0063 STR CMEM ..STORE IT IN CHARACTER MEMORY
0038 ; 0064
0038 1918; 0065 INC CMEM; INC ROCT ..INCREMENT TO NEXT ROW OF 1ST CHARACTER BIT PATTERN
003A ; 0066 ..AND KEEP A COUNT ON THE NUMBER OF ROWS
003A BFBF08; 0067 GLO ROCT; XRI #08 ..SEE IF PUT IN 8 ROWS OF BIT PATTERNS
003D 3A34; 0068 BNZ BTLD ..IF NOT GO BACK FOR ANOTHER
003F ; 0069
003F FB00AB; 0070 LDI #00; PLO ROCT ..RESET THE ROW COUNT TO 0
0042 17; 0071 INC CHST ..INCREMENT FOR THE NEXT CHARACTER BIT PATTERN
0043 87FB40; 0072 GLO CHST; XRI #40 ..SEE IF STORED ALL 64 CHARACTER PATTERNS
0046 3A30; 0073 BNZ CHLD ..IF NOT GO BACK
0048 ; 0074
0048 ; 0075 ..BIT PATTERN ARE NOW STORED IN CHARACTER MEMORY
0048 ; 0076
0048 FB00BA; 0077 LDI #80; PHI PPTR ..TURN OFF THE CHARACTER MEMORY ACCESS
0048 FB0BA5; 0078 LDI #C8; PLO PPTR ..MODE
004E 65; 0079 OUT 5
004F ; 0080 ..NOW WE CLEAR THE SCREEN
004F ; 0081
004F FBFFBA; 0082 LDI #FF; PHI PPTR ..POINT THE PAGE POINTER AT THE TOP
0052 AA; 0083 PLO PPTR ..OF THE PAGE MEMORY
0053 FB00; 0084 CLPG: LDI #00 ..GET THE CHARACTER MEMORY ADDRESS FOR A SPACE
0055 3455; 0085 B1 * ..WAIT
0057 73; 0086 STXD ..STORE IT IN THE PAGE MEMORY AND POINT TO NEXT

```

```

005B ;
005B 9AFBF7; 008B GHI PPTR; XRI #F7 ..LOWER PAGE MEM LOCATION
005B 3A53; 0089 BNZ CLFG ..SEE IF REACHED THE BOTTOM OF PAGE MEMORY
005D ; 0090 ..IF NOT GO BACK
005D 1A; 0091 INC PPTR ..MOVE THE PAGE POINTER TO 1ST LOCATION THAT IS
005E ; 0092 ..UPPER LEFT CORNER OF THE SCREEN
005E ; 0093
005E ; 0094
005E ; 0095 ..*****
005E ; 0096 ..INITIALIZATION COMPLETE
005E ; 0097 ..NOW WE ARE READY TO BRING IN CHARACTERS
005E ; 0098 ..*****
005E ; 0099
005E ; 0100
005E ; 0101 AIN=#0F ..REG F WILL CONTAIN THE ASCII BYTE INPUTTED
005E ; 0102 TSTR=8 ..REG 8 WILL BE NEEDED FOR TEMPORARY STORAGE
005E ; 0103 COCT=9 ..REG 9 WILL KEEP COUNT OF THE COLUMN IN WHICH THE
005E ; 0104 ..CURSOR IS
005E ; 0105
005E FB00A9; 0106 LDI #00; PLO COCT ..ZERO OUT THE COLUMN COUNTER
0061 ; 0107
0061 ; 0108
0061 ; 0109 ..*****
0061 ; 0110 ..THIS ROUTINE SAVES THE CHARACTER IN THE NEW LOCATION
0061 ; 0111 ..AND DISPLAYS THE CURSOR THERE
0061 ; 0112 ..*****
0061 ; 0113
0061 ; 0114
0061 0A08; 0115 NEWC: LDI PPTR; PLO TSTR..PUT THE CHARACTER INTO TEMPORARY STORAGE
0063 FB3F; 0116 LDI #3F ..LOAD THE CHEN ADDRESS OF THE CURSOR
0065 3465; 0117 B1 * ..WAIT
0067 5A; 0118 STR PPTR ..DISPLAY THE CURSOR
0068 ; 0119
0068 FB3BA3; 0120 REEN: A.0(READ)->UTCT.0 ..LOAD THE ADDRESS OF READ ROUTINE INTO
006B FB81B3; 0121 A.1(READ)->UTCT.1 ..UTILITY'S PROGRAM COUNTER
006E D3; 0122 SEP UTCT ..DO THE READ
006F ; 0123
006F EA; 0124 SEX PPTR ..RESET X
0070 9F; 0125 GHI AIN ..GET THE ASCII CHARACTER FROM REG F
0071 FA60C2010A; 0126 ANI #60; LBZ CTLC ..SEE IF BITS 5 AND 6 ARE LOW, IF SO MUST BE A
0076 ; 0127 ..CONTROL CHARACTER SO GO TO THAT ROUTINE
0076 FB603268; 0128 XRI #60; BZ REEN ..SEE IF BITS 5 AND 6 ARE HIGH, IF SO WE DON'T
007A ; 0129 ..NEED IT SO GO BACK FOR A NEW ENTRY
007A ; 0130
007A ; 0131 ..WE MUST HAVE A DISPLAYABLE CHARACTER SO DO IT
007A ; 0132
007A 9F; 0133 GHI AIN ..GET THE ASCII VALUE
007B FF20; 0134 SMI #20 ..SUBTRACT HEX 20 TO MAKE IT EQUAL TO ITS
007D ; 0135 ..CHARACTER MEMORY ADDRESS
007D 347D; 0136 B1 * ..WAIT
007F 5A; 0137 STR PPTR ..DISPLAY IT
0080 ; 0138
0080 ; 0139
0080 ; 0140 ..*****
0080 ; 0141 ..REMAINDER OF THE PROGRAM TAKES CARE OF THE POSITION ON
0080 ; 0142 ..THE SCREEN FOR THE NEXT CHARACTER
0080 ; 0143 ..AND THE SCROLLING
0080 ; 0144 ..*****
0080 ; 0145
0080 ; 0146
0080 ; 0147 ..HAS THE PAGE POINTER REACHED END OF LAST LINE ?
0080 ; 0148
0080 ; 0149 LLCK:
0080 BAFF7F; 0150 GLO PPTR; SMI #7F ..SEE IF THE PAGE POINTER IS AT THE LAST DISPLAYABLE
0083 3A91; 0151 BNZ INCR ..LOCATION IF NOT WE CAN GO AND INCREMENT IT
0085 9AFFFF; 0152 GHI PPTR; SMI #FF
008B 3A91; 0153 BNZ INCR
008A ; 0154
008A AAA9; 0155 PLO PPTR; PLO COCT ..MUST HAVE BEEN AT LAST LOCATION SO RESET PAGE
008C FBF8BA; 0156 LDI #FB; PHI PPTR ..POINTER TO BEGINNING THAT IS FB00
008F 3099; 0157 BR USCK ..GO TO CHECK IF WE MUST SCROLL UP
0091 ; 0158
0091 ; 0159 ..INC PG PTR AND CHECK POSITION
0091 ; 0160
0091 1A; 0161 INCR: INC PPTR ..INCREMENT PAGE POINTER AND COLUMN COUNTER
0092 19; 0162 INC COCT ..TO NEXT LOCATION
0093 89; 0163 GLO COCT ..SEE IF WE REACHED THE 40TH COLUMN
0094 FF28; 0164 SMI #28
0096 3A61; 0165 BNZ NEWC ..IF NOT GO GET ANOTHER CHARACTER
0098 ; 0166
0098 A9; 0167 PLO COCT ..MUST HAVE BEEN 40TH COLUMN SO ZERO OUT COLUMN
0099 ; 0168 ..COUNTER
0099 ; 0169
0099 ; 0170
0099 ; 0171 ..*****
0099 ; 0172 ..SCROLL UP ROUTINE
0099 ; 0173 ..*****
0099 ; 0174
0099 ; 0175 ..SCROLL UP CHECK
0099 ; 0176
0099 E2; 0177 USCK: SEX STK ..POINT X TO STACK
009A B6FFC0; 0178 GLO HA; SMI #C0 ..SEE IF PRESENT HOME ADDRESS IS IN SECOND
009D 967FB; 0179 GHI HA; SMRI #FB ..PAGE OF PAGE MEMORY

```


ICAN-6953

```

00A0 33AC;      0180      BDF SCPG          ..IF SO GO TO SECOND PAGE ROUTINE
00A2 ;         0181
00A2 ;         0182          ..1ST PAGE ROUTINE
00A2 ;         0183
00A2 86FCC073; 0184 GLO HA; ADI #C0; STXD ..STORE AN OFFSET HOME ADDRESS ON THE STACK
00A6 967C03;   0185 GHI HA; ADCI #03      ..WHICH IS THE ADDRESS OF THE NEXT LOCATION
00A9 52;       0186 STR STK          ..JUST BELOW THE PRESENTLY DISPLAYED SCREEN
00AA 30BE;     0187 BR PFCK           ..GO TO THE PAGE POINTER CHECK
00AC ;         0188

00AC ;         0189          ..2ND PAGE ROUTINE
00AC ;         0190
00AC 8652;     0191 SCPG; GLO HA; STR STK ..SUBTRACT THE HOME ADDRESS FROM THE PAGE
00AE 8AF7;     0192      GLO PPTR; SM      ..POINTER TO SEE IF THE PAGE POINTER IS STILL
00B0 9652;     0193      GHI HA; STR STK   ..ON SCREEN
00B2 9A77;     0194      GHI PPTR; SMB
00B4 3361;     0195      BDF NEWC          ..IF SO CAN GET A NEW CHARACTER
00B6 ;         0196
00B6 86FFC0;   0197      GLO HA; SMI #C0    ..MIGHT BE OFF SCREEN SO STORE THE OFFSET
00B9 73;       0198      STXD              ..HOME ADDRESS ON THE STACK
00BA 967F03;   0199      GHI HA; SMBI #03
00BD 52;       0200      STR STK
00BE ;         0201          ..PAGE PTR CHECK
00BE ;         0202
00BE 12;       0203 PPCK; INC STK        ..SUBTRACT THE OFFSET HOME ADDRESS FROM
00BF 8AF7;     0204      GLO PPTR; SM      ..THE PAGE POINTER
00C1 22;       0205      DEC STK
00C2 9A77;     0206      GHI PPTR; SMB
00C4 12;       0207      INC STK
00C5 EA;       0208      SEX PPTR          ..RESET X TO THE PAGE POINTER
00C6 3B61;     0209      BL NEWC          ..IF PAGE POINTER STILL ON SCREEN GO BACK
00C8 ;         0210          ..FOR A NEW CHARACTER
00C8 ;         0211
00C8 ;         0212          ..WE WILL HAVE TO SCROLL
00C8 ;         0213          ..BUT FIRST CLEAR THE NEXT LINE
00C8 ;         0214
00C8 ;         0215 TEMP=7          ..NEED TEMPORARY STORAGE
00CB 89A7;     0216 GLO COCT; PLO TEMP ..FOR THE COLUMN COUNT
00CA 8932D1;   0217 ZERO; GLO COCT; BZ CLLN ..MOVE THE PAGE POINTER TO THE LEFT MARGIN
00CD 292A;     0218      DEC COCT; DEC PPTR
00CF 30CA;     0219      BR ZERO
00D1 ;         0220
00D1 FB00;     0221 CLLN; LDI #00        ..GET THE CMEM ADDRESS OF A SPACE
00D3 34D3;     0222      B1*              ..WAIT
00D5 5A;       0223      STR PPTR          ..STORE IT IN PAGE MEMORY
00D6 1A19;     0224      INC PPTR; INC COCT ..MOVE OVER ONE
00D8 89FF2B;   0225      GLO COCT; SMI #2B ..SEE IF WE DID THE WHOLE LINE
00DB 3AD1;     0226      BNZ CLLN          ..IF NOT GO BACK
00DD ;         0227
00DD 2A;       0228 REST; DEC PPTR       ..RESET THE PAGE POINTER TO THE BEGINNING
00DE 29;       0229      DEC COCT       ..OF THE LINE
00DF 89;       0230      GLO COCT
00E0 3ADD;     0231      BNZ REST
00E2 ;         0232
00E2 8732EA;   0233 RSTR; GLO TEMP; BZ USCL ..REPOSITION THE PAGE POINTER AND COLUMN
00E5 27;       0234      DEC TEMP          ..COUNTER TO THE PROPER POSITION ON THE NEW LINE
00E6 191A;     0235      INC COCT; INC PPTR ..WHEN DONE GO TO SCROLL UP ROUTINE
00E8 30E2;     0236      BR RSTR
00EA ;         0237
00EA ;         0238          ..SCROLL UP ROUTINE
00EA ;         0239
00EA ;         0240          ..BUT FIRST SEE IF WE HAD THE LAST LINE OF THE
00EA ;         0241          ..SECOND PAGE ON THE BOTTOM OF THE SCREEN
00EA ;         0242
00EA 86FF5B;   0243 USCL; GLO HA; SMI #5B ..SEE IF LAST LINE WAS ON BOTTOM
00ED 3AFB;     0244      BNZ NLST
00EF 9AFFFF;   0245      GHI HA; SMI #FF
00F2 3AFB;     0246      BNZ NLST          ..IF NOT GO TO NOT LAST LINE ROUTINE
00F4 A6;       0247      PLO HA              ..MUST HAVE BEEN LAST LINE SO SET UP
00F5 FBFB64;   0248      LDI #FB; PHI HA    ..HOME ADDRESS
00FB C00103;   0249      LBR OUTP          ..GO TO OUTPUT IT
00FB ;         0250
00FB ;         0251          ..NOT LAST LINE
00FB ;         0252
00FB 86FC2B;   0253 NLST; GLO HA; ADI #2B ..ADD HEX 2B TO THE HOME ADDRESS
00FE A6;       0254      PLO HA
00FF 967C00;   0255      GHI HA; ADCI #00
0102 B6;       0256      PHI HA
0103 E667;     0257 OUTP; SEX HA; OUT 7    ..SET THE NEW HOME ADDRESS
0105 26EA;     0258      DEC HA; SEX PPTR ..DEC THE HA CAUSE OUT INSTRUCTION INCR'S IT
0107 ;         0259
0107 C00061;   0260      LBR NEWC          ..GO GET THE NEXT CHARACTER
010A ;         0261
010A ;         0262
010A ;         0263 ..*****
010A ;         0264 .. THE FOLLOWING ROUTINE LOOKS AT THE CONTROL
010A ;         0265 .. CHARACTERS FOR CURSOR MOVEMENT OR SCROLL
010A ;         0266 .. CONTROL CHARACTER ROUTINE
010A ;         0267 ..*****
010A ;         0268
010A ;         0269

```

```

010A 88J      0270 CTLCL: GLO TSTR      ..RESTORE THE CHARACTER WHICH USED TO
010B 340BJ    0271      B1 #          ..BE UNDER THE CURSOR
010D 5AJ      0272      STR PPTR
010E J        0273
010E 9FJ     0274 GHI AIN          ..GET THE ASCII VALUE
010F FF0832A5J 0275 SMI #08J BZ DSCL      ..CONTROL H IS SCROLL DOWN
0113 FF01C200EAJ 0276 SMI #01J LBZ USCL ..CONTROL I IS SCROLL UP
0118 FF013235J 0277 SMI #01J BZ LIFD    ..THIS IS LINE FEED
011C FF01C2000BJ 0278 SMI #01J LBZ INIT   ..CONTROL K IS CLEAR SCREEN AND HOME CURSOR
0121 FF01C2000BJ 0279 SMI #01J LBZ LLCK ..CONTROL L IS CURSOR RIGHT
0126 FF013250J 0280 SMI #01J BZ CART   ..THIS IS CARRIAGE RETURN
012A FF0132BFJ 0281 SMI #01J BZ CULT   ..CONTROL N IS CURSOR LEFT
012E FF01325BJ 0282 SMI #01J BZ CUUP ..CONTROL O IS CURSOR UP
0132 C00061J 0283 LBR NEWC          ..NOT USING ANY OTHERS RIGHT NOW SO
0135 J        0284          ..GO BACK FOR NEXT CHARACTER
0135 J        0285
0135 J        0286
0135 J        0287 ..*****
0135 J        0288 ..          LINE FEED ROUTINE          ..
0135 J        0289 ..*****
0135 J        0290
0135 J        0291
0135 BAFF5BJ 0292 LIFD: GLO PPTR; SMI #58 ..SEE IF CURSOR IS ON LAST LINE OF SECOND PAGE
0138 9A7FFFJ 0293 GHI PPTR; SMBI #FF
0138 334BJ   0294      BDF LILL          ..IF SO GO TO LAST LINE ROUTINE
013D J       0295
013D BAF2BJ 0296 GLO PPTR; ADI #28 ..NOT ON LAST LINE SO ADD HEX 28
0140 AAJ     0297 PLO PPTR
0141 9A7C0BJ 0298 GHI PPTR; ADCI #00
0144 BAJ     0299 PHI PPTR
0145 C00099J 0300 LBR USCK          ..AND GO CHECK TO SEE IF WE NEED TO SCROLL UP
0148 J      0301
0148 J      0302          ..LAST LINE ROUTINE
0148 J      0303
0148 B9AAJ 0304 LILL: GLO COCT; PLO PPTR..PAGE POINTER IS F800 PLUS THE COLUMN COUNT
014A FBFBBAJ 0305      LDI #FB; PHI PPTR
014B C00099J 0306      LBR USCK          ..GO CHECK FOR SCROLL UP
0150 J      0307
0150 J      0308
0150 J      0309 ..*****
0150 J      0310 ..          CARRIAGE RETURN ROUTINE          ..
0150 J      0311 ..*****
0150 J      0312
0150 J      0313
0150 89J    0314 CART: GLO COCT          ..SEE IF CURSOR IS IN LEFT MARGIN

0151 C20061J 0315      LBZ NEWC          ..IF SO GO BACK FOR NEXT CHARACTER
0154 2A29J   0316      DEC PPTR; DEC COCT..IT WASN'T, MOVE LEFT TILL WE GET THERE
0156 3050J   0317      BR CART
0158 J       0318
0158 J       0319
0158 J       0320 ..*****
0158 J       0321 ..          CURSOR UP ROUTINE          ..
0158 J       0322 ..*****
0158 J       0323
0158 J       0324
0158 BAFF2BJ 0325 CUUP: GLO PPTR; SMI #28 ..SEE IF CURSOR IS IN 1ST LINE OF 1ST PAGE
0158 9A7FBJ 0326 GHI PPTR; SMBI #F8..OF PAGE MEMORY
015E 3B6AJ   0327      BNF CUFL          ..IF SO GO TO 1ST LINE ROUTINE
0160 J       0328
0160 J       0329          ..MOVE CURSOR UP
0160 J       0330
0160 BAFF2BJ 0331 GLO PPTR; SMI #28 ..NOT IN 1ST LINE SO SUBTRACT HEX 28
0163 AAJ     0332 PLO PPTR
0164 9A7F0BJ 0333 GHI PPTR; SMBI #00
0167 BAJ     0334 PHI PPTR
0168 3087J   0335 BR DSCK          ..CHECK FOR SCROLL
016A J       0336
016A J       0337          ..1ST LINE ROUTINE
016A J       0338
016A FBFBBAJ 0339 CUFL: LDI #FF; PHI PPTR ..PUT CURSOR IN PROPER POSITION OF LAST
016D B9FC5BJ 0340      GLO COCT; ADI #58 ..LINE OF SECOND PAGE OF PAGE MEMORY
0170 AAJ     0341      PLO PPTR
0171 J       0342          ..NOW CHECK FOR SCROLL
0171 J       0343
0171 J       0344
0171 J       0345 ..*****
0171 J       0346 ..          DOWN SCROLL ROUTINE          ..
0171 J       0347 ..*****
0171 J       0348
0171 J       0349          ..DID WE GO OFF SCREEN ?
0171 J       0350
0171 B6FB0BJ 0351 DSFC: GLO HA; XRI #00 ..IS THE HOME ADDRESS F800 ?
0174 3AB3J   0352      BNZ GTCH
0176 96FBFBJ 0353      GHI HA; XRI #FB
0179 3AB3J   0354      BNZ GTCH          ..IF NOT DON'T SCROLL
017B BAFF5BJ 0355      GLO PPTR; SMI #5B ..ARE WE ON THE LAST LINE OF
017E 9A7FFFJ 0356      GHI PPTR; SMBI #FF..2ND PAGE ?
0181 338BJ   0357      BGE DSFL          ..IF SO, SCROLL
0183 EAJ     0358 GTCH: SEX PPTR
0184 C00061J 0359      LBR NEWC          ..GO BACK FOR MORE
0187 J       0360

```


Using the COSMAC Microboard Battery-Backup RAM, CDP18S622

by P. H. Merl

The RCA-CDP18S622, COSMAC Microboard 8-Kilobyte Battery-Backup RAM,¹ pictured in Fig. 1, is a Microboard RAM card equipped with rechargeable batteries. The batteries are maintained in the charged state either by the user's system power supply or by an entirely separate, external dc or ac supply. Provisions have been made on the card for an on-board rectifier and regulator and the necessary capacitors to accommodate this external supply, Fig. 2. There are also provisions for an additional, fourth, battery. Linking options provide the user with the ability to power a small system from the battery pack.² Logic diagrams of the battery-backup card are provided in Figs. 3 and 4. This Note discusses the application of the board as a standard-power backup medium, a nonvolatile transport medium, and as an efficient means of aiding the testing of new or prototype boards.

APPLICATIONS

Standard-Power Backup

Perhaps the primary application of the CDP18S622 Microboard is in systems subject to power failure. As power in the system fails, the on-board voltage comparator disables memory read (MRD) and memory write (MRW), and all information on the board remains intact in spite of the power failure. The drivers to the bus are held in the high-impedance state, so that no current is supplied to the system backplane. Battery drain is negligible and the information in memory is safe for days because of the low-power requirements of the CMOS RAM. The MRD and MRW functions are actively driven to the false state with system power removed so that the battery-backup RAM card can be removed from the system, transferred elsewhere, installed in another system, and run.

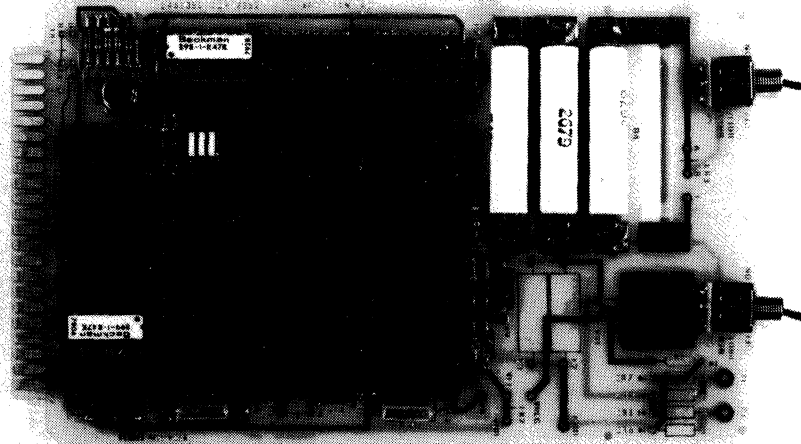
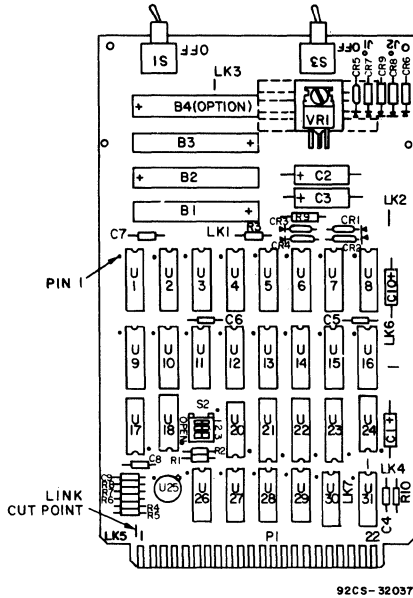


Fig. 1 - The CDP18S622, COSMAC Microboard 8-Kilobyte Battery-Backup RAM.



Parts List

- B1 - B3 = nickel-cadmium, 180 mAh, AAA
- *B4 = nickel-cadmium, 180 mAh, AAA
(Panasonic NR-AAA-U)
- C1, C10 = 15 μ F, 50 V
- *C2, C3 = 220 μ F, 20 V (Sprague 137D227C7020F2)
- C4 = 0.33 μ F, 50 V
- C5 - C8 = 0.1 μ F, 50 V
- C9 = 22 pF
- CR1 - CR4 = 1N270
- *CR5 = 1N270
- *CR6 - CR9 = 1N4001 (RCA D1201F)
- *H1 = Heat sink (Thermalloy 6070B)
- J1, J2 = terminal, optional 12.6 V ac
- R1, R2, R3, R6, R7 = 47 k Ω , 1/4 W, 5%
- R4 = 5.1 M Ω , 1/4 W, 5%
- R5 = 10.0 M Ω , 1/4 W, 5%
- R8 = 1.1 M Ω , 1/4 W, 5%
- R9 = 15 Ω , 1/2 W
- R10 = 100 k Ω , 1/4 W, 5%
- S1, S3 = SPDT
- S2 = 3-rocker DIP
- U1 - U16 = MWS5114E
- U17 = resistor module
22 k Ω , 16 pin
- U18 = CD4001BE
- U20 = CD4070BE
- U21 = CDP1867CE
- U22, U23 = CDP1866CE
- U24 = CD4075BE
- U25 = CA3078S
- U26, U27 = CDP1856CE
- U28, U29 = CD4050BE
- U30 = resistor module
22 k Ω , 14 pin
- U31 = CD4093BE
- *VR1 = 5-V voltage regulator
(Fairchild 7805)
- *User-supplied components for optional power supply.

Fig. 2 - Layout diagram for the CDP18S622. B4, VR1, CR5-CR9, C2 and C3 are optional items not supplied with the board.

The voltage comparators allow the system voltage to drop below battery voltage before the comparator disables the RAM. This feature assures that small power-supply fluctuations will not gate the RAM off during normal operation, and that only an actual power loss will disable the RAM. Conversely, on power-up, the voltage is allowed to rise above battery voltage before the comparator gates the RAM on. Hysteresis in the comparator circuit prevents oscillation on, and diminishes the noise sensitivity of, the enable signal when power-supply and battery voltage are nearly equal.

The CDP18S622 battery-backup board is involved in system power configurations through five different options. The first option makes use of the board as shipped, with three batteries and no regulator components. In this configuration, the batteries are maintained in the charged state by the user's system power supply.

When the system power is off, the batteries provide power for the CDP18S622 only and memory contents are preserved. The second option uses the optional external regulator circuit, which permits an external ac or dc supply to be connected directly to the battery-backup board; this external supply is, again, separate from the user's system supply. In this mode of operation, the external supply maintains the battery pack in the charged state and provides power for the CDP18S622, but not the rest of the system. With a power failure, backup power would be supplied to the battery-backup board only. The third option also makes use of the external supply feature. However, in this option, the entire system is powered from the external supply with the battery-backup capability available to the entire system. The fourth option uses the external supply to power the entire system, but the battery-backup feature applies only to the CDP18S622 RAM. The fifth option

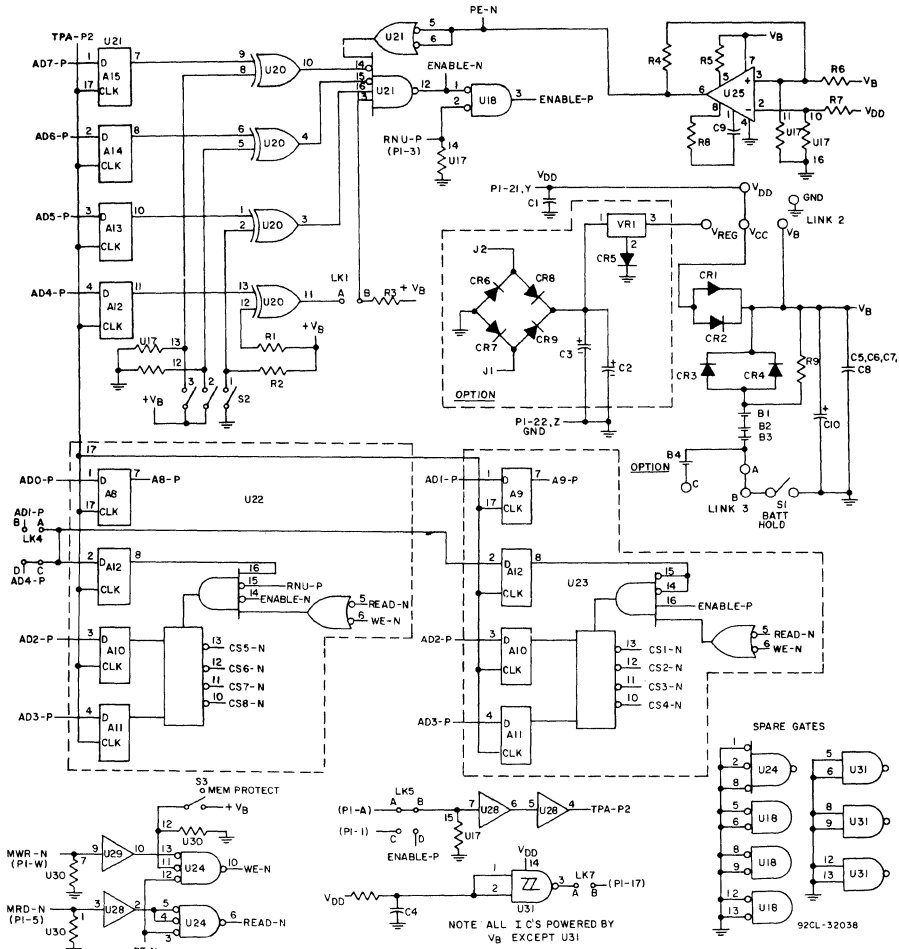


Fig. 3 - Logic diagram of the CDP18S622: control portion and optional power supply.

makes use of the user's system power supply to power the entire system; the battery-pack supply power is available to the entire system during a power failure.²

When operating an entire system from the battery pack, it is advisable to install the fourth battery. Installation and linking provisions have been provided on the card.²

Non-Volatile Transport Medium

A second application of the battery-backup Microboard is as a non-volatile transport medium. As stated above, the board can be withdrawn from a system, transferred elsewhere, installed in another system, and run without loss of data. Standard conductive packaging should **not** be used during the transfer as this packaging will short out signal, power, and ground lines on the RAM card and run the batteries down prematurely and/or cause stored data to be lost.

As an example of the use of the battery-backup Microboard as a transport medium,

consider the case of a development system, such as the CDP18S007 or CDP18S008, for which software has been developed using disk storage, keyboard, and display, software that is now to be run on a remote system that has none of these features. It would take time to program an EPROM, which ultimately might develop bugs, for use on the remote system. The backup Microboard software could be easily debugged with the aid of the Micromonitor, however, and the board inserted in the remote system in running condition.

An example of a typical debugging procedure involving use of the battery-backup Microboard is as follows:

1. Substitute the CDP18S622 RAM card for the RAM in the development system that would normally hold the program.
2. Assure that the memory-protect switch is in the off position
3. Down-load the program into the CDP18S622 RAM card.

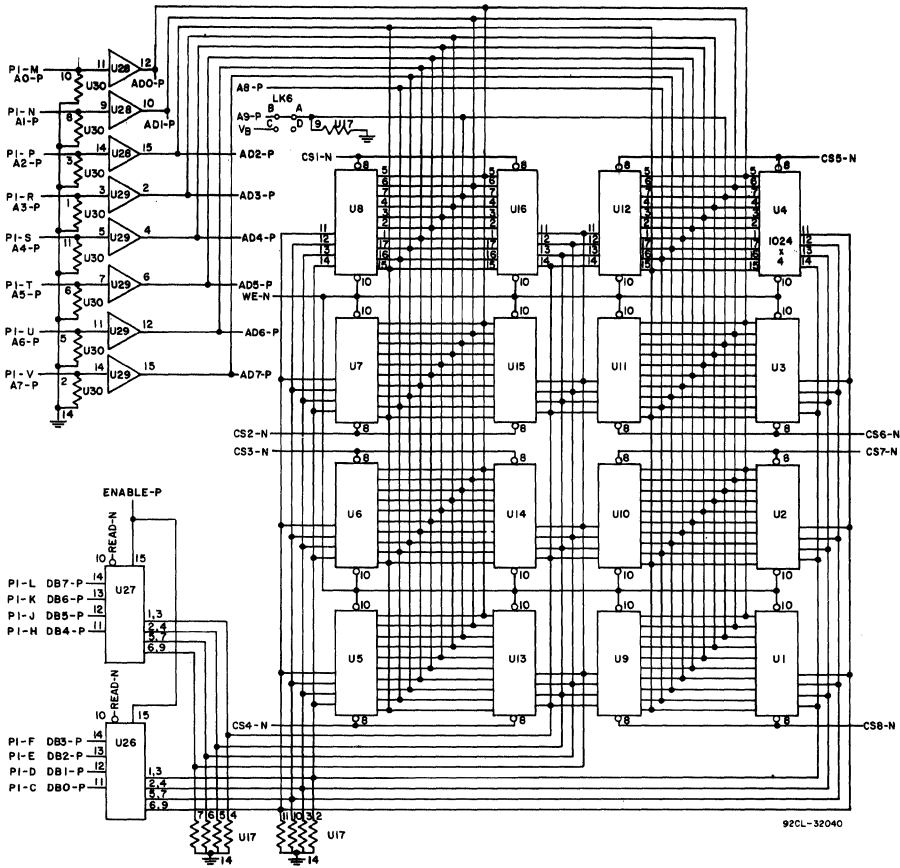


Fig. 4 - Logic diagram for the CDP18S622: memory and buffer portion.

4. Place the memory-protect switch in the on position.
5. Remove the CDP18S622 RAM card from the development system.

Note: Power need not be removed from the development system when removing or installing the battery-backup RAM cards, although power should be removed when reinstalling the original nonbattery-backup RAM. It is recommended that the development system be placed in the step mode, by using the Step/Continuous switch, before removing the CDP18S622 RAM card; system execution will probably halt if this is not done. After removal of the board, place the switch in the continuous mode and execution will resume.

6. Transport the CDP18S622 battery-backup RAM card to the target

system while observing these precautions: Make sure the back of the card does not contact a conductive surface, including Velostat[®] bags. Do not expose the board to large amounts of static electricity.

7. To change the memory address of the CDP18S622 at the target system, simply change the board switch settings³ and install the battery-backup RAM card into the target system. See note under step No. 5, above.
8. Reset the target system and run the transported program. At this point it is important to note that if the transported program requires RAM within the bounds of the battery-backup card, the memory-protect switch will have to be placed in the off position.

³Trademark of 3M Company.

The manual memory-protect switch already mentioned above is a significant advantage of the battery-backup RAM card. To test a program written to reside in a ROM or EPROM, the user need not program an EPROM or have a test-run ROM made since a flip of the switch turns the "read/write" memory card into a "read only" memory card. Bugs can be sought out in a simulated ROM environment.

It is important to note that the battery-backup Microboard is not designed to save an executing program if power fails during operation. If the CPU is executing code in the battery-backup RAM card, and if the card is not already in the memory-protect mode, a power outage will cause the memory read and memory write to be disabled even before the CPU stops executing code. Therefore, a system restart will have to be performed whenever power is lost. However, by using the optional power-up reset capability, adding some hardware to detect power failure, and writing a short, fast, interrupt routine, system status could be saved on the battery-backup board so that on the return of power the system could resume operation precisely where the program was interrupted with the power failure.

Testing Other Boards

When testing newly manufactured or prototype boards, the CDP18S622 battery-backup card is a time-saving device. It

saves the time spent in reloading the test program for each board tested because it saves the program code; this savings can be substantial in the testing of many boards. The procedure given here should be followed when using the battery-backup Microboard in this application:

1. Put the battery-backup board into the memory area where the test program is to reside.
2. Load the test program from disk, paper tape, magnetic tape, etc. This step is the time consumer!
3. Position the memory-protect switch to on and run the test program, noting hardware failures.
4. Turn off the system power and make the necessary repairs.
5. Return the repaired board to the system or put the next card to be tested into the system, turn the power on, and rerun the test.

Again, no program load phase need be performed because the battery-backup card saves the code.

REFERENCES

1. "RCA COSMAC Microboard 8-Kilobyte Battery-Backup RAM CDP18S622," RCA Solid State publication MB-622.
2. See ref. 1 for parts list and directions for powering an entire system from the battery pack.
3. See ref. 1 for switch settings.

ICAN-6957

CDP1804 and CDP1805 Processors Improve System Performance and Lower Chip Count

by J. Paradise

The CDP1804 and CDP1805 processors are RCA's new introductions to the growing CDP1800 family of micro-processor and memory devices. These chips extend the capability of the CDP1802 microprocessor, both in higher performance and additional system functions, while maintaining upward software and hardware compatibility. To give the system designer a choice between flexibility and minimum chip count, two parts are offered: the CDP1805 for moderate cost and off-the-shelf availability, and the premium CDP1804 for custom VLSI system integration.

FUNCTIONAL DESCRIPTION

The CDP1804 is a CMOS, 8-bit, register-oriented micro-computer designed for use in a wide variety of general-purpose computing and control applications. It contains a 2048-byte mask-programmable ROM, 64 bytes of RAM, an 8-bit presettable down counter, and the same architecture as the CDP1802. The CDP1805 is identical to the CDP1804, with the exception that the ROM is left out for reasons of economy and flexibility. Both devices are capable of dc to 4-MHz operation at 5 volts over the commercial temperature range of -40°C to $+85^{\circ}\text{C}$, and both have a voltage-range capability of from 4 to 10.5 volts. These added hardware and performance features, in addition to an enhanced instruction set, make the CDP1804 or CDP1805 a suitable choice for customers up-grading present CDP1802 systems for higher system performance or greater system integration or considering the CDP1800-series family for the first time. A block diagram of the CDP1804/CDP1805 processors is shown in Fig. 1.

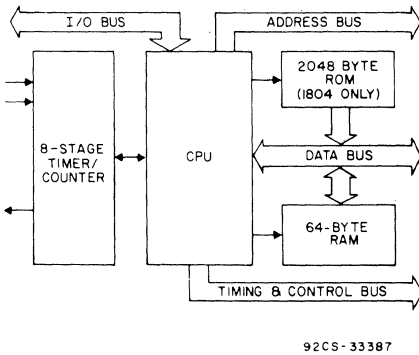


Fig. 1 - CDP1804/CDP1805 block diagram.

CDP1800 Architecture In Summary

This section of the Note is designed for potential CDP1804/05 users unfamiliar with CDP1800-series architectural features, and explores the software and hardware aspects of data transfer and manipulation, and the control and timing interface to support devices. While established users of other 8-bit machines may find this architecture initially perplexing because of the extreme flexibility of assignments within register and memory space, familiarity with the device and its capabilities will pay off in compact code generation and efficient use of memory and I/O in most control-oriented applications.

Scratchpad Register Array - The CDP1804/05 devices provide an indirect means of addressing memory through register assignment. Both devices contain sixteen 16-bit internal scratchpad registers (in addition to 64 bytes of RAM) that are user-programmable as program/subroutine counters, memory pointers, or stack pointers for memory addressing, Fig. 2. In addition, these same registers can hold data transferred to or from the accumulator (D register) by means of software instructions, Fig. 3. Finally, the register contents can be incremented or decremented for software loops or time delays.

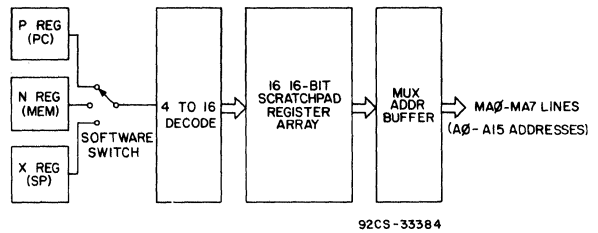


Fig. 2 - Common CDP1802/CDP1804/CDP1805 scratchpad register model, addresses.

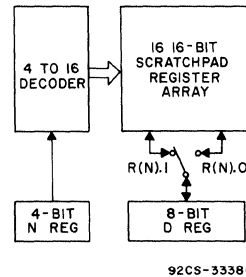


Fig. 3 - Common CDP1802/CDP1804/CDP1805 scratchpad register model, data transfer.

Memory Addressing - When used to address memory, the registers are selected by software instructions that load 4-bit values into register selectors. The 4-bit P register selects a 16-bit scratchpad as the program counter, the 4-bit X register selects a scratchpad as the stack pointer, and the 4-bit N register selects a scratchpad as the memory pointer during an external data transfer, or as an operand during an internal data transfer (register-accumulator, register-register). Note that all 16 scratchpads are capable of addressing any of the 64K memory locations available to the CDP1804/05 processors; thus, stack space is unrestricted, scratchpads can point to subroutines located anywhere in memory space, and most registers can be dedicated for specific data storage or addressing tasks during subroutine or interrupt processing.

Addressing Modes - As a result of this register-oriented structure, addressing modes include direct, paged direct, immediate, indirect, and inherent. The direct mode applies to all branch instructions, which can cause conditional or unconditional jumps within the current page or anywhere in memory space. Indirect addressing allows transfer of data to or from either general memory or stack, depending on user allocation of memory space. The inherent mode allows for internal register modification with external memory inactive.

Instruction Set - In addition to memory reference instructions, the CDP1804/05 instruction set has a full complement of arithmetic and logic operations, conditional page and long branch instructions that test the contents of the accumulator and carry flag, register instructions that modify the contents of the internal scratchpads or register selectors, and I/O and interrupt handling instructions, Table I.

Table I — Breakdown of 91 Instructions Common to CDP1802/04/05

MEMORY TRANSFER	7
INTERNAL REGISTER	7
LOGIC	10
ARITHMETIC	12
UNCONDITIONAL JUMPS	4
CONDITIONAL JUMPS	27
CONTROL	7
INTERRUPT CONTROL	3
I/O TRANSFER	14

92CS-33392

Bus Structure - The CDP1804/05 processors use a multiplexed address bus to address external memory. The high byte is generated first, with a TPA pulse provided to latch the byte into an external latch or a bus-compatible memory-support chip. The data bus is nonmultiplexed, with a TPB pulse provided to latch stable data into an I/O device. A separate, 3-bit, I/O address bus provides address selection for external peripheral chips.

Control and Status Pins - Control pins are provided for DMA transfer (with register R(0) as the DMA counter), interrupt requests (with register R(1) storing the interrupt vector), four testable flags, and a single-bit software-controlled output port. Two state code lines provide machine status. Separate READ and WRITE signals are provided for memory control, Fig. 4.

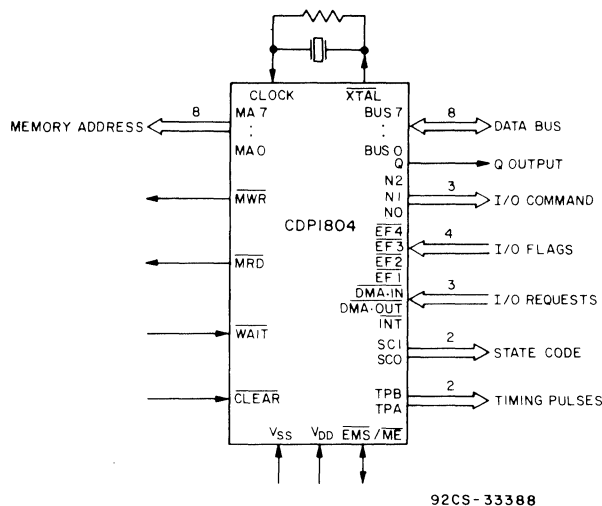


Fig. 4 - CDP1804/CDP1805 functional pinout.

Execution Speed - System timing is derived from an external crystal. The crystal is divided so as to generate eight clock cycles per machine cycle. A single FETCH and EXECUTE (16 clock cycles total) is all that is required for two-thirds of the instructions available with the CDP1804/05 devices. The result is a minimum instruction time of four microseconds at maximum frequency, Fig. 5.

CDP1800 ARCHITECTURE - USER ADVANTAGES

The register-based orientation of the CDP1800-series architecture discussed in the previous section is its dominant feature, and one not commonly found in the world of microprocessors. Once the user has mapped out a plan to assign registers to perform specific tasks, this flexibility provides him with a rich variety of software techniques for performing his required function. Flexible register assignment results in such structures as multiple program counters for quick subroutine calls, multiple stack pointers for independent data and I/O stacks, and multiple memory pointers that facilitate memory data transfers and the adaptability of the microprocessor to interpretive languages.

In addition, all CDP1800-series processors possess some unique hardware features that reduce system parts count and speed data transfer. The on-chip DMA counter allows the CPU section to generate addresses and control signals for memory/I/O transfer in either a continuous or cycle-stealing mode. This DMA feature also allows the user to perform a real-time clock function without tying up significant software or execution time, Fig. 6. The I/O structure allows for stack transfer of data directly to and from memory, again bypassing the CPU in the process, and the special I/O lines, Flags and Q, allow for software polling of external events and single-bit output control, as well as the use of the lines in combination for bit-banging serial I/O.

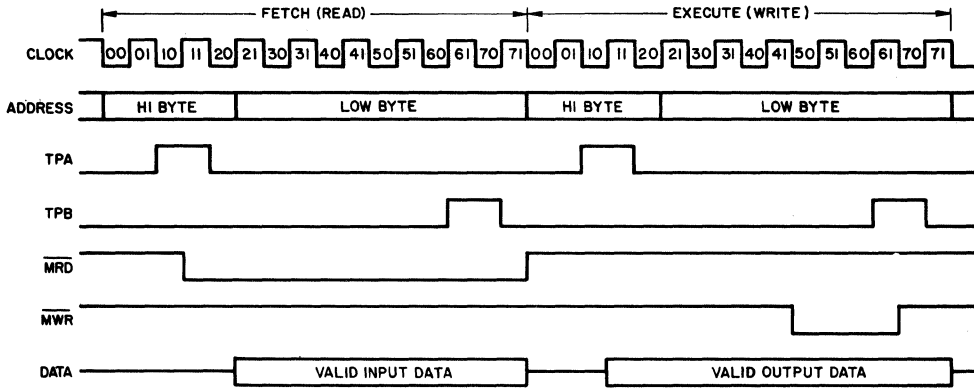
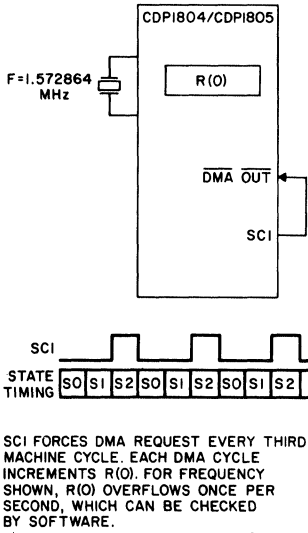


Fig. 5 - Basic dc timing diagram, one instruction cycle.

92CM-33382



92CS-33389

Fig. 6 - Application of DMA feature to generate real-time clock reference.

CDP1800 SERIES HARDWARE/SOFTWARE SUPPORT

An equally important advantage to potential CDP1804/05 users, besides architectural performance capabilities, is the availability of a complete line of compatible memory and I/O devices for efficient hardware designs, and easy-to-use tools to aid software development. The CDP1800-series is presently the broadest CMOS LSI line in the industry, and the software and debug support available allows users to reduce the significant software burden required to program a microprocessor family.

The list of support devices, Table II, for the CDP1800-series, which includes the CDP1802, CDP1804, and CDP1805 because of their compatibility, is highlighted by 12 RAM devices with capacities of from 32 to 4096 bits; ROM chips compatible with the CDP1800-series multiplexed bus structure, with user-programmed address decoders that uniquely define memory space without external decoding, and with identical pinout for system upgrade without board changes; EPROM's for system prototyping;

Table II — Major CDP1800-Series Support Components

RAMS		SIMPLE I/O	
CDP1821	1K X 1	CDP1852	I/O PORT
CDP1822	256 X 4	CDP1853	DECODER
MWS5101	256 X 4	CDP1856	BUFFER
CDP1823	128 X 8	CDP1857	BUFFER
CDP1824	32 X 8	CDP1858	LATCH
CDP1825	1K X 4	CDP1859	LATCH
MWS5114	1K X 4	CDP1863	COUNTER
* CDP1826	64 X 8	CDP1866	LATCH
		CDP1867	LATCH
		CDP1872	I/O PORT
		CDP1873	DECODER
		CDP1874	I/O PORT
		CDP1875	I/O PORT
ROMS		COMPLEX I/O	
CDP1831	512 X 8	CDP1851	PROG I/O
CDP1832	512 X 8	CDP1854A	UART
CDP1833	1K X 8	CDP1855	MDU
CDP1834	1K X 8	CDP1869	CRT CONTROLLER
* CDP1835	2K X 8	CDP1870	CRT CONTROLLER
* MWS5316	2K X 8	CDP1871	KEYBOARD ENCODER
		CDP1876	CRT CONTROLLER
		* CDP1877	INTERRUPT CONTROLLER
EPROMS			
CDP18U42	256 X 8		
* MWS57U58	1K X 8		

* RESERVED NUMBER FOR 1981 PRODUCTION DEVICES

92CS-33383

and a wide variety of I/O devices ranging from simple buffers and latches to complex peripherals such as multiport programmable I/O, UART's, math chips, and CRT controllers.

Software support is available from products that range from simple prototyping kits to complete development systems. High-level languages, including microFORTH, PL/M, and BASIC, are available to reduce software development time. In-circuit emulation, through the use of the "Micromonitor," provides comprehensive debug and evaluation capability, either in conjunction with a development system or for standard-alone use in the field. Finally, a complete line of CMOS single-board computers and peripheral subsystems, designed around products in the 1800-series family, are available to further ease the cost and the turn-around time of hardware implementation, Table III.

Table III — Major CDP1800-Series System Support

DEVELOPMENT SYSTEM

- COSMAC SYSTEM IV (CDP18S008)
 - CRT-BASED SYSTEM CONTAINING:
 - 64K MEMORY
 - DUAL FLOPPY-DISK DRIVES
 - IN-CIRCUIT EMULATION (MICROMONITOR)
 - BUILT-IN PROM PROGRAMMER
 - COMPLETE DISK OPERATING SYSTEM
 - LEVEL I, LEVEL II ASSEMBLER
 - MACROASSEMBLER
 - FULL-SCREEN EDITOR AND TEXT EDITOR
 - MICROMONITOR OPERATING SYSTEM (MOPS)
 - PROM-PROGRAMMER OPERATING SOFTWARE

SOFTWARE (OPTIONAL)

- BASIC1 (FIXED POINT) INTERPRETER/COMPILER
- BASIC2 (FLOATING POINT) INTERPRETER
- PLM-1800 COMPILER
- FIXED AND FLOATING POINT MATH SUBROUTINES

MICROBOARDS — SINGLE BOARD MICROCOMPUTERS

- COORDINATED SET OF COMPUTER, MEMORY AND I/O BOARDS
 - CDP18S606-1804/1805 EVALUATION BOARD CONTAINS:
 - 1804 CPU W/2.47 MHZ CLOCK
 - 2K BYTES RAM (FOR 1804 ROM SIMULATION)
 - 2 EACH ROM/EPROM SOCKETS
 - 2 EACH PARALLEL I/O PORTS
 - RS232C SERIAL PORT (UART)

CDP1804/05 Enhancements

The preceding discussion has dealt with features and advantages that are common to the CDP1802, CDP1804, and CDP1805. The following sections describe specific enhancements to the CDP1802: increased memory, timer-counter implementation, standard call and return instructions, and enhanced 16-bit data manipulation, all of which should be of particular usefulness to those who are already familiar with CDP1802 capabilities and the advantages of hardware and software enhancements in their system designs.

Memory

A significant feature of the upgraded CDP1804 is its memory expandability, which is not compromised as in some other single-chip microprocessors. The same 64K memory address space is available, with 2K of ROM and 64 bytes of RAM on-board. The large on-board ROM size is sufficient for many application programs, and it can hold special firmware such as the CDP18S827 floating-point binary arithmetic subroutine, or a budget interpreter such as TINY BASIC. The internal RAM provides enough locations for stack and auxiliary scratchpad usage; data-acquisition applications can take advantage of the larger memory address space for outboard RAM. Note that both internal ROM and RAM have mask-programmable address spaces, with an EMS signal provided to indicate when external memory is being addressed.

The CDP1805 has the same RAM complement and expandability features as the CDP1804. Since the device is an off-the-shelf part, its RAM is accessed through a \overline{CE} input that replaces the CDP1804 EMS output. In general, the user will find the absence of ROM on the CDP1805 an advantage in many system applications because of the trend to larger and larger ROM programs in increasingly sophisticated systems. The absence of the ROM allows the user great

flexibility in designing his system, and may well provide the most cost-effective approach for the majority of applications. Thus, the CDP1805 should be chosen when anticipated ROM program space exceeds that of the capacity of the CDP1804, when prototyping an experimental application with EPROM, when the application software can change, or when the user is willing to trade off cost for increased chip count.

Timer/Counter

An additional hardware feature of the CDP1804/05 devices, besides on-board memory, is an 8-stage presetable down counter, Fig. 7. This is a full-function timer/counter, with

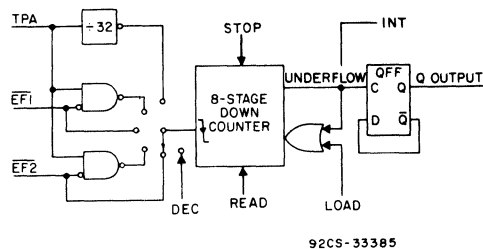


Fig. 7 - CDP1804/CDP1805 timer/counter model.

inputs available from an external source or a scaled internal clock, and output overflow indication through the external Q line or an internal counter interrupt request. The counter makes use of ten linked opcode instructions to perform its control functions and to program the counter for its operational modes. LOAD, READ, STOP, and DECREMENT control instructions provide manual control and allow the counter to be used to generate a programmable time delay.

Three running modes can be programmed with five additional instructions; the modes are:

1. **TIMER** - with the input derived from the CDP1804/05 TPA pulse divided by 32, which allows its use as a time base for real-time applications.
2. **EVENT COUNTER** - with the input from 1 of 2 flag lines (software selectable), for single or dual-channel event counting.
3. **PULSE DURATION MEASUREMENT** - with the input again from a flag line, with the counter value representing the pulse width at the input. Because two inputs can be sequentially applied, comparison measurements can be made.

In conjunction with each of these modes, a tenth instruction, ETQ, can generate a programmable square wave or provide control to external peripherals by toggling the Q output on every counter overflow.

In addition to the ten timer/counter functions, six additional instructions allow arbitration between external and counter interrupts. Two instructions each are provided for interrupt masking and unmasking, while two others allow for software polling to determine the interrupt source.

Standard Call and Return

The most significant software enhancement of the CDP1804/05 devices over the CDP1802 is the addition of single CALL and RETURN instructions, which save both software and time, and free-up internal scratchpad registers for other uses, Table IV. These instructions, SCAL and SRET, are slightly different than those of other microprocessor families, primarily to allow the user to pass in-line data from the main program or calling routine to the target subroutine. Direct addressing is incorporated as with other families; however, the main program counter is exchanged with a designated scratchpad, and it is the scratchpad contents that are saved on the stack. This technique allows the original PC to point to data following the call instruction without having to exchange pointers on the stack.

The SCAL and SRET instructions eliminate the need for dedicated call and return subroutines, as required with the CDP1802, as well as the need to allocate two registers to point to these subroutines. However, as with the CDP1802, the most efficient subroutine call for small programs that

do not use nested subroutines is still the SEP instruction, which uses the flexibility of register reallocation to switch program counters from main program to subroutine with one byte of code.

16-Bit Data Transfer

Sixteen-bit data transfers can be easily implemented on all CDP1800-series processors because of the presence of the 16-bit-wide scratchpad register array. In addition to 16-bit register increments and decrements, arithmetic and shift operations can be performed on 16-bit operators stored in the scratchpad registers with a sequence of CDP1804/05 software instructions. An additional hardware feature results from the CDP1804/05 I/O structure, which allows 16-bit data transfer in one machine cycle from scratchpad register R(X) to the address bus, with I/O control lines active to distinguish between this special I/O transfer and normal memory-address operations.

In addition to the above features common to the CDP1802, CDP1804, and CDP1805, the CDP1804/05 devices have four new instructions that supplement the 16-bit data-transfer operations:

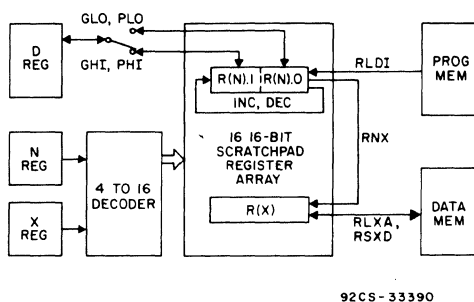
1. A register-load instruction that permits direct loading of any 16-bit scratchpad from two consecutive immediate program locations (RLDI).
2. A register-to-register transfer instruction that permits data transfer from any scratchpad to the R(X) stack pointer (RNX).
3. A register-to-memory transfer instruction that stores any register's contents into two consecutive memory locations (RSXD).
4. A memory-to-register transfer instruction that loads two consecutive memory locations into any scratchpad register (RLXA).

These additional instructions can be used to manipulate either 16-bit data or addresses on the CDP1804/05 devices because of the dual address/data capability of the scratchpad register array. Note that the accumulator (D register) is not involved in any of these new transfer operations, permitting its contents to be preserved without the need for additional manipulations. The enhanced data-transfer capability is illustrated in Fig. 8.

Table IV — Performance Comparisons-Subroutine Call and Return for CDP1802/04/05 Implemented with Various Techniques

	1802 SOFTWARE "SCRT" TECHNIQUE	1802 SOFTWARE "SEP" TECHNIQUE	1804/05 SOFTWARE SCAL / SRET INSTRUCTIONS	1804/05 SOFTWARE "SEP" TECHNIQUE
NUMBER OF MACHINE CYCLES - CALL	32	2	10	2
NUMBER OF MACHINE CYCLES - RETURN	24	4	8	4
CALL TIME (1802 @ 3.2 MHz) (1804/05 @ 4 MHz)	80 μs	5 μs	20 μs	4 μs
RETURN TIME (1802 @ 3.2 MHz) (1804/05 @ 4 MHz)	60 μs	10 μs	16 μs	8 μs
NUMBER OF BYTES SOFTWARE CALL + RETURN	45	4	6	4

92CS-33391



92CS-33390

Fig. 8 - Enhanced scratchpad register data-flow model unique to CDP1804/CDP1805. Compare this figure with Fig. 3.

WHY WAS I/O LEFT OUT?

The discussion of architectural details presented thus far requires mention of a significant functional block that is not contained within the CDP1804/05 device: a complement of full 8-bit I/O ports. The foremost design goal of the CDP1804/05 processors was to make them upward compatible in both software and hardware with the existing CDP1802 to ease the upgrade transfer for the system designer. This constraint eliminated any chance of adding I/O ports on-board. If the decision to add I/O at the expense of hardware compatibility had been made, then expandability would have been compromised. Since the power of

the CDP1804/05 devices lies within the 16-bit scratchpad array, a full 64K of addressable memory allows this power to be utilized to the fullest.

Some limited I/O is already present on the CDP1804/05 units in the form of the Flag and Q lines. Sixteen-bit data transfer via the address bus is another form of I/O that has already been discussed. External I/O can easily be mapped with CDP1804/05 I/O bus structure, and a wide variety of peripheral devices are available for this purpose. Finally, the high-volume customer can choose to integrate a custom I/O device that can be tailored to his specific needs and that can interface directly with the CDP1804/05 processors.

CONCLUSIONS

The CDP1800-series has been designed into a wide variety of new and existing applications, Table V; this has not been by accident. The announcement of the CDP1800 product line was made in 1976; products in the line are available in high volume. Users have taken advantage of the low power, traditional CMOS features of the line; its architecture, I/O handling capability, and ease of implementation have made the family popular in control-oriented applications. The addition of the more powerful CDP1804 and CDP1805 processors and an increasing array of memory and support chips will encourage users to develop increasingly sophisticated systems around the CDP1800 family, and should attract newcomers to this versatile, broad-based, well-established line.

Table V — CDP1800-Series Microprocessor Applications

SET BACK THERMOSTAT	BOARD TESTERS	MOTOR CONTROL
PORTABLE AIR QUALITY MONITOR	UTILITY METER	MINI PBSX
INDUSTRIAL POWER MONITOR	COIN CHANGER	HAND HELD MEDICAL TERMINAL
ENERGY MEASUREMENT	ARTIFICIAL BREAST	SPACE FLIGHT TAPE RECORDER
LOAD MANAGEMENT (ENERGY MANAGEMENT)	TEST INSTRUMENTS	MULTI LINK INTERCOM
PORTABLE METER READER	IRRIGATION CONTROLLER	AUTO ANTI-THEFT
PORTABLE BILL CALCULATOR	SALMON COUNTER	INDUSTRIAL CONTROLLERS (NUMERIC MACHINE CONTROL)
PORTABLE NOISE LEVEL MEASUREMENT	SEISMOGRAPH	REMOTE, HAND HELD, DATA ENTRY TERMINAL
AUTOMOTIVE SPARK CONTROL	ENVIRONMENTAL CONTROLS	AIRBORNE FLIGHT TEST INSTRUMENTATION
AUTOMOTIVE FUEL CONTROL	REMOTE GAUGES AND METERS	AIRCRAFT ATTITUDES DISPLAY
MILITARY RADIO COMMUNICATION	NAVIGATIONAL CONTROLS	FLOOD WATER MONITOR
ELECTRIC CAR	DATA ACQUISITION SYSTEMS	MOTOR-GENERATOR CONTROL SET
MOBILE TELEPHONE	CREDIT CARD VERIFIER	PORTABLE OIL EXPLORATION EQUIPMENT
ULTRASONIC WELD INSPECTION	SONAR SYSTEMS	HOME COMPUTER
HAND HELD MEDICAL MONITOR	FILM SPLICER	AIRCRAFT REMOTE CIRCUIT BREAKER CONTROL
FLOW METER	SOLAR POWERED DATA LOGGER	SOLAR WATER HEATER CONTROLLER
HOME SECURITY (FIRE & INTRUSION)	PORTABLE GAS ANALYZER	BEER TAP CONTROLLER
LOGGING (WELL DRILLING)	KIOSKS	TV GAMES
MISSILES	DEEP FAT FRYER	LANGUAGE TRANSLATOR
AUTOMOTIVE DIAGNOSTICS	VENDING MACHINE	SURVEY EQUIPMENT
TELEPHONE DIALER	COPIER CONTROL	DIVING EQUIPMENT
LINE PRINTER CONTROLLER	DIGITAL TUNING	UNDERGROUND MINE TELEPHONE
PRINTING ELAPSED TIME COUNTER	VEHICLE DIAGNOSTICS	
FREQUENCY SYNTHESIZER	TV CAMERA	
BATTERY CHARGER CONTROLLER		

New CMOS CDP1800-Series Processors Reduce Chip Count

by J. Paradise

INTRODUCTION

In today's world of microminiaturization, there is a constantly increasing emphasis on portability and compactness in microprocessor-based designs for commercial and industrial applications. These factors can be satisfied by two new CMOS processor introductions from RCA, namely, the CDP1804 and CDP1805. These devices combine CPU, memory, and peripheral functions on a single chip, and provide a compact system design with the additional portability offered by battery-operation.

THE CDP1804

The CDP1804 is an 8-bit microcomputer enhancement of the well-established CDP1802 register-oriented microprocessor. The CDP1804 has the same architectural features as the CDP1802, with the addition of a 2048-byte mask-programmable ROM, 64 bytes of RAM, an 8-bit presetable timer/counter, and an enhanced instruction set for greater versatility and improved data handling. Process improvements have enhanced the performance of the CDP1804 over that of the CDP1802: speed capability is increased to dc to 4 MHz at 5 volts over the commercial temperature range of -40°C to $+85^{\circ}\text{C}$, with an operating voltage range capability of from 4 to 10.5 volts. Because the ROM in this device is tailored to a specific customer requirement, the device should be viewed as a premium part suitable for custom VLSI integration, where minimum chip count is the overriding factor in the system design. A block diagram of the CDP1804 is shown in Fig. 1.

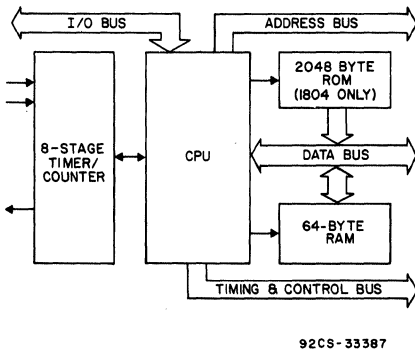


Fig. 1 - CDP1804 block diagram.

THE CDP1805

The CDP1805 is identical to the CDP1804, with the exception that the ROM is left out for reasons of economy and flexibility. In general, because of the trend to larger and larger ROM programs in new system designs, the absence of ROM on the CDP1805 should be viewed as an advantage. Greater design flexibility is provided for cases where anticipated ROM program space is undefined, where ROM patterns may change, where the substitution of only a ROM can generate a new end product, or where experimental applications are to be prototyped with EPROM. Even though total chip count may increase, the system cost may be reduced: two smaller chips can often generate higher equivalent yields than a single large die.

The diagrams in Fig. 2 show the equivalent implementation of a microcomputer function using the CDP1802, CDP1804, and CDP1805 in the system design.

CDP1804/CDP1805 FEATURES AND ENHANCEMENTS

Internal Architecture - The architectural features of the CDP1804 and CDP1805 are identical to those of the CDP1802. Internally, an array of sixteen 16-bit scratchpad registers provides control over memory addressing and internal housekeeping. The user, employing software control, indirectly programs this scratchpad by using 4-bit register designators (P,N,X). The scratchpad registers can be programmed as program counters, memory pointers, stack pointers, or storage locations for variable data. Once the user has mapped out a register assignment plan, a rich variety of software techniques are possible. Multiple program counters for quick subroutine calls, multiple stacks for I/O, data, and context switching, and multiple memory pointers for low-software-overhead data transfers can be facilitated.

The D register (accumulator) is used in most data transfer and arithmetic operations. Internal and I/O control flags facilitate software test and branch operations. The internal structure of the CDP1804 and CDP1805 allows a variety of software instruction types and memory addressing modes, as shown in Tables I and II.

External Interfacing - Externally, the CDP1804 and CDP1805 have the same bus and control structure as the CDP1802, right down to a nearly identical pinout arrangement, Fig. 3. An 8-bit multiplexed address bus is used to generate one of 64K possible addresses to internal and external memory; a separate bidirectional bus is used for data transfer. The I/O structure is unique, with a separate I/O address bus, four flag inputs and one interrupt input, and a single Q-bit output port. The I/O structure allows for stack transfer of data directly to and from memory, while the flag and Q lines can be used as I/O pins under software

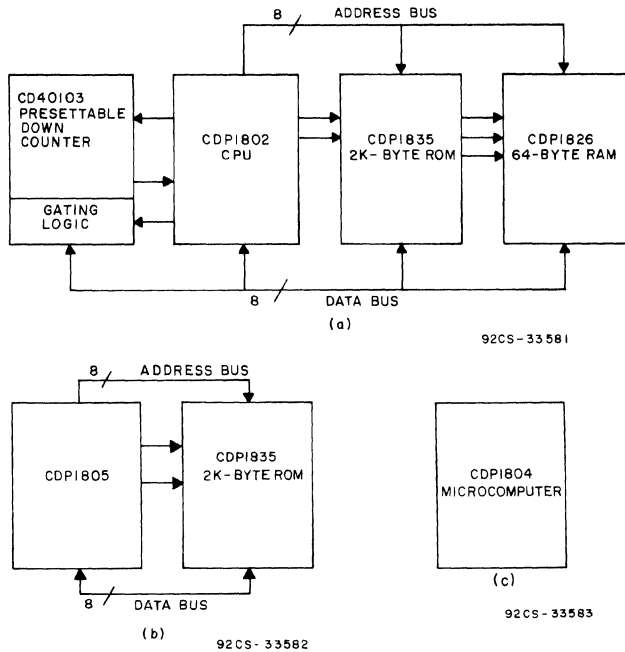


Fig. 2 - Equivalent microcomputer implementation:
 (a) CDP1802-based design, (b) CDP1805-based design,
 (c) CDP1804-based design.

Table I - Breakdown of 91 Instructions Common to CDP1802/CDP1804/CDP1805

MEMORY TRANSFER	7
INTERNAL REGISTER	7
LOGIC	10
ARITHMETIC	12
UNCONDITIONAL JUMPS	4
CONDITIONAL JUMPS	27
CONTROL	7
INTERRUPT CONTROL	3
I/O TRANSFER	14

Table II - CDP1804/CDP1805 Addressing Modes

DIRECT	- FOR JUMPS ANYWHERE IN MEMORY SPACE
PAGED DIRECT	- FOR JUMPS WITHIN A CURRENT PAGE
IMMEDIATE	- FOR DATA TO ACCUMULATOR OR SCRATCHPADS
INDIRECT	- FOR ALL MEMORY ADDRESSING
INHERENT	- FOR INTERNAL REGISTER CONTROL

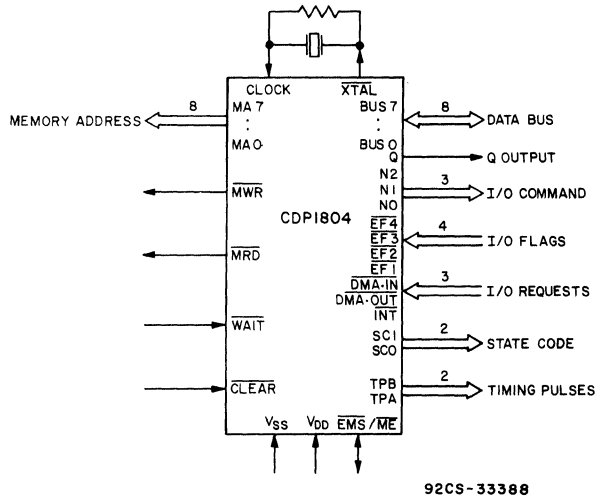


Fig. 3 - CDP1804/CDP1805 functional pinout.

control, or in serial I/O applications. In addition to the above, DMA request inputs are provided for control of an on-board DMA counter. The DMA feature allows the CPU section to generate address and control signals for memory/I/O transfer in a continuous or cycle-stealing mode, and can also be used to provide a real-time clock function under hardware-update control.

Hardware Enhancements - The chip count savings in using the CDP1804 or CDP1805 instead of a CDP1802 (or equivalent 8-bit micro) results from the on-board memory and timer/counter function. The CDP1804 on-board ROM is sufficient for many applications; it can alternately hold special firmware such as math routines, video display character generator sets, or a scaled-down high-level-language interpreter. The RAM on the CDP1804 and CDP1805 provides enough storage locations for stack and variable data in minimum systems, with the memory addressing capability useful for off-board RAM in memory-intensive applications.

The full-function timer-counter, Fig. 4, is an 8-stage preset-

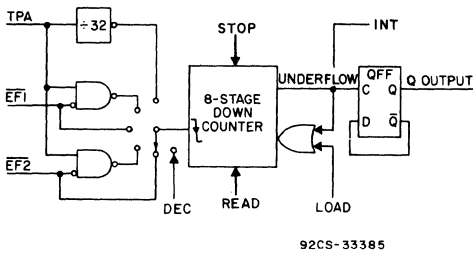


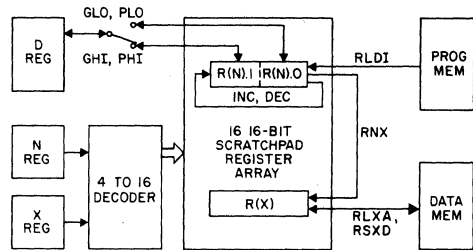
Fig. 4 - CDP1804/CDP1805 timer/counter model.

table down counter, with inputs available from an external source or a scaled internal clock. An output is available via the Q line or in software through the use of an internal counter interrupt request. Sixteen CDP1804/CDP1805 instructions are provided for counter mode control, manual counter operation, and arbitration between external and counter interrupts. The instructions allow the counter to be used to count iterations in software loops, for time-base referencing, for external-event counting, or for pulse-width measurements.

Software Enhancements - Six instructions additional to the sixteen timer-counter instructions are used for improved CDP1804/CDP1805 throughput. Single CALL and RETURN instructions are provided to save software time and free-up internal scratchpad registers for other uses. In comparing these instructions against equivalent techniques using the CDP1802, a 5:1 improvement in execution speed is realized.

The remaining four instructions enhance the 16-bit data transfer capability of the CDP1804/CDP1805 by extending the capability of the 16-bit wide scratchpad register array. New instructions have been added for immediate 16-bit scratchpad loading, 16-bit register-to-register transfers, and 16-bit register-to-memory and memory-to-register transfers. Fig. 5 shows a model of scratchpad register-based data transfer.

The twenty-two new instructions of the CDP1804 and CDP1805 processors, Table III, by allowing the user to realize savings in both software and execution speed, match these new processors to applications in which code length or throughput may have precluded the use of the CDP1802.



92CS-33390

Fig. 5 - Enhanced scratchpad-register data-flow model unique to CDP1804/CDP1805.

Table III - New CDP1804/CDP1805 Instructions

TIMER/COUNTER

- LDC - LOAD
- GEC - READ
- STPC - STOP
- DTC - DECREMENT
- STM - TIMER MODE
- SCM1 - EVENT COUNTER VIA EF1
- SCM2 - EVENT COUNTER VIA EF2
- SPM1 - PULSE WIDTH VIA EF1
- SPM2 - PULSE WIDTH VIA EF2
- ETQ - COUNTER OUTPUT VIA Q

INTERRUPT CONTROL

- XIE - ENABLE EXTERNAL INT
- XID - DISABLE EXTERNAL INT
- CIE - ENABLE COUNTER INT
- CID - DISABLE COUNTER INT
- BCI - BRANCH ON COUNTER INT
- BXI - BRANCH ON EXTERNAL INT

SUBROUTINE AND DATA TRANSFER

- SCAL - STANDARD CALL
- SRET - STANDARD RETURN
- RLDI - REGISTER LOAD IMM.
- RNX - REG. TO REG. TRANSFER
- RLXA - MEM. TO REG. TRANSFER
- RSXD - REG. TO MEM. TRANSFER

CONCLUSIONS

The potential user of a CDP1804 or CDP1805 should look beyond the specific architectural details of the device in designing a minimum-parts-count system. These two devices are members of a complete family of devices supplemented by easy-to-use tools that aid in system and software development. The versatile devices in the line make it possible for the user to realize efficient hardware designs, and the software and debug support available allows the user to reduce the significant software burden required to program a microprocessor family.

The user can look with confidence to this high-volume, established family from which the wide variety of new and existing applications shown in Table IV have been designed. In the past, the architecture and I/O features of the CDP1802 have supplemented its traditional low-power, high-noise-immunity characteristics in control-oriented designs. The CDP1804 and CDP1805 are extensions of this family, and these enhanced devices, along with new, high-density memories and complex I/O support devices, should continue to make the CDP1800 family a primary choice in future CMOS microprocessor-based systems.

**Table IV - A Sampling of Typical
CDP1800-Series Microprocessor Applications**

- SET BACK THERMOSTAT
- PORTABLE AIR QUALITY MONITOR
- INDUSTRIAL POWER MONITOR
- ENERGY MEASUREMENT
- LOAD MANAGEMENT (ENERGY MANAGEMENT)
- PORTABLE METER READER
- PORTABLE BILL CALCULATOR
- PORTABLE NOISE LEVEL MEASUREMENT
- AUTOMOTIVE SPARK CONTROL
- AUTOMOTIVE FUEL CONTROL
- MILITARY RADIO COMMUNICATION
- ELECTRIC CAR
- MOBILE TELEPHONE
- ULTRASONIC WELD INSPECTION
- HAND HELD MEDICAL MONITOR
- FLOW METER
- HOME SECURITY (FIRE & INTRUSION)
- LOGGING (WELL DRILLING)
- MISSILES
- AUTOMOTIVE DIAGNOSTICS
- TELEPHONE DIALER
- LINE PRINTER CONTROLLER
- PRINTING ELAPSED TIME COUNTER
- FREQUENCY SYNTHESIZER
- BATTERY CHARGER CONTROLLER
- SONAR SYSTEMS
- FILM SPLICER
- SOLAR POWERED DATA LOGGER
- PORTABLE GAS ANALYZER
- KIOSKS
- DEEP FAT FRYER
- VENDING MACHINE
- COPIER CONTROL
- DIGITAL TUNING
- VEHICLE DIAGNOSTICS

Understanding and Using The CDP1855 Multiply/Divide Unit

by G. Johnson

The CDP1855 Multiply/Divide Unit,¹ MDU, can be an efficient hardware replacement for the software-only implementation of arithmetic and signal-processing algorithms. The software is required with most 8-bit microprocessors because they are fabricated without multiply and divide instructions. Software routines that perform multiply and divide operations are generally complex, and involve valuable memory space and time in their implementation. In applications that require many arithmetic operations, where real-time execution speed and code compactness are important, as in signal processing algorithms involving sums of products, the CDP1855 can effectively perform a mathematical task in hardware to supplement the capability of the system processor, and thereby eliminate the need for elaborate software routines and improve system speed and capability. Up to four CDP1855's can be cascaded for multiple-precision operations. The MDU is fabricated by means of the CMOS technology, which means that it can offer dramatic power savings when used in place of math chips of other technologies.

GENERAL DESCRIPTION

The CDP1855 MDU has three 8-bit registers (X, Y, and Z), which are loaded with operands prior to the multiply or divide operation, and which contain a product or quotient when the process is complete. In addition, an 8-bit control register defines and initiates the operation, while a single-bit status register (bit 0) is used for overflow indication (also available at the CO pin). The MDU may be addressed in I/O or memory space.

Addressing—I/O Space

If connections are made as shown in Fig. 1, the MDU is addressed as an I/O device, and responds to input and output instructions. Table 1 lists the I/O instruction codes and N line addresses that access the various MDU registers.

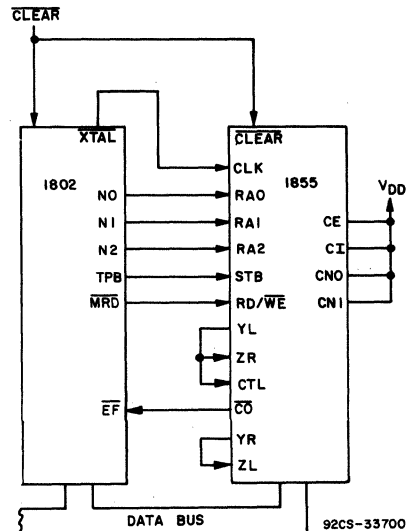


Fig. 1 — Circuit configuration for MDU addressed as an I/O device.

Addressing—Memory Space

Fig. 2 shows the required connections for memory-mapped addressing of the MDU. MWR is used as the read/write signal, and address bits (latched high address bits if used) drive the RAO, RA1, and RA2 pins. Certain non-memory instructions, such as PHI and GLO, cause the contents of internal CPU registers to appear on the address bus. As a result, the read and write signals (MRD and MWR) are OR'ed to produce a valid chip-enable signal (CE) only during actual memory operations.²

Control Byte

The control-byte format is shown in Fig. 3. Bits 0 and 1 define the type of operation: multiply, divide, or no operation. If bit 2 is a logic 1, the contents of the Z register will be reset to zero. Similarly, if bit 3 is a logic 1, the contents of the Y register will be reset to zero. Bits 4 and 5 define the number

Table 1 — I/O Instruction Codes and N Line Addresses Accessing Various MDU Registers

CDP1855 OPERATION	I/O INST.	C2	N2	N1	N0	R/W
Load X register	64	1	1	0	0	0
Load Z register	65	1	1	0	1	0
Load Y register	66	1	1	1	0	0
Load Control Byte	67	1	1	1	1	0
Read X register	6C	1	1	0	0	1
Read Z register	6D	1	1	0	1	1
Read Y register	6E	1	1	1	0	1
Read Status Byte	6F	1	1	1	1	1
No-Operation	XX	0	X	X	X	X

X = Don't Care

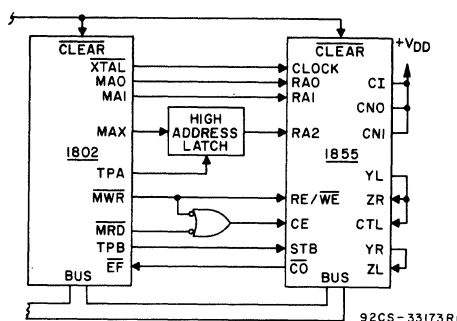


Fig. 2 — Required connection for memory-mapped addressing of the MDU.

BIT	7	6	5	4	3	2	1	0
	CLOCK PRESCALER	RESET SEQUENCE COUNTERS	TWO BIT CODE FOR NUMBER OF MDU'S USED		CLR Y	CLR Z	DIV	MULT.

Fig. 3 — Control-byte format.

92CS-33695

of MDU units in cascade. A logic 1 in bit 4 will reset the internal sequence counters. Bit 7 selects the built-in clock prescaler, which divides by 2, 4, or 8 depending on the number of MDU's cascaded.

Sequence Counters

When multiple units are used, the loading and reading of each X register is sequential: the first selection of the X register accesses the most significant CDP1855; the second time the X register is selected, the X register of the next significant unit is accessed, and so on. Y, Z, and the status registers are all independently sequenced in the same manner. Before loading the X, Y and Z registers with operands, the sequence counters must be reset by loading a control byte with a logic 1 in bit 6, or by clearing the device through the use of the clear pin.

The control registers do not sequence, but are all loaded at the same time.

MULTIPLY OPERATION

In a multiply operation, the 8-bit operands are loaded into the X and Z registers. When the operation is complete, the product is in the Y and Z registers, with Y being the more significant half and Z the less significant half; the X register will be unchanged after the operation. The original contents of the Y register are added to the product of X and Z. Bit 3 of the control byte will reset Y to zero if no addition factor is desired.

The above operation can be described by the following formula:

$$X * Z + Y \rightarrow YZ$$

Multiply Example

An 8x8-bit hardware multiply operation can be performed with the CDP1800-based system shown in Fig. 1 by using the program shown in Fig. 4. The first portion of the program allocates registers of the CPU for memory and stack pointers. The subroutine begins with the loading of the control registers through the execution of an OUT 7 instruction followed by an immediate byte, in this case F0 (11110000). Logic highs on bits 4 and 5 define the number of cascaded MDU's as one. Bit 6 resets the internal sequence counters and bit 7 selects the optional clock prescaler, which causes the shift frequency to be one-half the clock frequency.

Once the control byte has been loaded, the CDP1855 X register is loaded by executing an OUT 4 instruction. The data, stored in the stack, is put out on the data bus and loaded into the X register of the MDU. The OUT 5 instruction loads the Z register in the same manner. The control register is reloaded with F9 (11111001). Bit 3 resets register Y to zero, and the code of 01 on bits 1 and 0 causes the multiply operation to begin.

The MDU automatically performs the 8x8 multiply and stores the most significant half of the 16-bit answer in register Y, and the least significant half in the Z register. The time required to perform the actual multiply operation is 8N+1 shifts, where N is the number of cascaded MDU's, one in

this case. Therefore, the multiply operation takes 9 shift cycles in this example, and since the prescaler is being used, this number of cycles translates to 18 clock pulses of the CPU. To provide enough time for the multiply operation, software should be arranged so that at least one instruction occurs between the multiply command (OUT 7, F9) and reading of the answer. The answer is read from the Y and Z registers by performing an INP 6 and an INP 5 instruction; the result is stored in two stack locations in memory.

DIVIDE OPERATION

In a divide operation, the divisor is loaded into the X register. The dividend is loaded into the Y and Z registers with the more significant half in the Y register and the less significant half in the Z register. When the operation is complete, the quotient is in the Z register and the remainder is in the Y register. The above operation can be described by the following formula:

$$Y Z \div X \rightarrow Z + Y_{REM}$$

If the answer exceeds the size of the Z register (8N bits, where N is the number of MDU's) the CO/OF pin on the most significant MDU goes low. If desired, the Z register can be cleared by loading a control byte with bit 2 high, and another divide operation can be performed on the remainder.

```

!M
0000 FB00;          0001 BEGIN LDI A.1(MULT)  ..LOAD HI-ADD OF MULT
0002 BE;            0002 PHI RE      ..INTO RE.1
0003 F80E;          0003 LDI A.0(MULT)  ..LOAD LO-ADD OF MULT
0005 AE;            0004 PLO RE      ..INTO RE.0
0006 FB00;          0005 LDI A.1(STACK) ..LOAD HI-ADD OF STACK
0008 BF;            0006 PHI RF      ..INTO RF.1
0009 FB1D;          0007 LDI A.0(STACK) ..LOADS LO-ADD OF STACK
000B AF;            0008 PLO RF      ..INTO RF.0
000C DE;            0009 SEP RE      ..GOES TO MULT SUBROUT.
000D ;              0010
000D ;              0011
000D D0;            0012 RETU SEP R0    ..RETURN TO MAIN PROG.
000E EE;            0013 MULT SEX RE   ..X=P=RE
000F 67F0;          0014 OUT 7; DC OF0H ..LOADS MDU CON.REG.
0011 EF;            0015 SEX RF      ..P=RE, X=STACK
0012 64;            0016 OUT 4      ..LOAD X REG. OF MDU
0013 65;            0017 OUT 5      ..LOAD Z REG. OF MDU
0014 EE;            0018 SEX RE      ..X=P=RE
0015 67F9;          0019 OUT 7; DC OF9H ..LOAD MDU CON.REG.
0017 EF;            0020 SEX RF      ..P=RE, X=RF (STACK)
0018 6E;            0021 INP 6      ..READ Y REG. OF MDU
0019 60;            0022 IRX         ..INCR. REG X
001A 6D;            0023 INP 5      ..READ Z REG. OF MDU
001B 300D;          0024 BR RETU    ..GOTO RETU
001D ;              0025
001D ;              0026 STACK DS 4    ..THIS STACK CONTAINS DATA
0021 ;              0027 ..TO BE MULTIPLIED TOGETHER
0021 ;              0028 ..FIRST 2 BYTES SHOULD BE
0021 ;              0029 ..THOSE TO BE MULTIPLIED
0021 ;              0030 ..THE NEXT 2 WILL BE THE
0021 ;              0031 ..ANSWER HI AND LO, AFTER
0021 ;              0032 ..MULTIPLICATION
0021 ;              0033 END
0000
    
```

Fig. 4 — Program used to perform 8x8-bit hardware multiply operation.

Divide Example

A 16÷8-bit division can be performed with the CDP1800-based system shown in Fig. 1 by using the subroutine given in Fig. 5. The program is very similar to the multiply program (Fig. 4) except that all three registers are loaded (X, Y and Z) and the answer is 8 bits in length (Z register) with an 8-bit remainder (X register).

```

!M
0000 F800;          0001 BEGIN LDI A.1(DIVID) ..LOAD HI-ADD OF DIVIDE
0002 BE;           0002 FHI RE ..INTO RE.1
0003 F80E;          0003 LDI A.0(DIVID) ..LOAD LO-ADD OF DIVIDE
0005 AE;           0004 FLD RE ..INTO RE.0
0006 F800;          0005 LDI A.1(STACK) ..LOAD HI-ADD OF STACK
0008 BF;           0006 FHI RF ..INTO RF.1
0009 F81E;          0007 LDI A.0(STACK) ..LOADS LO-ADD OF STACK
000B AF;           0008 FLD RF ..INTO RF.0
000C DE;           0009 SEP RE ..GOES TO DIVIDE SUBROUT.
000D ;             0010
000D ;             0011
000D DO;           0012 RETU SEP RO ..RETURN TO MAIN PROG.
000E EE;           0013 DIVID SEX RE ..X=P=RE
000F 67FC;         0014 OUT 7; DC OFCH ..LOADS MDU CON.REG.
0011 EF;           0015 SEX RF ..P=RE, X=STACK
0012 64;           0016 OUT 4 ..LOAD X REG. OF MDU
0013 66;           0017 OUT 6 ..LOAD Y REG. OF MDU
0014 65;           0018 OUT 5 ..LOAD Z REG. OF MDU
0015 EE;           0019 SEX RE ..X=P=RE
0016 67F2;         0020 OUT 7; DC OF2H ..LOAD MDU CON.REG.
0018 EF;           0021 SEX RF ..P=RE, X=RF (STACK)
0019 6D;           0022 INF 5 ..READ Z REG. OF MDU
001A 60;           0023 IRX ..INCR. REG X
001B 6E;           0024 INF 6 ..READ Y REG. OF MDU
001C 300D;         0025 BR RETU ..GOTO RETU
001E ;             0026
001E ;             0027 STACK DS 5 ..THIS STACK CONTAINS DATA
0023 ;             0028 ..FIRST BYTE IS 8 BIT DIVISOR
0023 ;             0029 ..SECOND BYTE IS MSB DIVIDEND
0023 ;             0030 ..THIRD BYTE IS LSB DIVIDEND
0023 ;             0031 ..FOURTH BYTE IS 8 BIT ANSWER
0023 ;             0032 ..FIFTH BYTE IS REMAINDER
0023 ;             0033 END
0000

```

Fig. 5 — Program used to perform 16÷8-bit division.

A SOFTWARE ALGORITHM COMPARISON

Consider a CDP1800-based system without the CDP1855 MDU, where the 8x8-bit multiply operation is implemented by means of a software algorithm of add and shift right, Fig. 6. Normal overhead, initializing the CPU registers, is done in the main program. Since 9 shifts are required to complete the multiply operation, 15 instructions of the subroutine must be repeated 9 times, resulting in a total of 284 machine cycles. Assuming a system clock frequency of 3 MHz, the time required to complete the subroutine of Fig. 6 (no MDU) is 757 microseconds. The time required to complete the subroutine in the program of Fig. 4 (using the MDU) is only 69 microseconds. The MDU does all of the shifting and adding; therefore, the actual 13-instruction subroutine need be executed only once. A comparison of the loop times of the subroutines of Figs. 4 and 6 indicates that CDP1800-based systems using the CDP1855 hardware multiply-divide unit can perform an 8x8-bit

multiplication in less than 10 percent of the machine time required by systems using a software-multiply algorithm.

CASCADED MDU'S

The use of cascaded CDP1855 units in a system can provide even more dramatic improvements in computing power. For example, a CDP1800-based system incor-

porating two MDU's can perform a 16x16-bit multiply operation or a 32÷16-divide operation 25 times faster than the Fixed-Point Binary Arithmetic Subroutine software package offered by RCA.³ Fig. 7 is a schematic diagram showing four MDU's in cascade with connector wiring for interfacing to an RCA CDP18S008 Development or Microboard system.

The program shown in Fig. 8 is a universal program that facilitates multiplication with addition, or division operations. A 32x32 multiplication resulting in a 64-bit product may be performed. Operations involving numbers less than 32 bits are accomplished by loading the more significant bits with zeros. A number may be added to the product by preloading the Y registers with up to 32 bits before the multiply operation begins. The final results appear in Y (most significant half) and Z (least significant half). Register X is unaltered by the operation, so that repeated multiplications by a constant number are possible without having to reload the X register each time.


```

!M
0000 FB00;      0001 BEGIN  LDI 00H  ..R0 IS PROG. COUNTER
0002 B7;        0002     PHI R7   ..R7 IS PRODUCT REG.
0003 FB00;      0003     LDI A.1(TEMP)
0005 BB;        0004     PHI R8
0006 FB32;      0005     LDI A.0(TEMP)
0008 AB;        0006     PLO R8   ..R8=TEMP.STORAGE
0009 FB17;      0007     LDI A.0(MULT)
000B AE;        0008     PLO R8
000C FB00;      0009     LDI A.1(MULT)
000E BE;        0010     PHI R8   ..R8 IS SUBROUT. POINTER
000F FB00;      0011     LDI A.1(STACK)
0011 BA;        0012     PHI RA    ..DATA STACK POINTER
0012 FB33;      0013     LDI A.0(STACK)
0014 AA;        0014     PLO RA
0015 DE;        0015     SEP R8   ..GOTO MULT SUBROUT.
0016 DO;        0016 RETU  SEP R0   ..RETURN TO MAIN PROG. COUNTER
0017 FB09;      0017 MULT  LDI 09H  ..PRESETS LOOP COUNTER TO 9
0019 A6;        0018     PLO R6   ..R6 IS LOOP COUNTER
001A FE;        0019     SHL      ..PRESETS DF=0
001B 4A;        0020     LDA RA    ..LOAD MULTIPLICAND
001C 58;        0021     STR R8   ..INTO 00FF
001D 4A;        0022     LDA RA    ..LOAD MULTIPLIER
001E A7;        0023     PLO R7   ..INTO R7
001F EB;        0024     SEX R8
0020 97;        0025 CONT  GHI R7
0021 76;        0026     SHRC     ..SHIFT LO BYTE
0022 B7;        0027     PHI R7
0023 B7;        0028     GLO R7
0024 76;        0029     SHRC     ..SHIFT HI BYTE
0025 A7;        0030     PLO R7
0026 3B2B;      0031     BNF INDEX ..BRANCH IF DF=0
0028 97;        0032     GHI R7
0029 F4;        0033     ADD      ..ADD M'PLIER TO R7.1
002A B7;        0034     PHI R7   ..RETURN SUM TO R7.1
002B 26;        0035 INDEX  DEC R6   ..INDEX LOOP COUNTER
002C B6;        0036     GLO R6
002D 3216;      0037     BZ RETU  ..IS LOOP COUNTER=9?
002F A6;        0038     PLO R6
0030 3020;      0039     BR CONT  ..CONTINUE LOOPING
0032 ;          0040 TEMP  DS 1
0033 ;          0041 STACK DS 2
0035 ;          0042     END
0000
    
```

Fig. 6 — Software algorithm of add and shift right used to implement an 8x8-bit multiply operation.

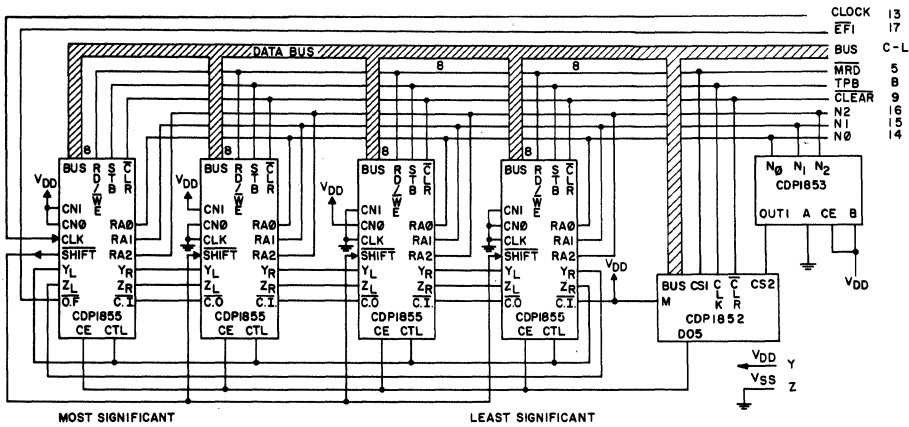


Fig. 7 — Cascaded CDP1855 configuration.

92CL-33701

```

;M
0000 6120;      0001      OUT 1;DC 20H ..ENABLE CDS TWO-LEVEL I/O
0002 FB33;      0002      LDI A,0(STACK)
0004 AF;        0003      PLO RF
0005 FB00;      0004      LDI A,1(STACK)
0007 BF;        0005      PHI RF
000B EF;        0006      SEX RF
0009 67;        0007 MULDIV OUT 7      ..LOAD CONTROL REGISTER W/CC
000A 64;        0008      OUT 4      ..LOAD X REG. OF MS UNIT
000B 64;        0009      OUT 4
000C 64;        0010      OUT 4
000D 64;        0011      OUT 4      ..LOAD X REG. OF LS UNIT
000E 65;        0012      OUT 5      ..LOAD Z REG. OF MS UNIT
000F 65;        0013      OUT 5
0010 65;        0014      OUT 5
0011 65;        0015      OUT 5      ..LOAD Z REG. OF LS UNIT
0012 66;        0016      OUT 6      ..LOAD Y REG. OF MS UNIT
0013 66;        0017      OUT 6
0014 66;        0018      OUT 6
0015 66;        0019      OUT 6      ..LOAD Y REG. OF LS UNIT
0016 67;        0020      OUT 7      ..LOAD MDU CONT. REG'S
0017 ;          0021      ..WITH C1 FOR MULT. OR
0017 ;          0022      ..C2 FOR DIVIDE.
0017 C4;        0023      NOP      ..ALLOW 33 MACHINE CYCLES
001B C4C4C4;    0024      NOP;NOP;NOP .. FOR THE MULT/DIV. OPER-
001B C4C4C4;    0025      NOP;NOP;NOP ..ATION BEFORE READING
001E C4C4C4;    0026      NOP;NOP;NOP
0021 C4;        0027      NOP
0022 6E60;      0028      INP 6; IRX      ..READ Y REG. OF MS UNIT
0024 6E60;      0029      INP 6; IRX
0026 6E60;      0030      INP 6; IRX
0028 6E60;      0031      INP 6; IRX      ..READ Y REG. OF LS UNIT
002A 6D60;      0032      INP 5; IRX      ..READ Z REG. OF MS UNIT
002C 6D60;      0033      INP 5; IRX
002E 6D60;      0034      INP 5; IRX
0030 6D60;      0035      INP 5; IRX      ..READ Z REG. OF LS UNIT
0032 00;        0036      IDLE      ..STOP OR RET. TO MAIN PROG
0033 CC;        0037 STACK DC 0CCH ..CONTROL BYTE CC
0034 ;          0038      DS 12      ..OUTPUT DATA STACK
0040 C1;        0039      DC 0C1H ..C1 FOR MULT,C2 FOR DIV.
0041 ;          0040      DS 8       ..INPUT DATA STACK
0000

```

Fig. 8 — Universal program that facilitates multiplication, multiplication with addition, or division operations.

A 64÷32-bit division is possible, and will yield a 32-bit result (2 registers) and a 32-bit remainder (Y registers). The CO pin of the most significant MDU is used as an overflow indicator. The status register also contains a bit for overflow indication, and can be read by executing an INP 3 instruction. Overflow occurs if the result of the division is more than 32 bits in length.

FASTER MDU THROUGHPUT USING DMA

When using the CDP1855 MDU in the normal I/O mapped configuration shown in Fig. 1, a large portion of the overall throughput time is consumed by loading and unloading data and control bytes. The CPU must execute six I/O instructions to facilitate a multiply or divide operation, Fig. 4. Throughput speed for an 8x8-bit multiply can be increased by 45 percent by using a DMA in/out technique to load and unload the MDU. The DMA-enhanced-MDU technique requires only one I/O instruction, leaving more unused I/O mapping space for larger systems.

A continuous multiplier system that employs the DMA-enhanced-MDU technique is shown in Fig. 9. A more detailed description of the hardware shown in the dashed box of Fig. 9 is given in Fig. 10. The software employed by the system is shown in Fig. 12.

Hardware Description

Fig. 10 shows the logic hardware needed in addition to the CDP1855 to facilitate a DMA transfer of MDU data and control bytes. Once initiated with a single output instruction, the added hardware automatically asserts DMA IN and DMA OUT signals, sequentially addresses the MDU registers, and controls the timing of the read/write signal. Normally both flip-flops are in a reset condition, and no DMA action is occurring.

A (0111) code is jammed into the CD4516, keeping the MDU deselected (CE = 0). When an OUT 2 instruction occurs, the

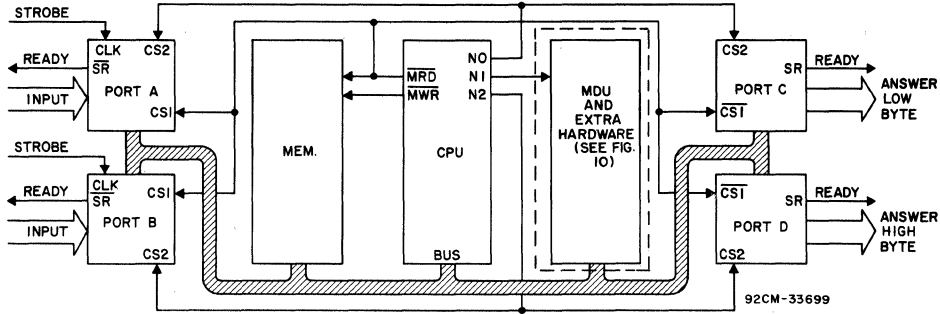


Fig. 9 — Continuous multiply system using DMA-enhanced-MDU technique.

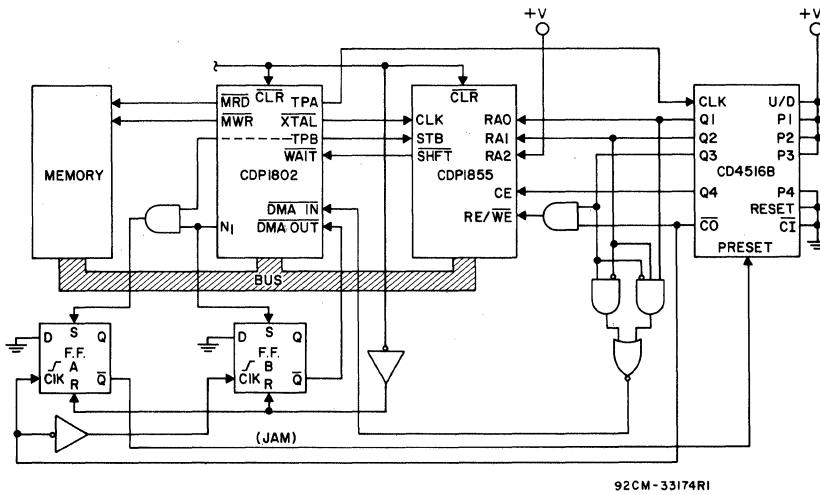
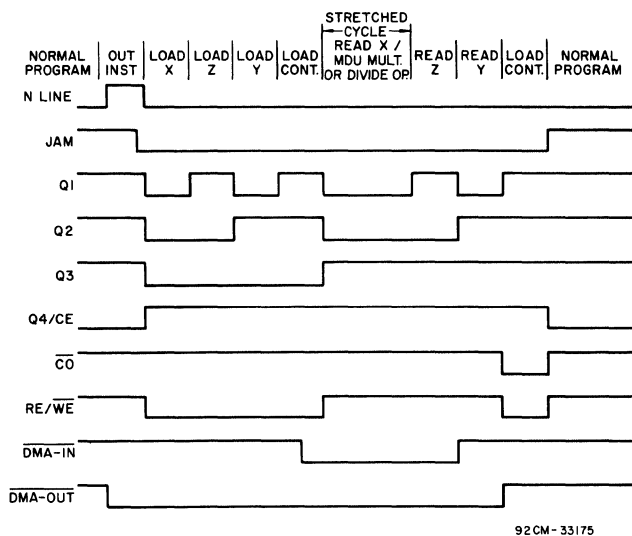


Fig. 10 — MDU and extra hardware needed to implement DMA-enhanced-MDU technique.

N1 line goes high, setting flip-flop B, which asserts a DMA OUT signal at the CPU. Later in the output cycle, during TPB, flip-flop A is set, which removes the preset (JAM) signal from the CD4516. The CD4516 merely counts up at each TPA pulse, sequentially addressing the MDU. Additional logic decodes the CD4516 output to produce the DMA-IN signal (DMA-IN has priority over DMA-OUT). At the end of the address sequence, the carry out (CO) signal restores the flip-flops to their original states, ending the DMA action and deselecting the MDU. The MDU SHIFT pin is connected to the CPU WAIT pin; this connection causes a stretched CPU cycle during the multiply/divide operation, so that the MDU receives at least 18 clock pulses before the answer is read from the Z and Y registers, Fig. 11.

Software Description

The program in Fig. 12 starts by initializing CPU register R3, designating it as the program counter, and freeing R0 for the DMA pointer. RX is also set to R0 so that the stack can be loaded/unloaded using INP/OUT instructions. The program then enters the multiply routine, and loops continuously through successive multiply operations. The looping scheme was chosen to demonstrate the throughput rate of the continuous multiplier system, Fig. 9, in terms of multiplications per second. The program of Fig. 12 can be modified to facilitate the use of branch and interrupt techniques to get in and out of the multiply loop. With the addition of two SEP instructions, the multiply operation can become a one-shot subroutine.



92CM-33175

Fig. 11 — Timing diagram for DMA transfer of MDU data and control bytes.

```

!M
0000 FB07;      0001 BEGIN  LDI A.0(INIT)  ..R0=PROG.COUNTER
0002 B3;        0002      PHI R3
0003 FB00;      0003      LDI A.1(INIT)  ..R3 IS LOADED
0005 A3;        0004      PLO R3      ..TO BECOME PROG.
0006 D3;        0005      SEP R3      ..COUNTER
0007 E0;        0006 INIT   SEX R0      ..R0=DMA STACK POINTER
0008 FB1D;      0007      LDI A.0(STACK1)
000A A0;        0008      PLO R0
000B FB00;      0009      LDI A.1(STACK1)
000D B0;        0010      PHI R0
000E FB1F;      0011 MULT  LDI A.0(STACK2)
0010 A0;        0012      PLO R0
0011 69;        0013      INP 9
0012 20;        0014      DEC R0
0013 6C;        0015      INP 0CH  ..PORT B DATA TO STACK
0014 20;        0016      DEC R0
0015 62;        0017      OUT 2      ..INITIATE DMA AND
0016 FB23;      0018      LDI A.0(STACK3)  ..MULTIPLY ACTION
0018 A0;        0019      PLO R0
0019 61;        0020      OUT 1      ..STACK DATA TO PORT C
001A 64;        0021      OUT 4      ..STACK DATA TO PORT D
001B 300E;      0022      BR MULT
001D 00;        0023 STACK1 DC 00H  ..DUMMY BYTE
001E ;          0024      DS 1      ..LOAD X REGISTER
001F ;          0025 STACK2 DS 1      ..LOAD Z REGISTER
0020 00;        0026      DC 00H  ..LOAD Y REGISTER W/ 00
0021 F1;        0027      DC 0F1H  ..LOAD CONT. REG. W/ F1
0022 ;          0028      DS 1      ..READ X REGISTER
0023 ;          0029 STACK3 DS 2      ..READ Z REG. (1ST BYTE)
0025 ;          0030      DS 2      ..READ Y REG. (2ND BYTE)
0025 FC;        0031 STACK4 DC 0FCH  ..LOAD CONT. REG. W/ FC
0026 ;          0032      END
0000

```

Fig. 12 — Software required for implementation of DMA transfer of MDU data and control bytes.

COMPARISON OF MULTIPLICATION TECHNIQUES

Table II shows a comparison of multiplication times for 8x8, 16x16, and 32x32-bit multiply operations using software-only routines, MDU routines, and a DMA-enhanced-MDU routine.

DIGITAL FILTERING

Digital circuitry has opened new horizons in signal processing, especially in the field of digital filtering.⁴ Digital filters have distinct advantages over analog filters; the advantages include:

- No drift with temperature
- Immunity to power-supply aberrations
- Little or no adjustment ("tweaking") required
- Software programmable response
- Reliable repeatability

A variety of algorithms for digital-signal processing have been developed; however, they have several things in common: all require many multiplications, additions, and delay operations.

In a strictly software algorithm (no MDU), the number of multiply or divide operations is usually minimized to reduce the throughput time; the result is a very complex algorithm. However, if the MDU is utilized, the algorithm can be simple and repetitive, involving many additions and multiplications. There are two types of basic digital-filtering algorithms:

- Non-recursive filters (also called finite-impulse filters), usually with many feed-forward terms, composed of multiply, add and delay elements, Fig. 13.
- Recursive filters (also called infinite-impulse response filters) with feed-forward and feed-back terms; Fig. 14.

Fig. 15 shows a simple low-pass analog filter composed of a resistor (R) and a capacitor (C). The relationship of output voltage to input voltage is expressed in the following equation:

$$\frac{dV_O}{dt} + \frac{V_O(t)}{RC} = \frac{V_I(t)}{RC} \quad (1)$$

Table II Comparison of Multiply Speeds (In CPU Machine Cycles)

Operation	8x8	16x16	32x32
Software	284 cycles using program in Fig. 6.	Approximately 1000 cycles using the "RCA Fixed Point Binary Arithmetic Subroutine."	Approximately 5000 cycles using the "RCA Fixed Point Binary Arithmetic Subroutine."
MDU	26 cycles using program in Fig. 4 and one MDU.	41 cycles using program in Fig. 8 and two MDU's.	89 cycles using program in Fig. 8 and four MDU's.
MDU-with-DMA	11 cycles using program in Fig. 11 and one MDU, as shown in Fig. 9.		

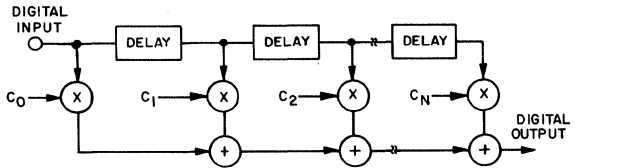


Fig. 13 — Symbolic representation of a nonrecursive filter.

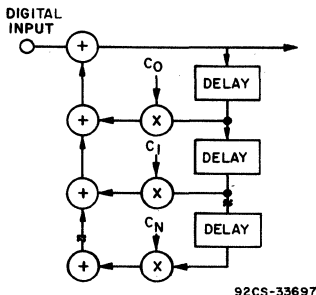


Fig. 14 — Symbolic representation of a recursive filter.

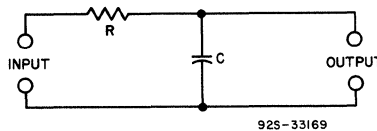


Fig. 15 — Simple low-pass analog filter.

Solving the equation by integration using Euler's method of sampling results in a new expression:

$$V_{O(n)} = kV_{I(n)} + mV_{O(n-1)} \quad (2)$$

where n is any specific sample period. The two constants, k and m, are functions of the R and C values as well as of the sampling time t:

$$k = \frac{1}{1 + RC/t}, \quad m = \frac{1}{1 + t/RC}$$

Implementation with the CDP1855

Euler's sampling algorithm can be implemented with a recursive digital filter that uses multiply, add, and delay functions. Fig. 16 is a symbolic representation of this filter, and shows multiplication operations performed by the MDU. The second operation adds the result of the first operation to its product. Feedback delay is obtained by storing the output byte in a scratchpad memory for use in the next operation.

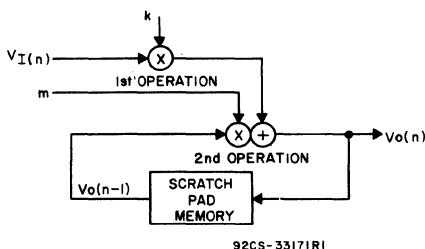


Fig. 16 — Symbolic representation of recursive digital filter that uses multiply, add, and delay functions.

Table III — Digital Filter Response

No. of Sample Periods	0	1	2	3	4	5	6	7	8	9	10
Input Voltage V _I (volts)	0	1	1	1	1	1	1	1	1	1	1
Output Voltage V _O (volts)	0	0.20	0.36	0.49	0.69	0.67	0.74	0.79	0.83	0.86	0.90

Assuming that the component values in Fig. 15 are R = 30 kilohms and C = 0.1 microfarad, and that t = 75 microseconds:

$$k = \frac{1}{1 + 300/75} = 0.2 = 0.00110011 \text{ binary} = 33 \text{ hexadecimal}$$

and

$$m = \frac{1}{1 + 75/300} = 0.8 = 0.11001100 \text{ binary} = CC \text{ hexadecimal}$$

Assume now that the input signal goes from 0 to a full-scale voltage of 1 volt, and remains at 1 volt for 10 sample periods. The corresponding output voltages, determined by using equation 2, are listed in Table III; if plotted, they would produce a response curve similar to that of the analogous RC circuit.

Fig. 17 is a block diagram of a digital-signal processing system. Incoming analog signals are digitized by an A/D converter, processed by the digital filter, and reconstructed by a D/A converter. A flowchart of the required steps in the software program is given in Fig. 18.

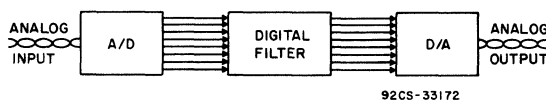


Fig. 17 — Block diagram of a digital-signal processing system.

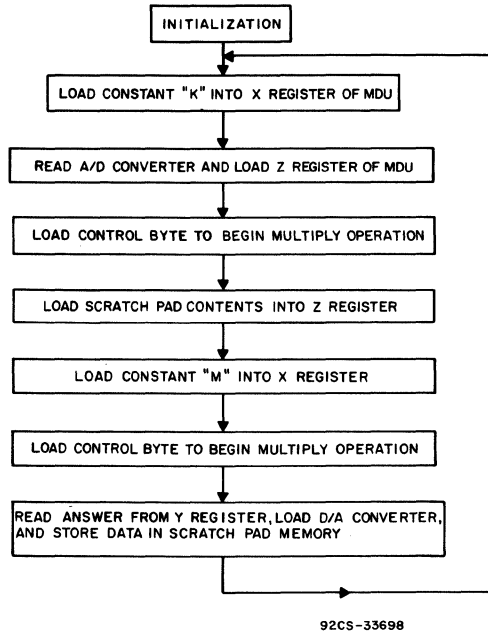


Fig. 18 — Program flowchart for system of Fig. 17.

Acknowledgment

The author thanks J. Pintaske and H. Tweddle of RCA Brussels for their contributions to this Note.

References and Bibliography

1. CDP1855 data sheet, "8-Bit Programmable Multiply/Divide Unit" RCA Solid State File Number 1053.
2. The IRX instruction is a special case; software should be arranged to avoid executing an IRX instruction when the memory pointer, R(X), is pointing to any of the MDU address locations. The IRX instruction not only presents the contents of the R(X) register on the address bus, but also asserts a read signal (MRD) that will enable and inadvertently address the MDU.
3. "Binary Arithmetic Subroutines for RCA COSMAC Microprocessors," RCA Solid State Publication MPM-206.
4. "Implementing Digital Filters Efficiently," Richard J. Karwoski, **Electronic Design**, Sept. 1, 1979.

New CMOS CDP1800-Series Processors Enhance System Performance

by R. M. Vaccarella

INTRODUCTION

Since the introduction of integrated circuits in the 1960's, the trend has been towards the inclusion of more devices, in more complex configurations, on a single slice of silicon. The result has been an extraordinary reduction of room-size computers to near-microscopic components. But the accomplishments have not been limited to the shrinking of silicon real estate; new technologies and design innovations have expanded the application of electronics into many new areas, and have greatly increased reliability.

The MOS technology, the CMOS technology in particular, has played a significant role in the miniaturization of electronic circuits and has, perhaps, contributed most to the application of microelectronics to new and previously untapped areas of the commercial and industrial marketplace. As exemplified by the recent introduction of many newly developed CMOS microprocessors and microcomputers by the major semiconductor manufacturers, the CMOS technology is in great demand. The low operating power, wide voltage range, and competitive computation speed of the CMOS technology often provides the only practical design solution in many battery-powered and other industrial and automotive applications. The wide operating-temperature range and high noise immunity of CMOS devices encourages their use in difficult and extreme environments, and improves overall product reliability and performance.

The RCA CMOS product-line, introduced in 1968, and expanded most recently by the introduction of the CDP1800-series products, has been acknowledged in the industry as a reliable, low-power IC line. A mature and widely accepted offering of devices, ranging from simple gates to microcomputers, is currently available in high volume and continues to expand with the addition of many memory, I/O, and hardware/software support products. The newest introductions to the well-established CDP1800-series microprocessor family are the CDP1804 and CDP1805 microprocessors. The bulk of this Note, following a review of CDP1800-series features and advantages, is devoted to a discussion of the attributes of the CDP1804 and CDP1805, both of which are hardware and software upward-compatible with the CDP1802.

REVIEW OF CDP1800-SERIES MICROPROCESSOR FEATURES AND ADVANTAGES

In addition to the advantages inherent in the CMOS technology, those pointed out in the introduction, above, the CDP1800-series microprocessors contain some special features that are of value in a wide variety of applications. Fig. 1 shows the internal arrangement of the various registers and the control circuitry of the CDP1802, CDP1804, and CDP1805.

Register Allocation - The basic strength of the CDP1802 is its register-oriented architecture, which allows flexible assignment of register, memory, and I/O space. An array of sixteen 16-bit general-purpose scratchpad registers is provided. The contents of these registers may be directed to external memory, the D register, or the increment/decrement logic. The various register operations are summarized in Table I.

Table I - Internal Register System

D	8 Bits	Data Register (Accumulator)
DF	1 Bit	Data Flag (ALU Carry)
R	16 Bits	1 of 16 Scratchpad Registers
P	4 Bits	Designates which register is Program Counter
X	4 Bits	Designates which register is Data Pointer
N	4 Bits	Holds Low-Order Instr. Digit
I	4 Bits	Holds High-Order Instr. Digit
T	8 Bits	Holds old X, P after interrupt (X is high nibble)
IE	1 Bit	Interrupt Enable
Q	1 Bit	Output Flip Flop
CH	8 Bits	Holds Counter Jam Value

Memory Addressing - Through the use of the 16-bit scratchpad registers, numerous addressing modes are available. The contents of any of the sixteen registers are accessible by either the P, N, or X registers for use as a program counter, memory pointer for data handling, or as a stack pointer. In addition, multiple program counters, memory pointers, and stack pointers can be created to facilitate subroutine calls and data manipulations, a technique not practical with other types of processor architecture. Table II summarizes the memory-addressing modes.

Bus Timing - The internal 16-bit address bus, which is capable of handling up to 65,536 bytes of memory, is multiplexed on eight address-output lines. The eight higher-order bits are sent out first and are latched externally by the TPA pulse. The eight lower-order bits then follow on the same lines and remain valid for the rest of the machine cycle.

The internal 8-bit data bus is sent out directly on the eight bidirectional bus-output lines. Internally, the data bus is used to route data to and from memory, scratchpad registers, the D register, the ALU, and the control registers.

A 3-bit code is sent out on the N-line outputs, and is used for I/O address selection, with the TPB pulse used to latch valid data. These lines are active only during input/output instructions, and are valid for the entire machine cycle.

Instruction Timing - An external crystal supplies system timing and is divided internally to provide eight clock cycles for each machine cycle. One FETCH cycle and one EXECUTE cycle, consisting of a total of sixteen clock

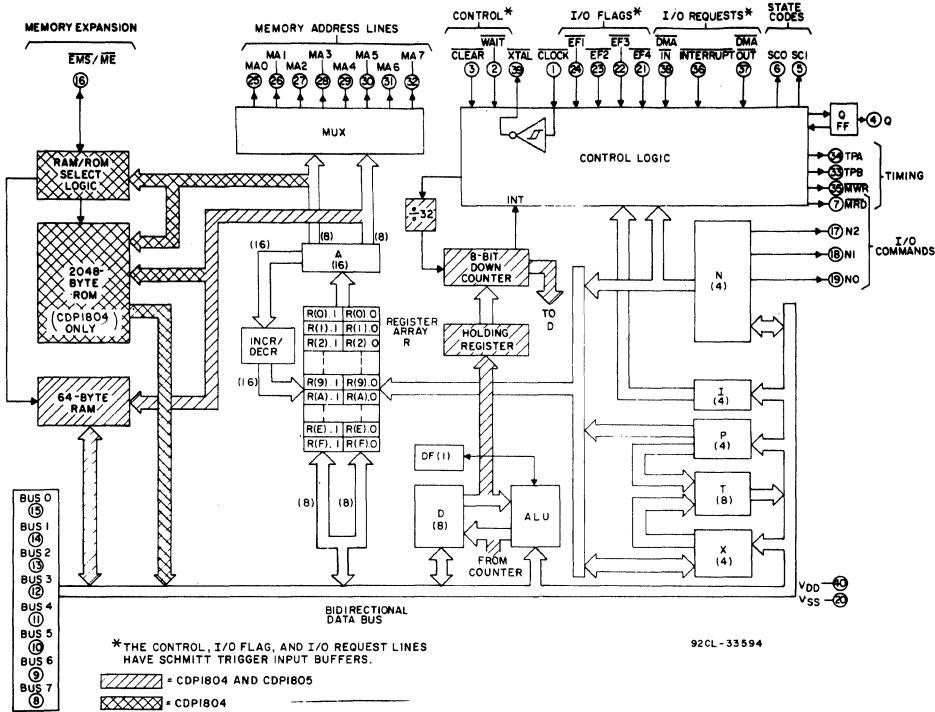


Fig. 1 - Block diagram of the CDP1802, CDP1804, and CDP1805 microprocessors.

Table II - Memory Addressing Modes

Mode	Instruction	Register	Memory	Operation	Comments
Immediate	Operand			Value is contained in the instruction	Load Immediate (LDI)
Direct Address	Address	→ Operand		Value is contained in the location pointed to by the address in the instruction	Short and Long Branch (BR, LBR)
Register	Register Address	→ Operand		Value is contained in the register pointed to by the instruction	Register Data Manipulation (GLO, GHI)
Indirect Register	Register Address	→ Address → Operand		Value is contained in the location whose address is in the register pointed to by the instruction	Data to and from RAM (LDN, LDX)

Table III - Breakdown of Ninety-One Instructions Common to CDP1802/04/05 Microprocessors

MEMORY TRANSFER	7
INTERNAL REGISTER	7
LOGIC	10
ARITHMETIC	12
UNCONDITIONAL JUMPS	4
CONDITIONAL JUMPS	27
CONTROL	7
INTERRUPT CONTROL	3
I/O TRANSFER	14

cycles, are used in over 60 percent of the available instructions. This consistent timing allows accurate software timing loops to be programmed. Table III shows a breakdown of the CDP1800-series instruction set.

Special Functions and Status - A maskable interrupt input is provided that automatically designates R1 as storage for the vector address and R2 as the stack pointer. This input is useful for asynchronous system control, such as the initiation of a software routine to store data in the event of a power failure.

A DMA input and a DMA output control automatic transfer of data directly to and from memory, using R0 as the DMA counter. These functions are useful for fast data handling, since the internal D register is bypassed.

Four flag inputs (EF1-EF4), which may be software tested, and a Q output, which is software controlled, are also available. These functions are useful for external hardware control of software routines.

Machine status and direct system control are provided by two state-code outputs (SC0, SC1), separate READ (MRD) and WRITE (MWR) outputs, and WAIT and CLEAR inputs.

Hardware Support - A large support capability exists for the CDP1800-series product line. Many of the parts are specifically tailored to the CDP1800-series microprocessors to reduce pin counts and to simplify interfacing. Memory products include RAM and EPROM, for prototyping work, and mask-programmable ROM's for high-volume production work. Complex I/O devices which range from UART's to CRT controllers are also available. These parts can significantly reduce the size and power requirements of a large system. Simple I/O devices, such as buffers, latches, and decoders, plus CDP1800-series-compatible devices from the standard RCA CMOS product line, complete the CMOS system support capability. A complete listing of support products is shown in Table IV.

Table IV - Major CDP1800-Series Support Components

RAMS		SIMPLE I/O	
CDP1821	1K X 1	CDP1852	I/O PORT
CDP1822	256 X 4	CDP1853	DECODER
MWSS101	256 X 4	CDP1856	BUFFER
CDP1823	128 X 8	CDP1857	BUFFER
CDP1824	32 X 8	CDP1858	LATCH
CDP1825	1K X 4	CDP1859	LATCH
MWSS114	1K X 4	CDP1863	COUNTER
* CDP1826	64 X 8	CDP1866	LATCH
		CDP1867	LATCH
ROMS		COMPLEX I/O	
CDP1831	512 X 8	CDP1872	I/O PORT
CDP1832	512 X 8	CDP1873	DECODER
CDP1833	1K X 8	CDP1874	I/O PORT
CDP1834	1K X 8	CDP1875	I/O PORT
* CDP1835	2K X 8		
* MWSS316	2K X 8	CDP1851	PROG I/O
		CDP1854A	UART
EPROMS		CDP1855	MDU
CDP18U42	256 X 8	CDP1869	CRT CONTROLLER
* MWSS7U58	1K X 8	CDP1870	CRT CONTROLLER
		CDP1871	KEYBOARD ENCODER
		CDP1876	CRT CONTROLLER
		* CDP1877	INTERRUPT CONTROLLER

* RESERVED NUMBER FOR 1981 PRODUCTION DEVICES

92CS-33383

Software Support - To complement the extensive hardware product line, a comprehensive software-support line is offered. Products in this line range from low-cost prototyping kits to complete development systems, and are intended to aid in system evolution from the design stage, through the product development stage, and into the field-support stage.

The Evaluation Kit, a simple single-board system, is useful for low-end development of small systems and micro-processor educational purposes.

The COSMAC Development Systems are used for larger and more complicated design development. The most recent introduction, the COSMAC System IV, is a CRT-based system offering 64K of memory, a full-screen editor and text editor, a macro-assembler, and easy interfacing with floppy-disks, PROM programmers, printers, and other computer peripherals. The companion software includes BASIC1 and 2, PLM-1800, DOS, MOPS and fixed and floating-point math subroutines.

An in-circuit emulator, the Micromonitor, is available for debug and evaluation work. The Micromonitor may be used to enhance the basic features of the COSMAC Development System or in a stand-alone mode for field-service work. It is supported in software by the Micromonitor Operating System (MOPS) for complete system integration and production-type testing.

In the area of single-board computers, RCA offers the Microboard computer line. This line consists of a set of small-size, low-power computer, memory, and I/O boards that allow quick turn-around time and modular product development. These boards feature LSI CDP1800-series components and standard backplane wiring for convenient interfacing.

The major CDP1800-series system support products are shown in Table V. All are user-oriented for simple, time-saving product development.

Table V - Major CDP1800-Series System Support

DEVELOPMENT SYSTEM
• COSMAC SYSTEM IV (CDP18S008)
• CRT-BASED SYSTEM CONTAINING:
• 64K MEMORY
• DUAL FLOPPY-DISK DRIVES
• IN-CIRCUIT EMULATION (MICROMONITOR)
• BUILT-IN PROM PROGRAMMER
• COMPLETE DISK OPERATING SYSTEM
• LEVEL I, LEVEL II ASSEMBLER
• MACRO ASSEMBLER
• FULL-SCREEN EDITOR AND TEXT EDITOR
• MICROMONITOR OPERATING SYSTEM (MOPS)
• PROM-PROGRAMMER OPERATING SOFTWARE
SOFTWARE (OPTIONAL)
• BASIC 1 (FIXED POINT) INTERPRETER/COMPILER
• BASIC 2 (FLOATING POINT) INTERPRETER
• PLM-1800 COMPILER
• FIXED AND FLOATING POINT MATH SUBROUTINES
MICROBOARDS — SINGLE BOARD MICROCOMPUTERS
• COORDINATED SET OF COMPUTER, MEMORY AND I/O BOARDS,
• CDP18S606 - 1804/1805 EVALUATION BOARD CONTAINS:
• 1804 CPU W/2.47 MHZ CLOCK
• 2K BYTES RAM (FOR 1804 ROM SIMULATION)
• 2 EACH ROM/EPROM SOCKETS
• 2 EACH PARALLEL I/O PORTS
• RS232C SERIAL PORT (UART)

CDP1804/CDP1805 FEATURES AND ADVANTAGES

The same register-based architecture used in the CDP1802 microprocessor is found in the CDP1804 and CDP1805, but with significant enhancements that upgrade performance and flexibility. The additional features include on-board ROM (CDP1804 only), on-board RAM, an 8-bit timer-counter, and twenty-two new instructions. Performance is improved by increasing the operating speed to 4 MHz at 5 volts. Except for pin 16, which is used for external memory control on the CDP1804 and CDP1805, these parts are pin-compatible with the CDP1802. The multiplexed address bus is still available to access up to 65,536 bytes of external memory.

Although the CDP1804 and CDP1805, shown in Fig. 1, are functionally identical, except for on-board memory capability, their intended applications are quite different. The CDP1804 contains 2048 bytes of mask-programmable /mask-selectable ROM and 64 bytes of mask-selectable RAM. Because of the mask-programmable ROM feature, this part is well-suited for large-volume products, those for which no software changes are anticipated. The CDP1805

contains only the 64 bytes of on-board RAM, which is selected via memory enable (\overline{ME}) input. This part is suited for more general applications. When the CDP1805 is used for system prototyping, software may be developed in RAM or EPROM, and later replaced with ROM at the production phase. Since the ROM would be an external part, any software changes or additions could be implemented by simply replacing the ROM. In systems requiring more than the 2K bytes of ROM offered on the CDP1804, the CDP1805 would be the more effective choice.

Hardware Improvements - The 2K bytes of on-board ROM offered on the CDP1804 may be used to hold an operating system, special subroutines, or a higher-level language such as BASIC. The remaining memory space outside the mask-programmable 2K block is available to the user via the address bus, with the EMS signal indicating when external memory is being accessed. The 64 bytes of RAM, which are present on both the CDP1804 and the CDP1805, are convenient for use as a stack or data storage area. When using the CDP1804, two RAM modes are available. In the RAM/ROM mode, the RAM functions as ROM, using internal mask-decoding for selection. The EMS signal indicates when external memory is being accessed. In the RAM-only mode, the internal mask-decoding is disabled. The \overline{ME} signal is used as the internal RAM select input. The latter mode is also used by the CDP1805.

The other important hardware enhancement of the CDP1804 and CDP1805 is the on-board timer-counter. This 8-stage presettable down counter, shown in Fig. 2, is software

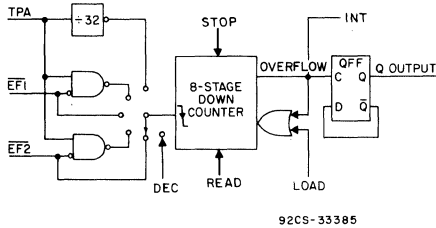


Fig. 2 - CDP1804/CDP1805 timer/counter model.

controllable in three basic modes, with the clock provided externally from either of two flag inputs or internally by TPA divided by 32. In addition, counter overflow may be indicated externally via the Q output or internally with a software interrupt request. The counter is a useful alternative to the decrement register N instruction used with the CDP1802, since a scratchpad register is saved and the D register does not have to be saved or tested for zero.

Timer Mode - The input source to the 8-bit preset counter is supplied by a divide-by-32 prescaler, which is decremented on each low-to-high transition of TPA, thereby instituting a counter clock rate equal to the system clock divided by 256. When the preset counter has decremented to the count of zero, the internal counter-interrupt request is latched and the initial preset value is reloaded into the counter. The operation is repeated on subsequent counts. This mode is useful as an accurate time-base in real-time applications.

Event-Counter Mode - The input source to the 8-bit preset counter is supplied by either the $\overline{EF1}$ or $\overline{EF2}$ flag input. The internal prescaler is not used and the counter is decremented on each high-to-low transition of the flag. On the count of zero, the counter-interrupt request is latched and the preset value is reloaded into the counter. The operation

is repeated on subsequent counts. This mode is useful as an external event counter or as a timer with a clock source other than TPA.

Pulse Width Mode - The input source to the 8-bit counter is supplied by TPA. The internal prescaler is not used and the counter is decremented by TPA if the input signal at the $\overline{EF1}$ or $\overline{EF2}$ flag is low. When the flag input goes high, the count is stopped, the mode is cleared, and the counter-interrupt request is latched. The counter value represents the width of the pulse at the flag input. Since the two flag inputs may be used together, this mode is useful for comparison-type measurements. The flag inputs may still be tested with the existing CDP1802 branch instructions.

Software Improvements - Twenty-two new instructions have been added to the ninety-one existing CDP1802 instructions. The new instructions are intended to enhance the flexibility and performance of the CDP1800-series architecture in the areas of register, interrupt, and sub-routine control, and to optimize the use of the on-board timer-counter. All of the new instructions use a linked "68" OPCODE, which requires a minimum of three machine cycles, including two S0 cycles and one or more S1 cycles. The instruction format is: 68 (linked OPCODE) XX (instruction OPCODE) YY, ZZ (one or two immediate bytes, if required). Table VI is a summary of the new instructions.

Table VI - New CDP1804/CDP1805 Instructions

TIMER-COUNTER CONTROL	
LDC	LOAD COUNTER
GEC	READ COUNTER
STPC	STOP COUNTER
DTC	DECREMENT COUNT
STM	START TIMER MODE
SCM1	START EVENT COUNTER MODE 1 (EF1)
SCM2	START EVENT COUNTER MODE 2 (EF2)
SPM1	START PULSE WIDTH MODE 1 (EF1)
SPM2	START PULSE WIDTH MODE 2 (EF2)
ETQ	ENABLE TOGGLE Q
INTERRUPT CONTROL	
XIE	ENABLE EXTERNAL INTERRUPT
XID	DISABLE EXTERNAL INTERRUPT
CIE	ENABLE COUNTER INTERRUPT
CID	DISABLE COUNTER INTERRUPT
BCI	BRANCH ON COUNTER INTERRUPT
BXI	BRANCH ON EXTERNAL INTERRUPT
REGISTER CONTROL	
RLDI	REGISTER LOAD IMMEDIATE
RNX	REGISTER-TO-REGISTER COPY
RLXA	MEMORY-TO-REGISTER COPY AND ADVANCE
RSXD	REGISTER-TO-MEMORY COPY AND DECREMENT
SUBROUTINE CONTROL	
SCAL	STANDARD CALL
SRET	STANDARD RETURN

Counter-Timer Control - Ten instructions are available to control the various counter-timer modes described in the previous section under Hardware Improvements. The ten instructions are designed to simplify timer-counter operation by providing both internal and external control of the clock source.

Interrupt Control - Six instructions are used to identify and control interrupts from either the internal counter or the existing CDP1802-type external interrupts. Masking and branching within the current memory page on a priority basis are possible on either interrupt.

Register Control - Four instructions are provided for 16-bit data and memory manipulations. These instructions execute faster and require less program storage space than the equivalent CDP1802 instructions. The D register is not

used for the internal operations, as in the CDP1802, which requires additional instructions to save the contents of D. The four new instructions include a register load immediate, in which two immediate program bytes are transferred to one of the 16-bit scratchpad registers; a register-to-register copy, in which the contents of one of the 16-bit scratchpad registers is transferred to another; a memory-to-register copy, in which two successive memory locations are transferred to one of the 16-bit scratchpad registers; and a register-to-memory copy, in which the contents of one of the 16-bit scratchpad registers is transferred to two successive memory locations. The latter two instructions complement each other and are useful for stack-type operations.

These four additional instructions extend the present CDP1800-series 16-bit register-based operations, in which 16-bit data transfers using the X register and I/O instructions are possible.

Subroutine Control - The two subroutine-control instructions are an instruction-implementation of the CDP1800-series Standard Call and Return Technique, which is used for unlimited nested-subroutine programming. In-line data parameters may also be passed from the calling program to the subroutine without additional instructions.

The standard call instruction (SCAL) is used to call a subroutine at any location in the 65K memory space while saving the present program counter. The location of the subroutine is specified in the two in-line bytes following the call instruction, and is followed by any data that is to be used by the subroutine.

The standard return instruction (SRET) is used to return from a subroutine or nested-subroutine to the main program or the calling program. All registers and the stack are automatically restored.

The SCAL and SRET instructions provide a clean and simple technique for the frequently used subroutine-programming methods commonly required in large software program. Scratchpad registers and considerable program space is saved when this method is used in place of the

existing CDP1802 software technique. The SCAL and SRET instructions execute five times faster than the equivalent CDP1802 technique, as shown in Table VII. However, the SEP technique is still the better choice for small programs that do not employ nested subroutines.

Additional On-Board I/O - Because the CDP1804 and CDP1805 were designed to enhance the CDP1802, but not to an extent that would preclude their use in upgrading existing system designs, no additional I/O capability has been provided on-board. To do so would require a significantly different pin-out for the newer parts, and would eliminate any hardware compatibility with the CDP1802.

The Flag and Q lines of the CDP1804 and CDP1805 still function as those in the CDP1802 for limited on-board I/O. External I/O is easily mapped with the CDP1800-series I/O bus and instruction capability. A variety of complex peripheral devices, which may be used for external I/O requirements, are available in the CDP1800-series product line. Therefore, the user need only add the I/O device necessary for his particular application.

CONCLUSION

The CDP1804 and CDP1805 offer the designer a means of applying state-of-the-art microprocessors to a large variety of product applications with the confidence that his work is supported by a mature product-line and efficient, easy-to-use hardware and software development tools. Many CDP1800-series parts are also available in Hi-Rel (high-reliability) versions for military and aerospace applications.

Present CDP1802 users have the opportunity to upgrade existing systems and add new products with a minimum of design effort; Fig. 3 shows how the CDP1804 and CDP1805 can simplify a CDP1802-based design. New users are offered the inherent advantages of the CMOS technology and the broad-based, well-established RCA CDP1800-series microprocessor product line. In either case, the enhanced features and excellent system performance of the CDP1804 and CDP1805 provide a strong building block for the more powerful and sophisticated system designs of the future.

Table VII - Performance Comparisons—Subroutine Call and Return for CDP1802/04/05 Implemented with Various Techniques

	1802 SOFTWARE "SCRT" TECHNIQUE	1804/05 SOFTWARE SCAL / SRET INSTRUCTIONS	1802 SOFTWARE "SEP" TECHNIQUE	1804/05 SOFTWARE "SEP" TECHNIQUE
NUMBER OF MACHINE CYCLES - CALL	32	10	2	2
NUMBER OF MACHINE CYCLES - RETURN	24	8	4	4
CALL TIME (1802 @ 2.5 MHz) (1804/05 @ 4 MHz)	102.4 μs	20 μs	6.4 μs	4 μs
RETURN TIME (1802 @ 2.5 MHz) (1804/05 @ 4 MHz)	83.2 μs	16 μs	12.8 μs	8 μs
NUMBER OF BYTES SOFTWARE CALL + RETURN	45	6	4	4

92CS-33616

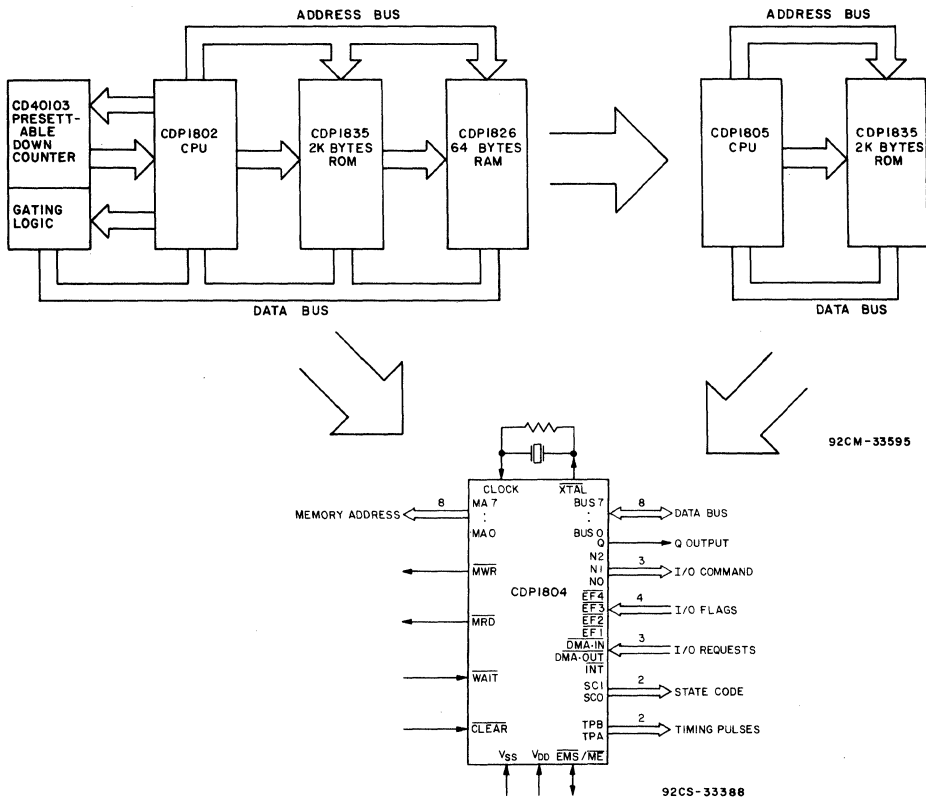


Fig. 3 - CDP1804/CDP1805 system integration.

A Slave CDP1802 Serial Printer Buffer System

by K. Nagy

This Note describes a CDP1802-based stand-alone line-printer buffer that links a master processor system to a serial printer through an RS-232C interface. The main buffer board consists of a CDP1802 Microprocessor (configured as a peripheral controller), a CDP1854A UART with baud-rate generation devices (configured in industry mode 0 and able to handle separate receive and transmit baud rates), and a CDP18U42 EPROM (to store a minimal user program). With the addition of RCA Microboards for system RAM (CDP18S620, CDP18S621, CDP18S622, CDP18S623, CDP18S624, or CDP18S625 in various configurations from 4K to 16K), a buffer system with a printer character storage capacity of up to 32K bytes can be implemented.

The line buffer is especially useful with interactive computer systems; it frees high-speed terminals for other tasks while a slower printer (connected in parallel to the terminal through a serial port, such as is used with RCA Development Systems) prints from buffer memory. The buffer circuitry also illustrates how a CDP1800 processor can be configured as a slave controller that communicates with a master system through a serial port. In this configuration, some of the special I/O features of CDP1800 architecture are highlighted; these features include DMA control for high-speed data transfer, I/O instructions for communication with peripheral chips such as the CDP1854A UART, and the availability of several input flag lines for event polling.

The low power consumption of the system makes it ideal for operation from a storage battery, which provides portability of user data, and also protects data integrity should a line power failure occur.

System Modes

The line buffer operates in the following modes:

1. Receive data—The print buffer is loaded at some (usually high) baud rate (through a serial interface under CDP1802 DMA control).
2. Transmit data—Data is transmitted, usually at a low baud rate (through a serial interface under CDP1802 programmed I/O control), to the printer when the printer is selected and ready.
3. Instant replay—Data is retransmitted to the printer upon request.

These modes permit the system to accommodate a wide range of input and output conditions, and to communicate with many different peripheral devices.

System Set Up

The line-buffer system input and output is connected to suitable sending and receiving units equipped with RS-232C interfaces by means of cables and connectors. The

system output contains a feedback (handshake) line that permits the printer to operate at its optimum speed; the feedback feature can be eliminated if necessary through software changes.

The printer-buffer prototype was evaluated using a CDP18S007 CDS Development System, an ADM-3A CRT Terminal, and an EPSON MX-80 Printer. However, the universal nature of the buffer design permits it to be used with a wide variety of master system and printer combinations.

Hardware

A block diagram of the system is shown in Fig. 1; detailed system interconnection diagrams are shown in Appendix A. The "U" designators used in the following text refer to the figures in Appendix A, and are identified in Table I and in the Appendix. Table II summarizes register assignments.

Table I — List of IC's Used in System

IC	Identity
U1	CDP1802 CPU
U2	CDP18U42 EPROM
U3	CDP1854A UART
U4	CD4013 D FF
U5, U6	CD4069 Hex Inverter
U7	CD4093 Quad 2-Input NAND
U8	CD4013 D FF
U9, U10, U11	CD40161 Sync Binary Counter
U12	MC1489 Quad Line Receiver
U13	MC1488 Quad Line Driver

Table II — Line-Buffer Register Use

R0	Initial Program Counter/DMA-IN Pointer
R3	Program Counter
R4	Printer Pointer
R5	End of Memory Pointer
R6	XOR Temporary Location Pointer
R7	Temporary DMA-IN Pointer
R8	Cold Start Flag

The system program resides in the CDP18U42, 256-byte UV Erasable PROM, U2, located in the lower 32K address space on the system map. The chip-select signal for the EPROM is generated by latching the high-order address byte of the MA7 signal by means of a CD4013 flip-flop (U4). The EPROM is selected when MA.1 is low ($\overline{CS1}$), MRD is low ($\overline{CS3}$), and CLR is high (CS2).

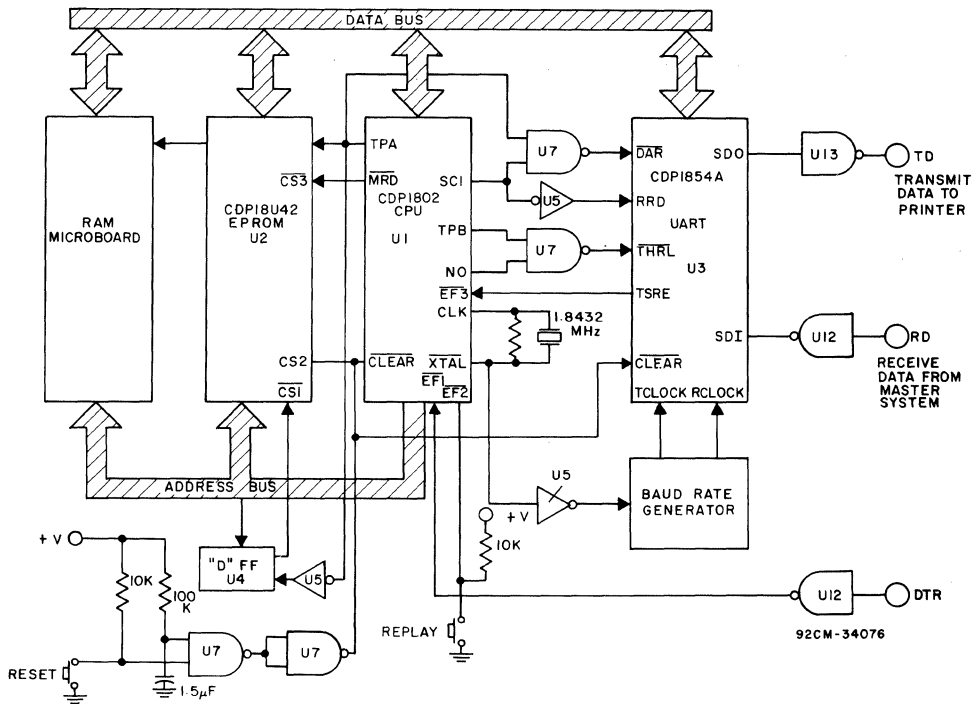


Fig. 1 - Block diagram of the serial printer buffer system.

The internal oscillator of the CDP1802 microprocessor is driven by a 1.8432-MHz external crystal, which acts as the system master clock. The crystal is connected between the CLOCK and XTAL terminals of the microprocessor. One inverter of the CD4069 (U5) provides a buffered signal called CLK OUT to the baud-rate generator divider chain for the UART.

Part of a CD4093 (U7) forms a power-on reset circuit that outputs a CLEAR signal to the microprocessor and the UART. An external reset signal can force the processor to restart program execution. An SPST switch connected to the EF2 flag-input terminal of the microprocessor provides the instant-replay request signal for the program. If this switch is in the replay mode, activation of the reset button will start the printing of the previously deposited contents of the RAM buffer. This feature relieves the host computer of the time-consuming task of producing copies on the printer.

A CDP4013, Dual D Flip-Flop (U8), forms a synchronous divide-by-three circuit; its output is connected to two CD40161B cascaded synchronous binary counters (U9 and U10). The Q outputs of U9 and U10 provide baud rates from 150 to 19,200 baud. The accuracy of these baud rates is comparable to that of the crystal oscillator. Another CD40161B synchronous binary counter (U11) is used in a divide by 11 configuration to generate the 110 baud used by some teletypes and other slow mechanical printers. Because it is possible to input to, or output from, this system at varying baud rates, the system is very useful in interactive applications or where baud-rate exchange is required.

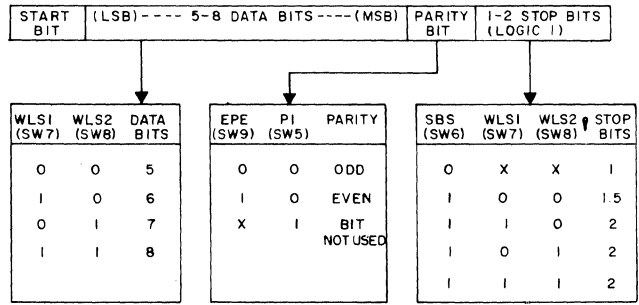
The CDP1854A UART (U3) has two modes of operation. Mode 1 is directly compatible with the CDP1800 family of microprocessors without additional interface circuitry; software techniques are used for programming in this

mode.¹ In Mode 0, the mode used in the serial printer buffer system, the CDP1854A is compatible with a great many other UART's, such as the TR1602A by Western Digital, and provides the user with hardware-option selection in the form of five independent switches. The functions of the switches are as follows:

- SW5-When open, the generation of parity is inhibited.
- SW6-When open, two stop bits are selected, when closed, one is selected. Selection of two stop bits with five data bits programmed selects 1.5 stop bits.
- SW7, SW8- These two switches select the character length (exclusive of parity) of five to eight data bits.
- SW9- This switch is closed for odd parity, opened for even parity.

Fig. 2 shows serial-word format-programming data. The UART has two independent byte-wide buses. Both receive and transmit buses are connected to the system data bus.

The system receive operation is initiated when a serial bit stream is sent via the RS-232 receiver interface to the UART. When the UART receiver assembles a full received character, the data available (DA) output goes high and initiates a DMA-IN cycle. During DMA, the CPU is forced into a DMA cycle (S2), which causes SC1 to go high; the result is a transfer of data from the UART receiver holding register to the data bus. The address and control signals needed to direct the flow of data into the RAM pointed to by DMA pointer R0 are generated during DMA. Additionally, hardware interconnections are made that cause the data available reset (DAR) flag to be reset. Reset is accomplished through a signal to the data available reset (DAR) input; the signal terminates the DMA-IN request before it is again sampled by the CPU. Finally, R0 is incremented at the end of the S2 cycle, and the system is ready for transfer of another byte of data when it is received.



92CS-34075

Fig. 2 - Description of serial word format programming.

The system transmit operation begins when the UART transmitter shift register is empty (TSRE high, polled via EF3) and the printer is selected (polled via EF1). The CPU issues an I/O command that signals the transmitter holding register load (THRL) input to transfer data from the data bus to the transmitter holding register. The data is serialized by the transmitter shift register and output, in the format selected by the user, through the serial data out (SDO) line and the RS-232 line driver to the printer. During the time that data is actively being transmitted by the UART, the TSRE signal will be low, preventing transmission of additional data. When the TSRE signal becomes high, and the printer is ready, a character can be transferred from the system data bus to the transmitter holding register. The microprocessor then outputs a low pulse to the UART transmitter holding register load (THRL) input, causing TSRE to go low. When the TSRE output signal goes high again, and EF1 is low (indicating that the printer is ready), another character can be loaded into the transmitter holding register for transmission. This process is repeated until all of the line-buffer contents are transmitted.

Baud-Rate Generation

The CDP1854A UART requires a clock signal that is 16 times the desired bit rate for proper operation. This 16X clock rate is applied to the UART receive clock (RCLOCK) and transmitter clock (TCLOCK) inputs. In the subject system, the RCLOCK and TCLOCK inputs are connected to separate frequencies (through SW 3 and SW 4). Both of these bit rates are derived from the same crystal oscillator and divider circuitry.

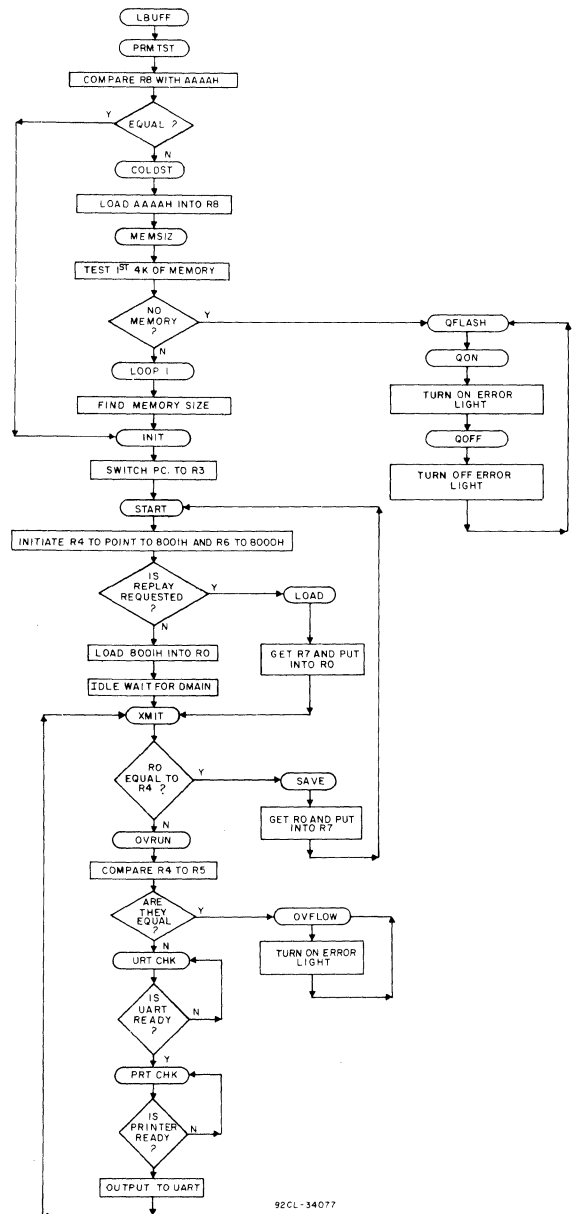
The bit rate and the number of bits per character determine the number of characters that can be transmitted in one second. If the UART receiver is required to operate at a data rate of 19,200 bits/s, then the receive clock (RCLOCK) frequency must be $16 \times 19,200 = 307,200$ Hz. Consequently, if the data transmission rate must be 300 bits/s, then the TCLOCK input of the UART must be connected to a clock frequency of $16 \times 300 = 4800$ Hz. Table III shows baud rates and frequencies supported by the system.

Software

The program flowchart is shown in Fig. 3. Upon power-on,

Table III — Baud Rates and Frequencies Supported by System

Baud Rate	16X Freq. (kHz)	Time (μs)
110	1.760	568.18
150	2.400	416.67
300	4.800	208.33
600	9.600	104.17
1200	19.200	52.08
2400	38.400	26.04
4800	76.800	13.02
9600	153.600	6.51
19200	307.200	3.26



92CL-34077

Fig. 3 - Program flowchart.

ICAN-6991

the power-on reset circuit initiates program execution from the first ROM location (located at 0000). After program execution has begun, the first part of the program, PRMTST, tests register R8 to determine if a "cold" or first start is being initiated. If the start is not a first start, then R8 contains AAAAH, which results in the execution of the routine INIT. If a cold start is encountered, R8.1 and R8.0 will contain random values, and the program will branch to the cold-start routine COLDST. This routine sets R8 to AAAAH, and passes control to the memory-size test routine, MEMSIZ, which determines the amount of RAM available in the system for line-buffer use. If there is no RAM connected to the system, the QFLASH routine is entered, and the error LED flashes on and off until the condition is corrected and the reset button pressed. Upon completion of the memory-size routine, register R5 will contain the size of the user RAM. The system then undergoes initialization under software control through execution of the routine INIT.

The INIT routine initializes the printer pointer and DMA pointer to 8001H (the first available RAM location) and the program counter to R3. A byte of RAM is set aside at location 8000H for use by the program. When initialization is complete, the program checks for the instant-replay request ($\overline{EF2}$ low); if it finds no request, it goes to idle and waits for the first byte received by the UART receiver to initiate a DMA-IN.

As the processor receives a DMA-IN request, it places the byte, now on the data bus, in the location pointed to by DMA pointer R0. When the DMA cycle is completed, the processing of the XMIT routine begins by comparing the DMA pointer (R0) to the printer pointer (R4). If these two pointers are not set to the same location in memory, the XMIT routine determines whether the printer pointer is set to an address within the valid boundaries of the memory buffer. If it is not, an error is indicated by an error light; if it is, program control is passed to the UART checking routine, URTCHK.

The URTCHK routine first checks the UART transmitter-shift-register-empty signal, TSRE. If TSRE is high, control goes to the printer checking routine, PRTCHK. When this routine receives a ready signal from the printer, it outputs a character to the printer through the transmitter side of the UART. After the byte has been transferred to the UART transmitter holding register, the system proceeds with the XMIT routine until the entire contents of the buffer are printed out.

In the subject system, all incoming data is deposited in memory in real time by DMA-IN within an execution time of one CPU machine cycle per byte. This method is similar but superior to that of an interrupt driven system because it does not have the overhead associated with the interrupt software routine and, therefore, can easily cope with high incoming baud rates.

At baud rates of 19,200 bits per second and 10 bits per character, the system has approximately 520 microseconds of time available for processing between two successive DMA-IN pulses. At a system clock rate of 1.8432 MHz (selected for the baud-rate generation circuits for the UART), the system has enough time to execute approximately 120 machine cycles before the next DMA-IN cycle comes in.

The software assembly listing in ASM 4 is shown in Appendix B.

Summary

In summary, this system is especially useful in applications requiring interaction with a variety of types of equipment as it is capable of taking input or providing output at varying baud rates. The instant-replay feature relieves a host computer of the time consuming task of producing copies on a printer. The system UART, when used in mode 0, constitutes a peripheral device that makes interfacing of the line-buffer system with other systems a simple matter. The low power consumption of the line-buffer system makes it an ideal one for operation from a storage battery, which provides portability of user data and protection of this data in the event of a line-power failure.

References and Bibliography

1. "Use of the CDP1854 UART with RCA Microprocessor Evaluation Kit CDP18S020 or EK/Assembler-Editor Design Kit CDP18S034," R.G. Ott, RCA Solid State Application Note ICAN-6632.
2. "COS/MOS Memories, Microprocessors, and Support Systems," RCA Solid State Databook SSD-260.
3. "COS/MOS Integrated Circuits," RCA Solid State Databook SSD-250.
4. "COS/MOS Integrated Circuits Manual," RCA Solid State publication CMS-272.
5. "User Manual for the CDP1802 COSMAC Microprocessor," RCA Solid State publication MPM-201.

Appendix A

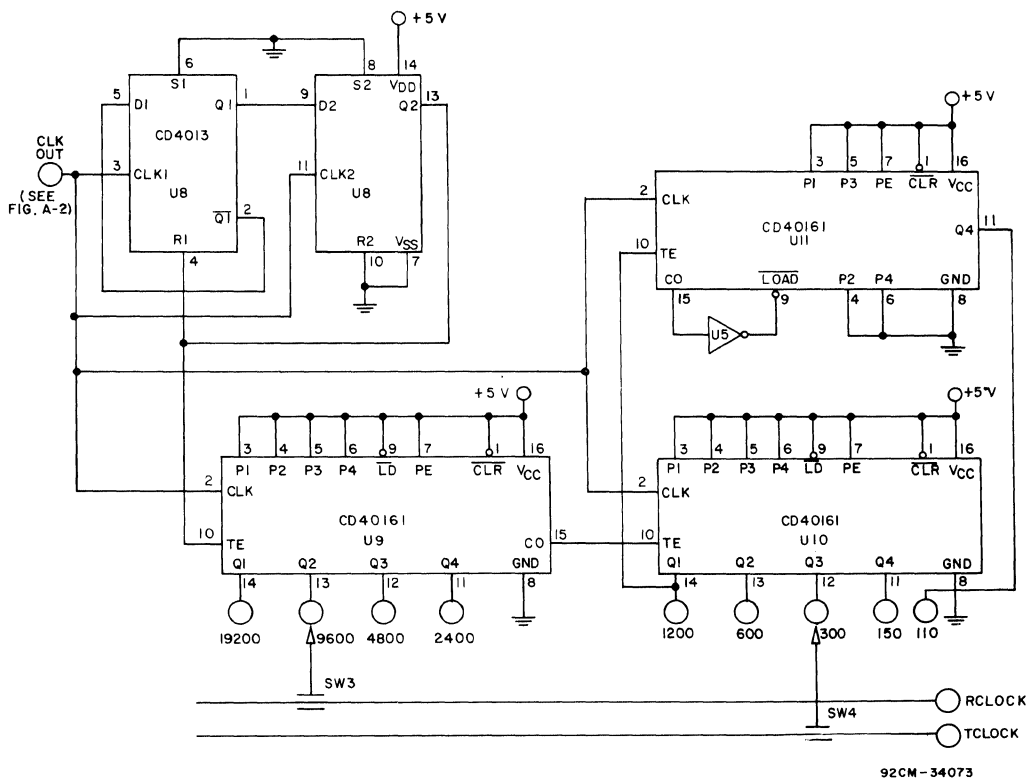


Fig. A-1 - Interconnection diagram for line-buffer baud-rate generator.

Table A-I — Microboard Computer Bus Interface (Fig. A-2)

Pin	Signal	Pin	Signal
A	TPA-P	1	DMA1-N
B	TPB-P	2	DMA0-N
C	DB0-P	3	RNU-P
D	DB1-P	4	INT-N
E	DB2-P	5	MRD-N
F	DB3-P	6	Q-P
H	DB4-P	7	SC0-P
J	DB5-P	8	SC1-P
K	DB6-P	9	CLEAR-N
L	DB7-P	10	WAIT-N
M	A0-P	11	-5 V/-15 V
N	A1-P	12	SPARE
P	A2-P	13	CLOCK OUT
R	A3-P	14	N0-P
S	A4-P	15	N1-P
T	A5-P	16	N2-P
U	A6-P	17	EF1-N
V	A7-P	18	EF2-N
W	MRW-N	19	EF3-N
X	EF4-N	20	+12 V/+15 V
Y	+5 V	21	+5 V
Z	GND	22	GND

Table A-II — List of IC's Used in System

IC	Identity
U1	CDP1802 CPU
U2	CDP18U42 EPROM
U3	CDP1854A UART
U4	CD4013 D FF
U5, U6	CD4069 Hex Inverter
U7	CD4093 Quad 2-Input NAND
U8	CD4013 D FF
U9, U10, U11	CD40161 Sync Binary Counter
U12	MC1489 Quad Line Receiver
U13	MC1488 Quad Line Driver

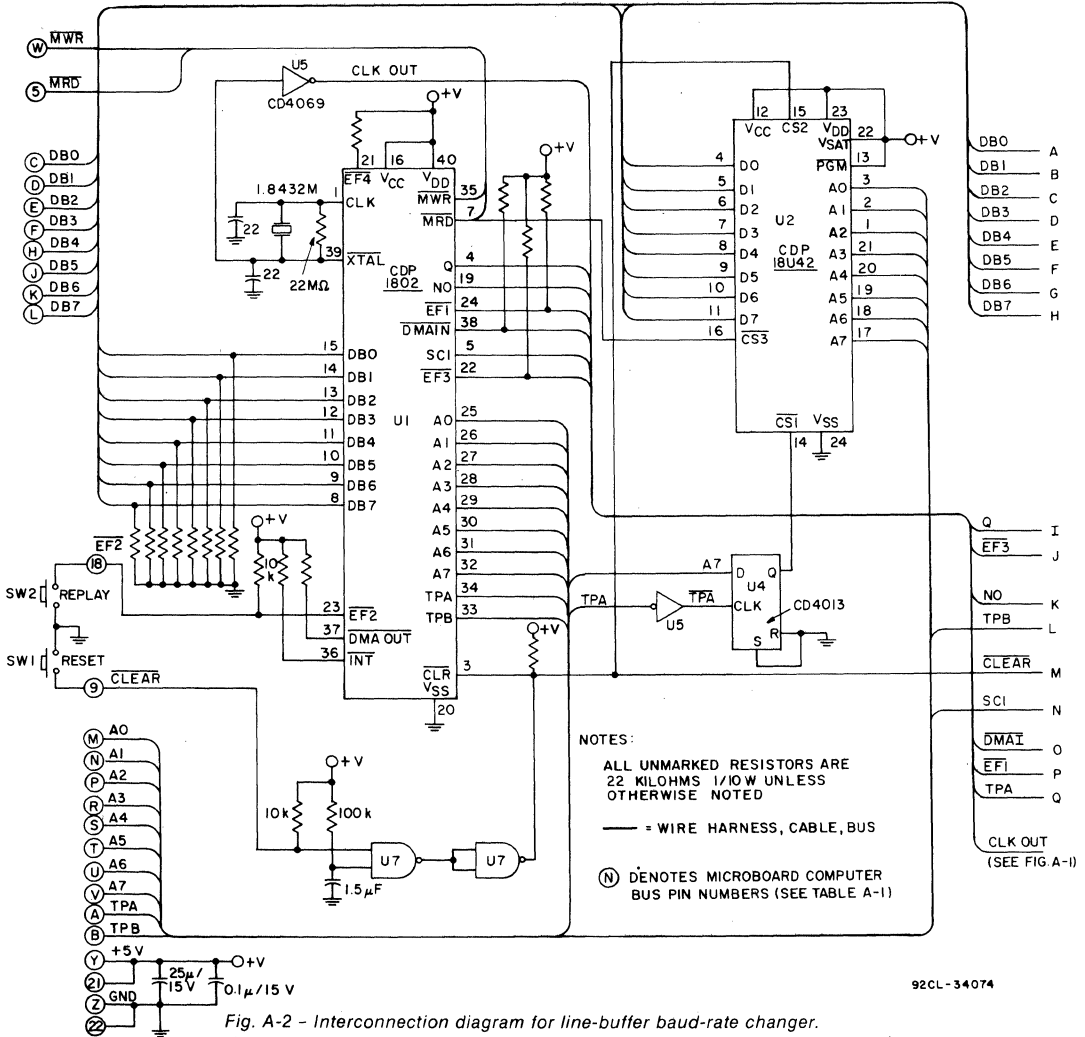


Fig. A-2 - Interconnection diagram for line-buffer baud-rate changer.
 (Letters A through Q on right match this figure with Fig. A-3 on facing page.)

Appendix B — Program Listing

```

0000 ;
0000 ;
0000 ;
0000 ;
0000 ;
0000 ;
0000 ;
0000 ;
0000 ;
0000 ;
0000 F8B0;
0002 B5;
0003 F800;
0005 A5;
0006 E5;
0007 98;
0008 55;
0009 FBAA;
000B F5;
000C 3AB9;
000E 88;
000F 55;
0010 FBAA;
0012 F5;
0013 3AB9;
0015 3044;
0017 F8BF;

0001 ..*****
0002 ..*
0003 ..* * * LBUFF 3-15-81 K. NAGY * * *
0004 ..*
0005 ..*****
0006 ..
0007 ..
0008
0009 PRMTST: LDI #80 ..CHECK FOR COLD START
0010 PHI R5
0011 LDI #00
0012 PLO R5
0013 SEX R5
0014 GHI R8
0015 STR R5
0016 LDI #AA
0017 SD
0018 BNZ COLDST
0019 GLO R8
0020 STR R5
0021 LDI #AA
0022 SD
0023 BNZ COLDST
0024 BR INIT
0025 MEMSIZ: LDI #8F ..CHECK FOR MEMORY SIZE
    
```

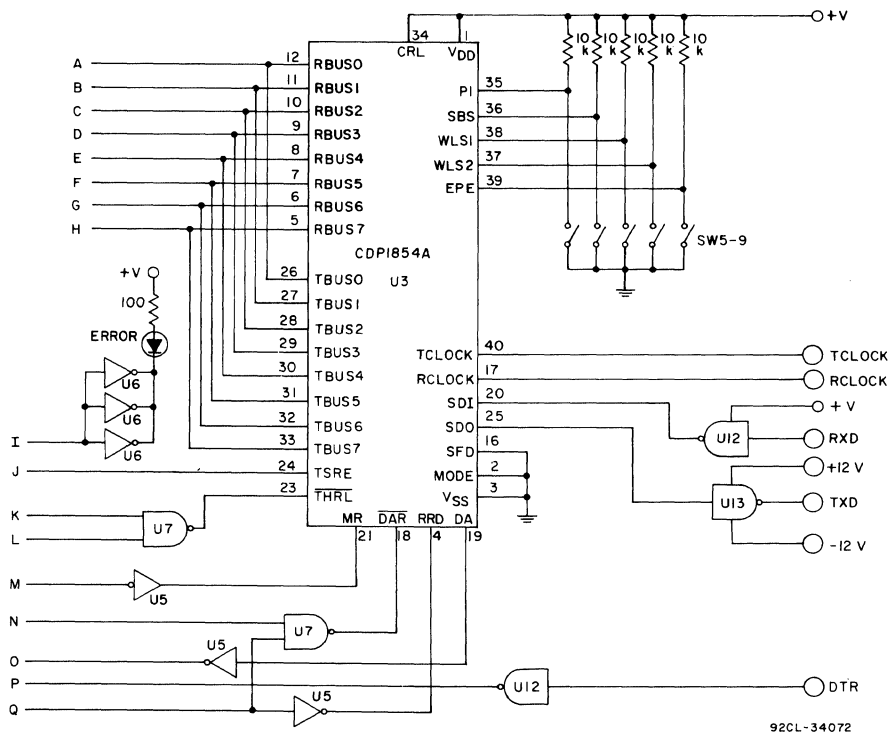


Fig. A-3 - Interconnection diagram for line-buffer receiver/transmitter.
 (Letters A through Q on left match this figure with Fig. A-2 on facing page.)

Appendix B (Cont'd)

```

0019 B5;          0026      PHI R5          ..IN THE SYSTEM MIN. 4K.
001A FBFF;       0027      LDI #FF          ..MAX. 32K.
001C A5;         0028      PLO R5
001D E5;         0029      SEX R5
001E FBAA;       0030      LDI #AA
0020 55;         0031      STR R5
0021 F5;         0032      SD
0022 3236;       0033      BZ LOOP1
0024 7B;         0034 OFLASH: SEQ          .. IF SYSTEM MEMORY IS
0025 FB20;       0035      LDI #20          ..LESS THAN 4K FLASH
0027 B5;         0036      PHI R5          ..ERROR LITE
0028 25;         0037 QON:   DEC R5
0029 95;         0038      GHI R5
002A 3A28;       0039      BNZ QON
002C 7A;         0040      REQ
002D FB20;       0041      LDI #20
002F B5;         0042      PHI R5
0030 25;         0043 QOFF:  DEC R5
0031 95;         0044      GHI R5
0032 3A30;       0045      BNZ QOFF
0034 3024;       0046      BR OFLASH
0036 95;         0047 LOOP1: GHI R5
0037 FC10;       0048      ADI #10
0039 B5;         0049      PHI R5
003A FBAA;       0050      LDI #AA
    
```

Appendix B (Cont'd)

```

003C 55;          0051      STR R5
003D F5;          0052      SD
003E 3236;       0053      BZ LOOP1
0040 95;          0054      GHI R5
0041 FF10;       0055      SMI #10
0043 B5;          0056      PHI R5          ..R5=MEMORY SIZE
0044 F800;       0057 INIT:  LDI #00
0046 B3;          0058      PHI R3          ..SWITCH PC TO R3
0047 A6;          0059      PLO R6
0048 F84C;       0060      LDI A.0(START)
004A A3;          0061      PLO R3
004B D3;          0062      SEP R3
004C F8B0;       0063 START:  LDI #B0          ..INIT.PRINTER POINTER - R4
004E B6;          0064      PHI R6          ..TO BEGINNING OF RAM: 8001H
004F B4;          0065      PHI R4          ..AND XOR TEMPORARY HOLDING
0050 FB01;       0066      LDI #01          ..LOCATION - R6 TO 8000H
0052 A4;          0067      PLO R4
0053 3583;       0068      B2 LOAD          ..CHECK FOR INSTANT REPLAY
0055 A0;          0069      PLO R0          ..REQUEST. REPLAY IF 0
0056 F8B0;       0070      LDI #B0
0058 B0;          0071      PHI R0          ..INITIALIZE DMA POINTER - R0
0059 F800;       0072      LDI #00          ..TO 8001H
005B 00;          0073      IDL
005C E6;          0074 XMIT:  SEX R6          ..ALL INPUT PRINTED?
005D 94;          0075      GHI R4
005E 56;          0076      STR R6
005F 90;          0077      GHI R0
0060 F3;          0078      XOR
0061 3A69;       0079      BNZ OVRUN
0063 B4;          0080      GLO R4
0064 56;          0081      STR R6
0065 B0;          0082      GLO R0
0066 F3;          0083      XOR
0067 327D;       0084      BZ SAVE
0069 95;          0085 OVRUN:  GHI R5          ..OVER-RUN TEST ON
006A 56;          0086      STR R6          ..PRINT BUFFER
006B 94;          0087      GHI R4
006C F3;          0088      XOR
006D 3A75;       0089      BNZ URTCHK
006F B5;          0090      GLO R5
0070 56;          0091      STR R6
0071 B4;          0092      GLO R4
0072 F3;          0093      XOR
0073 328F;       0094      BZ OVFLOW
0075 3675;       0095 URTCHK:  B3 URTCHK          ..WAIT FOR UART READY
0077 3C77;       0096 PRTCHK:  BN1 PRTCHK          ..PRINTER READY TEST
0079 E4;          0097      SEX R4          ..READY IF HI
007A 61;          0098      OUT 1
007B 305C;       0099      BR XMIT
007D 90;          0100 SAVE:  GHI R0
007E B7;          0101      PHI R7
007F B0;          0102      GLO R0
0080 A7;          0103      PLO R7
0081 304C;       0104      BR START
0083 97;          0105 LOAD:  GHI R7
0084 B0;          0106      PHI R0
0085 B7;          0107      GLO R7
0086 A0;          0108      PLO R0
0087 305C;       0109      BR XMIT
0089 FBAA;       0110 COLDST:  LDI #AA
008B B8;          0111      PHI R8
008C AB;          0112      PLO R8
008D 3017;       0113      BR MEMSIZ
008F 7B;          0114 OVFLOW:  SEQ          ..PRINT LENGTH EXCEEDED AMOUNT
0090 308F;       0115      BR OVFLOW          ..OF AVAILABLE RAM IN SYSTEM.
0092 ;           0116      END          ..ERROR LITE IS ON STEADY.
0000

```

New CDP1805 Microprocessor Upgrades CDP1800-Based Systems

by J. Paradise

A new microprocessor, the CDP1805, speeds throughput and reduces chip count while retaining all the advantages of the register-based CDP1800-series architecture.

The RCA CDP1800 microprocessor series is a well-tried LSI product line that implements control-oriented microcomputer functions in CMOS technology. The preeminent device of the series, the CDP1802, has been in production since 1977, with yearly output now exceeding one million devices per year. Finding its way into a wide range of applications that require low power for portability, high noise immunity for satisfactory operation in industrial environments, or a wide operating temperature and voltage range to assure optimum performance under a wide variety of ambient conditions, the CDP1802 has established itself as the leading CMOS microprocessor. With the introduction of the CDP1805 by RCA, CDP1800-based designs can be further enhanced, and proposed new or improved system designs whose performance requirements previously exceeded the capability of the CDP1802 can now be implemented.

The CDP1800-Series Microprocessor Architecture

This section will help readers unfamiliar with CDP1800-series architecture to understand its fundamentals and salient features, so that the enhancements achieved in the CDP1805 can be more fully appreciated.

As shown in Fig. 1, the central feature of the CDP1800-series microprocessor is an array of sixteen 16-bit scratchpad registers used to provide control over memory addressing and internal housekeeping. When used to address memory, the registers are selected by software instructions that load 4-bit values into register selectors (P,N,X). These selectors program the scratchpad registers as program counters, memory pointers, and stack pointers. This assignment flexibility allows the use of multiple-pointer and context-switching techniques, and provides quick subroutine-call implementation and efficient stack and interrupt handling in real-time control applications. In addition, these same registers can be used for variable data storage, providing up to 30 bytes of data memory on-chip.

The 16-bit-wide register-array matrix provides two advantages in its dual use in address and data operations. A 2^{16} or 65,536-byte memory address range is possible for RAM /ROM addressing in main program, subroutine, and stack operations. For data storage, the contents of each 16-bit register can be used as a 16-bit data word, with instructions for increment, decrement, register-to-register, and register-to-memory manipulation of 16-bit operands. Thus, 16-bit arithmetic and logic operations can be supported by this internal architectural configuration.

Externally, CDP1800-series devices interface with a variety of bus and I/O pins to external memory and peripheral functions. Memory addresses are generated via a multiplexed address bus; a latching TPA signal is provided for the high-order byte. I/O addressing is accomplished via a separate 3-bit I/O bus with dedicated instructions for data transfer between memory and peripherals.

Other I/O lines are provided for on-board DMA address and control-line generation, for interrupt vectoring, for use as flag lines for polling, and for implementation of a Q-line-output port for control of external devices. The flag and Q lines, in conjunction with a software-driver routine, can also be used as a serial port to external I/O.

The remainder of this Note describes specific CDP1805 features and explains how they are optimally applied in a microcomputer system.

CDP1805 - THE SPECIFICS

On-Board RAM Complement

The CDP1805 includes, on-board, a 64-byte RAM array in addition to the 16×16 scratchpad register array described above. The RAM's six address lines are internally connected to the CPU section; the eight data lines are part of the CPU's external bus. The chip-select function for the RAM is available through pin 16 (the V_{CC} connection on the

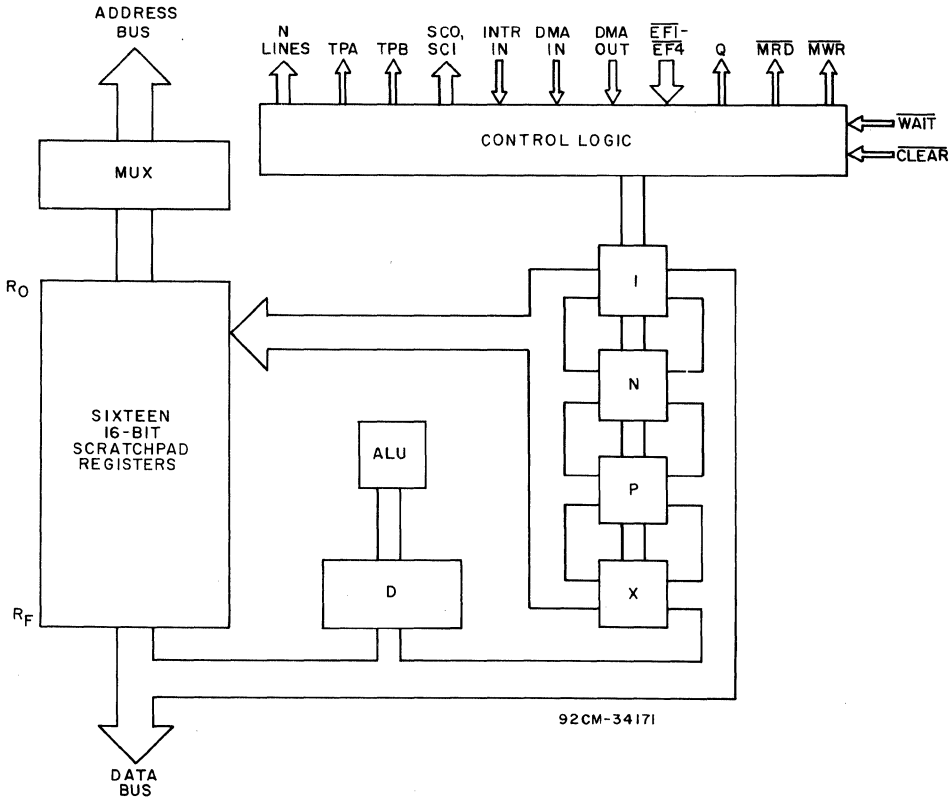


Fig. 1 - CDP1800-series architecture features a scratchpad register array for address and data manipulation.

CDP1802). Thus, the RAM is essentially configured as independent memory, with its 64-byte block locatable anywhere in memory space, and with its data lines available to external I/O for reading and writing. The 64-byte block is of adequate size for a stack area, being able to handle many levels of subroutine and interrupt nesting, as well as providing sufficient space for a modest data stack for main program or I/O routines. Figure 2 shows a typical connection of the CDP1805 in a CDP1800-based system.

On-Board Counter-Timer

The CDP1805 contains an 8-bit presettable down-counter on-chip. The counter is configurable in a wide variety of modes through the use of new CDP1805 software instructions (Fig. 3); external input and output lines are available via the flag and Q pins. The counter features an internal counter interrupt, which is activated on counter underflow, and internal execution similar to the external interrupt mechanism found on the CDP1802.

The counter is loaded with a desired count, and preset to its intended operating mode, by means of software instructions. Counter contents may be loaded into the CPU D register and manually decremented, a function that can replace software loops in timing applications. In its dynamic operating modes (of which there are five) the counter-timer may receive its internal decrement-clock signal from TPA divided by a ÷ 32 prescaler, or from flag lines EF1 and EF2 (Fig. 4).

In the timer mode, the effective clock input is the crystal frequency divided by 256 ($XTAL \div 8 = TPA, TPA \div 32 =$ prescaler output). Interrupts are generated on each counter underflow; the preset counter value is reloaded into the counter after each interrupt. In the timer mode, the counter-timer can function as an accurate time base in real-time applications. An ETQ instruction causes Q to toggle with each interrupt, creating a programmable-frequency square wave for external-device control applications.

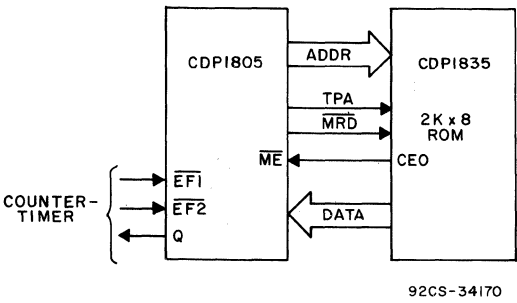


Fig. 2 - Minimum connection diagram for a CDP1800-based system that includes CPU, 2-kilobyte ROM, 64-byte RAM, and a counter-timer.

ICAN-7009

LDC	—	LOAD
GEC	—	READ
STPC	—	STOP
DTC	—	DECREMENT
STM	—	TIMER MODE
SCM1	—	EVENT COUNTER VIA EF1
SCM2	—	EVENT COUNTER VIA EF2
SPM1	—	PULSE WIDTH VIA EF1
SPM2	—	PULSE WIDTH VIA EF2
ETQ	—	COUNTER OUTPUT VIA Q

Fig. 3 - CDP1805 counter-timer instructions for manual control and selection of one of five dynamic modes.

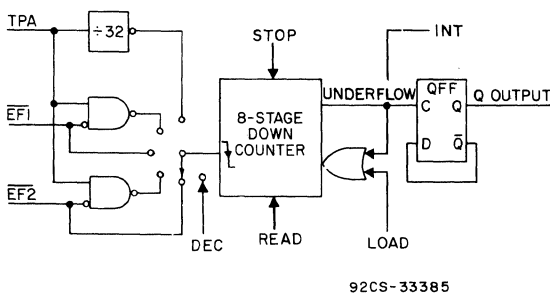


Fig. 4 - CDP1805 counter-timer model. The 8-bit down-counter has a variety of possible inputs and an output available from the Q line.

Event counting can be performed in the SCM1 and SCM2 modes (Fig. 3) by bypassing the internal TPA clock and feeding inputs directly from $\overline{EF1}$ and $\overline{EF2}$. This arrangement still permits the flag lines to retain their normal functions. Again, the counter generates an interrupt when a preset number of external events have occurred.

$\overline{EF1}$ and $\overline{EF2}$ can also be used to make pulse-width measurements in the SPM1 or SPM2 modes (Fig. 3). In either of these modes, $\overline{EF1}$ and $\overline{EF2}$ are used as a gate to the TPA clock without the prescale divider. An interrupt is generated, with the remaining count frozen, on the trailing-edge transition of the flag line. Thus, the counter value represents the width of the pulse appearing at $\overline{EF1}$ or $\overline{EF2}$. Since the two flag inputs may be used together, these modes are useful for comparison-type measurements.

These generalized explanations give some idea of how the counter can be used in control applications. The specific examples given below employ some of these counter-timer functions and illustrate other powerful features of CDP1800-series I/O.

Enhanced Hardware DMA: The Counter-Timer

On-board DMA control is a useful I/O feature of CDP1800 architecture, especially in data transfer operations where high speed is essential. Instead of requiring an external controller to provide address and memory read/write signals, the CDP1800 processor performs these functions automatically, through register R(0), when DMA-In or

DMA-Out pins are activated. The counter-timer, along with a single inexpensive CD4011, a CMOS NAND gate, keeps track of the desired number of DMA transfers, with the peripheral only required to issue a DMA request in the form of a single short pulse. The modest internal software initialization and maintenance required with this method results in an acceptable trade-off with external hardware techniques.

In the configuration shown in Fig. 5, the CDP1805 itself keeps track of DMA transfers. The counter-timer is placed in event-counting-mode SCM1, and counts TPA transitions that occur during the time that DMA is active. When the desired number of transfers is complete, Q toggles, ending the DMA mode.

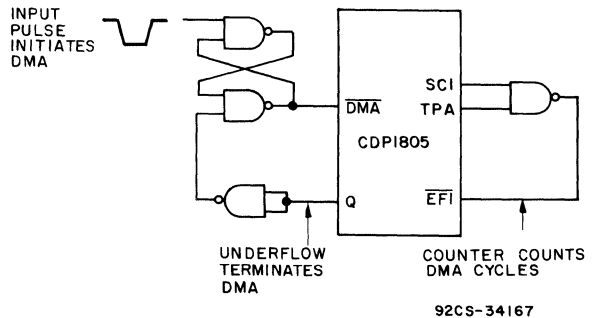


Fig. 5 - CDP1805-enhanced DMA operation using on-board counter-timer. The counter-timer can be preset to count from 1 to 255 DMA transfers.

Software DMA with the CDP1805 Counter-Timer

When properly initialized, the CDP1805 can be used to generate software-driven DMA (Fig. 6). In this application, the counter-timer is set to pulse-mode SPM1. An SEQ instruction, locatable anywhere in software, forces DMA to begin. The counter counts internal TPA pulses until underflow, at which time Q toggles and terminates the DMA mode. The process may be repeated anywhere in the software sequence; the number of transferred bytes is determined by the count loaded into the counter-timer.

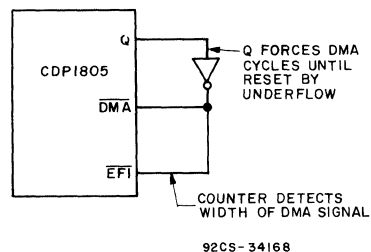


Fig. 6 - Software-controlled DMA using counter-timer. The user has full control over any DMA transfer.

Adding a Second Interrupt to the CDP1805

The internal counter-timer interrupt that is generated within the CDP1805 can be used to advantage as a second external interrupt, complete with a dedicated handshake line. Further-

more, this interrupt is edge rather than level sensitive, and the interrupt-acknowledge signal can be reset manually within the service routine to provide accurate status information to a peripheral.

For this operation, shown in Fig. 7, the counter-timer is set to the counter mode, with a count of 01 preloaded, and with ETQ enabled. The first high-to-low input-signal transition causes a counter underflow, generating an interrupt and creating ACK2. The interrupt source is arbitrated within the interrupt-service routine, during which time ACK2 can be reset.

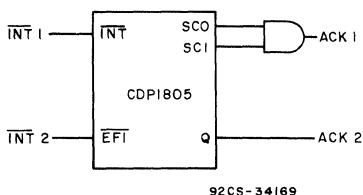


Fig. 7 - CDP1805 multiple-interrupt capability with handshaking using counter-timer. The additional interrupt is edge-sensitive and has its own handshake line.

Enhanced Interrupt Control

The type of interrupt action described in the counter-timer discussion above is made possible because of the enhanced interrupt-control logic and instructions of the CDP1805. Since two interrupts must be dealt with internally in the CDP1805, six additional linked instructions, shown in Fig. 8, have been provided for arbitration and control. In addition to standard CDP1800-series RET and DIS instructions, which provide master interrupt control, separate XIE, XID, CIE, and CID instructions are provided for independent control of external and counter interrupts. In addition, both external and counter interrupts are pollable by means of BXI and BCI short-branch instructions.

- XIE — ENABLE EXTERNAL INT**
- XID — DISABLE EXTERNAL INT**
- CIE — ENABLE COUNTER INT**
- CID — DISABLE COUNTER INT**
- BCI — BRANCH ON COUNTER INT**
- BXI — BRANCH ON EXTERNAL INT**

Fig. 8 - CDP1805 interrupt-control instructions. These instructions arbitrate between external and counter interrupts.

The advantages to the user of this structure are that he can enable or disable a counter or external interrupt request with a simple instruction, and avoid the indirect programming structure of RET and DIS for simple enable or disable functions. Although not latched, the external interrupt is pollable, making it useful as a fifth flag line, if desired. Finally, the latched counter interrupt can be tested as a real-time or polled event by selective use of the CIE, CID, and BCI instructions.

RC-Oscillator Capability

The addition of a Schmitt trigger in the CDP1805 oscillator section provides crystal or RC-oscillator capability (Fig. 9). An RC oscillator can have several advantages over a crystal

in noncritical timing applications. The most obvious advantage is in cost; the crystal typically costs ten times more than the resistor-capacitor combination. A temperature-compensated capacitor can be purchased to improve stability further when required by the application.

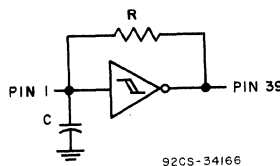


Fig. 9 - CDP1805 oscillator configuration using RC time constant. The internal Schmitt trigger allows crystal or RC operation.

The RC structure also permits easier change of frequency. The resistor can be replaced with a simple user-adjustable potentiometer. Further, step changes in frequency for specific applications, time-base changes, or lower power considerations can be accommodated through simple RC-switching techniques. By way of providing basic design guidelines, Fig. 10 matches resistor and capacitor values with frequency.

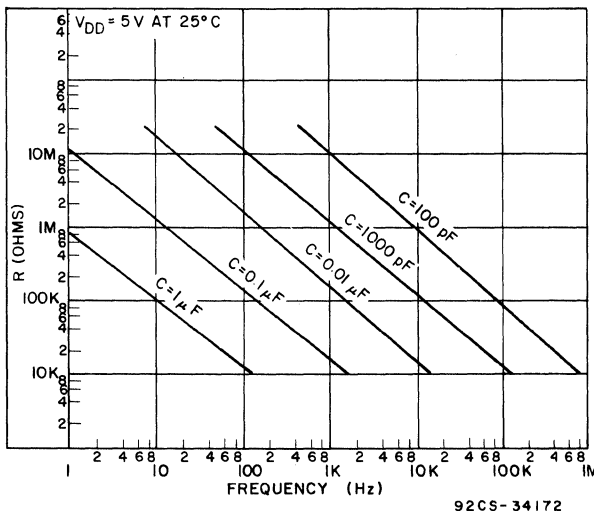


Fig. 10 - Component values as a function of frequency for the RC oscillator. The graph shows that frequencies up to 1 MHz are achievable.

High Clock Rate

The 4-MHz clock speed of the CDP1805, a 25-percent increase over the 3.2 MHz of the CDP1802A, can give the CDP1805 the performance edge in certain applications. This performance edge, together with the enhanced data instructions described below, can produce fast throughput in both real-time and response-critical applications.

Table I — Performance Comparisons-Subroutine Call and Routine for CDP1802 and CDP1805 Implemented with Various Techniques

	1802 SOFTWARE "SCRT" TECHNIQUE	1802 SOFTWARE "SEP" TECHNIQUE	1804/05 SOFTWARE SCAL / SRET INSTRUCTIONS	1804/05 SOFTWARE "SEP" TECHNIQUE
NUMBER OF MACHINE CYCLES - CALL	32	2	10	2
NUMBER OF MACHINE CYCLES - RETURN	24	4	8	4
CALL TIME (1802 @ 3.2 MHz) (1804/05 @ 4 MHz)	80 μ s	5 μ s	20 μ s	4 μ s
RETURN TIME (1802 @ 3.2 MHz) (1804/05 @ 4 MHz)	60 μ s	10 μ s	16 μ s	8 μ s
NUMBER OF BYTES SOFTWARE CALL + RETURN	45	4	6	4

92CS-3339I

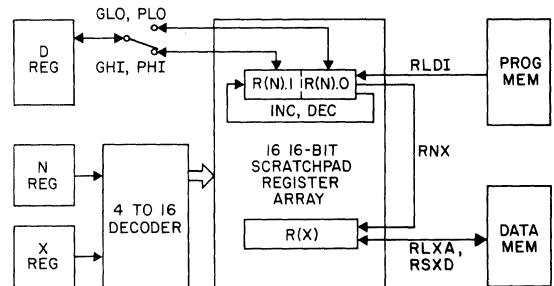
SCAL and SRET Instructions

Of all new CDP1805 instructions, the new call and return instructions, SCAL and SRET, provide the most significant improvement in throughput. These 4-byte instructions perform the same function as the SCRT technique outlined in existing user manuals. However, the savings in code and execution time is significant (Table I), especially where subroutines are frequently used.

Besides providing subroutine call-and return capability in a single instruction, SCAL and SRET support the additional parameter-passing feature. Parameters are passed from main program to subroutine by saving the main program counter in a scratchpad register, rather than on a stack, and using a memory-load instruction to link data following the SCAL instruction to the subroutine.

16-Bit Data Instructions

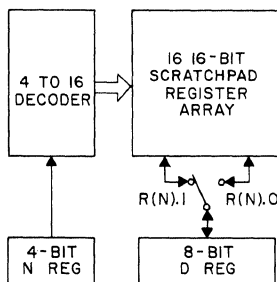
Four instructions, RLDI, RLXA, RSXD, and RNX, were added to the CDP1805 to handle enhanced 16-bit operations involving the scratchpad registers. Figures 11 and 12



92CS-33390

Fig. 12 - Enhanced CDP1805 scratchpad-register data-transfer model. The CDP1805 allows direct register-memory transfers.

illustrate the enhancements made possible through these new instructions. For example, they permit 16-bit loads of data from program memory to a designated scratchpad, 16-bit loads and stores between scratchpads and memory, and transfer of the contents of any scratchpad to R(X).



92CS-33386

Fig. 11 - CDP1802 scratchpad-register data-transfer model. The register array is accessed via the D register.

Conclusion

The architecture and performance improvements afforded by the new CDP1805 microprocessor have been reviewed. For each improvement, an advantage or benefit to the user in a real system application has been pointed out. For customers requiring any or all of these performance improvements, the CDP1805 may prove to be the appropriate choice in a new or modified CMOS system design.

Multimicroprocessor-based Transistor Test Equipment

by W. J. Hepp and R. H. Isham

As the capability of power devices has improved, it has become increasingly important to accurately test not only for the traditional parameters, but for characteristics—such as thermal resistance and forward-biased second breakdown—directly related to the application in which the device is used. In addition, some customers require special-use tests. Switching tests and high-temperature parameter verification are also being required in an increasing number of applications. Traditionally, these testing needs have been satisfied by multiplexed computer-controlled test sets that measure standard parameters, and a multiplicity of special-purpose test sets that measure those characteristics that must be controlled in specific applications or that cannot be measured effectively in conventional test sets. The need to reduce as much as possible the number of insertions, that is, the number of different types of test sets that a device must pass through, has also increased in importance. The passage of a device through a multitude of test sets increases its cost by adding handling costs, and increases the possibility that it will be mis-segregated at least once. These factors are particularly important in the testing of automotive ignition transistors, where very high quality levels are demanded, and where pricing is very competitive. The Note discusses a modern test system that meets these demands through its ability to perform multiple-temperature testing of traditional and custom parameters in one insertion.

History

Computer-controlled test equipment was first introduced to the RCA Solid State Division (SSD) power-manufacturing facility in the form of SCOPE test systems.¹ More than 35 of these hardware-multiplexed minicomputer systems were built between 1969 and 1976, and 25 are still being used in various SSD locations throughout the world for routine device testing.

The marketing of high-voltage Darlington transistors for use in automotive ignition systems required testing to quality levels unprecedented for commercial devices. Some customers also insisted that tests be performed to simulate distributor misfiring, reversed battery polarity, battery disconnect transients, and fouled spark-plug operation. One-hundred-percent testing of both switching time at room temperature and dc parameters at several high temperatures was also required. Traditional testing methods would have involved at least seven insertions, making the achievement of the required quality levels difficult.

A method of internally heating a power transistor by dissipating energy had been demonstrated through the use of an engineering version of a SCOPE test system. The method worked well, but required more test time on a multiplexed, multisocket SCOPE system than the maximum 200 to 300 milliseconds that was economically feasible. It was recognized that a nonmultiplexed SCOPE system

would not suffer the same restriction since all of the hardware would be dedicated to a single socket. In addition, removal of the multiplex circuitry, with its long leads and high stray capacitance, would allow a primitive switching-time circuit and the special customer tests to be added.

A prototype of a nonmultiplexed system, the first GALT test station,² was developed and placed on line in 1975. It used the basic SCOPE concepts, implemented in state-of-the-art hardware, and was optimized for single-insertion multiple-temperature testing of automotive Darlington transistors. It was connected to the minicomputer in an existing SCOPE test system and operated by "stealing" CPU time. Calculations had shown that more than 20 GALT stations could be added to the system with no degradation in the performance of the SCOPE system. This prototype station is still in use in SSD's plant in Malaysia.

Introduction of the SwitchMax high-speed transistor families in 1977 brought a need for accurate testing of switching time at high temperatures. Using experience gained with the prototype GALT system, a new station optimized for high-speed switching time was designed. In this station, the minicomputer was replaced with a COSMAC development system as the station controller. This replacement was possible because the minicomputer in a typical SCOPE station is idle about 99 percent of the time during testing, and has a machine-cycle time only ten-times faster than a COSMAC CPU. To keep the cost down, the only peripheral included with the GALT station was an RCA Microterminal, which was to be used primarily for maintenance purposes. A central computer system was used to load test programs and to log results over a serial data link.

A second COSMAC development system with disk-operating software and four test-station interfaces was originally used for the central computer. More recent GALT sets have tied to existing Hewlett-Packard minicomputer-based systems, and have employed RCA Microboard prototyping systems as the station controllers.

A total of three of these high-temperature-switching-time test stations (Fig. 1) have been built to date. One is in operation in SSD's Mountaintop location and the other two are operating in Malaysia. In addition to testing switching time, the stations are configured to test forward-biased second-breakdown and several conventional parameters.

System Description

Unlike most other commercial test-system manufacturers, RCA has generally grouped test hardware functions on boards; one board contains all of the drivers needed for gain measurements, another those needed for leakage measurements, and so on. This practice allows customizing of the response of each driver to suit the task, and allows modification or addition of one class of test without affecting others.



Fig. 1 - GALT V switching-time console (with door removed).

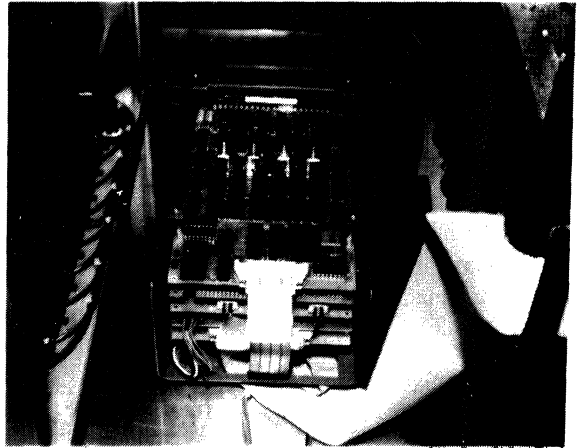


Fig. 2 - Microboard station controller (with cover removed).

Until recently, there had been little control at the board level. All information needed to run a test was placed in parallel on the test backplane, timing signals were generated, and results read under control of the central computer. More complex tests, such as switching time, required more information than could be placed on a test backplane or transmitted over a parallel data link. This predicament suggested a board-level controller that could accept data in serial/parallel fashion, run the test, and return results. A system of this type provides a clean break between the logic that performs the test and the logic that decides which test to perform. Thus, the testing hardware developed can be used in different systems with minimum trouble, and can be easily expanded.

The heart of the test station developed to satisfy the needs just described is an RCA Microboard computer system (Figs. 2, 3, and 4). Fig. 2 is a photograph of the Microboard station controller itself. Fig. 3 depicts the system Microboard complement, and Fig. 4, the block diagram of the switching-time console. As shown in Fig. 2, the system employs four standard cards: the CDP18S601 CPU, the CDP18S641 UART, the CDP18S640 control and display board, and the CDP18S622 battery-backed 8-kilobyte RAM card. A custom parallel I/O card completes the system.

Test programs are loaded from a central station at 2400 baud through the UART card. No further external communication is needed unless data taking is requested. The

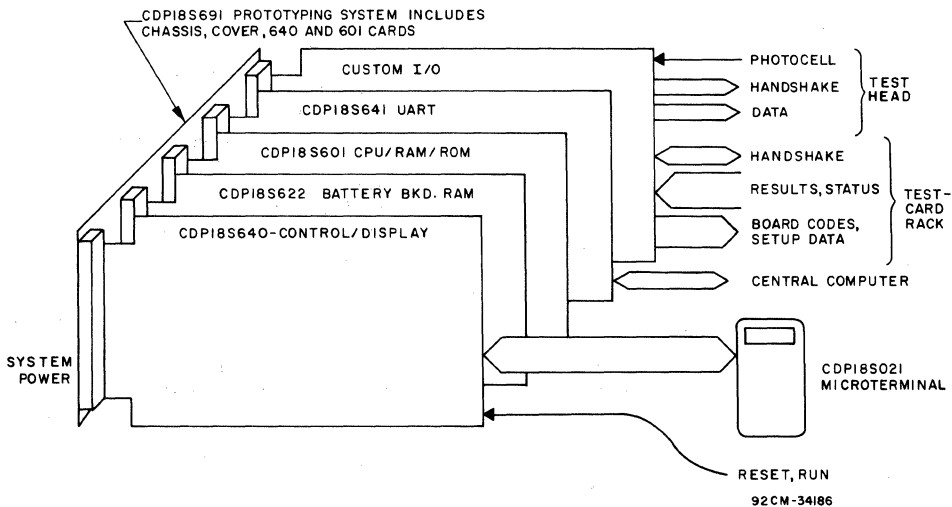
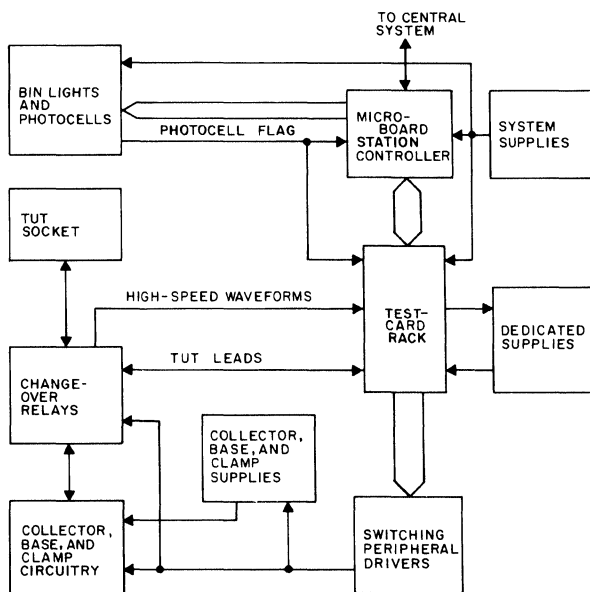


Fig. 3 - The Microboard system.



92CM-34187

Fig. 4 - Block diagram of the switching-time console.

battery-backed RAM provides protection from power outages; the control and display card provides the interface to an RCA Microterminal (a pocket-sized keyboard and display unit) used for servicing. The custom I/O card provides communications with the set of test boards and with the test head.

The test head houses pass/fail LED's as well as hexadecimal readouts for displaying bin selection. As in RCA SCOPE systems, light and photocell pairs are grouped around the test socket to detect the operator's hands. The readouts are latched internally and driven by address, data, and timing signals from the custom I/O card. The "hand-clear" signal from the test-head photocells is critical; the operator's safety depends on it. It is used to signal the system to start a test sequence and to provide a hardware reset to all test boards, overriding any software. The signal is also generated during power up to clear all boards.

Each class of tests is accommodated on a 11 by 17-inch board with two 43-pin edge connectors. One connector handles large signals: transistor-under-test leads, dedicated power-supply connections, and high-power system supplies. The second connector handles all low-power system supplies, data, and handshaking lines. Fig. 5 shows a typical board with its two connectors, and Fig. 6 shows a typical test-board assembly or "cage."

Each test board contains a small CDP1802-based system (Fig. 7) that controls the hardware on the board and communicates with the test-station controller. The system includes 1 kilobyte of ROM, 32 bytes of RAM a CDP1851 programmable input/output port, and some handshaking and address-decoding logic. A self-calibration feature added to some boards requires the addition of a CDP1855 multiply-divide chip. Because of the limited I/O decoding capability of the CDP1802, eight memory addresses are decoded for the on-board hardware interface.

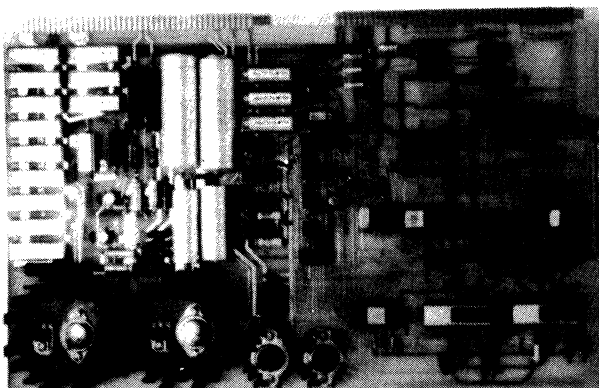


Fig. 5 - Forward-drop board.

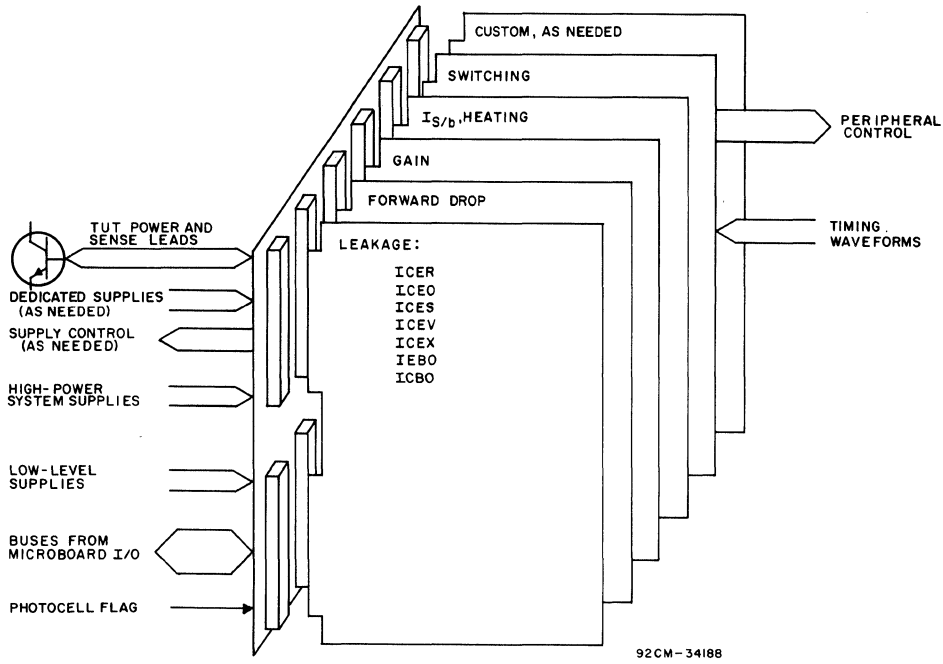


Fig. 6 - Typical test board assembly or "cage."

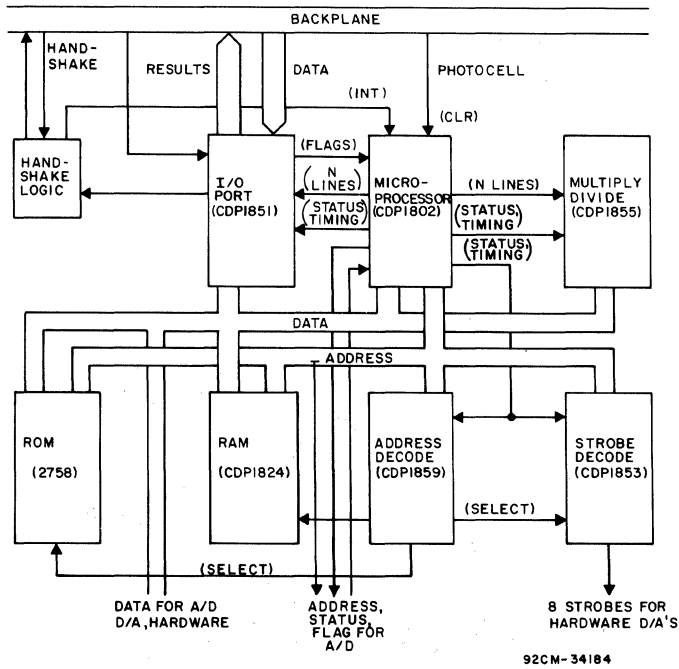


Fig. 7 - Board-level processor system.

The CDP1802 was chosen as the main component of the board-level processor system for several reasons. Designer familiarity and its low cost were important factors, but CMOS noise immunity was the deciding one. There has, indeed, been very little trouble with "glitches", despite the electrically noisy environment of the test stations.

Two data buses connect the test boards to the system controller: one to pass "board codes" or test set-up data to the boards, and one to pass back results and status. The "hand-clear" signal resets each board at the start of a test sequence. All output ports to the backplane are three-stated; the boards initialize (and self-calibrate) and wait in an idle loop. The station controller places the address of the desired test (board code) on the backplane and generates a signal to interrupt all boards. All boards compare the "board code", and the addressed board responds with a flag. Sufficient data to run the test is then passed over the bus using a simple handshaking scheme. The board compiles the set-up data, runs the test, and signals the system controller to retrieve the results. Again, using a simple handshaking scheme, two bytes of results and status are passed back. The board then idles, waiting for another interrupt.

The leakage board (Fig. 8) typifies recent hardware in that it uses no off-board components and is self-calibrating. The microprocessor front-end interfaces to the hardware through relay drivers, a data latch, and A/D and D/A converters. The D/A converter is time multiplexed to set both base-bias conditions and collector or emitter voltage.

When the board has received all data needed to specify a test, the relay-driver bits are determined and outputted. During the relay-settling time, the base and then the collector D/A-converter values are calculated and loaded (the D/A converter has two internal registers). The base bias is isolated from the D/A converter, the collector voltage is

"rippled through", and the collector-supply inverter is activated. After a further settling time, which depends on current measuring range, the A/D converter is strobed. Supplies are shut off in reverse order, and the results are returned.

As an example of a noise source present in the test set, the leakage supply is a 125-kHz push-pull inverter with a 1-kilovolt output. The inverter is located within several inches of the microprocessor "front end."

Switching-Time Measurements

The measurement of power-transistor switching parameters over a wide range of conditions requires more than one board of hardware; Fig. 9 shows the set-up used. The transistor under test must be disconnected from the normal test-board backplane and connected in a low-inductance loop with a collector load and a current-sensing transformer. A second low-inductance loop is composed of the base-drive circuitry and a second current transformer. A third low-inductance loop contains high-speed switching diodes and bypass capacitors that clamp collector voltage when inductive loads are switched.

Base- and collector-current and collector-voltage waveforms (Fig. 10) are fed back to the switching-time board. These signals then pass through programmable gain amplifiers and comparators into the measurement circuitry. Measurement is made by time-interval averaging,³ which produces a resolution of five nanoseconds. All comparator levels are set under program control, an arrangement that allows the measurement points to vary.

The processor system is somewhat larger than on other test boards, with more RAM and 32 decoded output lines. A separate rack is provided for all supply drivers, relay drivers, the base-drive timing driver, and so on.

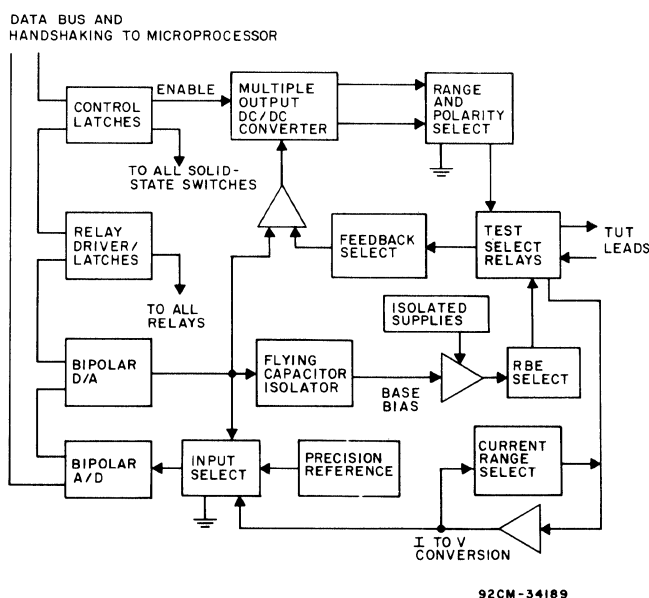


Fig. 8 - Leakage-board output section.

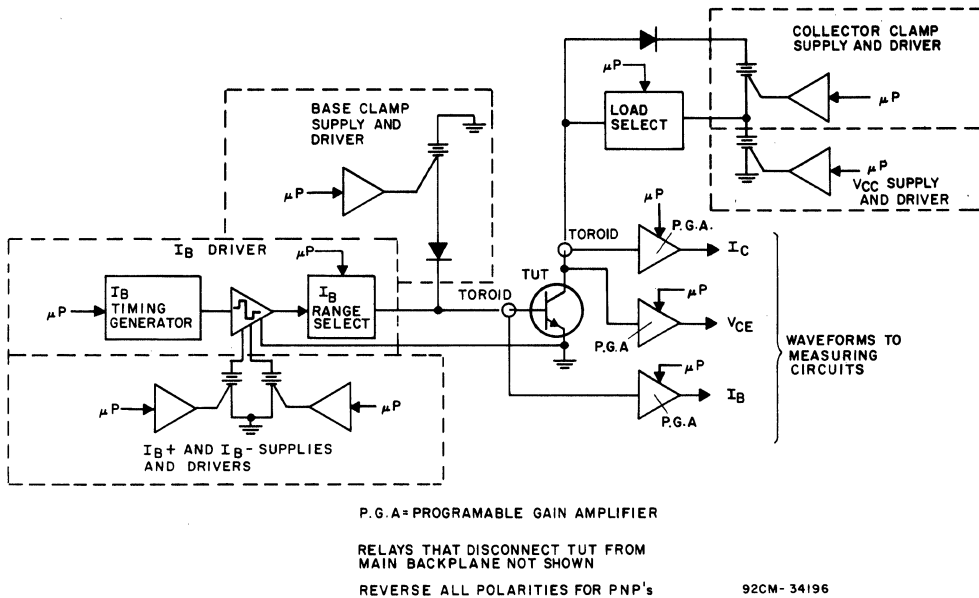


Fig. 9 - Switching-time measurement system.

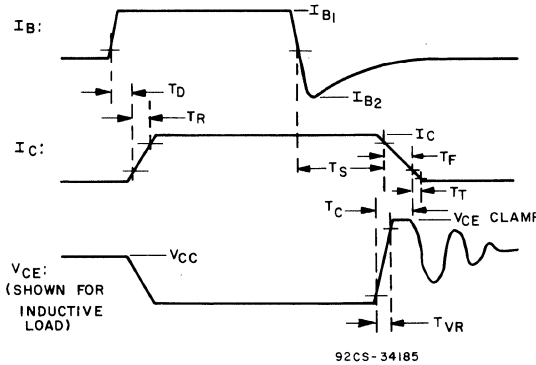


Fig. 10 - Base- and collector-current and collector-voltage waveforms supplied to the switching-time board during switching-time measurements.

The most difficult hardware functions to implement in the multimicroprocessor test set included the base-drive circuitry and the programmable amplifiers. The base drive must output a reversing current pulse of 5- to 50-microseconds duration and of 1-milliampere to 6-amperes magnitude. Rise and fall times of less than 20 nanoseconds are produced at low currents, with socket and wiring inductance limiting the highest current pulses to 50 nanoseconds. The drive is entirely bipolar; both n-p-n and p-n-p transistors can be accommodated. The programmable amplifiers must operate over a gain range of 1 to 2000, with 1, 2, 5 sequence. Rise and settling times total 25 nanoseconds, accuracy is one percent, and inverting or noninverting gain is provided.

The main collector and collector-clamp supplies were also somewhat difficult to realize. They must sink or source

currents as high as one-half ampere at voltages up to 450 volts, also under program control, and can be stacked to a maximum of 900-volts clamp voltage.

The appearance of the switching-time circuitry to the test-console controller is similar to any other test board, but approximately 20 bytes are required to specify the test. Nine different parameters can be measured in one test, but as described earlier, only one result of up to 16 bits is returned per test. To avoid unnecessary test repetition, the board first compares all set-up conditions to those of the previous test. If the conditions compare exactly (except for the result requested) and no other board has been called since the last test, the board will go directly to returning results.

If a new test is required, the first function of the system is to decode and energize the required relays. The starting

values for all supplies are then calculated, and after a waiting period for relay settling, the supplies are programmed up. Reference values for all D/A converters are outputted along with other set-up conditions, such as base-drive pulse width. After the waiting period for supply ramp-up, the base drive is activated at a 2-millisecond repetition rate. At this time a supply-adjustment procedure begins.

Switching parameters can be very sensitive to small changes in set-up conditions; therefore, these conditions must be maintained as accurately as possible. Some parameters, such as reverse base current during turn-off, maintain their peak value for less than 50 nanoseconds. These constraints make conventional regulation techniques difficult, and it is for this reason that the processor has been made part of the feedback loop. Comparators are set with the desired peak values of collector current, forward- and reverse-base current, and collector-clamp voltage. During a switching cycle, latches are set if any value exceeds that desired. The processor examines the latches after each cycle, and increases or decreases each supply as needed. This adjustment is made after every cycle for 100 cycles, and then reduced to one adjustment on every fourth cycle. The measurement circuitry is then enabled, and the switching, adjusting, and measuring continue for another 100 cycles. Supplies are then disabled, results calculated, possible errors checked, and the result phase entered.

Operating-System Software

The operating-system software in any test system should provide the means for testing devices in the minimum amount of time necessary to bin (or sort) a device accurately. When multiple temperatures are used, the problem is complicated, since it is time consuming to cool a device once it has been heated. For this reason, all tests that might be needed at a given temperature must be performed before the unit under test is heated to the next temperature. Those tests not needed must be excluded to save time.

The Microboard computer contained in the GALT stations has been programmed to perform testing along an optimum path. Units are tested so that the device will fall into the highest priority bin for which it is qualified, with no unnecessary testing. The test program contains a list of test conditions to be used by the hardware to perform the tests. There is one entry for each parameter at each set of conditions. The program also contains lists of tests required for each bin. Each entry contains a pointer to the proper entry in the test condition list and the limits to which the results of the test are to be compared. Limit comparison is done in software, since each test returns an actual result. This is a departure from almost all commercial test systems, which make only analog hardware comparisons. A unit is assumed to be eligible for all bins before testing begins. An eligibility flag for each bin is maintained in memory during testing, and is used by the steering logic to determine which test to perform next.

Testing begins when a device is plugged into the test head and the "hand-clear" flag is set by the photocell hardware. The first test performed is a continuity test to assure that the unit is making proper contact in the socket. If this test fails, the continuity-fail code is displayed on the bin lights, and testing is complete. If this test passes, all of the eligibility flags are set to "eligible", and all of the test-result locations are set to indicate that no tests have been performed. The pointer for the first test of the highest-priority bin (the one which must be filled first) is then retrieved from the required test list, and the test is performed. The results of the test are stored in the memory location reserved for that

test, overwriting the test-not-performed flag. The results are then compared to the specifications for the highest-priority bin. If the test has failed, the eligibility flag for that bin is set to "reject," and no further tests are performed for that bin. If the test passes, the eligibility flag remains unchanged, and the pointer for the next test in that bin is retrieved.

Testing continues, as long as a fail does not occur, until the last room-temperature test for the highest-priority bin is performed. At this point, the program looks ahead to see if any tests are required at higher temperatures. If they are, the eligibility code remains unchanged and testing continues with the next bin. If no tests are required at higher temperature, the eligibility code is set to "qualified."

Before the room temperature tests for the next bin are started, the list of tests for that bin is scanned to see if any of the tests have already been performed. The previously performed tests are compared to the required limits, and a pass/fail decision is made for each. If a failure occurs, the eligibility code is set to "reject" (as above) for that bin, and no further tests are performed for that bin. If no failures are found, and there are remaining room-temperature tests, they are performed. This look-ahead feature prevents further testing of a device that has already failed an identical test, and prevents repeating tests already performed to the same set of conditions.

Room-temperature testing proceeds for the remaining bins or until a bin is found for which the device is fully qualified. The program then checks to see if the device is eligible for higher-temperature testing for a bin of a higher priority. If it is not, the number of the bin for which the device qualifies is displayed and testing is complete. If the device is eligible for higher-priority bin testing at a higher temperature, the proper amount of energy is dissipated in the device to heat it to the next temperature. The test-result flags are reset to show no tests done, and testing continues. The process is repeated for each temperature. If, at any point, all eligibility codes are set to "fail", the reject code is displayed on the bin lights and testing is complete.

The method of heating devices through the use of internal heating is a time-consuming one. Therefore, there is a tendency on the part of the test programmer to heat the device as fast as possible by setting the dissipation as close to the capability of the device as he can. When this is done, however, some of the tested devices may experience forward-biased second breakdown. If this condition occurs, the device does not reach the proper temperature and further testing is inaccurate. The control program handles this problem by displaying a failed heating code on the bin lights and by considering testing complete. Since this condition is not a true failure, the device can be placed in a bin for retesting with lower-energy heating pulses.

Some of the tests used to classify power devices require the dissipation of energy sufficient to damage a device and make the previous test results invalid. A feature has been included in the subject test-system control program to code such tests. If at any time the device fails one of these potentially destructive tests, a special code is displayed on the bin lights, and testing is complete.

Some of the microprocessor-controlled test boards have provisions for self-diagnostic checks. When one of these checks fails, a code is included in the test results sent to the Microboard computer. When the control program senses this code, it immediately stops the testing process and displays a flashing code on the bin lights to indicate a board failure. The display includes both the board number and

ICAN-7020

type of failure. The test station is locked up at this time. After the board failure is noted, the station can be reset to allow testing of the next device, although board failure will be detected if it occurs again.

When one test board is switched off and a different one called, there is the possibility that a relay may fail to open. Since this situation is potentially destructive, the software calls in a "hung-relay" check each time the boards are switched. If a hung relay is found, the system locks-up in a manner similar to that for a failed board.

Utility Software

The control software contains two methods of testing in addition to the optimum path described above. A user can define a data-test program that contains a list of tests in a specified sequence. This method is used in conjunction with a data-analysis program in the central computer to record actual test results. In this mode, the testing begins when the serial number of the device to be tested is sent from the central computer to the test station. The serial number is displayed on the bin lights and the test station is enabled. When a "hand-clear" flag is received, the tests are performed as specified. The results are sent to the central computer, which processes them and, when ready, sends back another serial number. The process continues until the file of serial numbers is depleted and the central computer sends a test-complete code in place of a serial number. This signal causes the test-complete code to be displayed on the bin lights, and testing is complete.

The second method of testing is a very simple test mode designed for maintenance purposes. This mode employs the station's Microterminal for I/O. The necessary control words for a single test are loaded into a scratchpad area of memory. When operating in this mode, the control program repeats the test defined in the scratchpad continuously and displays the results on the Microterminal. Repetition times can vary from 12 milliseconds to 30 seconds. This mode can be used to perform a test once only by setting the repetition factor to zero. Utility software that checks the operation of the bin lights, photocells, and communications link has been included.

The software also contains routines for the loading of test programs from the central computer. While communication is proceeding, the photocell flag is ignored. This arrangement prevents an operator from testing while a program is being changed.

Maintainability

The most useful tool in facilitating repair of the subject transistor test system has been the RCA Microterminal used with the utility software described above. Because of the intelligence available at the board level, the words needed to run a test are simple to compile by hand. By manually loading a single test and using the appropriate control software, a board can easily be observed in operation.

Because it is known that the setting up of A/D and D/A converters is time consuming (and often ignored when one of these functions is being replaced), a way was sought to bypass these operations without compromising testing accuracy. The method decided on involves the placement of an accurate voltage reference on each board along with facilities to connect the A/D converter to the reference and to the D/A converters. Immediately after the "hand-clear" flag is set by the photocell hardware, each board reads ground and the reference voltage through its A/D converter. Gain and offset factors are calculated and stored for use in correcting the A/D-converter readings. Each D/A converter is then driven to ground and to a value near the

end of the scale. Using the corrected A/D converter, gain and offset factors for the D/A converters can be calculated. During a board's operation, the factors are recalled and used to correct the D/A-converter outputs. Calculations are done during the three-millisecond relay-settling time, so that no test time is added. The result is corrected at the end of the test, which adds several hundred microseconds to the test time.

The test-board circuitry is designed to be minimally affected by offsets or other op-amp parameters. This fact, together with the self-calibration feature, allows the boards to meet one-percent accuracy specifications with no set-up at all, thus reducing labor content considerably.

Some limited self-diagnosis is starting to be included on test boards, with status bits being returned along with test results. A simple diagnostic test, such as for forward-biased second breakdown (IS/b) may report that a supply had run out-of-compliance, thus negating the result. A complex board, such as the switching-time board, may report such problems as counter overflow, supplies adjusted out of expected range, or improper base-drive response. It must be stressed that several problems may have identical symptoms; the diagnostics can only suggest the general location of the trouble.

Summary

The placement of several levels of intelligence in a test console has provided many new test-system features. The extreme modularity of the set, in both hardware and software aspects, allows a very flexible system design. The addition of a capability to an existing set is as simple as plugging-in a board. The system software is independent of the boards used.

The optimum-path software provides a powerful and time-saving determination of test sequence. The multiple-temperature, multiple-binning capability in one insertion provides a cost-effective way of meeting ever more stringent custom-testing requirements.

Utility software, Microterminal access, self-calibration, and self-diagnosis speed the understanding and solution of problems. Self-calibration and self-diagnosis decrease the possibility of mistesting the product. The construction and check-out time of boards is also greatly reduced, and the need for minor adjustments ("tweaking") has been eliminated.

The multimicroprocessor-based transistor test system is a flexible system that is adaptable to almost any custom-test requirement. As a bonus, the cost to fabricate the system internally can be as little as one-fifth of the purchase price of commercially available test systems.

Acknowledgments

The authors thank Don Burke for his support during the development of the GALT stations, Wally Williams for clearing administrative requirements, Clyde Van Horn for his contributions to the control and communications software, and Jim Hoshowsky for constructing the hardware from prints that no one else could read.

References

1. Cutshaw, E., "SCOPE—The Power Testing Paladin," RCA Engineer, Vol. 21, No. 5 (Feb./Mar. 1976).
2. Rand, Ayn, "Atlas Shrugged," Random House, New York (1957). The term, GALT, was taken from this book.
3. "Time Interval Averaging," Hewlett Packard, Application Note 162-1.

CDP1800-Series Peripherals — Building Blocks of a Complete Processor Family

by J. Paradise

Advances in CMOS technology in recent years have demonstrated that the system designer no longer needs to compromise his design performance goals when he chooses CMOS in his microprocessor-based system. Speed and integration levels have evolved to the point where CMOS functions have begun to closely match their NMOS counterparts. Microprocessors and memories are usually the functions that come to mind when making these comparisons. With the introduction in the last few years of a host of CMOS peripheral functions from RCA, the CDP1800 series now offers a complete array of auxiliary chips that make available to the system designer the functions, flexibility, and performance levels that once were only achievable with NMOS.

CDP1800 SERIES I/O STRUCTURE

The processor functions in the CDP1800 series feature a register-based, control-oriented architecture and instruction set geared to mid-range performance capability. The well-established CDP1802, the venerable patriarch of CMOS processors, has found its way into numerous system designs that take advantage of two of its dominant features—low power for portability, and I/O handling for control applications. The obvious classical CMOS advantages aside, CDP1800 architecture is well-suited to interface to complex external I/O devices, with performance and flexibility beyond what can be practically achieved in a single-chip microprocessor with today's technology.

The diagram in Fig. 1 illustrates the I/O interface pins that

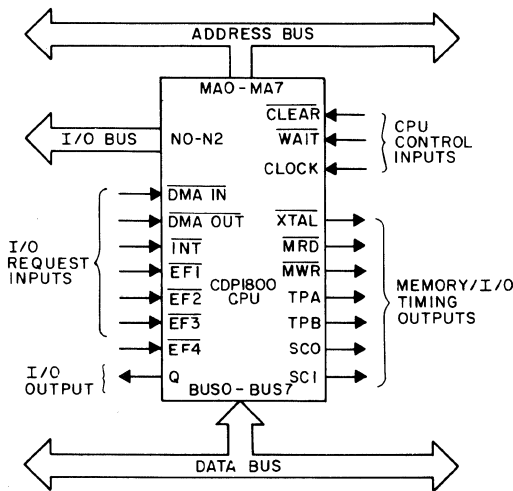


Fig. 1 - Common CDP1800-series CPU interface pins.

appear on all processors in the CDP1800-series product line. Featured are a separate 3-bit I/O address bus, on-board DMA control, interrupt and flag input lines, and a single, software-controlled, Q output bit. Peripheral devices can be mapped in I/O space using the I/O bus, or memory mapped anywhere within the 64K address space of the CPU. The interrupt and flag lines allow a peripheral to request service, while the DMA feature allows for high-speed memory <—> peripheral transfers in systems where the peripheral has access to the system bus. The flag and Q lines can form a serial I/O interface, or, in the case of the enhanced CDP1804 and CDP1805, can also provide access to the internal 8-bit counter-timer.

When used in I/O space, peripheral data transfer is accomplished by means of INPUT and OUTPUT instructions. These instructions transfer data between a memory stack and the peripheral, bypassing the CPU in the process (the INPUT instruction loads data into both memory and the CPU accumulator). Because of its register-based architecture, a CDP1800-series CPU can maintain several memory stacks within its 64K address space, dedicating storage areas for I/O transfer at reasonably high speeds.

The INPUT and OUTPUT instructions set bits on the I/O address bus for selection of peripheral functions, with the selection code implicit within the instruction opcode. The 3-bit I/O bus thus allows seven opcode combinations each for INPUT and OUTPUT instructions, plus an all-zero code to distinguish a memory transfer. The I/O bus may be used directly to select up to three devices, or it can be decoded, as shown in Fig. 2, to provide seven unique select signals for

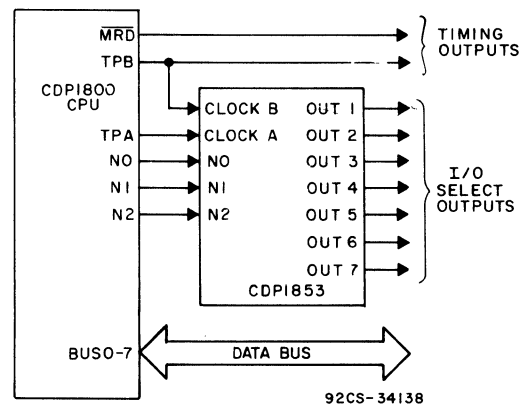


Fig. 2 - CDP1800 expanded I/O interface. I/O select outputs are active only during I/O instruction execute cycle.

ICAN-7023

either an INPUT or OUTPUT transfer. An alternate scheme shown in Fig. 3 allows latched single-bit control of I/O devices, with bits set or reset only when an instruction is issued.

For I/O intensive systems, a two-level I/O approach can be used, as shown in Fig. 4. In this configuration, two I/O instructions are issued, the first as a group selector and the second to select an I/O device within the group. With this minimal hardware addition, I/O capability is expanded to 48 input or 48 output combinations. With additional circuitry, this concept may be expanded to provide more than 12,000 unique codes.

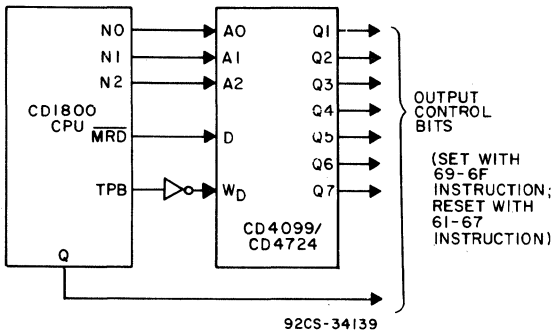


Fig. 3 - CDP1800-series 8-bit I/O output control. Each bit is individually set or reset by means of software instructions.

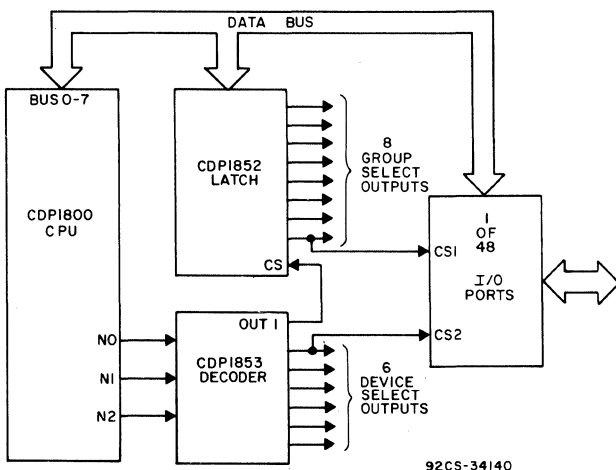


Fig. 4 - Generalized two-level I/O technique.

PERIPHERAL FUNCTIONS AVAILABLE

Table I lists all of the currently available and soon-to-be announced I/O functions in the CDP1800 series. These functions range from simple buffers, decoders, and latches to complex special and general-purpose devices.

Table I — Twenty-Seven I/O Functions that Support the CDP1800-Series Product Line

I/O Ports	Buffers
CDP1851*	CDP1856
CDP1852	CDP1857
CDP1872	
CDP1874	Video Control
CDP1875	CDP1861
	CDP1862
Memory/I/O Decoders	CDP1864
CDP1853	CDP1869
CDP1858	CDP1870
CDP1859	CDP1876
CDP1866	
CDP1867	Keyboard Interface
CDP1868	CDP1871
CDP1873	
UART	Timer Functions
CDP1854A*	CDP1863
	CDP1878*
Multiply/Divide	CDP1879*
CDP1855*	Interrupt Control
	CDP1877

*Discussed in detail in this note.

This discussion is focused on the more complex general-purpose components (marked with an asterisk in the table), which contain a number of addressable registers and which can operate in a variety of programmable modes. These types of components and their functions more closely match equivalent NMOS offerings found in recent catalogs, and all have the feature of being able to interface with any general-purpose bus-oriented microprocessor or expandable microcomputer. (The diagram in Fig. 5 shows a typical interface between CMOS multiplexed address/data bus processors and complex CDP1800-series I/O devices). All of these featured devices will operate in a 5-MHz CDP1800 system, with access times on the order of 500 nanoseconds (at 5 volts). Operation is over a voltage range of 4 to 10.5 volts, and over a temperature spread of -40 to +85°C.

The remainder of this note deals individually with these components. In addition to functional descriptions, the interface requirements, typical system configurations, and viable applications for the devices are discussed.

CDP1851, Programmable I/O Expander

The CDP1851 is a general-purpose programmable I/O device that has 20 I/O lines that may be used in several different modes of operation. Two full 8-bit ports, complete with handshaking lines, are provided for efficient interfacing between a parallel CPU bus and peripheral functions. The CDP1851 is programmed by the CPU, by means of its control register, to define port mode, interrupt enabling, I/O bit assignment, bit masking, etc.

Table II shows a summary of the CDP1851 programming modes with corresponding pin configurations. Simple input or output-port functions with ready and strobe handshaking control lines may be selected, or more complex bidirectional and bit-programmable modes may be utilized.

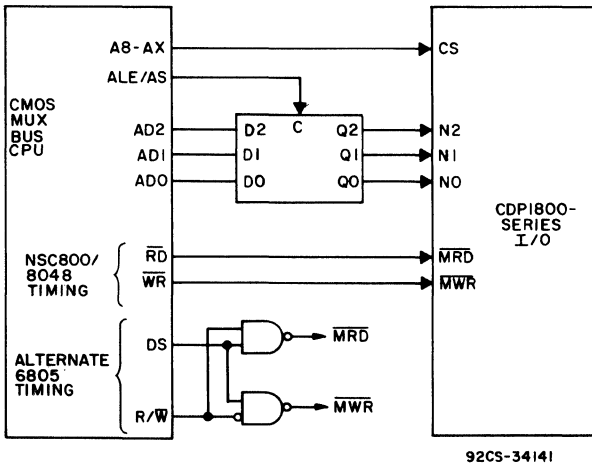


Fig. 5 - General interface requirements; CMOS multiplexed-bus CPU to CDP1800-series I/O.

In the bidirectional mode, the handshake control lines maintain proper bus flow discipline and allow the CDP1852 to interface to a master or slave bidirectional bus. In the bit-programmable mode, individual lines can be designated as input or output lines, including the handshake pins, which are not used for ready and strobe in this mode.

Separate interrupt lines are available for each port, with or-tie capability for single interrupt CPU inputs. Interrupts may be enabled or disabled, and can be read from an on-board status register. In the bit-programmable mode, interrupts are generated by logic conditions (AND, OR, NAND, NOR) programmed on bit inputs. Thus, a bit-programmable port can act as an interrupt expander in the OR mode, or can respond to the coincidence of several input conditions in the AND mode.

The CDP1851 interfaces without additional components to all CDP1800-series processors in either I/O or memory space. In I/O space, the N lines are connected directly to CDP1851 inputs, with different N-code combinations selecting the CDP1851 registers and ports (see Table III). In memory space, an on-board latch is provided to create a chip-select from any selected high-order address line on the CDP1800 multiplex bus.

Fig. 6 illustrates an application that utilizes the complex capabilities of the CDP1851 in a multiprocessor application. In this configuration, one or more CDP1851 devices can interface to a CPU or shared master memory. Port A is used as a bidirectional port, with handshake lines controlling bus transfer. Port B is configured as a bit-programmable port, and is used to accept interrupt requests from the master controller for proper sequencing and placement of data, which is transferred by the CDP1851. Because master bus interfacing is done under interrupt control, individual slave CPU's can perform independent dedicated tasks and ultimately increase total system throughput and performance capability.

CDP1854A, UART

The CDP1854A is a CMOS Universal Asynchronous Receiver/Transmitter (UART) circuit. It provides the necessary formatting and control for interfacing between parallel and serial data paths. It is capable of half or full-duplex operation, has a double-buffered receiver and transmitter section, and features a programmable data word, parity bit, and stop-bit length. The receiver is capable of checking parity and the occurrence of a stop bit with parity, overrun, and framing-error output indication.

The CDP1854A can be programmed to operate in one of two modes through the use of single control pin. In mode 1, it is fully CDP1800 compatible, as shown in Fig. 7, and features programmability options made available through a control register that is accessed with N lines. In Mode 0, the CDP1854A is compatible with industry-type 1602 devices, and utilizes hard-wired external pins for data formatting and control.

The CDP1854A is useful in any serial communication circuit where hardware parallel-to-serial or serial-to-parallel operation is desired. Performance is superior to equivalent bit-banging software techniques, with typical data rates in excess of 250K bits per second possible.

This UART device can provide an interface in loosely coupled systems to other processors, with transmit and receive UART's isolating local buses from each other; the diagram in Fig. 8 shows such an application. In the configuration shown, the CDP1802 acts as an intelligent controller in interfacing a master system to a print buffer, and frees the host CPU for other tasks during the spooling operation. In the figure, the CDP1854A is configured in Mode 0, with handshaking to both the CPU and printer for data transfer. As data is received by the UART into its

Table II - CDP1851 Programming Modes

Mode	(8) Port A Data Pins	(2) Port A, Hand- shaking Pins	(8) Port B Data Pins	(2) Port B, Hand- shaking Pins
Input	Accept input data	Ready, Strobe	Accept input data	Ready, Strobe
Output	Output data	Ready, Strobe	Output data	Ready, Strobe
Bidirectional (Port A only)	Transfer input/ output data	Input hand- shaking for Port A	Must be previously set to bit-programmable mode	Output hand- shaking for Port A
Bit- Programmable	Programmed individually as inputs or outputs	Programmed individually as inputs or outputs	Programmed individually as inputs or outputs	Programmed individually as inputs or outputs

Table III — CDP1851 I/O Space Register Assignment

N	Line Code	Instruction	Action	Register
NO	N1			
1	0	INP 1	READ	STATUS REGISTER
1	0	OUT 1	LOAD	CONTROL REGISTER
0	1	INP 2	READ	PORT A
0	1	OUT 2	LOAD	PORT A
1	1	INP 3	READ	PORT B
1	1	OUT 3	LOAD	PORT B

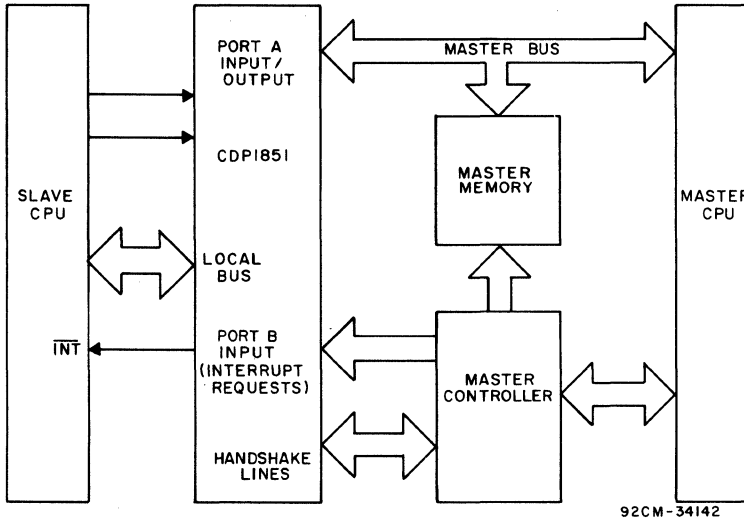


Fig. 6 - CDP1851 master/slave multiprocessor application.

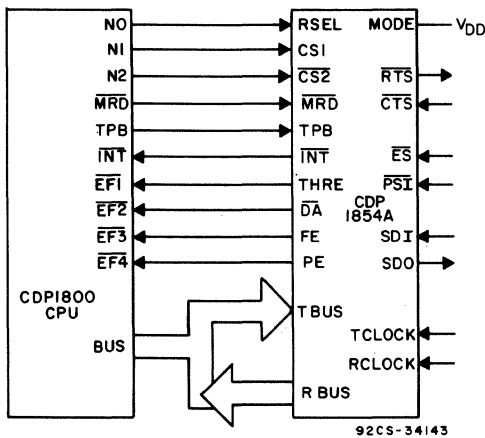


Fig. 7 - CDP1854A mode 1 interface to CDP1800-series CPU.

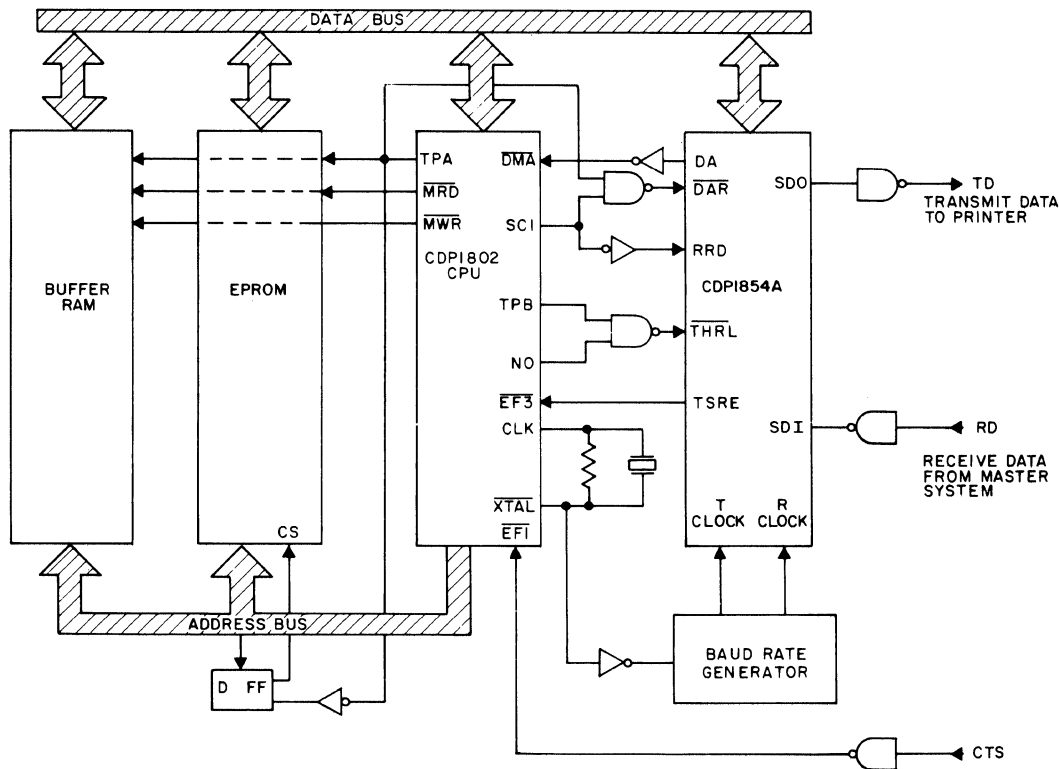
receiver holding register, it flags the CDP1802 for transfer to buffer memory under CPU DMA control. The CDP1802 also polls the printer for data transfer from buffer memory to printer through the transmit section of the same UART. Since the receiver and transmitter functions are independent, the data exchange rates can match those of both the higher-speed host system and the slower printer.

CDP1855, MDU

The CDP1855 is a Multiply-Divide Unit (MDU) that can be an efficient hardware replacement for the software-only implementation of arithmetic and signal-processing algorithms. It performs multiply and divide operations on unsigned 8-bit data with an add-and-shift type hardware implementation, and is structured to permit cascading of identical units to handle operands up to 32 bits.

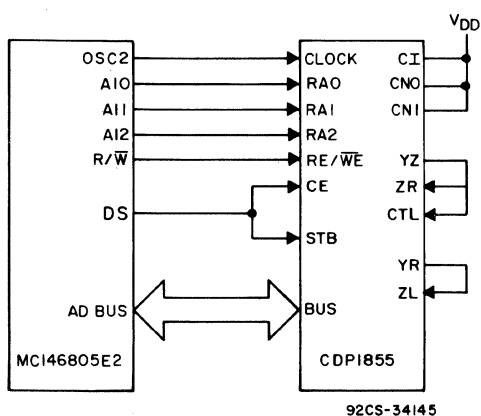
The MDU has three 8-bit registers, X, Y, and Z, which are loaded by the CPU with operands prior to the arithmetic operation, and which contain a product or quotient when the process is complete. The actual hardware operation typically requires only five microseconds for an 8-bit calculation, with additional software overhead time for loading and unloading registers. An 8-bit control register defines and initiates the operation, with a status register available for overflow indication.

The MDU can typically be mapped in I/O space, with eight instructions required to address its X,Y,Z, control and status registers. The device also easily maps into the memory space of other processors, as shown in Fig. 9, where an MDU is interfaced to an MC146805E2.



92CM-34144

Fig. 8 - CDP1802/CDP1854A slave controller application-printer buffer interface.



92CS-34145

Fig. 9 - Minimum 6805/CDP1855 memory-mapped (in 4K) interface.

The CDP1855 can be used in any application that requires fast multiply or divide throughput, or where the CPU would be required to perform real-time tasks during a long arithmetic routine. MDU efficiency increases as word size increases, as shown in the chart of Table IV. In addition, in low-frequency signal-processing applications, the MDU is suitable for use as a recursive digital filter in conjunction with A/D and D/A conversion circuitry.

Table IV - Software vs Hardware Multiply Time Comparisons for the CDP1855 MDU

Word Size	Approximate Time in Machine Cycles	
	Software	Hardware
8 x 8	280	26
16 x 16	1000	41
32 x 32	5000	89

CDP1878, Counter-Timer

The CDP1878 is a dual 16-bit counter-timer with a variety of operational modes. It is a general-purpose device that can produce an assortment of output formats usable in real-time or control-oriented applications. It handshakes with the CPU through an interrupt line that is independent of the timer outputs. This mechanism provides efficient operation with a minimum of software overhead, and without compromising the wave shape required by an external device.

Each section of the CDP1878 consists of a 16-bit programmable down counter with a separate control register, programmable-level gate, true and complement outputs, and a maskable interrupt request that can appear on a shared output pin and in a status register. In operation, the user jams a desired counter value in two 8-bit sequences, and then selects the desired counter mode and initiates timing by writing to the control register. This control register, which programs gate level and interrupt enabling, can also be used to stop the counter at any time, and can assure a stable counter readout by freezing the present count into a separate holding register.

ICAN-7023

The five counter modes of the CDP1878 are shown in Table V along with typical applications for each. These modes, along with different combinations of gating levels, output polarity, and underflow interrupt indication, provide a complete array of timing, pulse-forming, and event-counting programming for efficient use of the device in system designs.

The CDP1878 can be mapped into CDP1800 I/O space, or into the memory space of CDP1800 or other general-purpose processors by means of an external control pin. Fig. 10 shows a typical application for the device, which, in the figure, is mapped in the I/O space of the CDP1805. The circuit provides a means for generating spark advance and dwell timing based on engine speed and a lookup-table of engine constants for various load conditions. Half of the CDP1878 is used to monitor engine speed as a high-resolution, 16-bit count based on time period, while the other half outputs a variable-duty-cycle phase-shifted pulse to the ignition circuitry, with proper formatting for correct dwell and spark-advance settings.

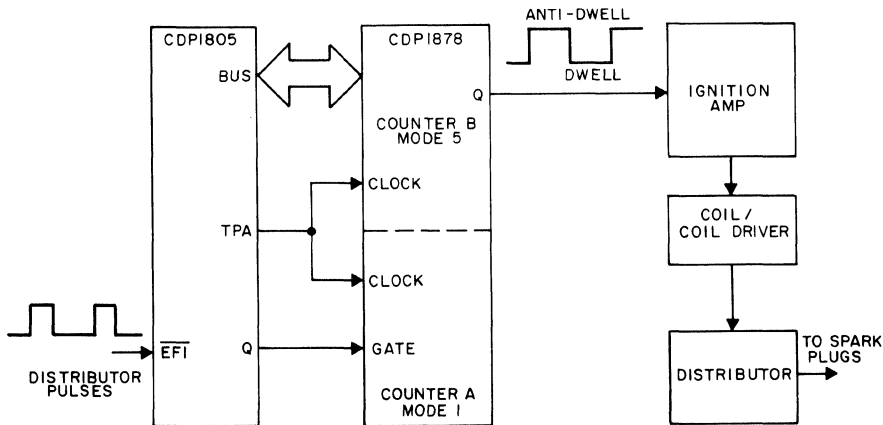
CDP1879, Real-Time Clock

The CDP1879 is a time-of-day clock/calendar chip useful in a variety of real-time applications. The device counts seconds, minutes, hours, date, and month. It operates with

a variety of crystal frequencies, has a separate clock output, and features an interrupt alarm that has a one-second resolution within a 24-hour period.

This real-time clock device can be thought of as a programmable divider chain. One of four crystal frequencies, from 32 kHz to 4 MHz, is selected as the clock source. An on-board control register selects the appropriate prescaling to produce a one-second pulse that is fed to a chain of five programmable counters. The prescaler and divider chain can be tapped to generate 50-percent duty-cycle pulses (subsecond or one per second, minute, hour, or day) at the clock output along with an interrupt request. Counters may be written to or read from in BCD format through individual addresses. Special circuitry allows reading "on-the-fly," even if the counter chain is rippling through a clock pulse at the instant a read attempt is made. Separate second, minute, and hour alarm registers generate an interrupt request when their values match those of the counters. A status register keeps track of alarm and clock status when interrupts are disabled.

The CDP1879, like the CDP1878, interfaces in I/O space to all CDP1800 CPU's as well as in memory space to CDP1800 or other general-purpose processors, through a single control pin.



92CM-34137

Fig. 10 - CDP1878 engine-control application.

Table V — CDP1878 Counter-Timer — Modes of Operation

Name	Function	Application
1 Timeout	Outputs change when clock decrements counter to zero.	Event counter
2 Timeout strobe	One clock-wide output pulse when clock decrements counter to zero.	Trigger pulse
3 Gate controlled one-shot	Outputs change when clock decrements counter to zero. Retriggerable	Time-delay generation
4 Rate generator	Repetitive clock-wide output pulse	Time-base generator
5 Variable duty cycle	Repetitive output with programmed duty cycle	Motor control

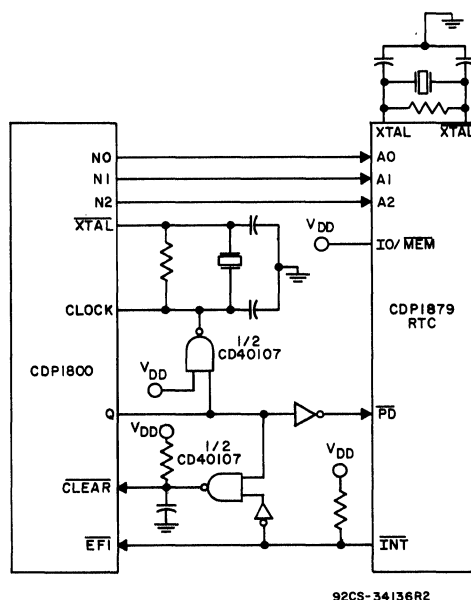
A powerful application for this real-time clock is as a wake-up control to a CPU to reduce total system power in intermittent-use systems. A hookup diagram illustrating this feature is shown in Fig. 11. In this configuration, the alarm and power-down features of the CDP1879 are utilized in the control of the sleep and wake-up states of the CPU. A typical shut-down/start-up sequence for this system could proceed as follows:

1. The CPU has finished a current task and will be inactive for the next six hours.
2. The CPU loads the CDP1879 alarm registers with the desired wake-up time.
3. The CDP1800 Q output is set high, which stops the CPU oscillator (as an alternative, in an NMOS system, power to all components except the clock chip could be shut off).
4. This Q output signal is received by the CDP1879 as a power-down signal.
5. The CDP1879 tri-states all pins (to accommodate powered-down chips).
6. The CDP1879 eventually times out, and sets an alarm by driving the INT output low.
7. The alarm signal resets the CPU (to avoid oscillator start-up problems) and flags the processor for a warm-start routine.
8. The CPU, once into its normal software sequence, writes to the CDP1879 control register to reset the interrupt request.

Because of the versatility of the CDP1879, it is not restricted to use with CMOS processors. Any processor capable of writing to and reading from the clock chip can utilize its low-power capability.

SUMMARY

That there are many versatile I/O devices in the CDP1800-series family, some of which are unique in function in the CMOS world, has been demonstrated in this note. The components featured are general enough in function and interface requirements to work with a wide variety of bus-



92CS-34136R2

Fig. 11 - CPU wake-up circuit using the CDP1879 real-time clock.

oriented microprocessors of different technologies. The components described can expand the I/O capability of any processor design, and bring to the market a variety of powerful control and interface features that were formerly not possible in a low-power design.

Microboard Equipment Control

by W. Schilp, Jr.

The purpose of the project described was to build a piece of equipment to demonstrate the use of RCA Microboards in specialized manufacturing-test equipment. The specific example selected was the testing of some active parameters of power transistors. The equipment was required to give a pass/fail response to preprogrammed limits for one type of transistor and to optionally enter a diagnostic mode and record statistics. The system was designed to be user interactive and to be flexible and expandable, so that additional tests, limits, or transistor types could be handled. All information is displayed by the system on a color monitor; the program comes-up running in the pass/fail mode. Since portability is a requirement, a 10-milliampere limit has been placed on current testing to keep the power requirement low.

System Configuration

In the typical factory power-transistor test set there are four major blocks: central computer, station controller, board-level controller, and transistor under test. In the demonstration system, the central computer was replaced by a video terminal and keyboard because only one transistor type was to be tested and there was no need for mass storage of test programs. Fig. 1 is a block diagram of the demonstration system. It shows a multiprocessor design with the processor in the system controller handling the keyboard inputs, video output, pass/fail criteria and diagnostics, while the test-subsystem processor controls the test conditions for the transistor under test and the reading of the test results. The system is designed so that additional test subsystems for tests such as leakage current, breakdown voltage, switching parameters, and energy capability can easily be added.

Why Use Microboards?

Only a few units of a piece of equipment like the subject tester are usually built, and those, in many cases, by people familiar with the product to be tested rather than microprocessor circuit design. One way of assuring a more successful product under these conditions is the use of standard board-level products, in this case the RCA Microboards.

Some of the more common reasons given for using Microboards include:

- they can be used by those who lack microprocessor circuit-design expertise;
- they give fast turnaround;
- low volume cannot justify in-house development;
- they require low initial investment;
- they overcome manpower limitations;
- there is no manufacturing capacity; and
- they minimize risk.

The advantages provided by the RCA Microboards are:

- the high noise immunity standard with all CMOS devices;
- low-power operation and, consequently, low-cost power-supply requirements;
- the compactness of the 4.5-by-7.5-inch boards;
- easysystem modification through the use of the universal backplane (described below); and
- good support from a full line of compatible hardware and software products.

The Universal Backplane

Since the requirements of the test system include flexibility and expandability, the board interconnections must exhibit the same properties. The RCA Universal Backplane is of an industry-standard, 44-pin, card-edge design that accommodates all pins on all boards in a standard configuration.

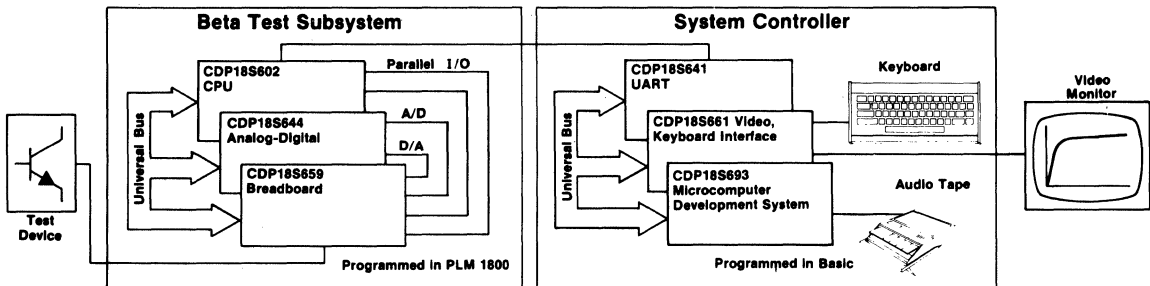


Fig. 1 - Block diagram of the test system.

All control signals of the CDP1802 are available on the backplane, an advantage that minimizes software manipulation as different boards are accessed. The parallel signal, and pin configuration of all RCA Microboards allows the use of the simple yet rugged printed-circuit board backplane that makes hardware changes easy; any Microboard can simply be added or removed as system memory of I/O requirements change. All of the I/O connections that are specific to an individual Microboard are brought out at the opposite end of the board from the backplane connection, again with industry-standard connectors. Fig. 2, a diagram of a typical Microboard configuration, illustrates the board connections.

Circuit Operation

On power up, both the system controller and test-subsystem processors are reset, and an initialization program for each is stored in PROM. The operator is prompted to insert a transistor in the test socket and key an input on the keyboard to start testing. The system-controller calculates test words, which are sent over a serial line to the test subsystem. The test-subsystem microprocessor decodes these words, sets up the proper conditions on the transistor and causes a measurement (reading) to be taken. Based on the reading, the test-subsystem processor may change some of the conditions and take additional readings. If a proper reading is achieved, data words are sent back; if not, an error code is returned. The system-controller processor then stores the reading and sends another set of data words. This action continues until all of the tests are performed. The system-controller calculates the beta, compares it to preprogrammed limits, and outputs a pass or fail message to the video display. The operator can then have the equipment enter a diagnostic mode, can retest the same device, or can insert a new one.

SYSTEM CONTROLLER

Hardware

The system controller is designed to calculate and transmit the data words to the test subsystem, receive and decode the data words, calculate pass/fail information, receive inputs from the keyboard, and update the video monitor as new information becomes available. Because the controller must be interactive with the user and easily expanded or changed, an interpretive language was chosen as its means of communications with the system. A requirement of this language was that it have the floating-point arithmetic capability required to perform arithmetic calculations. RCA's BASIC3 meets these criteria and, in addition, is available in the Microcomputer Development System Microboard, CDP18S693. A CDP18S641 UART (universal asynchronous receiver/transmitter) Microboard is used to provide communication (it transmits test and data words) over the RS232 link with the remainder of the system. A CDP18S661A Audio, Video, Keyboard-Interface Microboard is used for keyboard and color-video-monitor control; all of the video is in color. These boards, together with those in the MCDS (microcomputer development system), the CDP18S601 CPU Microboard and the CDP18S652 ROM, RAM, Cassette-Interface Microboard, complete the board complement of the system controller.

All system Microboards are housed in a CDP18S676 chassis with cover; the dc voltages—+15, -15 and +5 volts—are brought into the rear of the chassis. The RS232 link cable plugs into the CDP18S602 board of the test-subsystem.

Software

During the initial software development for the system controller, the developmental version of BASIC3 was used,

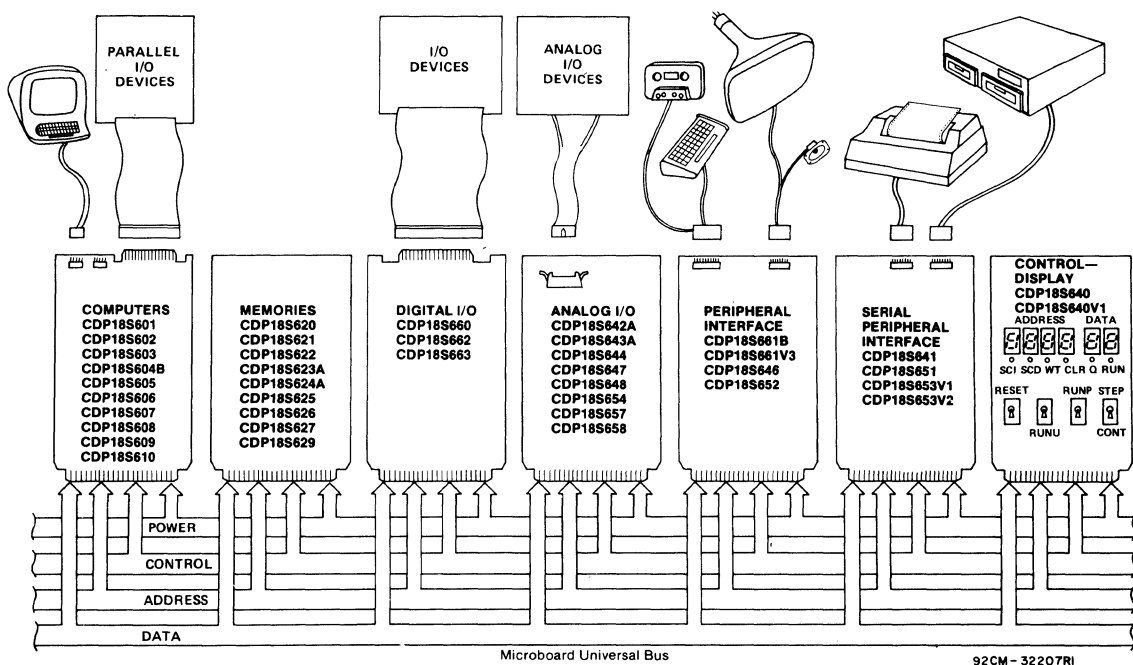


Fig. 2 - Diagram of typical Microboard-system complement.

so that it was possible to encode and run the program in real-time. Once the code was completed, it was burned into EPROM and the Run Time BASIC PROM's were substituted for the developmental version. The interpreter PROM's occupy 12 kilobytes (kbytes) of memory, the BASIC program occupies 2 kbytes, and the text for the video output takes 4 kbytes. Approximately 1 kbyte of RAM is used for stack, variable storage and work area.

On reset, the program initializes the UART board and the video board, outputs a message on the video monitor and waits for command input from the operator to start testing. On command, the system controller calculates and sends test words to the test subsystem and receives and stores the data words. After the testing is complete, the beta of the transistor is calculated for each test point and compared to a limit. A pass or fail message is output to the video, and the system waits for an input telling it to retest or enter the diagnostic mode. Fig. 3 is a portion of the BASIC3 program, specifically, the single-step diagnostic mode.

Diagnostics

Three choices of diagnostic mode are available: single step, table of values, and graph of beta-versus-collector current. In the single-step mode, the operator is prompted to input a collector-emitter voltage and collector current. From these values, the system controller calculates the test words and transmits them to the test subsystem, which performs the test and returns the data words. The data is decoded and beta is calculated; the base current and beta are displayed for the operator. The second option will

display a table of the value of collector current, base current, and beta as found during the pass/fail testing. The final option presently uses the graphics capability of the CDP18S661A Video Microboard. A graph of beta-versus-collector current is calculated from the pass/fail test data and displayed one video dot at a time by this board, a practice that products a smooth and continuous curve.

BETA-TEST SUBSYSTEM

Hardware

The beta-test portion of the test system is required to accept and decode test words, set up the test parameters for the transistor under test, read the results, reset the test parameters, if necessary, and finally transmit the results. A latch was needed to store the emitter-current-range bits for the test. A power-on reset assures that the system comes-up running, and ROM and RAM are used to store the program and provide for stack area. All of these capabilities are available on the CDP18S602 CPU Microboard. The transistor under test is connected into the system through a CDP18S659 Microboard breadboard.

The V_{CE} voltage-test word and the emitter-current word must be converted into analog signals before they can be applied to the transistor. In the subject system, eight bits of data are sufficient for the desired accuracy. The two D/A channels in the CDP18S644 A/D and D/A converter Microboard handle this conversion. The CDP18S644 also has sixteen 8-bit A/D channels, one of which is used to read the voltage across the base sense resistors to provide a base-current multiplier. The values of the multiplier and the

```

FILE: ATOD1.BAS          DISK: NEW BASIC

10 PRINT CHR$(4)
20 DEFINT H
30 PRINT "ENTER A COLLECTOR VOLTAGE BETWEEN .1 AND 10 VOLTS"
40 INPUT X
50 IF X>10X=10
60 A=INUM(X*25)
70 PRINT "ENTER A COLLECTOR CURRENT BETWEEN .001 AND 10 MILLIAMPS"
80 INPUT V:W=V
90 IF W>=10W=9.990
100 IF W<.001W=.001
110 Y=.434*LOG(1000*W)
120 B=4*INT(Y)
130 C=INUM(250*(W/10^(FNUM(INT(Y))-2)))
140 OUT (2,3,*1D)
150 H=INP(2,2)
160 Q=0
170 OUT (2,2,B): GOSUB 340
180 OUT (2,2,C): GOSUB 340
190 OUT (2,2,A)
200 Q=1: GOSUB 340
210 D=INP(2,2): OUT (2,2,0)
220 GOSUB 340
230 E=INP(2,2): OUT (2,2,0)
240 S=(FNUM(E))/250*10^(D-3)
250 IF S=0 PRINT "BASE CURRENT IS BELOW RANGE"
260 IF S=0 GOTO 300
270 PRINT "AT A COLLECTOR TO EMITTER VOLTAGE OF ";X;" VOLTS, AND A"
280 PRINT "COLLECTOR CURRENT OF ";V;" MILLIAMPS, THE BASE CURRENT IS"
290 PRINT S;" MILLIAMPS, GIVING A BETA OF ";INUM(V/S)
300 PRINT : PRINT : PRINT : PRINT
310 INPUT "DEPRESS C TO CONTINUE OR E TO EXIT AND RETURN "A#
320 IF A#="C" GOTO 10
330 END
340 H=INP(2,3)
350 F=H AND 1
360 IF F<>1 GOTO 340
370 IF Q=1 RETURN
380 H=INP(2,2): RETURN
390 WFLN "ATOD1.BAS:0"
400 DOUT : LIST : CLOSE : TOUT
410 END
    
```

Fig. 3 - The single-step diagnostic-mode portion of the BASIC3.

base-current range are transmitted through the UART to the system controller, which outputs the value of the base current.

Software

The software program for this part of the system was written on a CDP18S008 CDSIV Development System in PLM 1800. PLM was chosen because of the ease of coding in the higher-level language, and particularly because of the I/O constructs in this language that permit direct communication with the I/O Microboards. On reset the

PLM program starts automatically, initializes the hardware, and waits for a signal from the UART to the effect that it has received data. After receiving the third data word, all three words are sent to the A/D and D/A converter board and the parallel I/O port of the CDP18S602 board, which sets up the voltage and current for the transistor. The A/D converter is instructed to read the sense resistor and convert the data, which is then transmitted to the system controller via the UART. The program then loops back and waits for another set of inputs. Fig. 4 is the PLM source code of the program. The assembled code occupies approximately 3 kbytes of

```

DO;
MAIN: PROCEDURE;
DECLARE OUT$LOOP BYTE INITIAL(0),
        CONTINUE$PROCESSING BYTE INITIAL(0),
        INNER$LOOP BYTE,
        STOP BYTE INITIAL(1),

        UART$BOARD BYTE INITIAL(1),
        UART$WORD BYTE INITIAL(10H),
        ATODTOA$BOARD BYTE INITIAL(30H),
        FIXED$CHANNEL BYTE INITIAL(0),
        START$CONVERSION BYTE INITIAL(0),
        PIO$BOARD BYTE INITIAL(8),

        TABLE(10) ADDRESS,
        IC$RANGE BYTE INITIAL(1),
        IC$MULTIPLIER BYTE INITIAL(2),
        VCE$VOLTAGE BYTE INITIAL(3),
        IB$RANGE BYTE INITIAL(4),
        IB$MULTIPLIER BYTE INITIAL(5),

        DUMMY BYTE,
        I BYTE;

WAIT: PROCEDURE;
DUMMY = 0;
DO WHILE DUMMY=0;
    DUMMY=INPUT(3) AND 1;
END;

END WAIT;
DO WHILE OUT$LOOP=CONTINUE$PROCESSING;
    TABLE(IC$RANGE)=0;
    OUTPUT(1)=UART$BOARD;
    OUTPUT(3)=UART$WORD;
    DUMMY=INPUT(2);
    DO I=IC$RANGE TO VCE$VOLTAGE;
        CALL WAIT;
        TABLE(I)=INPUT(2);
        OUTPUT(2)=0;
    END;
    OUTPUT(1)=ATODTOA$BOARD;
    OUTPUT(3)=TABLE(VCE$VOLTAGE);
    OUTPUT(4)=TABLE(IC$MULTIPLIER);
    INNER$LOOP=CONTINUE$PROCESSING;
    DO WHILE INNER$LOOP=CONTINUE$PROCESSING;
        OUTPUT(1)=PIO$BOARD;
        OUTPUT(2)=TABLE(IC$RANGE) OR TABLE(IB$RANGE);
        CALL TIME(2); /*WAIT FOR BOARD TO SETTLE DOWN*/
        OUTPUT(1)=ATODTOA$BOARD;
        OUTPUT(6)=FIXED$CHANNEL; /*CHANNEL 0 & SINGLE ENDED*/
        OUTPUT(5)=START$CONVERSION;
        DO WHILE (EF1=0) END; /*WAIT FOR CONVERSION TO COMPLETE*/
        TABLE(IB$MULTIPLIER)=INPUT(3);
        IF TABLE(IB$MULTIPLIER) > 0FBH
            THEN DO; /*IF OUT OF IB RANGE, GOTO OUT OF RANGE ROUTINE*/
                IF TABLE(IB$RANGE) NE 03H
                    THEN TABLE(IB$RANGE)=TABLE(IB$RANGE)+1;
                ELSE DO;
                    TABLE(IB$MULTIPLIER)=OFFH;
                    INNER$LOOP=STOP; END;
            END;
        ELSE INNER$LOOP=STOP;
    END;
    OUTPUT(1)=UART$BOARD;
    DO I=IB$RANGE TO IB$MULTIPLIER;
        OUTPUT(2)=TABLE(I);
        CALL WAIT;
        DUMMY=INPUT(2);
    END;
END;
END MAIN;
CALL MAIN;
END ATOD;
EOF

```

Fig. 4 - PLM source code of the program.

ICAN-7026

PROM and uses less than one page of RAM area.

System Expansion

With additional software only, the beta-test subsystem can be made to perform base-emitter voltage (V_{BE} active) testing. By adding additional CDP18S641 UART Microboards to the system controller, additional subsystems can be designed and added to measure other transistor parameters, such as emitter-base and emitter-collector saturation voltages, junction breakdown voltages, and junction leakages. Each subsystem can be designed and built as a

module and, since each has its own microprocessor, can, with the overall supervision of the system controller, test, make some first-level decisions, and pass back only final data, thus speeding up the overall process of testing.

Acknowledgments

The author appreciates the assistance of R. Isham in the form of discussions of general design concepts and, in particular, for his design of the test-transistor interface board.

LOW-POWER TECHNIQUES FOR USE WITH CMOS CDP1800-BASED SYSTEMS

by G. Johnson

RCA CDP1800-series memory and microprocessor products employ static memory cells in all data-storage registers with the result that the products are fully functional from dc to their maximum rated frequencies. It is the static capability of the CDP1800-series products that gives them an advantage when low power consumption is imperative. This Note describes various techniques for reducing the power requirements of microcomputer systems since battery life is so important in most portable applications and in systems having a RAM battery back-up provision.

CMOS POWER DISSIPATION

The total power consumption of a CMOS microcomputer system is the sum of the quiescent and dynamic power dissipation. The quiescent dissipation comprises the combined dc leakage of all devices in the system, and is usually very much less than the dynamic power dissipation of the system. Quiescent power losses can be kept to a minimum by selecting parts with lowest leakage and/or by using 10-volt-rated (non-"C"-suffix) parts, which have a lower maximum-leakage-current specification than suffix-C parts when operated at 5 volts.

Dynamic power dissipation has three components:

1. The dissipation that results from current that charges and discharges the internal node capacitances of the CMOS devices, a design and process-influenced parameter that varies with system clock frequency. This dissipation can be somewhat reduced by choosing memory devices that gate off their input buffers with a chip select signal.
2. The dissipation that results from current that charges and discharges the external load capacitance of the output buffers. The dissipation of each output buffer is equal to CV^2f , where C is the load capacitance, V is the supply voltage, and f is the switching frequency of that output. Careful board layouts and smaller system size will keep external load capacitances to a minimum.
3. The dissipation caused by the current spikes through the PMOS and NMOS transistors in series at the instant of switching. This component amounts to approximately 10 percent of the total dissipation given in the data sheet for most RCA CMOS integrated circuits. Fast transition times of input signals to all CMOS devices will reduce the power consumed by current spikes.

POWER-SAVING TECHNIQUES

The obvious method of saving power in any microcomputer system is to shut the system down completely when not in use. This method is fine if data retention in RAM or CPU registers is not required. However, complete system power-down is not always possible in intermittent-duty systems, systems that must remain at idle as they wait for an interrupt to signal the beginning of processing. Data-acquisition systems are a good example of intermittent duty systems: data is collected at regular intervals, stored in RAM, and eventually down-loaded into some mass-storage device, such as tape or disk.

In reality, there are numerous ways of operating a system with low power; Table 1 shows these methods and indicates the tradeoffs involved in each. The following paragraphs detail these techniques and include diagrams and flowcharts to ease implementation in a system design.

Program Idle

When an idle instruction is executed, the CPU continuously repeats execute cycles (S1) without advancing the program; however, most of the timing signals (TPA, TPB, and MRD) and the high and low address bytes continue to be asserted by the CPU. The power savings in program idle is typically 20 percent of the running power, Fig. 1.

Pause Mode

When the WAIT pin on the CPU is brought low (with CLEAR high), the CPU stops; output signals are held indefinitely and only the oscillator remains in dynamic operation. Consequently, as shown in Fig. 1, the dynamic power of the system is reduced to the sum of the total device leakage plus the dynamic power of the oscillator. When the WAIT pin is again made high, the system resumes running on the next negative-going clock transition.

Fig. 2 shows a circuit that allows hardware or software initiation of the pause mode. An external stop signal can be applied to the control flip-flop to activate the WAIT input or, alternatively, a CPU software instruction (INP, OUT, SEQ) can be issued to assert a Q or N line to clock the flip-flop to the pause state. The pause mode is terminated when an external start pulse resets the flip-flop and allows the CPU to resume normal operation. Fig. 3 contains program flowcharts showing two methods of implementation of the software-driven pause mode.

Table 1 - Analysis of Power Consumption in a Typical CDP1800-Based System

Operating State of System	Dynamic Power	Quiescent Power	Advantages	Disadvantages
Running	All Devices	All Devices	Fully Functional System	Constant Running Power
Idle Instruction	Slightly Reduced	All Devices	DMA and INT Requests Accepted	Does Not Save Much Power Over Running
Pause Mode (Wait Pin Held Low)	CPU Only	All Devices	Pause Driven By Software or External Input	DMA and INT Requests Not Accepted
Oscillator Stopped	None	All Devices	Easily Implemented, Software Driven	Large Software Overhead, CPU Reset Required
Oscillator Slowed	Greatly Reduced	All Devices	System Fully Functional at Reduced Speed	Good for RC Oscillators Only
Voltage Shut-Off to CPU & Other Parts	None	Powered Devices Only	RAM Data Maintained	Complex Circuitry Required
RAM Powered	RAM Only	RAM Only	Lowest Power to Maintain Memory Data	Loss of CPU Register Data
Total Shut-Down of System	None	None	Easily Implemented, Best Battery Life	Total Loss of RAM & CPU Data When Off

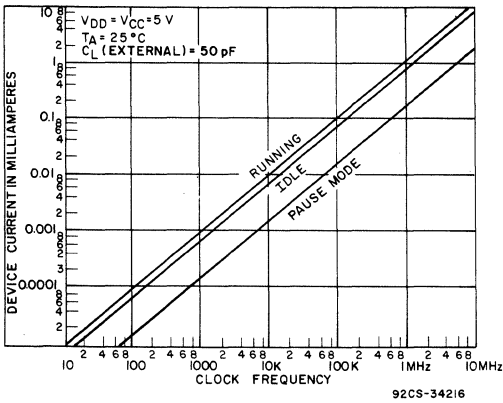


Fig. 1 - Typical power consumption of the CDP1802 as a function of clock frequency.

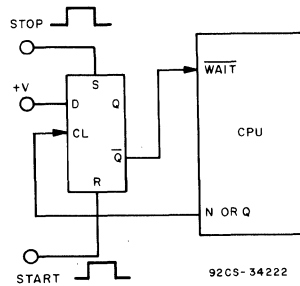


Fig. 2 - Simple pause-mode implementation.

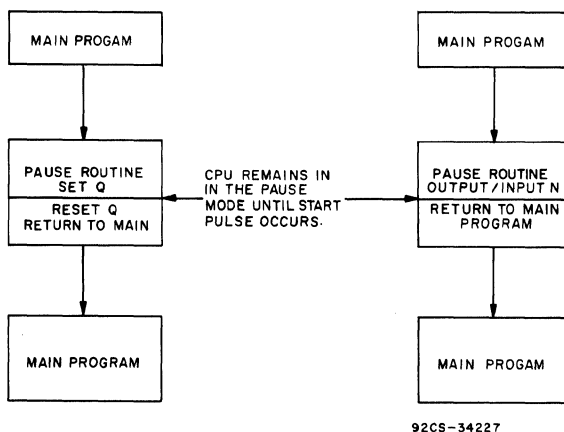


Fig. 3 - Program flowcharts for two methods of implementation of the software-driven pause mode.

Software-Driven Oscillator Stop

The flowchart for the software-driven oscillator-stop power-saving technique is shown in Fig. 4. The shutting down of the oscillator will save even more power than the application of the pause mode because power drain will then be strictly the result of device leakages. When the program reaches a point where the system is to be deactivated or "put to sleep," the Q output is set to a logic high, Fig. 5. The diode becomes forward biased and clamps the clock input high, and the oscillator stops. To activate or "wake up" the system, the CLEAR pin is brought low, which resets the CPU and the Q output as well. The diode is then back biased and the oscillator begins to function. The RC network on the CLEAR pin in Fig. 5 assures that the reset pulse continues until the oscillator has stabilized. An external signal from a real-time clock alarm can be used as the means to wake up the system.

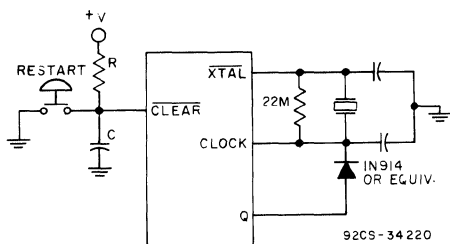


Fig. 5 - Oscillator shut-down with warm-start recovery.

An alternate circuit, shown in Fig. 6, uses a CD40107 open-drain dual-NAND buffer to clamp the clock when Q is set. A feature of this circuit is that it detects a warm-start condition by allowing the software to poll the flag line after reset for an external wake-up signal.

To avoid a floating-bus condition, and because the CPU tri-states its bus drivers during the SET Q instruction, pull-down or pull-up resistors should be used on all data lines.

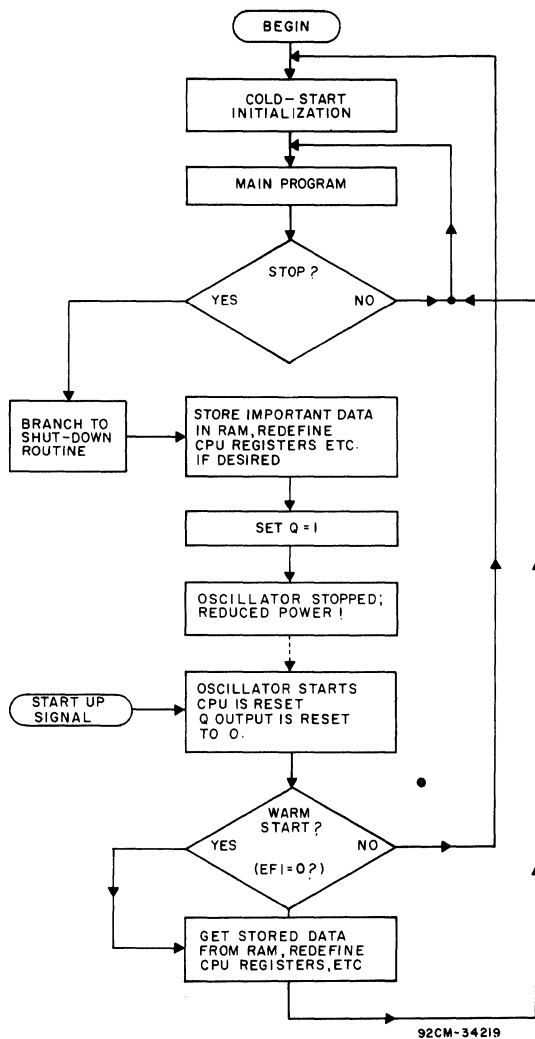


Fig. 4 - Software-driven oscillator-stop program flowchart. Refer to circuit of Fig. 6.

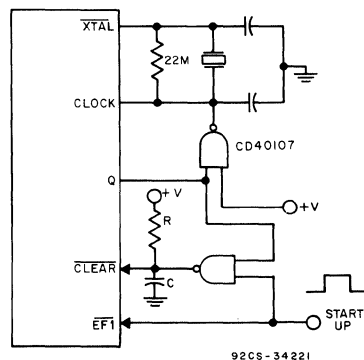


Fig. 6 - Oscillator stop/start circuit.

Oscillator Slow-Down

When a CDP1804 or CDP1805 is used in the system, the Schmitt-triggered oscillator buffer of these devices allows an RC circuit to be used to reduce the CPU clock frequency during periods of idle operation. (When using the CDP1802 microprocessor, an external RC oscillator must be used, as the clock input buffer is not Schmitt triggered). Fig. 7 shows a circuit that uses a bilateral transmission gate (CD4016 or CD4066) to switch an extra resistor in or out of the RC oscillator circuit. The transmission gate is driven by an idle-detector circuit consisting of logic and a simple RC delay (R3 and C2).

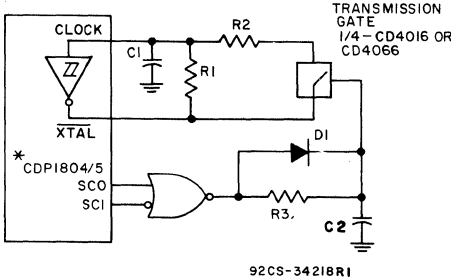


Fig. 7 - Variable-frequency power-saving circuit.

When an idle instruction is executed, the CPU continues to perform execute cycles (S1) until it is reset, or until a DMA or interrupt request is received. The logic detects the presence of the idle state and begins discharging C2 through R3. If execute cycles continue, the transmission gate is turned off; the result is a reduced oscillator frequency. During normal program operation, states other than S1 occur often enough to allow diode D1 to keep C2 charged, the transmission gate on, R2 switched in, and the oscillator running fast. R1 is a very high resistance (10 to 20 megohms); R2 is a relatively low resistance, but greater than 20 kilohms so that the amplifier gain is not reduced to the point where oscillation cannot occur. To avoid noise injection at the

clock pin, capacitor C1 should not be less than 100 picofarads in value.

If desired, the CPU Q output can be used to drive the transmission gate directly. During normal program operation, the Q output is set and remains high. When a slower frequency is desired, the Q output is reset low, which causes the transmission gate to open and the oscillator to slow down.

The advantage of the oscillator-slow-down power-saving method is that the system remains responsive to interrupt and DMA, and can continuously poll flag lines for external inputs.

Partial System Shut-Down

During long periods of system inactivity, and in cases where RAM data retention is required, considerable power can be saved by shutting off the supply voltage to the CPU as well as to the ROM and I/O devices. Fig. 8 shows a typical small system in which a software-driven signal is used to shut off VDD. In this circuit, transistor Q1 conducts and maintains VDD, provided that either the start-up signal is applied or the CPU Q output is present. The software program enters the power shut-down mode by resetting the Q output to a logic low. Q1 turns off and the inverter drives low to assure a fast shut-down of VDD as well as the presence of a path for any leakage current through transistor Q1. The CPU oscillator stops when VDD is removed, and the ROM and I/O device power down, leaving only the RAM's and some logic circuits to draw quiescent current from the power source.

To resume normal operation, a start-up signal is applied, Q1 is turned on, and VDD is restored. The RC circuit on the CLEAR pin of the CPU provides time for the crystal oscillator to reach full amplitude before allowing the CPU to begin processing. The software-driver program initially sets the Q output high so that VDD is locked on; once this event has taken place, the start-up signal can be removed. The Q output also provides the master chip select to the RAM devices so that the RAM is insensitive to any spurious addressing or write commands during the power-down and power-up sequences.

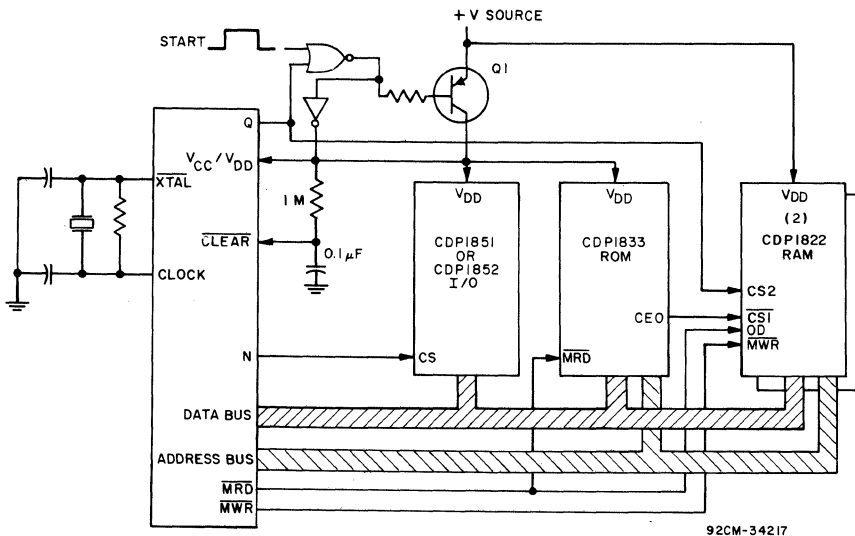


Fig. 8 - Partial power-shut-down technique.

Table II - CDP1800 Instructions That Float the Data Bus

<u>OP Code</u>	<u>Mnemonic</u>	<u>Instruction</u>
1N	INC	Increment
2N	DEC	Decrement
76	SHRC	Shift-Right W/Carry
7A	REQ	Reset Q
7B	SEQ	Set Q
7E	SHLC	Shift-Left W/Carry
F6	SHR	Shift Right
FE	SHL	Shift Left
6800	STPC	Stop Counter
6801	DTC	Decrement Counter
6802	SPM2	Set Pulse-Width Mode 2
6803	SCM2	Set Counter Mode 2
6804	SPM1	Set Pulse-Width Mode 1
6805	SCM1	Set Counter Mode 1
6807	STM	Set Timer Mode and STRT
6809	ETQ	Enable Toggle Q
680A	XIE	External-Interrupt Enable
680B	XID	External-Interrupt Disable
680C	CIE	Counter-Interrupt Enable
680D	CID	Counter-Interrupt Disable
686N	RLXA	Reg. Load Via X & Advance
688N	SCAL	Standard Call
689N	SRET	Standard Return
68AN	RSXD	Register STR Via X & Dec
68BN	RNX	Reg. N to Reg. X Copy
68CN	RLDI	Register Load Immediate

FLOATING DATA BUS AND USE OF PULL-UP/PULL-DOWN RESISTORS

The CPU data bus floats when certain nonmemory instructions are executed. Table II lists the instructions that cause a floating bus to occur during one or more execute cycles. The bus will also float during an interrupt cycle (S3). If the data bus is allowed to float without pull-down or pull-up resistors, the input buffers of all devices connected to the data bus may be subjected to a signal that is neither high nor low, but that, in fact, may be near the logic-transition region. Such a signal will cause the p and n transistors to conduct simultaneously; consequently, the device currents will be much greater than normal, with possible latch-up or device damage occurring as the result of overheating.

If pull-up or pull-down resistors are used, they will define the logic state of the data bus during the instructions listed in Table II, but they will also draw current of a magnitude dependent on their values. To assure that excessive current is not drawn, all signals to any CDP1800-series devices should have a rise or fall time of 1 microsecond or less. When the data-bus floats, the speed at which the pull-up/pull-down resistors can define the logical condition of the individual bus lines is equal to the time constant of the RC network made up of the pull-up/pull-down resistors and the load capacitance of the data line. Fig. 9 is a graph of recommended pull-up/pull-down resistance as a function of data-bus load capacitance.²

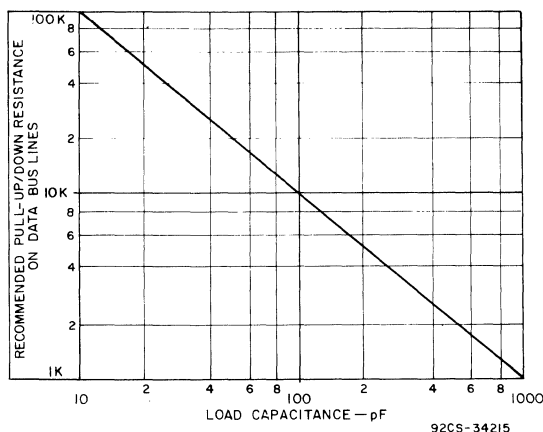


Fig. 9 - Recommended pull-up/pull-down resistance as a function of data-bus load capacitance.

REFERENCES AND BIBLIOGRAPHY

1. For further details on CMOS-device power dissipation: "Power-Supply Conditions for COS/MOS Devices", H.L. Pujol, RCA Application Note ICAN-6576.
2. "Designing Minimum/Nonvolatile Memory Systems with CMOS Static RAM's", R.M. Vaccarella, RCA Application Note ICAN-6943.
3. **RCA COS/MOS Memories, Microprocessors, and Support Systems**, RCA Solid State DATABOOK SSD-260.

ACKNOWLEDGMENT

The author thanks Dave Shand (RCA Quickborn, W. Germany) for information used in this Note.

CDP1800-BASED VIDEO TERMINAL USING THE RCA VIDEO INTERFACE SYSTEM, VIS

By R.M. Vaccarella

As a result of functional integration, the bringing together of many system functions into a single IC, the design of a sophisticated system to control the formatting and refreshing of a video display has been considerably simplified. The RCA Video Interface System, VIS, uses this technique to provide the designer with a powerful chip set (CDP1869, CDP1870, CDP1876) for use in video terminals, industrial displays, and broadcast-TV text overlays.

An important advantage of the VIS is its ability to operate independently of the CPU, and to provide all of the synchronization signals and refresh data for a standard NTSC (U.S.) or PAL (European) raster-scan display, while also supplying a clock input to the CPU. The result is a CPU free to handle other tasks, such as monitoring a keyboard; a primary requirement of the operating system used in the application discussed in this Note. The VIS is fully compatible with RCA CDP1800-series microprocessors and peripheral components. Direct interface with the CPU allows easy access to the VIS command registers and the dedicated screen-refresh page and character memories. The VIS sound output, programmable in frequency and amplitude, is convenient for audible keyboard feedback and system failure alerts.

The major features available with the VIS chip set are adequately demonstrated through its use in this Note in the implementation of a low-cost 40-character by 24-line video data terminal. Although the circuit design discussed is for a black-and-white display, gray-scale and color are easily added, as are the other VIS features detailed in the VIS data sheet.¹ The VIS chip set, a product of CMOS LSI technology, reduces the data-terminal system parts count to just sixteen IC's and a handful of passive components. The flexibility of the Video Interface System and the many on-board features included in the IC's in the chip set allow easy expansion of the basic design described. System software requires only a minimum amount of user program memory and is considerably simplified through the use of a unique interpretive language, the VIS Interpreter (described more fully below).

Although low-power requirements are not generally a serious problem in video-terminal applications, the standard CMOS advantages inherent in the VIS chip set, such as wide operating voltage (4 to 10.5 volts) and temperature ranges (-40 to +85°C), can be important in many industrial environments. These factors, coupled with the variety of programmable features, make the VIS a logical choice for all low-cost video applications.

REVIEW OF TERMS

The following information provides the reader with general definitions of some of the terms used in this Note, terms that apply to typical CRT controllers.

Raster-Scan Display - Raster-scan refreshing involves the simultaneous sweeping of an illuminated spot in both the horizontal and vertical directions, from left to right and top to bottom, on a cathode-ray tube (CRT). The operation, repeated at a rate fast enough to produce a flicker-free display and smooth motion (typically 60 times per second), requires each horizontal line of picture information to be followed by a horizontal synchronization pulse and each full screen (or field) of horizontal lines to be followed by a vertical synchronization pulse. The horizontal sync, vertical sync, and picture information are combined and transmitted as a single signal, to be separated by the appropriate circuitry in the video-monitor chassis. Fig. 1 shows a typical raster-scan display.

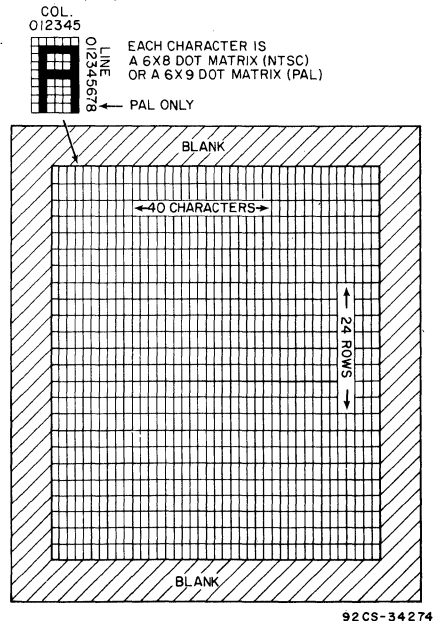


Fig. 1 - Typical raster-scan display.

Display Memory Requirements - The screen-refresh display memory requirements can be satisfied by employing either of two different methods. The first, a bit-map approach, requires that a copy of the actual character matrix be stored in the bit-map memory each time that character appears on the screen. This requirement results in a pixel (picture element) "map" of the display in memory. The bit-map approach is most efficient in displays with a large amount of nonrepeating graphics or characters.

The second method, the character-generator approach, requires the use of two separate storage areas, and is the method supported by the VIS. The two storage areas, page memory and character memory, perform specific display-related tasks. The page memory holds the ASCII (American Standard Code for Information Interchange) data for each display screen character, and is essentially a memory map of the location and type of character written to the CRT. Its size is determined by a one-to-one correlation of the number of characters per row times the number of rows displayed. Therefore, one page typically represents one full screen of text. It is common, however, for the page memory to hold multiple pages of text, in which case the CRT screen becomes a display window that scrolls through the page memory.

Each ASCII character requires seven bits of data. These seven bits are the data outputs of the page memory and become the higher address (or column) inputs of the character memory; these inputs select one of 128 different characters. An ASCII character is represented as a column and row matrix (i.e., 6 columns by 8 rows) in the character memory, Fig. 2. A 6-bit-wide memory, then, would require eight consecutive locations to store one ASCII character.

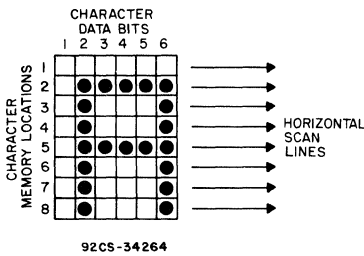


Fig. 2 - 6x8 character matrix.

The eight locations are selected by the lower address (or row) inputs, supplied by the CDP1869 (CMAO-CMA2), of the character memory.

The six data outputs for each of the eight memory locations from the character memory are sent to a parallel-to-serial shift register (within the CDP1870/76). The serial output of this shift register produces a series of dots for each horizontal scan line on the CRT, thus forming the ASCII character.

The advantage of using a separate page and character memory, as opposed to single bit-map memory, is that each character need only be defined once in character memory. Thereafter, only a single byte of page memory is required to call the character and display it on the screen. This method offers a considerable savings in memory size when a large number of characters are required, as in a text display.

Broadcast-TV Display - A broadcast-TV set is the type commonly found in the home to receive commercial TV broadcasts. The information received is transmitted over the air using an rf (radio frequency) carrier signal. The "front end" of the TV set, consisting of the rf tuner and if (intermediate frequency) sections, separates the video, sound, and synchronization information from the carrier signal. Further processing within the set results in a picture on the CRT that is complete and synchronized with the transmitted program. The broadcast-TV receiver is popular as a terminal display device for home computers and games because of its availability and relatively low cost. The output of a computer or electronic game can be connected directly to the antenna terminals, or a simple rf modulator circuit can be inserted between the TV and the source equipment. Most broadcast-TV receivers are also easily modified to accept a composite, single-signal video input that contains the horizontal sync, vertical sync, and video data but no rf carrier.

TV-Monitor Display - A TV monitor is basically a broadcast-TV set without the "front end" electronics; the source signal is a composite video signal input directly to the video section of the chassis. The TV monitor usually has a better quality CRT that provides slightly higher resolution than the CRT in a standard broadcast-TV receiver, but at a greater cost. The TV monitor may also provide scan rates not attainable with the standard broadcast-TV receiver.

RGB Monitor Display - Both broadcast-TV receivers and TV monitors can accommodate black-and-white or color CRT's. However, the use of color with this equipment requires transmission of the signal by another carrier-type method, a 3.58-MHz color subcarrier signal. Appropriate separator and demodulator circuitry contained in the TV chassis then processes the signal for display on the CRT. To eliminate this complex and bandwidth-limited technique, RGB color monitors are used to achieve very high color resolution for graphics-type applications. Separate red, green, and blue inputs are used to directly drive the corresponding CRT drive circuits without the use of the color subcarrier. However, because of the considerably higher cost and often nonstandard scan rates associated with RGB monitors, their use is limited to the most demanding display situations, or to applications where the CRT controller is part of the TV chassis. (The CDP1876 member of the VIS chip set is designed specifically for use with RGB color monitors.)

Text-Overlay Display - A text overlay is a display in which graphics or text is electronically superimposed on an existing display; for example, statistical information superimposed on a batter during a baseball game. This display technique requires that the synchronization signals of the two sources of information be locked together. In the case of the VIS, external horizontal and vertical sync inputs are provided for this purpose.

NTSC/PAL - The NTSC (National Television Standard Committee) standard is used for TV transmission in the U.S., Canada, Japan, and Mexico. The PAL (Phase Alternate Line) standard is used in the United Kingdom and continental Europe. The primary differences as far as CRT controllers are concerned are the line and field rates. The NTSC system has 525 horizontal lines in each picture field, which results in a horizontal sync frequency of 15,750 Hz. Each field is repeated at a 60-Hz rate. The PAL system has 625 lines (15,625 Hz) and a 50-Hz field rate. Since the PAL system has more horizontal lines than the NTSC, a different character aspect ratio results, and one more line of dots per character is needed so that the characters do not appear elongated. The VIS will operate with either NTSC or PAL systems.

GENERAL SYSTEM OPERATION

The VIS consists of a two-chip set of IC's: the CDP1869 address and sound generator, and the CDP1870/76 color video generator. The CDP1876 is a bond-out option of the CDP1870 in which the LUM (luminance), NTSC CHROM (chrominance) and PAL CHROM outputs are replaced by the RED, GREEN, and BLUE outputs to make the device compatible with RGB color monitors.

Fig. 3 is a functional block diagram of the CDP1869, which is used to control screen format, page and character memory addressing, and tone and white-noise generation. Software instructions and data are loaded into the CDP1869 directly from the CPU, with system timing controlled by hardware interaction with the CDP1870/76. Four command registers are available in the CDP1869 for screen resolution, mode control, and tone and white-noise programming.

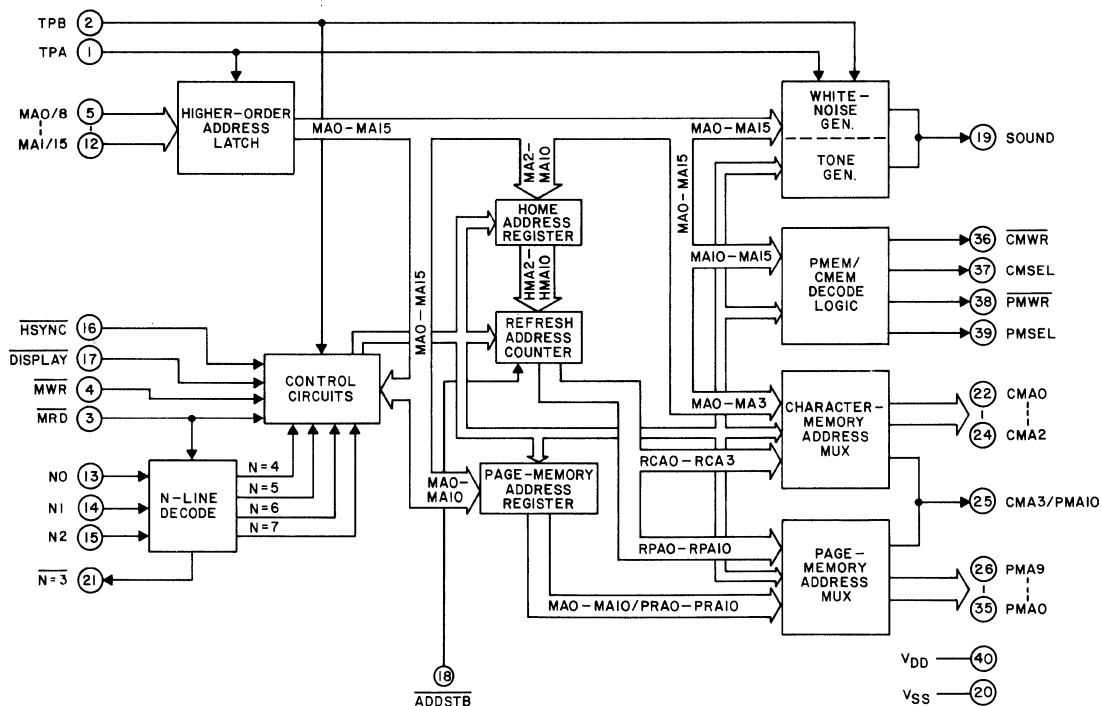
Fig. 4 is a functional block diagram of the CDP1870/76, which is used to control screen format, TV sync, video, and color. Sync timing to the CDP1869 and CPU, and access to the character-memory data, are also handled by the CDP1870/76. A single internal command register is available for horizontal screen resolution, display on/off, and color programming. All command and format instructions are executed prior to system operation, although command-register data may be changed asynchronously with respect to screen refreshing, with certain restrictions.¹

As shown in the system block diagram of Fig. 5, very few additional components are required to implement a complete color video display. The CPU and program memory provide overall system control, but are not involved in screen refreshing. The CDP1869 multiplexes page and character memory addressing from its own internal refresh

counter and the CPU when entering screen data. The refresh memory functions in a character-generator mode in holding screen and character data. The CDP1870 controls system timing and provides the composite-sync, luminance, and chrominance outputs needed to drive a standard TV monitor. Since the character memory may be ROM or RAM, the CDP1870 also controls multiplexing of the CPU data bus during operations involving the character RAM. The data bus may be routed to the CDP1870 command register or to the character data bits and character color bits for use by the character memory. The CDP1871 keyboard encoder is used to directly interface a SPST keyswitch keyboard to the CPU for data entry. The externally combined sync and video output may be used directly with a color monitor or with an rf-modulator circuit when the signal is to be fed to the antenna terminals of a broadcast-TV set. The sound output provides the buffering required to drive an eight-ohm speaker.

HARDWARE IMPLEMENTATION

The low-cost terminal described in this Note makes use of the basic design discussed above and shown in Fig. 5, with a few minor exceptions. Since it is a black-and-white display, the chrominance crystal (7.15909 MHz) and the capacitor on the CHROM output needed to implement color are not required. Gray-scale shading is obtainable, however, through the use of appropriate color commands. An RCA VP-611 ASCII (parallel output) membrane keyboard is used as the data input device, and the CDP1871 keyboard encoder is replaced by a CDP1852 configured as an input port. A power-on (R-C) reset and reset switch are also included. Fig. 6 is a block diagram of the video terminal; Figs A-1, A-2, and A-3 in Appendix A are detailed schematic diagrams.



92CM-34275

Fig. 3 - Functional block diagram of CDP1869.

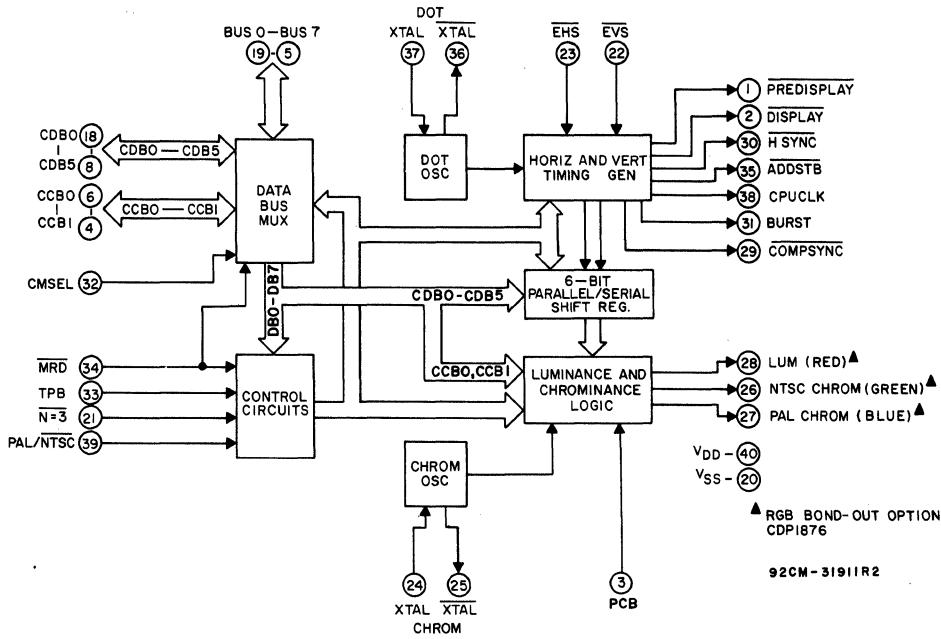


Fig. 4 - Functional block diagram of CDP1870/76.

The entire circuit can be constructed on a 4.5 by 7.5-inch pc board, such as the CDP18S659 Microboard Breadboard, Fig. A-4, Appendix A.

The CDP1802 CPU handles overall system control, interacting with the VIS and program memory as required. It provides initialization, keyboard control, and page memory up-dating, maintains the screen cursor position, and controls screen formatting.

Interfacing to the program and user memory is handled by the CDP1872 and the CDP1873. The CDP1872, an 8-bit

input port, latches the higher-order addresses from the CPU to provide address and chip-select data. A8, A9, and A10 are fed directly to the 2Kx8 memory devices. The CDP1875, a 1-of-8 decoder, uses the remaining addresses to provide eight chip-select signals in 2K blocks. Additional memory is easily added above address 2000₁₆, since the system uses only half of the available decoded 16K of memory. (A complete memory map is given in Appendix B.) The system program is stored in 5K of EPROM consisting of three 2716 2Kx8 devices (1K spare). User RAM consists of one 2Kx8 6116-type device.

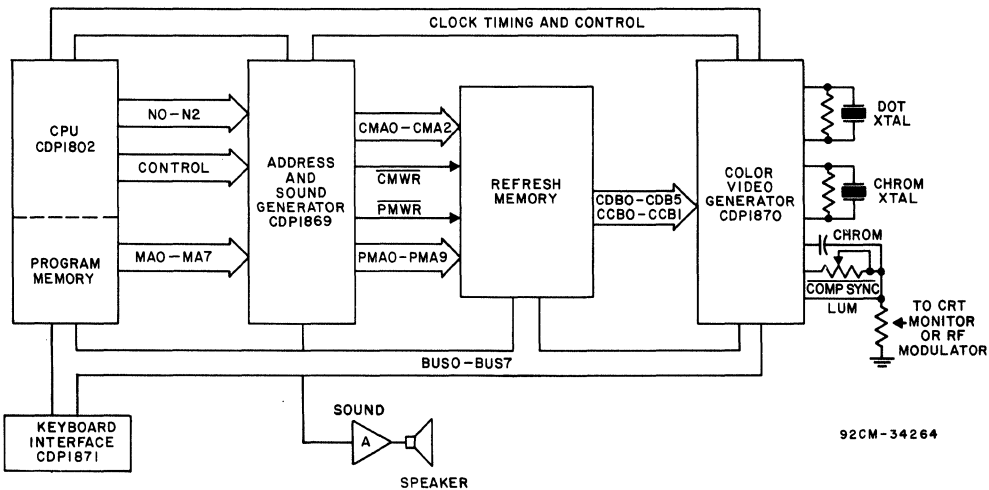


Fig. 5 - Video-terminal system diagram. (See Appendix A for detailed schematic diagrams.)

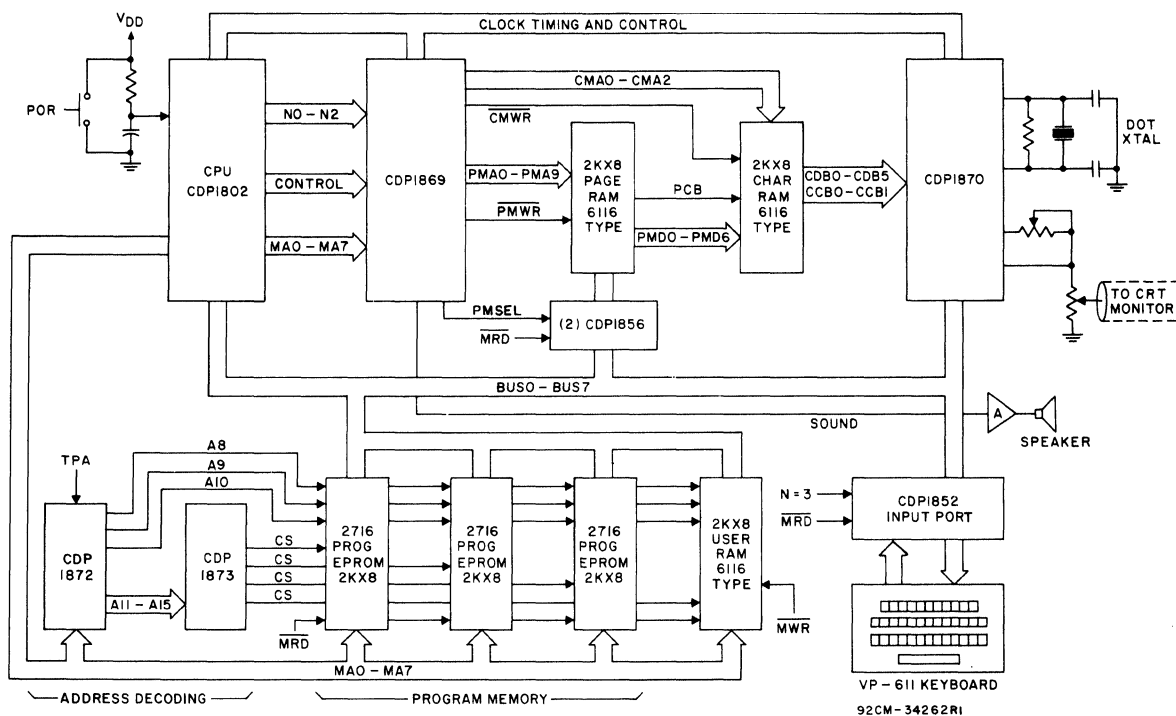


Fig. 6 - Block diagram of video terminal. (See Appendix A for detailed schematic diagrams.)

Page and character memory addressing and mapping are completely controlled by the VIS, and require only two CDP1856 4-bit bus buffer/separators to prevent bus contention and determine data direction to and from the page memory. The page memory is a 6116 2Kx8 RAM device, which allows double-page operation of the VIS. The character memory is also a 6116-type device, which offers some important system advantages. Since it is RAM, the standard ASCII character set is downloaded during initialization, and the characters may be modified during system operation. Further, because of the large memory capacity of the character RAM, an identical ASCII character set may be stored in reverse video format, with the PCB (page color bit) output (most-significant-bit (MSB)) of the page memory used to select either standard or reverse video format on a character-by-character basis. All of the memory devices used are pin-compatible with each other and with industry ROM devices, a condition that adds flexibility to the designer's choice of devices while simplifying decoding.

The data outputs of the character memory are connected to the CDP1870/76 dot and color inputs. Character data bits 0 through 5 provide the dot information for each character; character color bits 0 and 1 provide the color information or, in the subject application, gray-scale shading. Four levels of shading from white to black are available with CCBO and CCB1. A single external circuit composed of a crystal, a resistor, and two trimmer capacitors provides all system timing. The dot, address, and TV sync timing are derived from this 5.67 MHz (NTSC) crystal. The CDP1870/76 also provides a CPU clock output equal to one half the dot-crystal frequency and a PREDISPLAY synchronizing signal.

Two external resistors are used to adjust the ratio of the video and sync output levels, which can be connected directly to a standard TV monitor. The video (LUM) output supplies the background and character dot information; the sync (COMPSYNC) output contains the horizontal and vertical synchronization information. The sound output from the CDP1869 may be connected directly to an external amplifier or to the audio input of the TV monitor, if so equipped. A simple op-amp and transistor driver circuit may also be used to drive a small eight-ohm speaker. The sound output is used to supply an audio feedback to keystrokes, with a different tone each for normal and control characters and a third error tone for command format errors.

SOFTWARE IMPLEMENTATION

VIS Interpreter

The development of software for VIS applications is greatly simplified by the availability of a unique interpretive language. The VIS Interpreter (CDP18S835) is supplied on diskette and provides source code for use with RCA Development System assembly programs (ASM4, ASM8) in NTSC and PAL versions of the program. A table listing and a standard ASCII character-set source code are also included. After assembling or editing on a CDS (COSMAC Development System), programs may be stored in EPROM or ROM for nonvolatile system use.

The VIS Interpreter is an interpretive language, which runs in real time, that specifically supports the VIS with commands that control text, graphics, and motion on a CRT. It is an open-ended program, meaning that interpreter commands may be added or deleted as required by a user's

ICAN-7032

particular application. The program, as supplied by RCA, requires 3 kilo-bytes of ROM memory space and 64 bytes of RAM space. The VIS Interpreter, the 64-byte RAM, and the start of the user program may be located anywhere in memory by the use of the ORG, RAM, PROG equate statements in the source program.²

The VIS Interpreter terms and operating-system variables used in the subject CRT display application are given in Appendix C. A page-memory pointer (cursor), character-memory pointer, and main-memory pointer are maintained and easily loaded, incremented, and decremented, as are the accumulator and sixteen general-purpose variables. In the design for the subject application, the VIS Interpreter is stored in two 2716 EPROM devices located at addresses 0000-0BFF₁₆. After applying power to the system, or after a manual reset, the CPU automatically starts fetching instructions at address 0000₁₆. Therefore, the user is assured that the system always starts with the VIS Interpreter running. The remaining 1K of the second 2716 EPROM is used to store the standard ASCII character set (0C00-0FFF₁₆), which is downloaded, during initialization, to the VIS character RAM. The 64-byte RAM stack area required with the interpreter is located at address 1FC0-1FFF₁₆.

Operating System - When initialized, the VIS Interpreter searches for instructions at the location specified in the PROG equate statement. In the subject application, the program is the Operating System, which is stored in a 2716 EPROM device at location 1000-13FF₁₆. The entire Operating System uses only half of the capacity of the EPROM (one kilobyte); the remaining space is available to the user at his discretion. The Operating System contains the commands, routines, and subroutines necessary to control the video terminal and keyboard. Appendix D gives the command formats and explains each subroutine. Appendix E contains Operating-System flowcharts. A complete assembly listing containing object code and comments is given in Appendix F.

The Operating System is used to read and write to system RAM, input data from the keyboard, and to format and

change the display screen. Most of the standard terminal commands are available:

- carriage return
- line feed
- clear screen and home
- home cursor
- blinking cursor
- reverse video
- cursor direction (up, down, forward, back)

Additional commands and subroutines may simply be appended to the existing Operating System.

Since the Operating System program is written in VIS Interpreter language, the instructions are concise and easy to use. One routine of special interest is the RUN routine. It executes a user program, written in VIS Interpreter language, at the location specified by the USRAM equate statement of the Operating System source program, in this case location 1800₁₆. The Operating System requires only three dedicated variables: V9, VC, and VF. These variables should not be modified by a user program unless they are saved and subsequently restored. Other variables that may be used and modified by the Operating System routines are listed under each routine or subroutine heading in Appendix F.

REFERENCES and BIBLIOGRAPHY

1. "COS/MOS Video Interface System," RCA Solid State File Number 1197
2. **User Manual For The VIS Interpreter CDP18S835**, RCA Solid State Publication MPM-835
3. **COSMAC Microboard Computer Systems**, RCA Solid State publication CMB-250
4. **COS/MOS Memories, Microprocessors, and Support Systems**, RCA Solid State DATABOOK SSD-260
5. **COS/MOS Integrated Circuits**, RCA Solid State DATABOOK SSD-250
6. **Linear Integrated Circuits**, RCA Solid State DATABOOK SSD-240

Contents of Appendixes

Appendix	Content	Page
A	- PC board edge connector and keyboard connector pin assignments	7
	- Detailed Schematic Diagrams of Video Terminal System	8-9
	- Video terminal constructed on pc board	10
B	- Memory Map	10
C	- VIS Interpreter Terms and Operating System Variables	11
	Terms Used in VIS Interpreter Instructions	11
	Changes to Standard VIS Interpreter	11
	Operating System Equate Statements and Variable Usage	11
D	- Operating System Components	11
	Commands, Routines, Subroutines	11-13
E	- VIS Operating-System Flowcharts	14
	Main Program	14
	Memory Read	15
	Memory Write	16
	Run	16
	Clear Screen and Home	16
	Home Cursor	17
	Carriage Return/Line Feed	17
	Cursor Position	17
	Screen Resolution	17
	Control Flag	18
	Set Cursor	18
	Error	18
	Load Standard Character Set	18
	Input Character	19
	ASCII to Hex	19
	Display	19
	Tone Table	20
F	- VIS Operating System Listing	21

Appendix A

Table A-1 - P2 - 20-Pin Keyboard Connector*

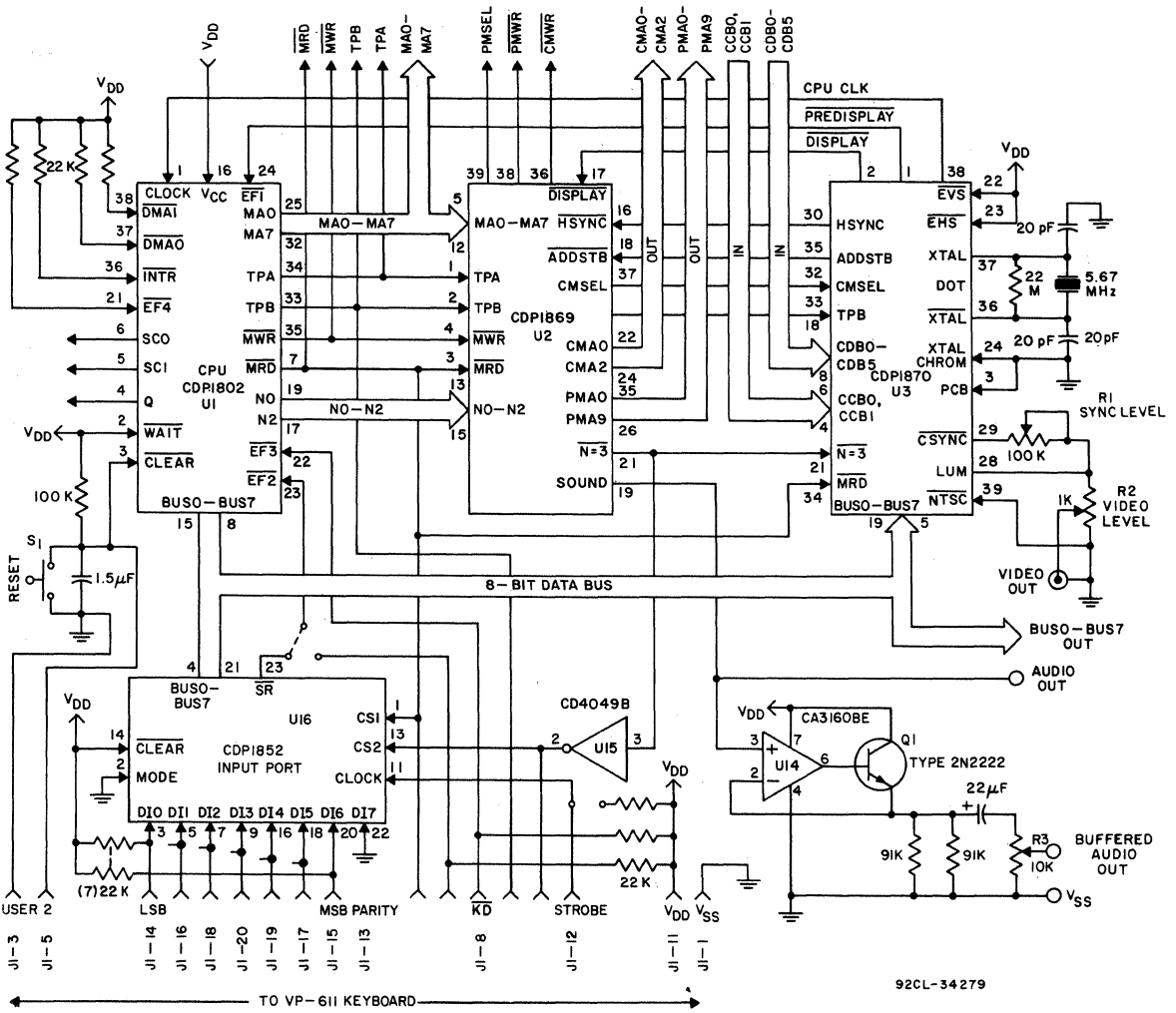
1 - GRD	11 - +5V
2 - POLARIZATION PIN	12 - STROBE
3 - USER 2	13 - B7 PARITY (N.C.)
4 - U + L CASE	14 - B0 LSB
5 - USER 2	15 - B6 MSB
6 - KD	16 - B1
7 - USER 1	17 - B5
8 - KD	18 - B2
9 - USER 1	19 - B4
10 - STROBE	20 - B3

*See Fig. A-1

Table A-2 - P1 - Edge Connector Wiring*

A - TPA - P	1 -
B - TPB - P	2 -
C - DB0 - P	3 -
D - DB1 - P	4 -
E - DB2 - P	5 - MRD - N
F - DB3 - P	6 -
H - DB4 - P	7 -
J - DB5 - P	8 -
K - DB6 - P	9 -
L - DB7 - P	10 -
M - A0 - P **	11 -
N - A1 - P **	12 -
P - A2 - P **	13 -
R - A3 - P **	14 - N0 - P **
S - A4 - P **	15 - N1 - P **
T - A5 - P **	16 - N2 - P **
U - A6 - P **	17 - EF1 - N
V - A7 - P **	18 - EF2 - N
W - MWR - N **	19 - EF3 - N
X -	20 -
Y - +5V -	21 - +5
Z - GRD -	22 - GRD

* See Fig. A-4. ** Must be wired on CPS I/O slot.



92CL-34279

Fig. A-1 - Video-terminal-system schematic diagram (part 1 of 3).

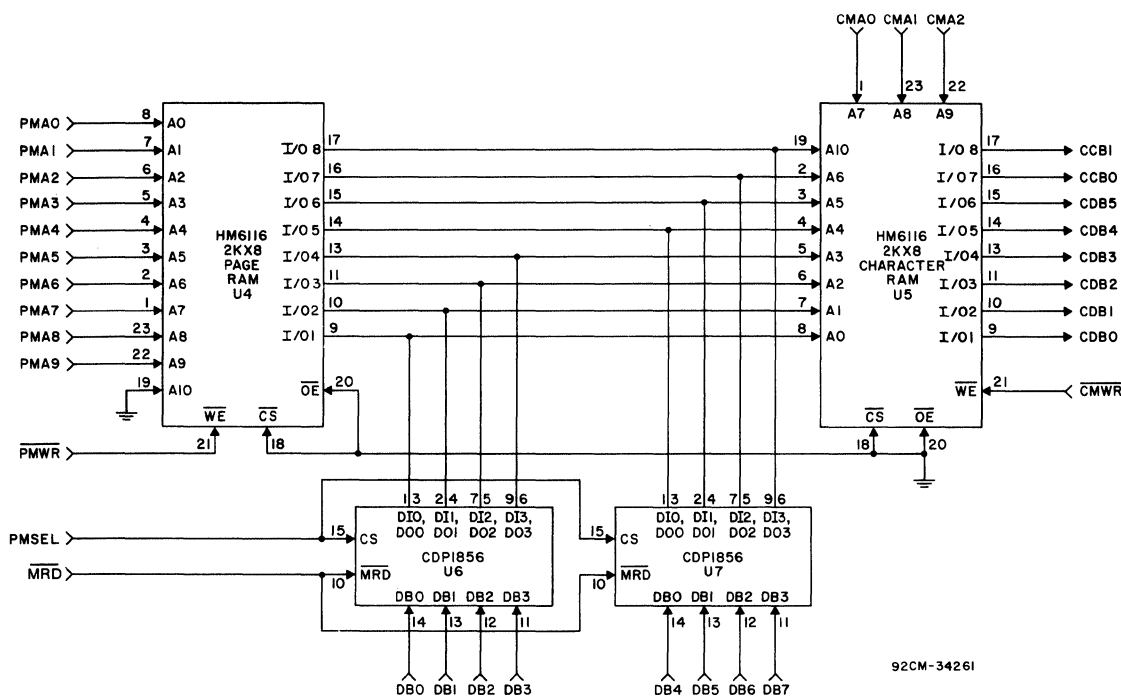


Fig. A-2 - Video-terminal-system schematic diagram (part 2 of 3).

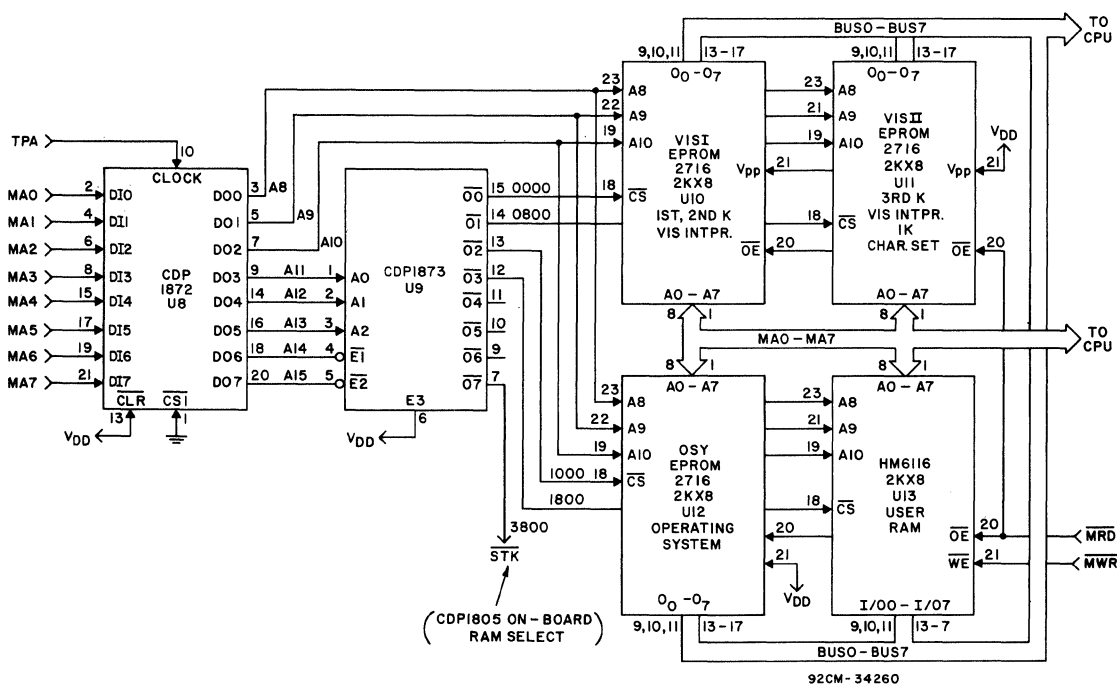


Fig. A-3 - Video-terminal-system schematic diagram (part 3 of 3).

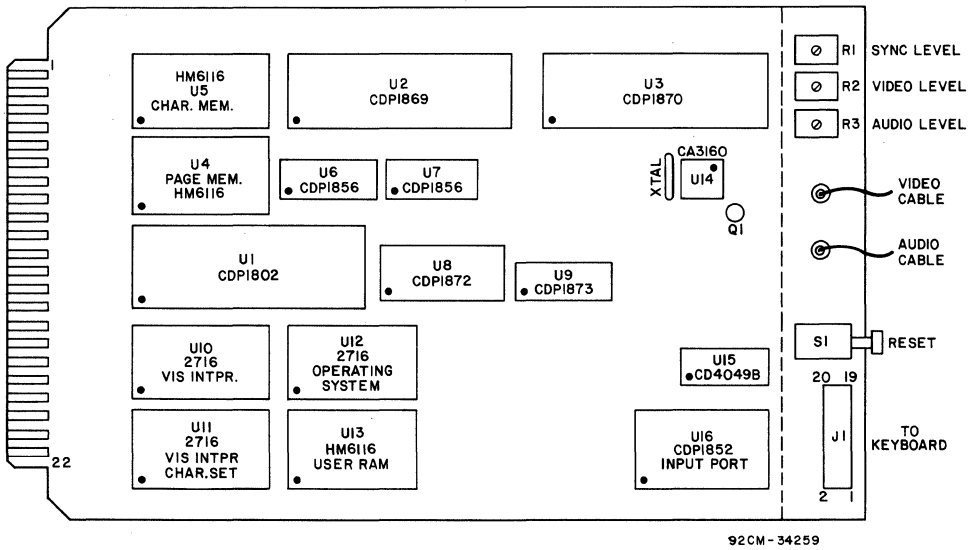


Fig. A-4 - The video terminal constructed on a 4.5 by 7.5-inch CDP18S659 Microboard Breadboard (component side shown).

Appendix B - Memory Map

VIS INTERPRETER 3K BYTES EPROM	0000 0BFF 0C00	<ul style="list-style-type: none"> - ONE CDP1872 - TO LATCH MA8 - MA15 - ONE CDP1873 - TO DECODE A11, A12, A13
STANDARD CHARACTER SET 1K BYTES EPROM	0FFF 1000	
OPERATING SYSTEM EPROM 1K BYTES	13FF 1400	
UNUSED 1K BYTES OF 2K EPROM	17FF 1800	
USER RAM (2K BYTES LESS STACK)	1FBF 1FC0	
STACK-64 BYTES RAM	1FFF 2000	
AVAILABLE 6K BYTES	37FF 3800	
AVAILABLE 2K BYTES	3FBF 3FC0	
MINIMUM INTERPRETER STACK 64 BYTES RAM	3FFF 4000	
NOT DECODED	F3FF F400	
CHARACTER RAM 1K BYTES	F7FF F800	<ul style="list-style-type: none"> - LATCHED AND DECODED BY CDP1869 - CHARACTER RAM USES ONLY 8 BYTES DURING CHAR. MEM. WRITE OPERATIONS. - FIRST OF SCREEN REFRESH RAM MAY BE USED WITHOUT STANDARD INTERPRETER. - ALL UNUSED SCREEN REFRESH RAM IS AVAILABLE TO CPU DURING NON DISPLAY TIME.
SCREEN REFRESH 1K BYTES (NOT USED)	FBFF FC00	
SCREEN REFRESH 960 BYTES RAM	FFBF FFC0	
64 BYTES (NOT USED)	FFFF	

92CM-34265

Fig. B-1 - Memory map.

Appendix C - VIS Interpreter Terms and Operating System Variables

Terms Used in VIS Interpreter Instructions

VX,VY - Sixteen 8-bit variables (VO-VF)
 C(VX) - Contents of the variable specified VX
 PMP - Page-memory pointer
 CMP - Character-memory pointer
 MMP - Main-memory pointer
 ACC - Accumulator
 K - Hex Value
 FLAG - Overflow flag

Changes To Standard VIS Interpreter

ORG = 0000 (VIS Interpreter-EPROM)
 RAM = 1FFF (Stack)
 PROG = 1000 (Operating System-EPROM)

Operating System Equate Statements and Variable Usage

ORG = 1000 - Start of Operating System
 USRAM = 1800 - Start of user RAM
 COLOR = 00 - Color/Gray-Scale byte
 CURS = 0F - VF holds cursor character
 CTRBLK = 0C - VC holds control/blink/reverse video flag;
 V9 holds display character for screen control

Appendix D - Operating System Components

The following is a description of the commands, routines, and subroutines contained in the Operating System. System commands are preceded by an asterisk (*) and may be executed directly from the keyboard, in real time, by following the format indicated in the command descriptions.

Any entry designated as a subroutine may be executed by the user program with the VIS Interpreter INTCAL, A.1, A.0 instruction, using the label of the desired subroutine. After execution of the subroutine, control is returned to the user program at the instruction immediately following the calling statement.

Commands, Routines, Subroutines

- 1 Main Program
- 2 * Memory Read (Routine)
- 3 * Memory Write (Routine)
- 4 * Run (Routine)
- 5 * Clear Screen and Home (Subroutine)
- 6 * Home Cursor (Subroutine)
- 7 * Carriage Return/Line Feed (Subroutine)
- 8 * Cursor Position (Subroutine)
- 9 * Screen Resolution (Subroutine)
- 10 * Control Flag (Subroutine)
- 11 * Set Cursor (Subroutine)
- 12 Error (Routine)
- 13 Load Standard Character Set (Subroutine)
- 14 Input Character (Subroutine)
- 15 ASCII to Hex (Subroutine)
- 16 Display (Subroutine)

ICAN-7032

MAIN PROGRAM (MAIN)

— Initializes system and waits for keyboard entry.

*MEMORY READ ROUTINE (MEMRD)

— Allows user to examine memory at any location (similar to CDS utility routine). The four hex digits of the address field of the command specify the starting memory location, and the four hex digits of the byte count field specify the number (in hex) of bytes to be displayed. The memory contents are displayed in lines up to 16-bytes, preceded by the starting address of the 16-byte block. The lines are further divided into 4-byte hex pairs separated by a space. The command is terminated by a carriage return (CR).

FORMAT — ?M[Address Field] [CR] Δ [Byte Count Field] (CR)

— Address field defaults to zero.

— Byte-count field defaults to one.

— Leading zeros assumed.

— If more than four hex digits are entered, only the last four are used.

— Δ = space

— (CR) = Carriage Return

— [] = optional

EXAMPLE - ?M1000 100

— Allows user to enter hex data into any RAM memory location (Similar to CDS utility routine). The four hex digits of the address field of the command specify the starting memory location. Data is entered into consecutive locations after each two hex digits are typed. One space is allowed between hex pairs. If an odd number of data digits is entered, the last digit is ignored. The command is terminated by a (CR).

FORMAT — !M[Address Field] Δ xx[Δ]xx[Δ]...[.][:]; (CR)

— Address field defaults to zero. Leading zeros assumed. If more than four hex digits are entered, only last four are used.

— xx = Hex digit pair

EXAMPLE — !M3002 12 4A 66AB7F

— Two options are available with the memory write command:

1. A string of data can be extended from line to line by typing a comma just before the normal (CR). A carriage return/line feed (CR/LF) occurs, and more data may be entered as described above.

EXAMPLE — !M110F 12 AA FF,
AB CF DECF 33

2. A string of data can be terminated by typing a semicolon just before the normal (CR). A (CR/LF) occurs, and more data may be entered at a new memory address. The new line must have the format of the !M command, but with the !M omitted.

EXAMPLE — !M2346 9A11 33;
2501 FF AA 67

*RUN ROUTINE (RUN)

— Executes a user program, written in VIS Interpreter language, at the memory location specified by the USRAM equate statement in the source program. (Location 1800, as supplied).

FORMAT — RUN(CR)

*CLEAR SCREEN AND HOME SUBROUTINE (CLSRHM)

— Turns off the display and loads the page memory with either normal or reverse-video null characters. The cursor is positioned at the upper-left corner of the screen (home position) and the display is again turned on.

FORMAT — (CTRL)C
(CTRL) = Control Character

*HOME CURSOR SUBROUTINE (HOMEC)

— Displays the last stored screen characters at the current cursor location, returns the PMP (page-memory pointer) to the upper-left corner of the screen, stores the character at that location, and displays the cursor.

FORMAT — (CTRL)A

*CARRIAGE RETURN/ LINE FEED SUBROUTINE (CRFEED)

— Displays the last stored screen character at the current cursor location (unless last stored character is cursor), returns PMP to left side of screen and down one character row, stores the character at that location, and displays the cursor.

FORMAT — (CR)

*CURSOR POSITION SUBROUTINE (CURSOR)

— The last stored screen character is displayed, the PMP is moved in the direction specified by the command, the character at that location is stored, and the cursor is displayed.

FORMAT — (CTRL)H [BS] - Moves cursor left one character
(CTRL)L [FF] - Moves cursor right one character
(CTRL)J [LF] - Moves cursor down one row
(CTRL)K [VT] - Moves cursor up one row

***SCREEN RESOLUTION
SUBROUTINE**
(RESOLN)

- The character following this command defines the screen resolution:
F = 40 character x 24 row display
H = 20 character x 12 row display

FORMAT — (ESC)-(CTRL)FF[H]
(ESC) = escape key

***CONTROL FLAG
SUBROUTINE**
(CTFLAG)

- The character following this command defines whether control characters will be displayed or suppressed.
Y = Display Control Characters
N = Do Not Display Control Characters

FORMAT — (ESC)(CTRL)BY[N]

***SET CURSOR SUBROUTINE**
(SETCUR)

- If the character following this command is (Y), the cursor will blink at a 0.5-Hz rate.
- If the character is (N), the cursor will not blink.
- If the character is (R), the reverse video flag is set.
- If the character is (S), the reverse video flag is reset.
- If the character is none of the above, (Y,N,R,S), the character itself becomes the new cursor.

FORMAT — (ESC)(CTRL)EY [N] [R] [S] [X]

- Yes, No, Reverse, Standard
- X = any ASCII character other than Y,N,R, or S

ERROR ROUTINE
(ERROR)

- Prints '?', followed by the cursor and sounds a tone if a format error is encountered during a command sequence.

**LOAD STANDARD
CHARACTER SET
SUBROUTINE**
(LDCSET)

- Downloads a standard ASCII character set from EPROM to the second 1K of character RAM. The same EPROM is also used, with an XOR technique, to load the first 1K of character RAM with a reverse video ASCII character set. The MSB of the page-memory data selects either the standard or reverse video characters.
- Upper and lower case characters.
- EPROM is at location 0C00, as supplied.

**INPUT CHARACTER
SUBROUTINE**
(INCHR)

- Waits for EF2 flag to go true and inputs a character from the keyboard to the accumulator (ACC). If the blink flag is set, the current screen character and the cursor are alternately displayed at a 0.5-Hz rate. If the blink flag is reset, only the cursor is displayed.
- If the character in ACC is not a control character, a tone is sounded, the character is displayed at the current cursor location, and the PMP is incremented. The character at that location is stored and the cursor is displayed.
- - If the character in ACC is a control character, the control flag (CTRBLK) is checked. If the control flag is set, a control tone is sounded and the character is displayed as above. If the control flag is reset, a control tone is sounded but the character is not displayed and the cursor position remains unchanged.

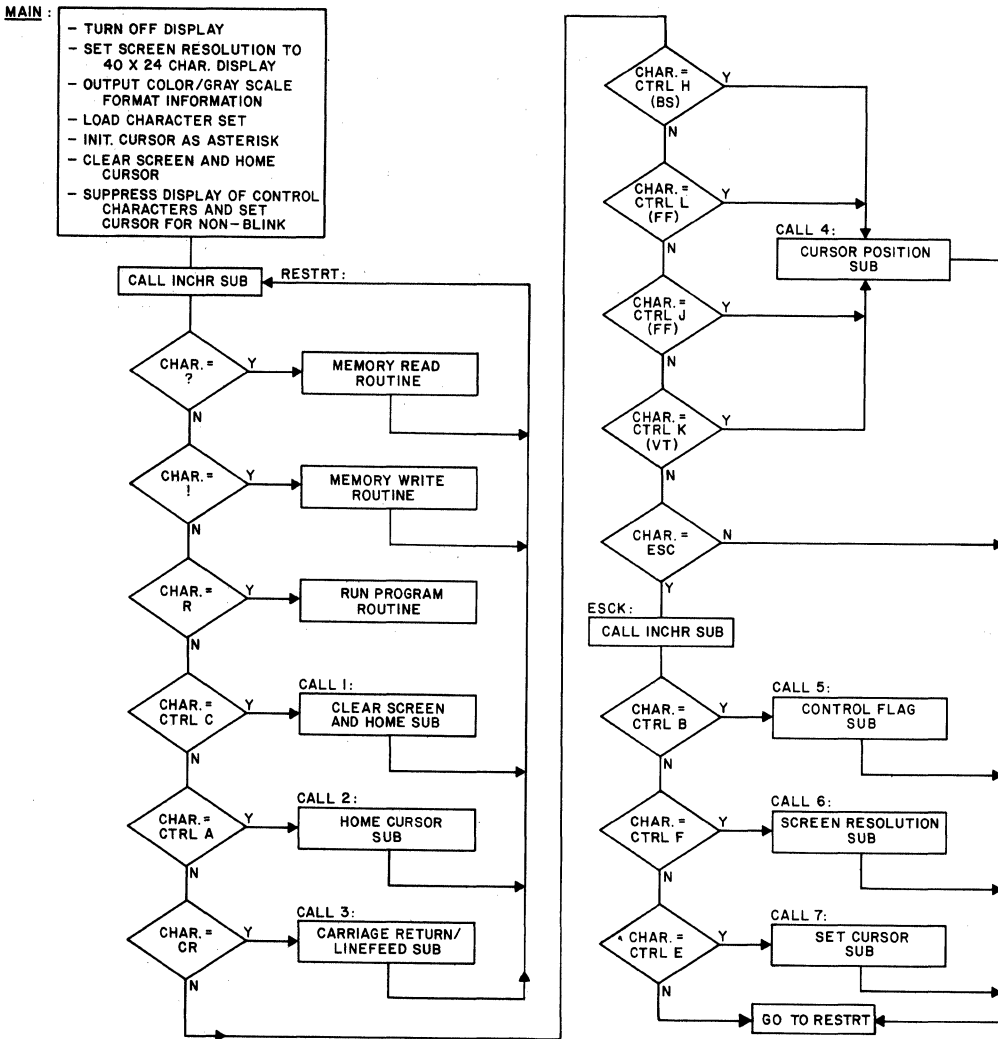
**ASCII TO HEX
SUBROUTINE**
(AS2HEX)

- Checks the ASCII character in ACC for a valid hex digit (0-F). If the character is hex, ACC is loaded with the hex equivalent of the ASCII character and the interpreter overflow flag (FLAG) is set. If the character is not hex, ACC retains the original ASCII character and the interpreter overflow flag is reset.

DISPLAY SUBROUTINE
(DISCUR, DISCHR)

- This subroutine has two entry points:
- DISCUR: ACC is set equal to the cursor. The reverse video flag is checked and ACC is displayed accordingly.
- DISCHR: The reverse video flag is checked and ACC is displayed accordingly.

Appendix E - VIS Operating-System Flowcharts



92CL-34367

Fig. E-1 - VIS Operating System flowchart - main program.

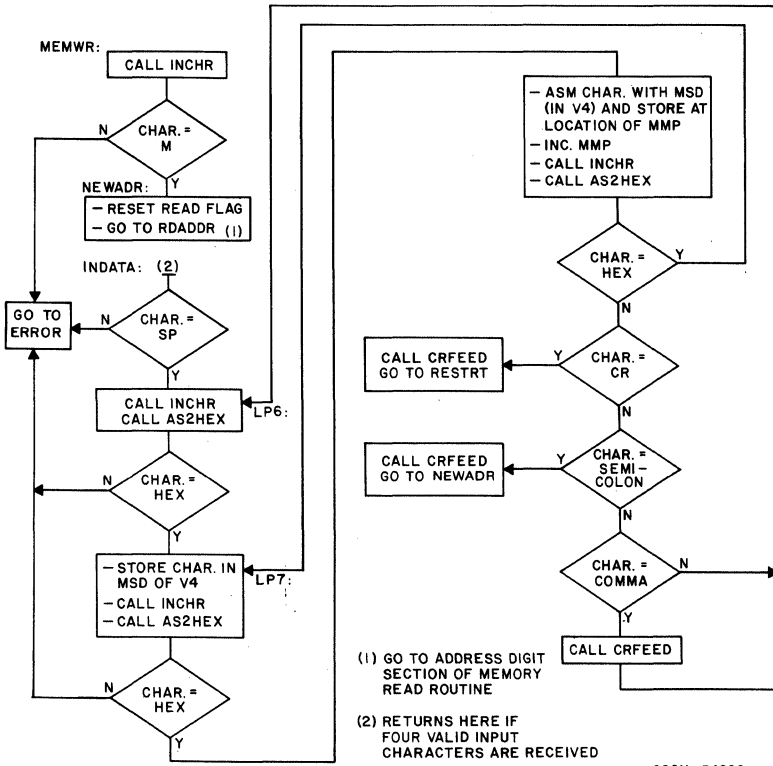


Fig. E-3 - Memory-Write routine.

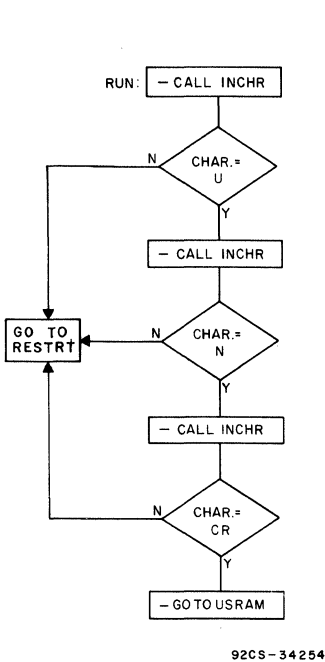


Fig. E-4 - Run routine.

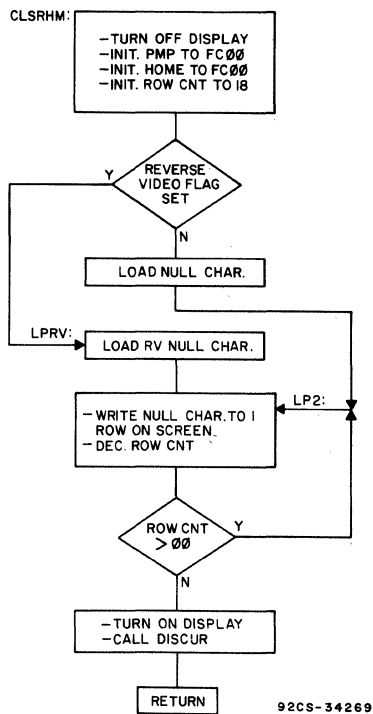


Fig. E-5 - Clear Screen and Home subroutine.

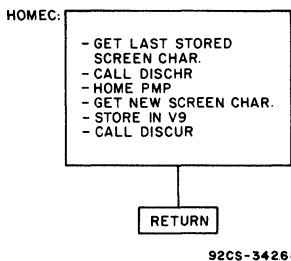


Fig. E-6 - Home-Cursor subroutine.

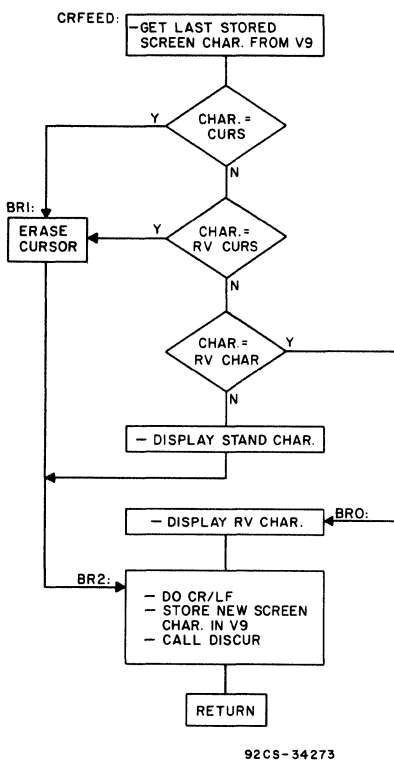


Fig. E-7 - Carriage Return/Line Feed subroutine.

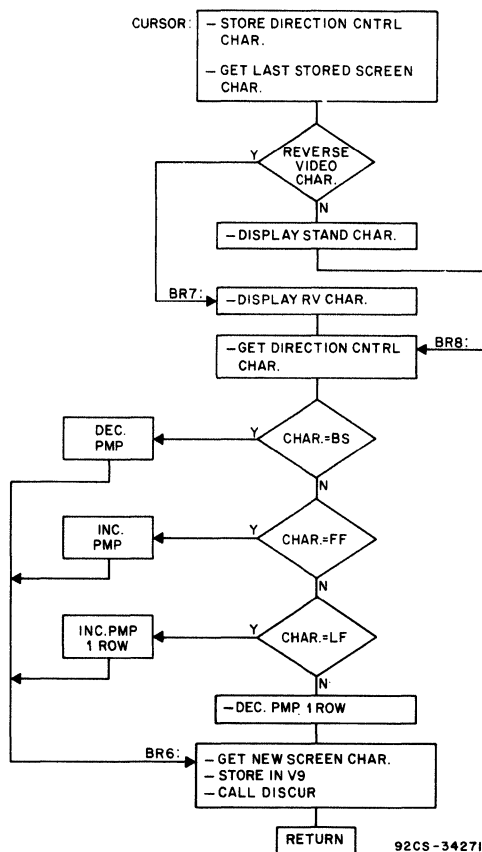


Fig. E-8 - Cursor-Position subroutine.

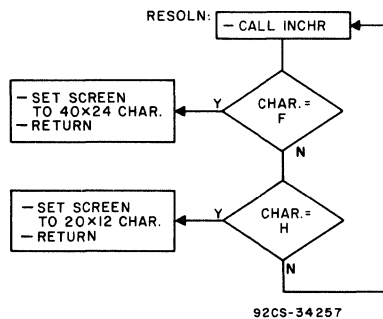


Fig. E-9 - Screen-Resolution subroutine.

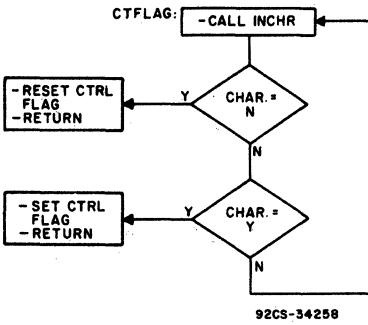


Fig. E-10 - Control-Flag subroutine.

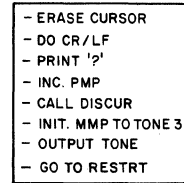


Fig. E-12 - Error routine.

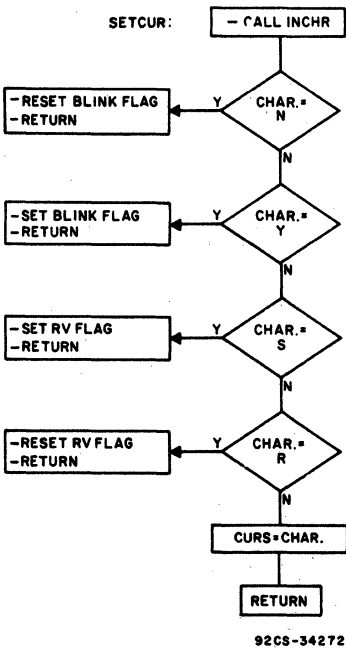


Fig. E-11 - Set-Cursor subroutine.

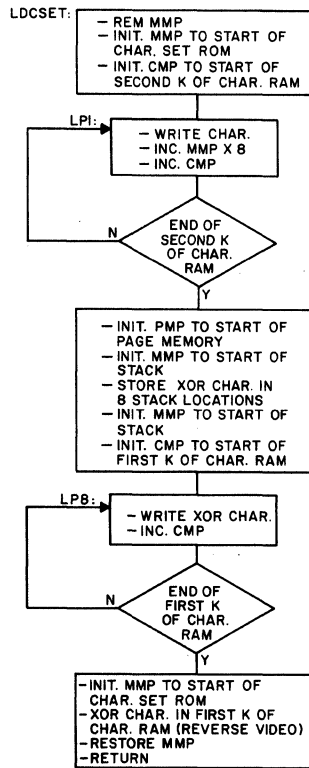


Fig. E-13 - Load Standard Character Set subroutine.

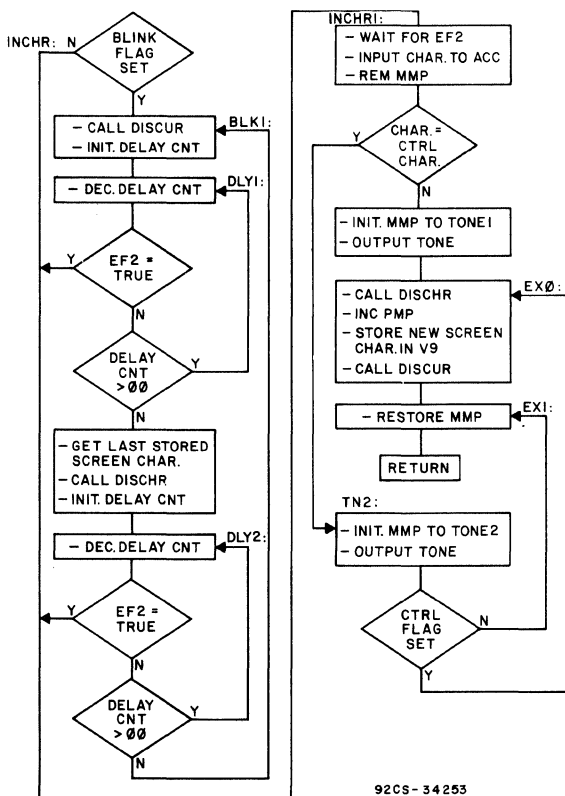
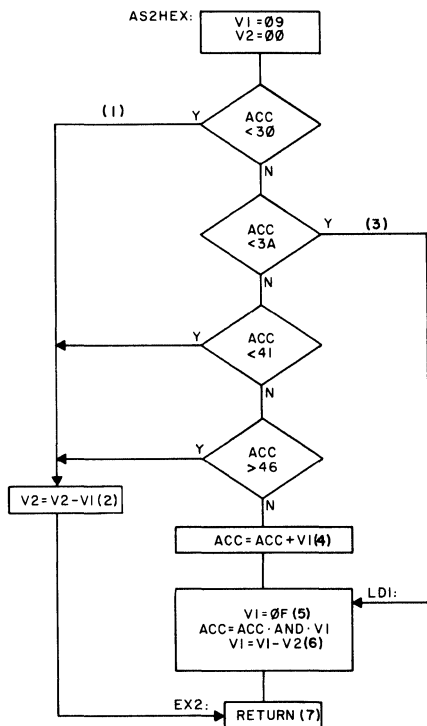


Fig. E-14 - Input Character subroutine.

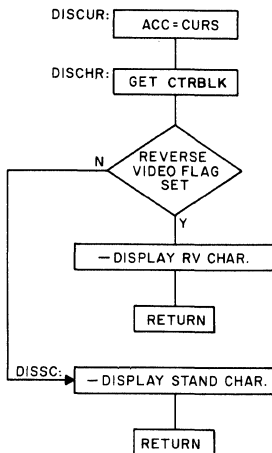
92CS-34253



- (1) CHAR. NOT 0-F
- (2) RESET FLAG
- (3) CHAR. IS 0-9
- (4) CHAR. IS A-F
ADD 9 TO CHAR.
- (5) STRIP OFF UPPER 4 BITS
- (6) SET FLAG
- (7) RETURNS WITH ACC = HEX
EQUIVALENT OF ASCII CHAR.
(0-F) AND FLAG SET, OR
WITH ACC = ORIGINAL ASCII
CHAR. AND FLAG RESET.

92CS-34277

Fig. E-15 - ASCII to Hex subroutine.



92CS-34255

Fig. E-16 - Display subroutine.

Table E-1 - Tone Table

TONE 1:	58	Divide by 88	1 kHz
	4F	88 kHz, Full Ampl.	
	02	2 x 60th TONE ON	
	00		
	80		
TONE 2:	01	Turn Tone OFF	2 kHz
	FF	Stop	
	2C	Divide by 44	
	4F	88 kHz, Full Ampl.	
	02	2 x 60th Tone ON	
TONE 3:	00		0.250 kHz
	80		
	01	Turn Tone OFF	
	FF	Stop	
	58	Divide by 88	
	2F	22 kHz, FULL AMPL.	
	06	6 x 60th Tone ON	
	00		
	80		
	01	Turn Tone OFF	
	FF	Stop	

Appendix F - VIS Operating System Listing

```

!M
0000 ;          0001
0000 ;          0002
0000 ;          0003 ..*****
0000 ;          0004 ..          VIS OPERATING SYSTEM USING
0000 ;          0005 ..          THE VIS INTERPRETER (ASM4)
0000 ;          0006 ..          PROGRAM IS FOR CDP1800-BASED
0000 ;          0007 ..          VIDEO TERMINAL APPLICATION
0000 ;          0008 ..*****
0000 ;          0009
0000 ;          0010
0000 ;          0011          ORG #1000          ..START OF OPERATING SYS
1000 ;          0012          CURS=#0F          ..VF HOLDS CURSOR CHAR
1000 ;          0013          USRAM=#1800        ..START OF USER RAM
1000 ;          0014          COLOR=#00         ..COLOR/GRAYSCALE BYTE
1000 ;          0015          CTRBLK=#0C        ..VC HOLDS CONTROL/BLINK/
1000 ;          0016                               ..REVERSE VIDEO FLAG
1000 ;          0017
1000 ;          0018                               ..V9 HOLDS DISPLAY CHAR
1000 ;          0019                               ..FOR SCREEN CONTROL
1000 ;          0020
1000 ;          0021 ..*****
1000 ;          0022 ..          MAIN PROGRAM
1000 ;          0023 ..*****
1000 ;          0024
1000 ;          0025
1000 FF;          0026          ,#FF          ..1ST BYTE MUST BE FF
1001 63;          0027 MAIN:          ,DISOFF        ..TURN OFF DISPLAY
1002 53;          0028          ,SETFUL        ..SET 40X24 RESOLUTION
1003 0B;          0029          ,LOADAC        ..
1004 00;          0030          ,COLOR        ..OUTPUT COLOR/GRAYSCALE
1005 5F;          0031          ,COLFOR        ..FORMAT INFORMATION
1006 07;          0032          ,INTCAL        ..LOAD STANDARD
1007 10B0;        0033          ,A(LDCSET)      ..CHARACTER SET
1009 0A;          0034          ,LOADVX        ..INIT
100A 0F;          0035          ,CURS          ..CURSOR AS
100B 2A;          0036          ,#2A          ..ASTERISK
100C 07;          0037          ,INTCAL        ..CLEAR SCREEN
100D 10DB;        0038          ,A(CLSRHM)     ..AND HOME CURSOR
100F 0A;          0039          ,LOADVX        ..SUPPRESS CONTROL
1010 0C;          0040          ,CTRBLK        ..CHARS AND SET
1011 00;          0041          ,#00          ..NON-BLINK
1012 07;          0042 RESTRT:        ,INTCAL        ..
1013 112A;        0043          ,A(INCHR)      ..INPUT CHARACTER
1015 40;          0044          ,ACCEQK        ..IF CHAR = ?
1016 3F;          0045          ,#3F          ..GO TO MEMORY
1017 11CB;        0046          ,A(MEMRD)     ..READ ROUTINE
1019 40;          0047          ,ACCEQK        ..IF CHAR = !
101A 21;          0048          ,#21          ..GO TO MEMORY
101B 129D;        0049          ,A(MEMWR)     ..WRITE ROUTINE
101D 40;          0050          ,ACCEQK        ..IF CHAR = R
101E 52;          0051          ,#52          ..GO TO RUN
101F 12FA;        0052          ,A(RUN)        ..PROGRAM ROUTINE
1021 40;          0053          ,ACCEQK        ..IF CHAR=CNTRL C
1022 03;          0054          ,#03          ..(ETX)
1023 1056;        0055          ,A(CALL1)     ..GO TO CALL1
1025 40;          0056          ,ACCEQK        ..IF CHAR=CNTRL A
1026 01;          0057          ,#01          ..(SOH)
1027 105C;        0058          ,A(CALL2)     ..GO TO CALL2
1029 40;          0059          ,ACCEQK        ..
102A 0D;          0060          ,#0D          ..IF CHAR=CR

```

ICAN-7032

```

102B 1062;          0061          ,A(CALL3)          ..GO TO CALL3
102D 40;           0062          ,ACCEQK           ..IF CHAR=CNTRL H
102E 08;           0063          ,#08             ..(BACK SPACE)
102F 106B;        0064          ,A(CALL4)         ..GO TO CALL4
1031 40;           0065          ,ACCEQK           ..IF CHAR=CNTRL L
1032 0C;           0066          ,#0C             ..(FORM FEED)
1033 106B;        0067          ,A(CALL4)         ..GO TO CALL4
1035 40;           0068          ,ACCEQK           ..IF CHAR=CNTRL J
1036 0A;           0069          ,#0A             ..(LINE FEED)
1037 106B;        0070          ,A(CALL4)         ..GO TO CALL4
1039 40;           0071          ,ACCEQK           ..IF CHAR=CNTRL K
103A 0B;           0072          ,#0B             ..(VERT TAB)
103B 106B;        0073          ,A(CALL4)         ..GO TO CALL4
103D 40;           0074          ,ACCEQK           ..
103E 1B;           0075          ,#1B             ..IF CHAR=ESC
103F 1044;        0076          ,A(ESCK)         ..GO TO ESCK
1041 2B;           0077          ,MEMGO           ..
1042 1012;        0078          ,A(RESTRT)       ..GO TO RESTRT
1044 07;           0079  ESCK:     ,INTCAL          ..
1045 112A;        0080          ,A(INCHR)        ..INPUT CHAR
1047 40;           0081          ,ACCEQK           ..IF CHAR=CNTRL B
1048 02;           0082          ,#02             ..(STX)
1049 106E;        0083          ,A(CALL5)         ..GO TO CALL5
104B 40;           0084          ,ACCEQK           ..IF CHAR=CNTRL F
104C 06;           0085          ,#06             ..(ACK)
104D 1074;        0086          ,A(CALL6)         ..GO TO CALL6
104F 40;           0087          ,ACCEQK           ..IF CHAR=CNTRL E
1050 05;           0088          ,#05             ..(ENQ)
1051 107A;        0089          ,A(CALL7)         ..GO TO CALL7
1053 2B;           0090          ,MEMGO           ..
1054 1012;        0091          ,A(RESTRT)       ..GO TO RESTRT
1056 07;           0092  CALL1:   ,INTCAL          ..CALL CLEAR SCREEN
1057 10DB;        0093          ,A(CLSRHM)       ..AND HOME SUBROUTINE
1059 2B;           0094          ,MEMGO           ..
105A 1012;        0095          ,A(RESTRT)       ..GO TO RESTRT
105C 07;           0096  CALL2:   ,INTCAL          ..CALL HOME
105D 131B;        0097          ,A(HOMEC)        ..CURSOR SUBROUTINE
105F 2B;           0098          ,MEMGO           ..
1060 1012;        0099          ,A(RESTRT)       ..GO TO RESTRT
1062 07;           0100  CALL3:   ,INTCAL          ..CALL CR/LF
1063 10FE;        0101          ,A(CRFEED)       ..SUBROUTINE
1065 2B;           0102          ,MEMGO           ..
1066 1012;        0103          ,A(RESTRT)       ..GO TO RESTRT
1068 07;           0104  CALL4:   ,INTCAL          ..CALL CURSOR
1069 1344;        0105          ,A(CURSOR)       ..POSITION SUBROUTINE
106B 2B;           0106          ,MEMGO           ..
106C 1012;        0107          ,A(RESTRT)       ..GO TO RESTRT
106E 07;           0108  CALL5:   ,INTCAL          ..CALL CONTROL
106F 132B;        0109          ,A(CTFLAG)       ..FLAG SUBROUTINE
1071 2B;           0110          ,MEMGO           ..
1072 1012;        0111          ,A(RESTRT)       ..GO TO RESTRT
1074 07;           0112  CALL6:   ,INTCAL          ..CALL SCREEN
1075 1377;        0113          ,A(RESOLN)       ..RESOLUTION SUBROUTINE
1077 2B;           0114          ,MEMGO           ..
107B 1012;        0115          ,A(RESTRT)       ..GO TO RESTRT
107A 07;           0116  CALL7:   ,INTCAL          ..CALL SET
107B 1389;        0117          ,A(SETCUR)       ..CURSOR SUBROUTINE
107D 2B;           0118          ,MEMGO           ..
107E 1012;        0119          ,A(RESTRT)       ..GO TO RESTRT
1080 ;           0120
1080 ;           0121
1080 ;           0122

```

```

1080 ;          0123 ..*****
1080 ;          0124 ..          LOAD STANDARD CHARACTER SET
1080 ;          0125 ..          SUBROUTINE (V1,V2,VA,VB)
1080 ;          0126 ..*****
1080 ;          0127
1080 ;          0128
1080 25;          0129 LDCSET: ,MMP1VX          ..
1081 0A;          0130          ,#0A          ..
1082 24;          0131          ,MMP0VX          ..
1083 0B;          0132          ,#0B          ..REM MMP
1084 0E;          0133          ,LDMMP          ..INIT MMP
1085 0C00;        0134          ,#0C00          ..TO CHAR SET ROM
1087 10;          0135          ,LDCMP          ..INIT CPM TO 2D K
1088 80;          0136          ,#80          ..OF CHAR RAM
1089 0A;          0137          ,LOADVX          ..
108A 01;          0138          ,#01          ..INIT
108B 00;          0139          ,#00          ..WRTCHR VARIABLES
108C 58;          0140 LP1:  ,WRTCHR          ..WRITE CHAR
108D 01;          0141          ,#01          ..FROM PAGE TO
108E 01;          0142          ,#01          ..CHAR MEMORY
108F 49;          0143          ,MMPINC          ..
1090 49;          0144          ,MMPINC          ..
1091 49;          0145          ,MMPINC          ..
1092 49;          0146          ,MMPINC          ..
1093 49;          0147          ,MMPINC          ..
1094 49;          0148          ,MMPINC          ..
1095 49;          0149          ,MMPINC          ..
1096 49;          0150          ,MMPINC          ..INC MMP 8 TIMES
1097 47;          0151          ,CMPINC          ..INC CMP
1098 1B;          0152          ,CMPAC          ..
1099 42;          0153          ,ACCGTK          ..CHECK FOR END
109A 80;          0154          ,#80          ..OF 2D K
109B 108C;        0155          ,A(LP1)          ..OF CHAR RAM
109D 68;          0156          ,INIT          ..INIT PMP
109E 0E;          0157          ,LDMMP          ..INIT MMP
109F 1FC0;        0158          ,#1FC0          ..TO STACK
10A1 0B;          0159 LPA:  ,LOADAC          ..STORE XOR
10A2 3F;          0160          ,#3F          ..CHAR ON STACK
10A3 0D;          0161          ,ACCMEM          ..
10A4 49;          0162          ,MMPINC          ..INC MMP
10A5 13;          0163          ,MMPOAC          ..CHECK MMP
10A6 41;          0164          ,ACCLTK          ..FOR 8 BYTES
10A7 C8;          0165          ,#C8          ..IF < #C8
10A8 10A1;        0166          ,A(LPA)          ..LOOP
10AA 0E;          0167          ,LDMMP          ..INIT MMP
10AB 1FC0;        0168          ,#1FC0          ..TO STACK
10AD 10;          0169          ,LDCMP          ..INIT CMP TO 1ST K
10AE 00;          0170          ,#00          ..OF CHAR MEMORY
10AF 58;          0171 LPB:  ,WRTCHR          ..STORE XOR
10B0 01;          0172          ,#01          ..CHAR
10B1 01;          0173          ,#01          ..IN CHAR RAM
10B2 47;          0174          ,CMPINC          ..INC CMP
10B3 1B;          0175          ,CMPAC          ..CHECK FOR END OF
10B4 41;          0176          ,ACCLTK          ..1ST K OF CHAR RAM
10B5 80;          0177          ,#80          ..IF < #80
10B6 10AF;        0178          ,A(LPB)          ..LOOP
10B8 0E;          0179          ,LDMMP          ..INIT MMP TO
10B9 0C00;        0180          ,#0C00          ..CHAR SET ROM
10BB 0A;          0181          ,LOADVX          ..
10BC 02;          0182          ,#02          ..
10BD 00;          0183          ,#00          ..INIT V2
10BE 06;          0184 LPC:  ,VXACC          ..SET C(PMP)
10BF 02;          0185          ,#02          ..EQUAL TO V2

```

ICAN-7032

```

10C0 4D;          0186          ,DISAC          ..
10C1 61;          0187          ,MOVBIT         ..XOR CHARS IN
10C2 01;          0188          ,#01           ..1ST K OF
10C3 01;          0189          ,#01           ..CHAR RAM
10C4 49;          0190          ,MMPINC        ..
10C5 49;          0191          ,MMPINC        ..
10C6 49;          0192          ,MMPINC        ..
10C7 49;          0193          ,MMPINC        ..
10C8 49;          0194          ,MMPINC        ..
10C9 49;          0195          ,MMPINC        ..
10CA 49;          0196          ,MMPINC        ..INC MMP
10CB 49;          0197          ,MMPINC        ..8 TIMES
10CC 01;          0198          ,VXINC         ..
10CD 02;          0199          ,#02           ..INC V2
10CE 3E;          0200          ,VXLTK         ..CHECK FOR END OF
10CF 02;          0201          ,#02           ..1ST K OF CHAR RAM
10D0 80;          0202          ,#80           ..IF < #80
10D1 10E;        0203          ,A(LPC)        ..LOOP
10D3 2A;          0204          ,VXMMP1       ..
10D4 0A;          0205          ,#0A           ..
10D5 29;          0206          ,VXMMP0       ..
10D6 0B;          0207          ,#0B           ..RESTORE MMP
10D7 0B;          0208          ,INTRTN       ..RETURN
10D8 ;           0209
10D8 ;           0210
10D8 ;           0211
10D8 ;           0212 ..*****
10D8 ;           0213 ..          CLEAR SCREEN AND HOME
10D8 ;           0214 ..          SUBROUTINE (V1,V8,V9)
10D8 ;           0215 ..*****
10D8 ;           0216
10D8 ;           0217
10D8 63;          0218 CLSRHM: ,DISOFF       ..TURN OFF DISPLAY
10D9 68;          0219          ,INIT          ..INIT PMP & HOME
10DA 0A;          0220          ,LOADVX       ..
10DB 01;          0221          ,#01           ..
10DC 18;          0222          ,#18           ..INIT ROW CNT
10DD 0A;          0223          ,LOADVX       ..
10DE 08;          0224          ,#08           ..
10DF 20;          0225          ,#20           ..INIT V8
10E0 2E;          0226          ,VXYAND       ..STRIP OFF
10E1 08;          0227          ,#08           ..UNUSED
10E2 0C;          0228          ,CTRBK        ..BITS OF CTRBLK
10E3 3F;          0229          ,VXGTK        ..
10E4 08;          0230          ,#08           ..
10E5 00;          0231          ,#00           ..CHECK REVERSE
10E6 10ED;        0232          ,A(LPRV)      ..VIDED FLAG
10E8 0B;          0233          ,LOADAC       ..
10E9 80;          0234          ,#80           ..LOAD NULL CHAR
10EA 2B;          0235          ,MEM60        ..
10EB 10EF;        0236          ,A(LP3)       ..GO TO LP3
10ED 0B;          0237 LPRV: ,LOADAC       ..LOAD RV
10EE 00;          0238          ,#00           ..NULL CHAR
10EF 05;          0239 LP3: ,ACCVX        ..
10F0 09;          0240          ,#09           ..INIT V9
10F1 52;          0241 LP2: ,FLROW1      ..WRITE 1 ROW
10F2 02;          0242          ,VXDEC        ..
10F3 01;          0243          ,#01           ..DEC ROW CTN
10F4 3F;          0244          ,VXGTK        ..IF ROW CTN
10F5 01;          0245          ,#01           ..IS NOT 0
10F6 00;          0246          ,#00           ..
10F7 10F1;        0247          ,A(LP2)       ..GO TO LP2

```

```

10F9 62;          0248          ,DISON          ..TURN ON DISPLAY
10FA 07;          0249          ,INTCAL        ..CALL DISPLAY
10FB 13BB;        0250          ,A(DISCUR)    ..CURSOR SUB
10FD 08;          0251          ,INTRTN       ..RETURN
10FE ;           0252
10FE ;           0253
10FE ;           0254
10FE ;           0255 ..*****
10FE ;           0256 ..          CARRIAGE RETURN/LINEFEED
10FE ;           0257 ..          SUBROUTINE (V1,V9,VF)
10FE ;           0258 ..*****
10FE ;           0259
10FE ;           0260
10FE 06;          0261 CRFEED: ,VXACC        ..GET LAST STORED
10FF 09;          0262          ,#09          ..SCREEN CHAR
1100 3A;          0263          ,VXEQAC       ..CHECK FOR
1101 0F;          0264          ,CURS         ..STANDARD
1102 1122;        0265          ,A(BR1)       ..CURSOR
1104 0A;          0266          ,LOADVX       ..
1105 01;          0267          ,#01          ..
1106 7F;          0268          ,#7F         ..INIT V1
1107 34;          0269          ,ACCAND       ..SET MSB
1108 01;          0270          ,#01         ..TO 0
1109 3A;          0271          ,VXEQAC       ..CHECK
110A 0F;          0272          ,CURS         ..FOR RV
110B 1122;        0273          ,A(BR1)       ..CURSOR
110D 06;          0274          ,VXACC        ..RELOAD OLD
110E 09;          0275          ,#09         ..SCREEN CHAR
110F 41;          0276          ,ACCLTK       ..
1110 80;          0277          ,#80         ..CHECK MSB
1111 1117;        0278          ,A(BR0)       ..DISPLAY
1113 6B;          0279          ,DISAWC       ..STANDARD CHAR
1114 2B;          0280          ,MEMGO        ..
1115 1118;        0281          ,A(BR2)       ..GO TO BR2
1117 4D;          0282 BR0:  ,DISAC        ..DISPLAY RV CHAR
1118 6A;          0283 BR2:  ,CRRET        ..
1119 44;          0284          ,ROWINC       ..
111A 01;          0285          ,#01         ..DO CR/LF
111B 1D;          0286          ,RDPMP        ..GET NEW SCREEN
111C 05;          0287          ,ACCVX        ..CHAR AND
111D 09;          0288          ,#09         ..STORE IN V9
111E 07;          0289          ,INTCAL       ..CALL DISPLAY
111F 13BB;        0290          ,A(DISCUR)    ..CURSOR SUB
1121 08;          0291          ,INTRTN       ..RETURN
1122 0B;          0292 BR1:  ,LOADAC       ..LOAD NULL
1123 00;          0293          ,#00         ..CHARACTER
1124 07;          0294          ,INTCAL       ..CALL DISPLAY
1125 13BD;        0295          ,A(DISCHR)    ..CHAR SUB
1127 2B;          0296          ,MEMGO        ..
1128 1118;        0297          ,A(BR2)       ..GO TO BR2
112A ;           0298
112A ;           0299
112A ;           0300
112A ;           0301 ..*****
112A ;           0302 ..          INPUT CHARACTER SUBROUTINE
112A ;           0303 ..          (V2,V7,V9,VA,VB,VC,VD)
112A ;           0304 ..*****
112A ;           0305
112A ;           0306
112A 0A;          0307 INCHR: ,LOADVX       ..
112B 02;          0308          ,#02         ..
112C 10;          0309          ,#10         ..INIT V2
112D 2E;          0310          ,VXYAND       ..STRIP OFF

```

ICAN-7032

112E 02;	0311	,#02	..UNUSED
112F 0C;	0312	,CTRBK	..BITS OF CTRBLK
1130 3D;	0313	,VXEQK	..
1131 02;	0314	,#02	..
1132 00;	0315	,#00	..CHECK
1133 115A;	0316	,A(INCHR1)	..BLINK FLAG
1135 07;	0317	BLK1: ,INTCAL	..CALL DISPLAY
1136 13BB;	0318	,A(DISCUR)	..CURSOR SUB
1138 0A;	0319	,LOADVX	..
1139 07;	0320	,#07	..INIT
113A FF;	0321	,#FF	..DELAY COUNT
113B 02;	0322	DLY1: ,VXDEC	..DEC
113C 07;	0323	,#07	..DELAY COUNT
113D 20;	0324	,BRINP	..CHECK
113E 115A;	0325	,A(INCHR1)	..KEYBOARD
1140 3F;	0326	,VXGTK	..
1141 07;	0327	,#07	..
1142 00;	0328	,#00	..CHECK
1143 113B;	0329	,A(DLY1)	..DELAY COUNT
1145 06;	0330	,VXACC	..DISPLAY OLD
1146 09;	0331	,#09	..SCREEN CHAR
1147 07;	0332	,INTCAL	..CALL DISPLAY
1148 13BD;	0333	,A(DISCHR)	..CHAR SUB
114A 0A;	0334	,LOADVX	..
114B 07;	0335	,#07	..INIT
114C FF;	0336	,#FF	..DELAY COUNT
114D 02;	0337	DLY2: ,VXDEC	..DEC
114E 07;	0338	,#07	..DELAY COUNT
114F 20;	0339	,BRINP	..CHECK
1150 115A;	0340	,A(INCHR1)	..KEYBOARD
1152 3F;	0341	,VXGTK	..
1153 07;	0342	,#07	..
1154 00;	0343	,#00	..CHECK
1155 114D;	0344	,A(DLY2)	..DELAY COUNT
1157 2B;	0345	,MEMGO	..
1158 1135;	0346	,A(BLK1)	..GO TO BLK1
115A 57;	0347	INCHR1: ,CHRIN	..WAIT FOR EF2;
115B 25;	0348	,MMP1VX	..CHAR INTO ACC
115C 0A;	0349	,#0A	..
115D 24;	0350	,MMP0VX	..
115E 0B;	0351	,#0B	..REM MMP
115F 41;	0352	,ACCLTK	..
1160 20;	0353	,#20	..CHECK FOR
1161 1179;	0354	,A(TN2)	..CONTROL CHAR
1163 0E;	0355	,LDMMP	..
1164 13CC;	0356	,A(TONE1)	..
1166 4C;	0357	,TONE	..TURN ON TONE1
1167 07;	0358	EXO: ,INTCAL	..CALL DISPLAY
1168 13BD;	0359	,A(DISCHR)	..CHAR SUB
116A 05;	0360	,ACCVX	..STORE ACC
116B 0D;	0361	,#0D	..TEMP
116C 43;	0362	,PMPINC	..
116D 01;	0363	,#01	..INC PMP
116E 1D;	0364	,RDPMP	..GET NEXT SCREEN
116F 05;	0365	,ACCVX	..CHAR AND
1170 09;	0366	,#09	..STORE IN V9
1171 07;	0367	,INTCAL	..CALL DISPLAY
1172 13BB;	0368	,A(DISCUR)	..CURSOR SUB
1174 06;	0369	,VXACC	..
1175 0D;	0370	,#0D	..RESTORE ACC
1176 2B;	0371	,MEMGO	..
1177 1188;	0372	,A(EX1)	..GO TO EXIT

```

1179 0E;          0373 TN2:      ,LDMMP          ..
117A 13D3;       0374          ,A(TONE2)       ..
117C 4C;         0375          ,TONE           ..TURN ON TONE2
117D 0A;         0376          ,LOADVX        ..
117E 02;         0377          ,#02           ..
117F 01;         0378          ,#01           ..INIT V2
1180 2E;         0379          ,VXYAND        ..STRIP OFF
1181 02;         0380          ,#02           ..UNUSED
1182 0C;         0381          ,CTRBLK        ..BITS OF CTRBLK
1183 3D;         0382          ,VXEQK        ..
1184 02;         0383          ,#02           ..
1185 01;         0384          ,#01           ..CHECK
1186 1167;       0385          ,A(EX0)        ..CONTROL FLAG
1188 2A;         0386 EX1:     ,VXMMP1        ..
1189 0A;         0387          ,#0A           ..
118A 29;         0388          ,VXMMP0        ..RESTORE MMP
118B 0B;         0389          ,#0B           ..RETURNS WITH
118C 0B;         0390          ,INTRTN        ..CHAR IN ACC
118D ;           0391
118D ;           0392
118D ;           0393
118D ;           0394 ..*****
118D ;           0395 ..                ASCII TO HEX SUBROUTINE
118D ;           0396 ..                (V1,V2)
118D ;           0397 ..*****
118D ;           0398
118D ;           0399
118D 0A;         0400 AS2HEX: ,LOADVX          ..
118E 01;         0401          ,#01           ..
118F 09;         0402          ,#09           ..
1190 0A;         0403          ,LOADVX        ..
1191 02;         0404          ,#02           ..
1192 00;         0405          ,#00           ..INIT FLAG VARIABLES
1193 41;         0406          ,ACCLTK        ..
1194 30;         0407          ,#30           ..IF CHAR IS NOT 0-F
1195 11AE;       0408          ,A(LD2)        ..GO TO LD2
1197 41;         0409          ,ACCLTK        ..
1198 3A;         0410          ,#3A           ..IF CHAR IS 0-9
1199 11A5;       0411          ,A(LD1)        ..GO TO LD1
119B 41;         0412          ,ACCLTK        ..
119C 41;         0413          ,#41           ..
119D 11AE;       0414          ,A(LD2)        ..
119F 42;         0415          ,ACCBTK        ..
11A0 46;         0416          ,#46           ..IF CHAR IS NOT A-F
11A1 11AE;       0417          ,A(LD2)        ..GO TO LD2
11A3 36;         0418          ,ACCADD        ..IF CHAR IS A-F
11A4 01;         0419          ,#01           ..ADD 9 TO ACC
11A5 0A;         0420 LD1:    ,LOADVX          ..
11A6 01;         0421          ,#01           ..
11A7 0F;         0422          ,#0F           ..
11A8 34;         0423          ,ACCAND        ..STRIP OFF
11A9 01;         0424          ,#01           ..UPPER 4 BITS
11AA 31;         0425          ,VXYSUB        ..
11AB 01;         0426          ,#01           ..
11AC 02;         0427          ,#02           ..SET FLAG
11AD 0B;         0428          ,INTRTN        ..RETURN
11AE 31;         0429 LD2:    ,VXYSUB          ..
11AF 02;         0430          ,#02           ..
11B0 01;         0431          ,#01           ..RESET FLAG
11B1 0B;         0432          ,INTRTN        ..RETURNS WITH ACC=HEX EQUIV
11B2 ;           0433          ..OF ASCII CHAR (0-F) AND
11B2 ;           0434          ..FLAG SET OR WITH ORIG

```


ICAN-7032

```

11B2 ;          0435          ..CHAR IN ACC AND FLAG RESET
11B2 ;          0436
11B2 ;          0437
11B2 ;          0438
11B2 ;          0439 ..*****
11B2 ;          0440 ..          ERROR ROUTINE
11B2 ;          0441 ..*****
11B2 ;          0442
11B2 ;          0443
11B2 0B;        0444 ERROR:  ,LOADAC          ..
11B3 00;        0445          ,#00          ..ERASE CURSOR
11B4 07;        0446          ,INTCAL          ..CALL SCREEN
11B5 13BD;      0447          ,A(DISCHR)       ..CHAR SUB
11B7 6A;        0448          ,CRRET          ..
11B8 44;        0449          ,ROWINC          ..
11B9 01;        0450          ,#01          ..DO CR/LF
11BA 0B;        0451          ,LOADAC          ..? INTO
11BB 3F;        0452          ,#3F          ..ACC
11BC 07;        0453          ,INTCAL          ..CALL DISPLAY
11BD 13BD;      0454          ,A(DISCHR)       ..CHAR SUB
11BF 43;        0455          ,PMPINC          ..
11C0 01;        0456          ,#01          ..INC PMP
11C1 07;        0457          ,INTCAL          ..CALL DISPLAY
11C2 13BB;      0458          ,A(DISCUR)       ..CURSOR SUB
11C4 0E;        0459          ,LDMMP          ..
11C5 13DA;      0460          ,A(TONE3)        ..
11C7 4C;        0461          ,TONE          ..TURN ON TONE3
11C8 2B;        0462          ,MEMGO          ..
11C9 1012;      0463          ,A(RESTRT)       ..DONE
11CB ;          0464
11CB ;          0465
11CB ;          0466
11CB ;          0467 ..*****
11CB ;          0468 ..          MEMORY READ ROUTINE
11CB ;          0469 ..          (V1,V2,V3,V4,V5,V6,VA,VB,VE)
11CB ;          0470 ..*****
11CB ;          0471
11CB ;          0472
11CB 07;        0473 MEMRD:  ,INTCAL          ..
11CC 112A;      0474          ,A(INCHR)        ..INPUT CHAR
11CE 40;        0475          ,ACCEQK          ..
11CF 4D;        0476          ,#4D          ..
11D0 11D5;      0477          ,A(RDO)          ..CHECK FOR M
11D2 2B;        0478          ,MEMGO          ..
11D3 11B2;      0479          ,A(ERROR)        ..IF NOT GO TO ERROR
11D5 0A;        0480 RDO:    ,LOADVX          ..
11D6 0E;        0481          ,#0E          ..
11D7 01;        0482          ,#01          ..SET READ FLAG
11D8 0E;        0483 RDADDR: ,LDMMP          ..
11D9 00;        0484          ,#00          ..
11DA 00;        0485          ,#00          ..INIT MMP
11DB 0A;        0486          ,LOADVX          ..
11DC 03;        0487          ,#03          ..
11DD 00;        0488          ,#00          ..
11DE 0A;        0489          ,LOADVX          ..
11DF 04;        0490          ,#04          ..
11E0 00;        0491          ,#00          ..INIT BYTE COUNT
11E1 07;        0492 RD1:   ,INTCAL          ..
11E2 112A;      0493          ,A(INCHR)        ..INPUT CHAR
11E4 40;        0494          ,ACCEQK          ..
11E5 0D;        0495          ,#0D          ..
11E6 120E;      0496          ,A(RDFLAG)       ..CHECK FOR CR

```

```

11E8 07;          0497          ,INTCAL          ..
11E9 11BD;       0498          ,A(AS2HEX)      ..
11EB 40;         0499          ,ACCEQK         ..
11EC 20;         0500          ,#20            ..
11ED 120E;       0501          ,A(RDFLAG)      ..CHECK FOR SPACE
11EF 66;         0502          ,BNFLAG         ..
11F0 11B2;       0503          ,A(ERROR)       ..CHECK FOR HEX
11F2 05;         0504          ,ACCVX          ..STORE NEW
11F3 05;         0505          ,#05            ..CHAR TEMP
11F4 14;         0506          ,MMP1AC         ..GET OLD MSD,
11F5 39;         0507          ,ACCSHL         ..
11F6 39;         0508          ,ACCSHL         ..
11F7 39;         0509          ,ACCSHL         ..
11F8 39;         0510          ,ACCSHL         ..SHIFT LEFT,AND
11F9 05;         0511          ,ACCVX          ..STORE
11FA 01;         0512          ,#01            ..IN V1
11FB 13;         0513          ,MMPOAC         ..GET OLD LSD
11FC 38;         0514          ,ACCSHR         ..
11FD 38;         0515          ,ACCSHR         ..
11FE 38;         0516          ,ACCSHR         ..
11FF 38;         0517          ,ACCSHR         ..SHIFT RIGHT,AND
1200 33;         0518          ,ACCOR         ..
1201 01;         0519          ,#01            ..ASM WITH MSD
1202 16;         0520          ,ACMMP1         ..STORE IN MMP.1
1203 13;         0521          ,MMPOAC         ..GET OLD LSD
1204 39;         0522          ,ACCSHL         ..
1205 39;         0523          ,ACCSHL         ..
1206 39;         0524          ,ACCSHL         ..
1207 39;         0525          ,ACCSHL         ..SHIFT LEFT,AND
1208 33;         0526          ,ACCOR         ..
1209 05;         0527          ,#05            ..ASM WITH NEW CHAR
120A 15;         0528          ,ACMMP0         ..STORE IN MMP.0
120B 28;         0529          ,MEMGO         ..
120C 11E1;       0530          ,A(RD1)         ..GO TO RD1
120E 3D;         0531 RDFLAG: ,VXEQK         ..
120F 0E;         0532          ,#0E            ..
1210 00;         0533          ,#00            ..CHECK READ FLAG
1211 12AD;       0534          ,A(INDATA)      ..GO TO INDATA
1213 40;         0535          ,ACCEQK         ..
1214 0D;         0536          ,#0D            ..
1215 1245;       0537          ,A(OUTR)        ..CHECK FOR CR
1217 07;         0538 RD2:  ,INTCAL         ..
1218 112A;       0539          ,A(INCHR)       ..INPUT NEXT CHAR
121A 40;         0540          ,ACCEQK         ..
121B 0D;         0541          ,#0D            ..IF CR,GO TO
121C 1245;       0542          ,A(OUTR)        ..OUTPUT ROUTIMNE
121E 07;         0543          ,INTCAL         ..
121F 11BD;       0544          ,A(AS2HEX)      ..
1221 66;         0545          ,BNFLAG         ..
1222 11B2;       0546          ,A(ERROR)       ..CHECK FOR HEX
1224 05;         0547          ,ACCVX          ..STORE NEW
1225 05;         0548          ,#05            ..CHAR TEMP
1226 06;         0549          ,VXACC         ..
1227 03;         0550          ,#03            ..GET OLD MSD,
1228 39;         0551          ,ACCSHL         ..
1229 39;         0552          ,ACCSHL         ..
122A 39;         0553          ,ACCSHL         ..
122B 39;         0554          ,ACCSHL         ..SHIFT LEFT,AND
122C 05;         0555          ,ACCVX          ..STORE
122D 03;         0556          ,#03            ..BACK IN V3
122E 06;         0557          ,VXACC         ..
122F 04;         0558          ,#04            ..GET OLD LSD,

```

ICAN-7032

```

1230 38;          0559          ,ACCSHR          ..
1231 38;          0560          ,ACCSHR          ..
1232 38;          0561          ,ACCSHR          ..
1233 38;          0562          ,ACCSHR          ..SHIFT RIGHT,AND
1234 33;          0563          ,ACCOR          ..
1235 03;          0564          ,#03            ..ASM WITH MSD
1236 05;          0565          ,ACCVX          ..
1237 03;          0566          ,#03            ..STORE IN V3
1238 06;          0567          ,VXACC          ..
1239 04;          0568          ,#04            ..GET OLD LSD
123A 39;          0569          ,ACCSHL          ..
123B 39;          0570          ,ACCSHL          ..
123C 39;          0571          ,ACCSHL          ..
123D 39;          0572          ,ACCSHL          ..SHIFT LEFT,AND
123E 33;          0573          ,ACCOR          ..ASM WITH
123F 05;          0574          ,#05            ..NEW CHAR
1240 05;          0575          ,ACCVX          ..
1241 04;          0576          ,#04            ..STORE IN V4
1242 2B;          0577          ,MEMGO          ..
1243 1217;        0578          ,A(RD2)         ..GO TO RD2
1245 0B;          0579 OUTR:    ,LOADAC          ..
1246 00;          0580          ,#00            ..ERASE CURSOR
1247 07;          0581          ,INTCAL         ..CALL DISPLAY
1248 13BD;        0582          ,A(DISCHR)      ..CHAR SUB
124A 6A;          0583 LP4:    ,CRRET          ..
124B 44;          0584          ,ROWINC         ..
124C 01;          0585          ,#01            ..DO CR/LF
124D 14;          0586          ,MMP1AC         ..OUTPUT UPPER ADDRESS
124E 4E;          0587          ,DISACC         ..AS HEX PAIR
124F 43;          0588          ,PMPINC         ..
1250 01;          0589          ,#01            ..INC PMP
1251 13;          0590          ,MMPOAC         ..OUTPUT LOWER ADDRESS
1252 4E;          0591          ,DISACC         ..AS HEX PAIR
1253 43;          0592          ,PMPINC         ..
1254 01;          0593          ,#01            ..INC PMP
1255 0B;          0594          ,LOADAC         ..
1256 3D;          0595          ,#3D            ..
1257 6B;          0596          ,DISAWC         ..OUTPUT = SIGN
1258 43;          0597          ,PMPINC         ..
1259 01;          0598          ,#01            ..INC PMP
125A 0A;          0599          ,LOADVX         ..
125B 06;          0600          ,#06            ..
125C 0F;          0601          ,#0F            ..INIT V6
125D 24;          0602          ,MMPOVX         ..
125E 01;          0603          ,#01            ..INIT LINE
125F 2E;          0604          ,VXYAND         ..LENGHT COUNT
1260 01;          0605          ,#01            ..TO LSD
1261 06;          0606          ,#06            ..OF MMP.O
1262 0A;          0607 DISCT1: ,LOADVX         ..
1263 02;          0608          ,#02            ..
1264 04;          0609          ,#04            ..INIT DISPLAY COUNT
1265 0C;          0610 DISCT2: ,MEMACC         ..OUTPUT 1 DATA BYTE
1266 4E;          0611          ,DISACC         ..AS HEX PAIR
1267 49;          0612          ,MMPINC         ..INC MMP
1268 43;          0613          ,PMPINC         ..
1269 01;          0614          ,#01            ..INC PMP
126A 01;          0615          ,VXINC          ..INC
126B 01;          0616          ,#01            ..LINE LENGHT COUNT
126C 02;          0617          ,VXDEC          ..DEC
126D 02;          0618          ,#02            ..DISPLAY COUNT
126E 3E;          0619          ,VXLTK         ..CHECK BYTE COUNT
126F 04;          0620          ,#04            ..(LSD)

```

```

1270 02;          0621          ,#02          ..FOR 0 OR 1
1271 128A;       0622          ,A(CKC1)     ..GO TO CKC1
1273 02;        0623          ,VXDEC       ..
1274 04;        0624          ,#04         ..DEC LSD
1275 3D;        0625 DISCT3: ,VXEQK       ..
1276 01;        0626          ,#01         ..CHECK
1277 10;        0627          ,#10         ..LINE LENGHT COUNT
1278 124A;      0628          ,A(LP4)     ..GO TO LP4
127A 3D;        0629          ,VXEQK       ..
127B 02;        0630          ,#02         ..CHECK
127C 00;        0631          ,#00         ..DISPLAY COUNT
127D 1282;      0632          ,A(SP1)     ..GO TO SP1
127F 2B;        0633          ,MEMGO       ..IF NOT 0
1280 1265;      0634          ,A(DISCT2)  ..GO TO DISCT2
1282 0B;        0635 SP1:   ,LOADAC      ..
1283 20;        0636          ,#20         ..
1284 6B;        0637          ,DISAWC     ..OUTPUT SPACE
1285 43;        0638          ,PMPINC     ..
1286 01;        0639          ,#01         ..INC PMP
1287 2B;        0640          ,MEMGO       ..
1288 1262;      0641          ,A(DISCT1)  ..GO TO DISCT1
128A 3D;        0642 CKC1: ,VXEQK       ..
128B 03;        0643          ,#03         ..CHECK BYTE
128C 00;        0644          ,#00         ..COUNT (MSD)
128D 1297;      0645          ,A(DONE)    ..GO TO DONE
128F 02;        0646          ,VXDEC       ..DEC BYTE
1290 03;        0647          ,#03         ..COUNT (MSD)
1291 0A;        0648          ,LOADVX     ..
1292 04;        0649          ,#04         ..
1293 FF;        0650          ,#FF         ..SET LSD
1294 2B;        0651          ,MEMGO       ..
1295 1275;      0652          ,A(DISCT3)  ..GO TO DISCT3
1297 07;        0653 DONE:  ,INTCAL     ..CALL
1298 10FE;      0654          ,A(CRFEED)  ..CRFEED
129A 2B;        0655          ,MEMGO       ..
129B 1012;      0656          ,A(RESTRT)  ..GO TO RESTRT
129D ;          0657
129D ;          0658
129D ;          0659
129D ;          0660 ..*****
129D ;          0661 ..          MEMORY WRITE ROUTINE
129D ;          0662 ..          (V4,VE)
129D ;          0663 ..*****
129D ;          0664
129D ;          0665
129D 07;        0666 MEMWR:  ,INTCAL     ..
129E 112A;      0667          ,A(INCHR)   ..INPUT CHAR
12A0 40;        0668          ,ACCEQK     ..
12A1 4D;        0669          ,#4D        ..
12A2 12A7;     0670          ,A(NEWADR)  ..CHECK FOR M
12A4 2B;        0671          ,MEMGO       ..IF NOT
12A5 11B2;     0672          ,A(ERROR)   ..GO TO ERROR
12A7 0A;        0673 NEWADR: ,LOADVX     ..
12A8 0E;        0674          ,#0E        ..
12A9 00;        0675          ,#00        ..SET WRITE FLAG
12AA 2B;        0676          ,MEMGO       ..GO TO ADDR DIG INP SECT
12AB 11DB;     0677          ,A(RDADDR)  ..OF MEMORY READ ROUTINE
12AD 40;        0678 INDATA: ,ACCEQK     ..RETURNS HERE
12AE 20;        0679          ,#20        ..
12AF 12B4;     0680          ,A(LP6)     ..CHECK FOR SPACE
12B1 2B;        0681          ,MEMGO       ..IF NOT
12B2 11B2;     0682          ,A(ERROR)   ..GO TO ERROR
12B4 07;        0683 LP6:   ,INTCAL     ..

```

ICAN-7032

```

12B5 112A;      0684          ,A(INCHR)      ..
12B7 07;        0685          ,INTCAL       ..
12B8 118D;      0686          ,A(AS2HEX)    ..INPUT 1ST DATA CHAR
12BA 66;        0687          ,BNFLAG       ..
12BB 11B2;      0688          ,A(ERROR)     ..CHECK FOR HEX
12BD 39;        0689 LP7:   ,ACCSHL       ..
12BE 39;        0690          ,ACCSHL       ..
12BF 39;        0691          ,ACCSHL       ..
12C0 39;        0692          ,ACCSHL       ..ASM 1ST DATA DIGIT AND
12C1 05;        0693          ,ACCVX        ..
12C2 04;        0694          ,#04          ..AND STORE IN V4
12C3 07;        0695          ,INTCAL       ..
12C4 112A;      0696          ,A(INCHR)     ..
12C6 07;        0697          ,INTCAL       ..
12C7 118D;      0698          ,A(AS2HEX)    ..INPUT 2D DATA DIGIT
12C9 66;        0699          ,BNFLAG       ..
12CA 11B2;      0700          ,A(ERROR)     ..CHECK FOR HEX
12CC 33;        0701          ,ACCR         ..ASM 2D DATA DIGIT
12CD 04;        0702          ,#04          ..WITH 1ST AND
12CE 0D;        0703          ,ACCMEM       ..STORE AT C(MMP)
12CF 49;        0704          ,MMPINC       ..INC MMP
12D0 07;        0705          ,INTCAL       ..
12D1 112A;      0706          ,A(INCHR)     ..
12D3 07;        0707          ,INTCAL       ..
12D4 118D;      0708          ,A(AS2HEX)    ..INPUT NEXT CHAR
12D6 67;        0709          ,BRFLAG       ..
12D7 12BD;      0710          ,A(LP7)       ..CHECK FOR HEX
12D9 40;        0711          ,ACCEQK       ..
12DA 0D;        0712          ,#0D          ..
12DB 12EB;      0713          ,A(CR1)       ..CHECK FOR CR
12DD 40;        0714          ,ACCEQK       ..
12DE 3B;        0715          ,#3B          ..
12DF 12EE;      0716          ,A(LD4)       ..CHECK FOR SEMICOLON
12E1 40;        0717          ,ACCEQK       ..
12E2 2C;        0718          ,#2C          ..
12E3 12F4;      0719          ,A(LD3)       ..CHECK FOR COMMA
12E5 2B;        0720          ,MEMGO        ..
12E6 12B4;      0721          ,A(LP6)       ..GO TO LP6
12E8 07;        0722 CR1:   ,INTCAL       ..CALL
12E9 10FE;      0723          ,A(CRFEED)    ..CR/LF SUBROUTINE
12EB 2B;        0724          ,MEMGO        ..
12EC 1012;      0725          ,A(RESTRT)    ..GO TO RESTRT
12EE 07;        0726 LD4:   ,INTCAL       ..CALL
12EF 10FE;      0727          ,A(CRFEED)    ..CR/LF SUBROUTINE
12F1 2B;        0728          ,MEMGO        ..
12F2 12A7;      0729          ,A(NEWADR)    ..GO TO NEWADR
12F4 07;        0730 LD3:   ,INTCAL       ..CALL
12F5 10FE;      0731          ,A(CRFEED)    ..CR/LF SUBROUTINE
12F7 2B;        0732          ,MEMGO        ..
12FB 12B4;      0733          ,A(LP6)       ..GO TO LP6
12FA ;          0734
12FA ;          0735
12FA ;          0736
12FA ;          0737 ..*****
12FA ;          0738 ..                RUN ROUTINE
12FA ;          0739 ..*****
12FA ;          0740
12FA ;          0741
12FA 07;        0742 RUN:   ,INTCAL       ..
12FB 112A;      0743          ,A(INCHR)     ..INPUT CHAR
12FD 40;        0744          ,ACCEQK       ..
12FE 55;        0745          ,#55          ..

```

```

12FF 1304;          0746          ,A(CKN)           ..CHECK FOR U
1301 2B;           0747          ,MEMGO           ..IF NOT
1302 1012;        0748          ,A(RESTRT)      ..GO TO RESTRT
1304 07;          0749 CKN:       ,INTCAL         ..
1305 112A;        0750          ,A(INCHR)       ..INPUT CHAR
1307 40;          0751          ,ACCEQK        ..
1308 4E;          0752          ,#4E           ..
1309 130E;        0753          ,A(CKCR)       ..CHECK FOR N
130B 2B;          0754          ,MEMGO           ..IF NOT
130C 1012;        0755          ,A(RESTRT)      ..GO TO RESTRT
130E 07;          0756 CKCR:     ,INTCAL         ..
130F 112A;        0757          ,A(INCHR)       ..INPUT CHAR
1311 40;          0758          ,ACCEQK        ..
1312 0D;          0759          ,#0D           ..
1313 1318;        0760          ,A(USER1)      ..CHECK FOR CR
1315 2B;          0761          ,MEMGO           ..IF NOT
1316 1012;        0762          ,A(RESTRT)      ..GO TO RESTRT
1318 2B;          0763 USER1:    ,MEMGO           ..
1319 1B00;        0764          ,A(USRAM)      ..RUN USER PROGRAM
131B ;            0765
131B ;            0766
131B ;            0767
131B ;            0768 ..*****
131B ;            0769 ..                HOME CURSOR SUBROUTINE
131B ;            0770 ..                (V1,V9)
131B ;            0771 ..*****
131B ;            0772
131B ;            0773
131B 06;          0774 HOMECL:  ,VXACC       ..GET LAST STORED
131C 09;          0775          ,#09           ..SCREEN CHAR
131D 07;          0776          ,INTCAL         ..CALL DISPLAY
131E 13BD;        0777          ,A(DISCHR)     ..CHAR SUB
1320 68;          0778          ,INIT          ..HOME
1321 1D;          0779          ,RDPMP         ..GET NEW SCREEN
1322 05;          0780          ,ACCVX        ..CHAR AND
1323 09;          0781          ,#09           ..STORE IN V9
1324 07;          0782          ,INTCAL         ..CALL DISPLAY
1325 13BB;        0783          ,A(DISCR)     ..CURSOR SUB
1327 08;          0784          ,INTRTN       ..RETURN
1328 ;            0785
1328 ;            0786
1328 ;            0787
1328 ;            0788 ..*****
1328 ;            0789 ..                CONTROL FLAG SUBROUTINE
1328 ;            0790 ..                (V2,VC)
1328 ;            0791 ..*****
1328 ;            0792
1328 ;            0793
1328 07;          0794 CTFLAG:  ,INTCAL         ..
1329 112A;        0795          ,A(INCHR)       ..INPUT CHAR
132B 40;          0796          ,ACCEQK        ..
132C 4E;          0797          ,#4E           ..
132D 133D;        0798          ,A(CTRRST)     ..CHECK FOR N
132F 40;          0799          ,ACCEQK        ..
1330 59;          0800          ,#59           ..
1331 1336;        0801          ,A(CTRSET)     ..CHECK FOR Y
1333 2B;          0802          ,MEMGO           ..IF NOT
1334 1328;        0803          ,A(CTFLAG)     ..LOOP
1336 0A;          0804 CTRSET:  ,LOADVX        ..
1337 02;          0805          ,#02           ..
1338 01;          0806          ,#01           ..INIT V2
1339 2D;          0807          ,VXYOR        ..

```

ICAN-7032

```

133A 0C;          0808          ,CTRBLK          ..SET
133B 02;          0809          ,#02            ..CONTROL FLAG
133C 08;          0810          ,INTRTN         ..RETURN
133D 0A;          0811 CTRRST: ,LOADVX         ..
133E 02;          0812          ,#02            ..
133F FE;          0813          ,#FE            ..INIT V2
1340 2E;          0814          ,VXYAND         ..
1341 0C;          0815          ,CTRBLK         ..RESET
1342 02;          0816          ,#02            ..CONTROL FLAG
1343 08;          0817          ,INTRTN         ..RET
1344 ;           0818
1344 ;           0819
1344 ;           0820
1344 ;           0821 ..*****
1344 ;           0822 ..                CURSOR POSITION SUBROUTINE
1344 ;           0823 ..                (V2,V9)
1344 ;           0824 ..*****
1344 ;           0825
1344 ;           0826
1344 05;          0827 CURSOR: ,ACCVX          ..STORE DIRECTION
1345 02;          0828          ,#02            ..CNTRL CHAR
1346 06;          0829          ,VXACC          ..DISPLAY OLD
1347 09;          0830          ,#09            ..STORED CHAR
1348 41;          0831          ,ACCLTK         ..
1349 80;          0832          ,#80            ..
134A 1350;        0833          ,A(BR7)         ..CHECK MSB
134C 6B;          0834          ,DISAWC         ..DISPLAY STAND CHAR
134D 2B;          0835          ,MEMGO         ..
134E 1351;        0836          ,A(BR8)         ..GO TO BR8
1350 4D;          0837 BR7:   ,DISAC          ..DISPLAY RV CHAR
1351 06;          0838 BR8:   ,VXACC          ..GET DIRECTION
1352 02;          0839          ,#02            ..CNTRL CHAR
1353 40;          0840          ,ACCEQK        ..
1354 08;          0841          ,#08            ..CHECK FOR BS
1355 1372;        0842          ,A(BR3)         ..
1357 40;          0843          ,ACCEQK        ..
1358 0C;          0844          ,#0C            ..CHECK FOR FF
1359 136D;        0845          ,A(BR4)         ..
135B 40;          0846          ,ACCEQK        ..
135C 0A;          0847          ,#0A            ..CHECK FOR LF
135D 136B;        0848          ,A(BR5)         ..
135F 46;          0849          ,ROWDEC        ..GO UP
1360 01;          0850          ,#01            ..1 ROW
1361 1D;          0851 BR6:   ,RDPMP          ..GET SCREEN
1362 05;          0852          ,ACCVX          ..CHAR AND
1363 09;          0853          ,#09            ..STORE IN V9
1364 07;          0854          ,INTCAL        ..CALL DISPLAY
1365 13BB;        0855          ,A(DISCUR)     ..CURSOR SUB
1367 08;          0856          ,INTRTN         ..RETURN
1368 44;          0857 BR5:   ,ROWINC        ..GO DOWN
1369 01;          0858          ,#01            ..1 ROW
136A 2B;          0859          ,MEMGO         ..
136B 1361;        0860          ,A(BR6)         ..GO TO BR6
136D 43;          0861 BR4:   ,PMPINC        ..GO RIGHT
136E 01;          0862          ,#01            ..1 CHAR
136F 2B;          0863          ,MEMGO         ..
1370 1361;        0864          ,A(BR6)         ..GO TO BR6
1372 45;          0865 BR3:   ,PMPDEC        ..GO LEFT
1373 01;          0866          ,#01            ..1 CHAR
1374 2B;          0867          ,MEMGO         ..
1375 1361;        0868          ,A(BR6)         ..GO TO BR6
1377 ;           0869
1377 ;           0870
1377 ;           0871

```

```

1377 ; 0872 ..*****
1377 ; 0873 ..          SCREEN RESOLUTION SUBROUTINE
1377 ; 0874 ..*****
1377 ; 0875
1377 ; 0876
1377 07; 0877 RESOLN: ,INTCAL          ..
1378 112A; 0878          ,A(INCHR)      ..INPUT CHAR
137A 40; 0879          ,ACCEQK        ..IF CHAR
137B 46; 0880          ,#46           ..IS F
137C 1387; 0881          ,A(ST2)       ..GO TO ST2
137E 40; 0882          ,ACCEQK        ..IF CHAR
137F 48; 0883          ,#48           ..IS H
1380 1385; 0884          ,A(ST1)       ..GO TO ST1
1382 2B; 0885          ,MEMG0         ..IF NOT
1383 1377; 0886          ,A(RESOLN)    ..LOOP
1385 60; 0887 ST1:      ,SETHAF        ..20 X 12
1386 08; 0888          ,INTRTN        ..RETURN
1387 53; 0889 ST2:      ,SETFUL        ..40 X 24
1388 08; 0890          ,INTRTN        ..RETURN
1389 ; 0891
1389 ; 0892
1389 ; 0893
1389 ; 0894 ..*****
1389 ; 0895 ..          SET CURSOR SUBROUTINE
1389 ; 0896 ..          (V2,VC,VF)
1389 ; 0897 ..*****
1389 ; 0898
1389 ; 0899
1389 07; 0900 SETCUR: ,INTCAL          ..
138A 112A; 0901          ,A(INCHR)      ..INPUT CHAR
138C 40; 0902          ,ACCEQK        ..
138D 4E; 0903          ,#4E           ..
138E 13A6; 0904          ,A(BLKRST)    ..CHECK FOR N
1390 40; 0905          ,ACCEQK        ..
1391 59; 0906          ,#59           ..
1392 139F; 0907          ,A(BLKSET)    ..CHECK FOR Y
1394 40; 0908          ,ACCEQK        ..
1395 53; 0909          ,#53           ..
1396 13B4; 0910          ,A(RVRST)     ..CHECK FOR S
1398 40; 0911          ,ACCEQK        ..
1399 52; 0912          ,#52           ..
139A 13AD; 0913          ,A(RVSET)     ..CHECK FOR R
139C 05; 0914          ,ACCVX         ..SET CURSOR EQUAL
139D 0F; 0915          ,CURS          ..TO NEW CHAR
139E 08; 0916          ,INTRTN        ..RETURN
139F 0A; 0917 BLKSET: ,LOADVX        ..
13A0 02; 0918          ,#02           ..
13A1 10; 0919          ,#10           ..INIT V2
13A2 2D; 0920          ,VXYOR         ..
13A3 0C; 0921          ,CTRBLK        ..SET
13A4 02; 0922          ,#02           ..BLINK FLAG
13A5 08; 0923          ,INTRTN        ..RETURN
13A6 0A; 0924 BLKRST: ,LOADVX        ..
13A7 02; 0925          ,#02           ..
13A8 EF; 0926          ,#EF           ..INIT V2
13A9 2E; 0927          ,VXYAND        ..
13AA 0C; 0928          ,CTRBLK        ..RESET
13AB 02; 0929          ,#02           ..BLINK FLAG
13AC 08; 0930          ,INTRTN        ..RETURN
13AD 0A; 0931 RVSET: ,LOADVX        ..
13AE 02; 0932          ,#02           ..
13AF 20; 0933          ,#20           ..INIT V2
13B0 2D; 0934          ,VXYOR         ..SET

```


ICAN-7032

```

13B1 0C;          0935          ,CTRBLK          ..REVERSE
13B2 02;          0936          ,#02           ..VIDEO FLAG
13B3 08;          0937          ,INTRTN       ..RETURN
13B4 0A;          0938 RVRST:    ,LOADVX       ..
13B5 02;          0939          ,#02           ..
13B6 DF;          0940          ,#DF          ..INIT V2
13B7 2E;          0941          ,VXYAND       ..RESET
13B8 0C;          0942          ,CTRBLK       ..REVERSE
13B9 02;          0943          ,#02           ..VIDEO FLAG
13BA 08;          0944          ,INTRTN       ..RETURN
13BB ;           0945
13BB ;           0946
13BB ;           0947 ..*****
13BB ;           0948 ..          DISPLAY SUBROUTINE
13BB ;           0949 ..          (V8,VC,VF)
13BB ;           0950 ..*****
13BB ;           0951
13BB ;           0952
13BB 06;          0953 DISCUR:  ,VXACC       ..SET ACC EQUAL
13BC 0F;          0954          ,CURS         ..TO CURSOR
13BD 0A;          0955 DISCHR:  ,LOADVX       ..
13BE 08;          0956          ,#08         ..
13BF 20;          0957          ,#20         ..INIT V8
13C0 2E;          0958          ,VXYAND       ..STRIP OFF
13C1 08;          0959          ,#08         ..UNUSED
13C2 0C;          0960          ,CTRBLK       ..BITS OF CTRBLK
13C3 3D;          0961          ,VXEQK       ..
13C4 08;          0962          ,#08         ..
13C5 00;          0963          ,#00         ..CHECK REVERSE
13C6 13CA;        0964          ,A(DISSC)     ..VIDEO FLAG
13C8 4D;          0965          ,DISAC       ..DISPLAY REVERSE CHAR
13C9 08;          0966          ,INTRTN       ..RETURN
13CA 6B;          0967 DISSC:   ,DISAWC       ..DISPLAY STANDARD CHAR
13CB 08;          0968          ,INTRTN       ..RETURN
13CC ;           0969
13CC ;           0970
13CC ;           0971
13CC ;           0972 ..*****
13CC ;           0973 ..          START OF TONE TABLE
13CC ;           0974 ..*****
13CC ;           0975
13CC ;           0976
13CC 58;          0977 TONE1:  ,#58         ..DIVIDE BY 88
13CD 4F;          0978          ,#4F         ..88KHZ,FULL AMPL
13CE 02;          0979          ,#02         ..2X60TH TONE ON
13CF 00;          0980          ,#00         ..
13D0 80;          0981          ,#80         ..
13D1 01;          0982          ,#01         ..TURN TONE OFF
13D2 FF;          0983          ,#FF         ..STOP
13D3 2C;          0984 TONE2:  ,#2C         ..DIVIDE BY 44
13D4 4F;          0985          ,#4F         ..88KHZ,FULL AMPL
13D5 02;          0986          ,#02         ..2X60TH TONE ON
13D6 00;          0987          ,#00         ..
13D7 80;          0988          ,#80         ..
13D8 01;          0989          ,#01         ..TURN TONE OFF
13D9 FF;          0990          ,#FF         ..STOP
13DA 58;          0991 TONE3:  ,#58         ..DIVIDE BY 88
13DB 2F;          0992          ,#2F         ..22KHZ,FULL AMPL
13DC 06;          0993          ,#06         ..6X60TH TONE ON
13DD 00;          0994          ,#00         ..
13DE 80;          0995          ,#80         ..
13DF 01;          0996          ,#01         ..TURN OFF TONE
13E0 FF;          0997          ,#FF         ..STOP
13E1 ;           0998          ..END OF PROGRAM

```

A CDP1800-Based CRT Controller

by R. M. Vaccarella

INTRODUCTION

As the RCA CDP1800 series of CMOS microprocessors continues to grow and find application in new areas of design, so also does its line of peripheral support devices. The register-based architecture and variety of control signals of the CDP1800-series microprocessors, along with the large number of memory and I/O support devices presently available, offer the user system design flexibility and expandability. A recent addition to this widely accepted support line is the Video Interface System, VIS. The VIS is a two-chip video-controller system consisting of the CDP1869 address and sound generator and the CDP1870/76 color video generator. Block diagrams of these devices are shown in Figs. 1 and 2. When used with a CDP1800-series microprocessor, these LSI CMOS devices provide all the screen-refresh and color-monitor signals needed to implement a raster-scan video display.

As an I/O system, the VIS takes advantage of the CDP1800-series microprocessor N-line input/output capability and the specific instructions provided for I/O-mapped peripherals. Using the 3-bit I/O address bus, the VIS interfaces directly with the CPU, with very few additional parts required.

VIS OPERATION

Fig. 3 shows a system diagram of a typical video system using the VIS. Those familiar with the older discrete-IC versions of CRT controller circuits will be impressed by the noticeably few components shown; the large number of features integrated on the VIS chips themselves is responsible for this advantage.

Any CRT controller system can typically be divided into three functional sections: the CPU interface, the video interface, and the screen-refresh interface. A basic under-

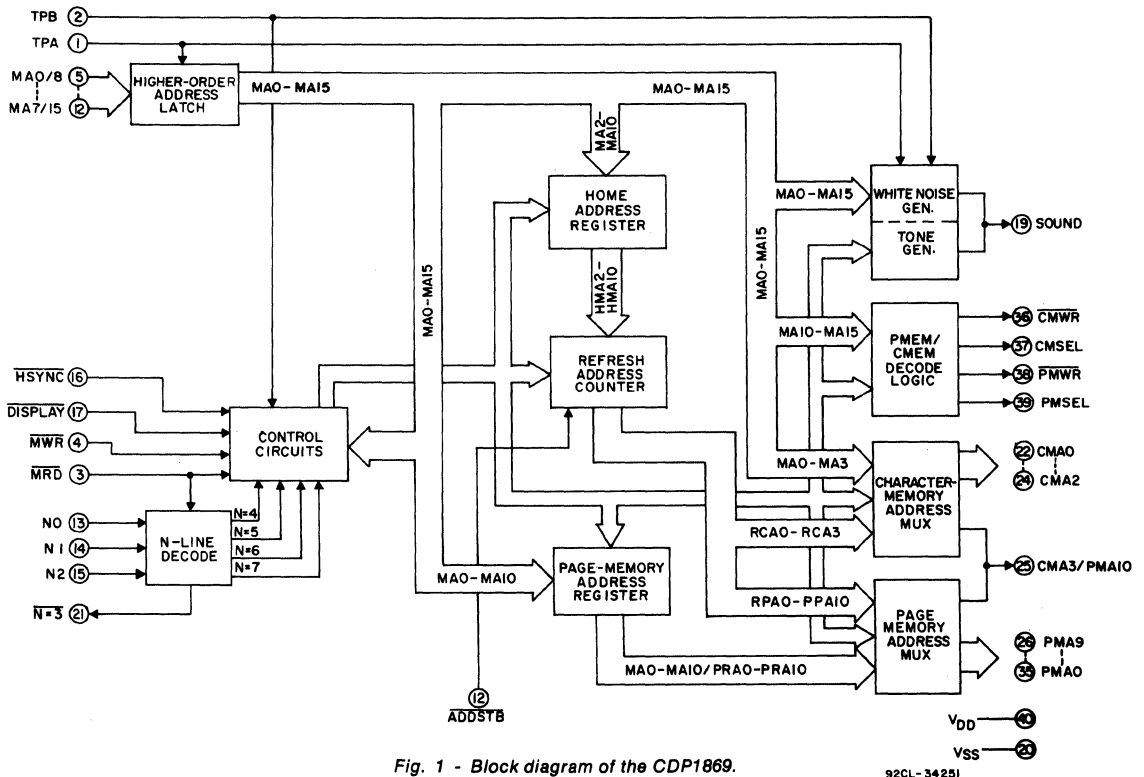


Fig. 1 - Block diagram of the CDP1869.

92CL-34251

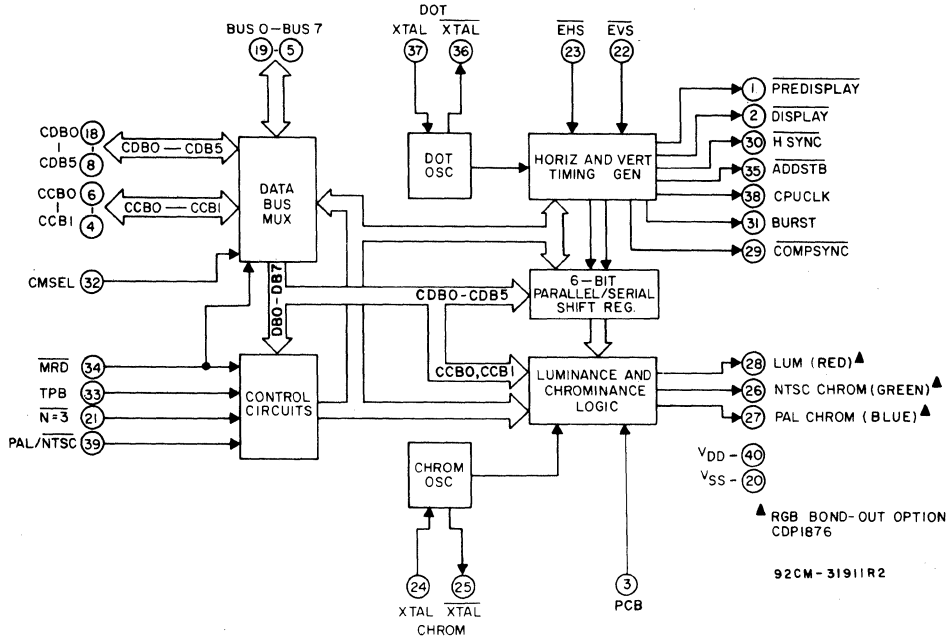


Fig. 2 - Block diagram of the CDP1870/CDP1876.

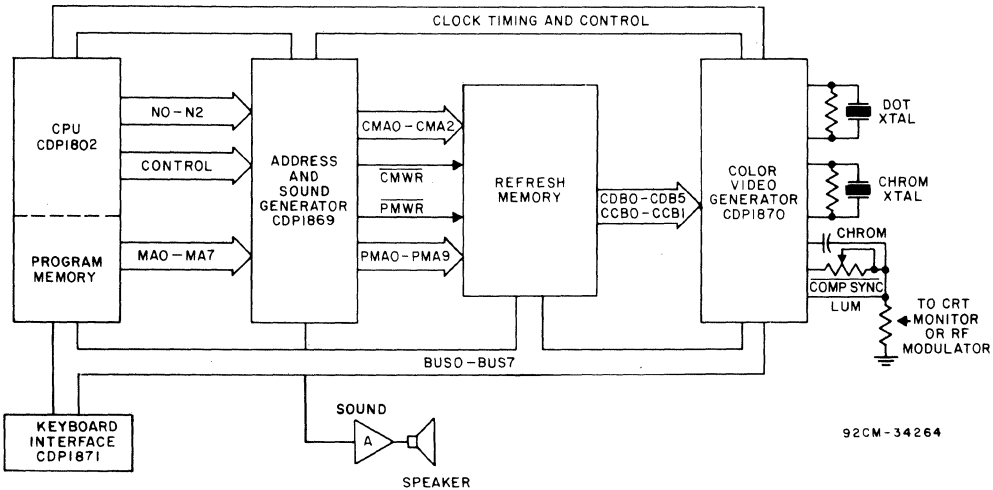


Fig. 3 - System diagram of a typical video system employing the VIS.

standing of each of these areas is necessary in video-system design. Each area is examined below using the VIS as the model system, but the basics apply to all CRT controller circuits.

CPU Interfacing

The CPU provides overall system control in the CRT controller circuit. When the system is turned on, the CPU initializes the VIS from the program memory by loading the appropriate command registers. These registers control the screen format, video and color information, TV synchronization, and the sound output. The CPU also handles the general program flow of the system by performing the tasks not associated with screen display control, such as

the keyboard interface. This job is especially simplified by the VIS since the CPU is not involved in the actual display-refresh procedure. Once initialized, the VIS handles all screen-related functions; the CPU is needed only to change formatting and to enter data in the page memory to update the display.

To simplify CPU interfacing, the VIS uses the I/O-oriented features of the CDP1800-series microprocessor. Peripheral systems devices, such as the VIS, can be mapped in I/O space using the 3-bit I/O address bus. Peripheral data transfer is accomplished by means of the INPUT and OUTPUT instructions and by use of a memory stack and the CPU internal 4-bit X designator.

Another method used in the VIS to simplify CPU interfacing is the register-based output technique. This method makes use of the 8-bit multiplexed address bus to output sixteen bits of data with a single software instruction. A CPU register, which normally supplies the 16-bit memory-address information, is loaded with data to be sent to one of the four command registers within the CDP1869. An OUTPUT instruction is used to transfer the data; X designates one of the sixteen 16-bit general-purpose CPU registers. The TPA pulse from the CPU is used to latch the eight higher-order data bits. Fig. 4 shows the timing for a CDP1800 CPU OUTPUT instruction.

The 8-bit data bus from the CPU connects directly to the CDP1870 and provides data to its single command register and the character memory. The bus is also connected, through tristate buffers, to the page-memory data lines and directly to the keyboard interface circuitry.

Video Interface

The most widely used video-display device is the raster-scan CRT monitor, either black-and-white or color. The monitor is essentially a high performance version of the home TV receiver, and accepts either separate horizontal sync, vertical sync, and black-and-white or color video signals, or a single composite signal consisting of all three.

Two on-board crystal oscillators in the CDP1870 generate all the necessary timing for a video monitor system. The 5.67-MHz dot crystal frequency is divided down to produce the required TV and VIS synchronization timing and also to provide a CPU clock at one-half the dot crystal frequency. The CPU may also run asynchronously at its own clock rate. The chrominance crystal runs at twice the color subcarrier frequency (7.159 MHz), and generates the necessary color timing.

Since all of the video circuitry is included on-board the CDP1870, only three external passive components (two resistors and a capacitor) are required to produce a composite color video signal.

Screen-Refresh Memory Interfacing

Perhaps the least defined area in the system design is the screen-refresh memory-interfacing procedure. Many methods and configurations are possible, but the two basic methods are the bit-map approach and the character-generator approach. The bit-map approach is a technique in which each vertical pixel (picture element) on the screen is defined by a single location in RAM memory while each horizontal pixel is defined by the data bits within that RAM location. The screen, then, is a one-to-one "map" of the refresh memory. Since each pixel must be defined at every screen location at which it appears, the bit-map approach is most efficient in displays with a large amount of non-repeating graphics or characters. The bit-map memory is addressed as a single storage area.

The character-generator approach, on the other hand, uses two types of memory storage areas. One, the page memory, defines the screen display in specific character blocks instead of individual pixels, and the other, the character memory, defines the actual character pixels within each block or matrix. In operation, the data outputs of the page memory become part of the address information of the character memory. The data outputs of the character memory are sent to a parallel-to-serial shift register and then to the CRT. Although two types of memories are required, the character generator approach is very efficient in displays with repeating graphics or characters, such as text displays. Each character block displayed, which is typically a 6x8 pixel matrix, requires only a single data byte in the page memory.

The VIS was designed primarily for the character-generator approach, although bit-mapping is possible. The refresh-memory interfacing of the VIS will be treated in more detail below.

COMMAND REGISTER OPERATION

CDP1869

As discussed above, the VIS uses the CDP1800 I/O structure

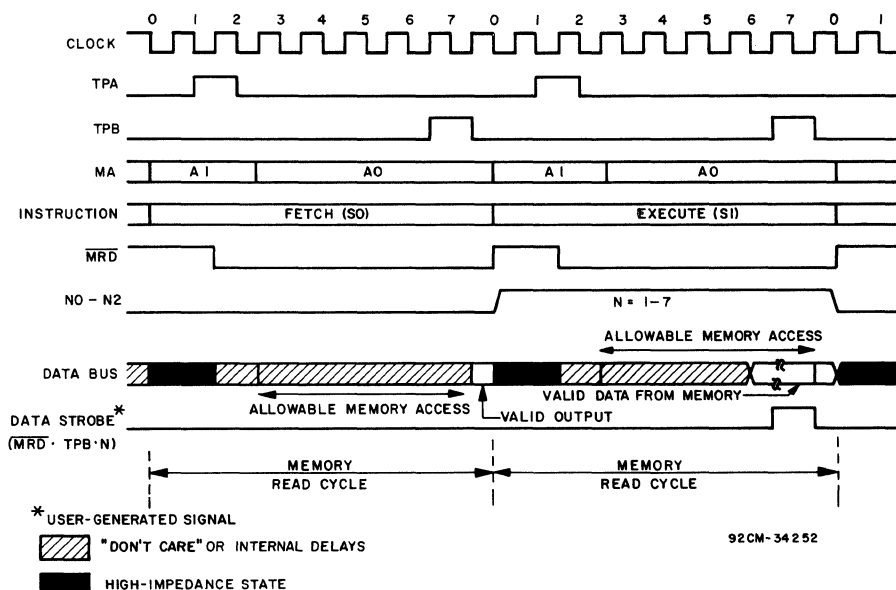


Fig. 4 - Timing diagram for the CDP1800-series CPU OUTPUT instruction.

Table I - VIS Command Register Functions

CDP1869	
OUT 4	Tone frequency and amplitude register
OUT 5	White noise (range and amplitude), screen format, character-memory access register
OUT 6	Page-memory address register
OUT 7	Home address register
CDP1870/76	
OUT 3	Screen format, display on/off, color format, background color register

Table II - CDP1869 Command Register Codes

CPU I/O Instruction	MA15	MA14	MA13	MA12	MA11	MA10	MA9	MA8	MA7	MA6	MA5	MA4	MA3	MA2	MA1	MA0
OUT 4	Tone 0*	Tone ÷ 2 ⁶	Tone ÷ 2 ⁵	Tone ÷ 2 ⁴	Tone ÷ 2 ³	Tone ÷ 2 ²	Tone ÷ 2 ¹	Tone ÷ 2 ⁰	Tone Off	Tone Freq SEL2	Tone Freq SEL1	Tone Freq SEL0	Tone Amp 2 ³	Tone Amp 2 ²	Tone Amp 2 ¹	Tone Amp 2 ⁰
OUT 5	WN Off	WN Freq SEL2	WN Freq SEL1	WN Freq SEL0	WN Amp 2 ³	WN Amp 2 ²	WN Amp 2 ¹	WN Amp 2 ⁰	FRES Vert	Double Page ***	16 Line Hi-Res ***	X	9 Line	X	X	CMEM Access Mode
OUT 6	X	X	X	X	X	PMA10 Reg	PMA9 Reg	PMA8 Reg	PMA7 Reg	PMA6 Reg	PMA5 Reg	PMA4 Reg	PMA3 Reg	PMA2 Reg	PMA1 Reg	PMA0 Reg
OUT 7	X	X	X	X	X	HMA10 Reg	HMA9 Reg	HMA8 Reg	HMA7 Reg	HMA6 Reg	HMA5 Reg	HMA4 Reg	HMA3 Reg	HMA2 Reg	X**	X**

X=Don't care.

*=Must be programmed low.

**=Always set low internally.

***=Must be programmed low during 9-line operation.

to simplify CPU interfacing. The 16-bit register-based output technique is used to load the four command registers within the CDP1869. Table I shows the function of each VIS command register and its associated CPU output instruction. Because the register sizes range from nine bits to sixteen bits, the 16-bit register-based output technique is used rather than the 8-bit data bus method, which would require two consecutive instructions. Table II shows the individual bit assignments for each CDP1869 command register.

The OUT 4 instruction uses 15 bits to control the tone output. 576 programmable tones covering eight octaves are available for sound effects, alarms, and audio feedback of system operations. The amplitude is also programmable from zero to 0.78 V_{DD} in sixteen steps.

The OUT 5 instruction uses eight of the sixteen available bits to control the white-noise output, in eight frequency ranges, for explosion-type sound effects. The tone and white noise are combined in a single output, but may be programmed and turned on and off individually. The remaining eight bits of this register are used for screen format and character-memory access control. The Full Resolution Vertical bit selects either twelve or twenty-four vertical rows of characters. The Double Page bit extends the addressing of the page memory to 2K bytes. The 16-Line

High-Resolution bit doubles the normal eight-line resolution of the 6x8 character matrix and allows individual programming of all sixteen lines. The 9-line bit allows operation with the European PAL TV standard, which requires a different character aspect ratio than the U.S. NTSC standard. The Character Memory Access Mode bit is used to set up operations requiring reading or writing to the character memory.

The OUT 6 instruction employs eleven bits to latch the page-memory addresses, and is used in conjunction with the Character Memory Access Mode bit for character-memory operations.

The OUT 7 instruction uses eleven bits to set the home address, which defines the upper-left starting position of a row of characters on the screen. By programming this register in multiples of twenty, in the 20-characters/row mode, or in multiples of forty, in the 40-characters/row mode, various roll and scroll screen operations are possible. For example, a display window can be set up to search through multiple pages of text.

CDP1870/76

The CDP1870/76 color video generator contains just one command register. Since this register has only eight bits, the more conventional CDP1800 I/O technique is used for

Table III — CDP1870 Command Register Code

CPU I/O Instruction	Bus 7	Bus 6	Bus 5	Bus 4	Bus 3	Bus 2	Bus 1	Bus 0
OUT 3	Fres Horz	COLB1	COLB0	Disp Off	CFC	Bkg Red	Bkg Blue	Bkg Green

the data transfer. The 8-bit CPU data bus supplies the command-register information for the screen-resolution and color-format control. Table III shows the bit assignments for the CDP1870/76 command register.

The OUT 3 instruction sets the eight bits required for this command register. The Red, Blue, and Green Background bits allow eight different background colors to be used, independent of the character dot colors. The Color Format Control bit, the Color Bit 0 and Color Bit 1, in combination with the page and character-memory color data bits, provide eight character dot colors.

In addition to providing the correct color subcarrier phase, the color information also affects the luminance (brightness) signal. Each color supplies a percentage of the luminance signal, from zero percent for black to 100 percent for white, so that in a black and white display, various brightness levels can be obtained to produce selective gray-scale shading of the displayed characters. The Chrom crystal is not required for a black and white display because it supplies only the chrominance phase information and not the luminance data.

The Display Off bit is used to disable the CRT display. When the display is off, the video information is held at the background color, but the TV sync signals continue to operate, providing a stable blank raster.

The Full Resolution Horizontal bit selects either 20-characters/row or 40-characters/row. This bit, in combination with the Full Resolution Vertical and the 16-Line High-Resolution bits of the CDP1869, can produce eight different character formats, as shown in Fig. 5.

The command registers within the CDP1869 and CDP1870 are write-only registers and must be initialized by the system for proper operation.

SCREEN-REFRESH MEMORY OPERATION

As above mentioned, the VIS uses a character-generator approach for the CRT display, an approach requiring separate page and character memories. The character memory can be ROM or RAM; characters can be modified during operation when using RAM. To update the display or redefine a character, the CPU must be able to access either the page or character memories, but the VIS must have priority over the CPU to maintain a flicker-free display. This priority is assured by having the VIS provide internal decoding and multiplexing of the page and character memory address lines and the character memory data lines. Only a bus separator is required externally for the page-memory data lines to prevent bus contention with the CPU. Fig. 6 shows how the internal structure of the VIS is used to route the proper address source signal to the page and character memories.

During display-refresh time, the addresses originate from the internal refresh counter of the CDP1869 and are multiplexed to the page and character memories. The bus separator is off at this time and the page-memory data outputs provide part of the character-memory addressing.

The character-memory data outputs are multiplexed, within the CDP1870, to the parallel-to-serial shift register and the chrominance/luminance logic to provide the screen video and color information.

During nondisplay time, which includes the vertical retrace period and the blank display period at the top and bottom of the screen, the addresses originate from the CDP1869 address latch and the CPU address bus. The eight higher-order addresses are latched by the CPU TPA pulse and are decoded by the CDP1869 for page and character-memory select signals. These same addresses are also multiplexed to the page and character-memory address inputs. The screen-refresh memory, then, functions as an extension of the CPU system memory. When the page memory is selected, the bus separator is active and the 8-bit data bus is directed to the page-memory data input/output lines. When the character memory is selected, the 8-bit data bus is multiplexed, within the CDP1870, to the character-memory data input/output lines. The memory-read (MRD) and memory-write (MWR) signals normally used with the CPU's system memory are also used during screen-refresh memory operations. To synchronize the system to the display and nondisplay periods, the VIS provides a PREDISPLAY signal, which is active one horizontal line before the start of the display time. The signal may be connected to either the interrupt or a flag input of the CPU.

This method of access-sharing of the page and character memories simplifies interfacing through the use of on-board multiplexers, and significantly reduces the number of external devices required.

APPLICATIONS

The most obvious application for the VIS is in a video-terminal system used to input data from a keyboard and to display information on a CRT. Fig. 7 shows such a system,² some of whose features include:

- Carriage Return
- Line Feed
- Clear Screen and Home
- Home Cursor
- Blinking Cursor
- Reverse Video
- Cursor Direction (up, down, forward, back)

Because this system is based on the VIS, only fifteen IC's and a handful of passive components are needed.

In applications where an RGB monitor is used, or when the CRT controller is part of the CRT chassis, the CDP1876 bond-out option of the CDP1870 can be used to increase color resolution and simplify interfacing. The CDP1876 replaces the LUM, PAL, CHROM, and NTSC CHROM outputs with RED, BLUE, and GREEN outputs, which are used to drive the CRT color amplifiers directly. In this mode of operation, the color subcarrier signal is not used and the CHROM crystal is not required. The overall system remains the same as with the composite output CDP1870, except for the video interfacing, as shown in Fig. 8.

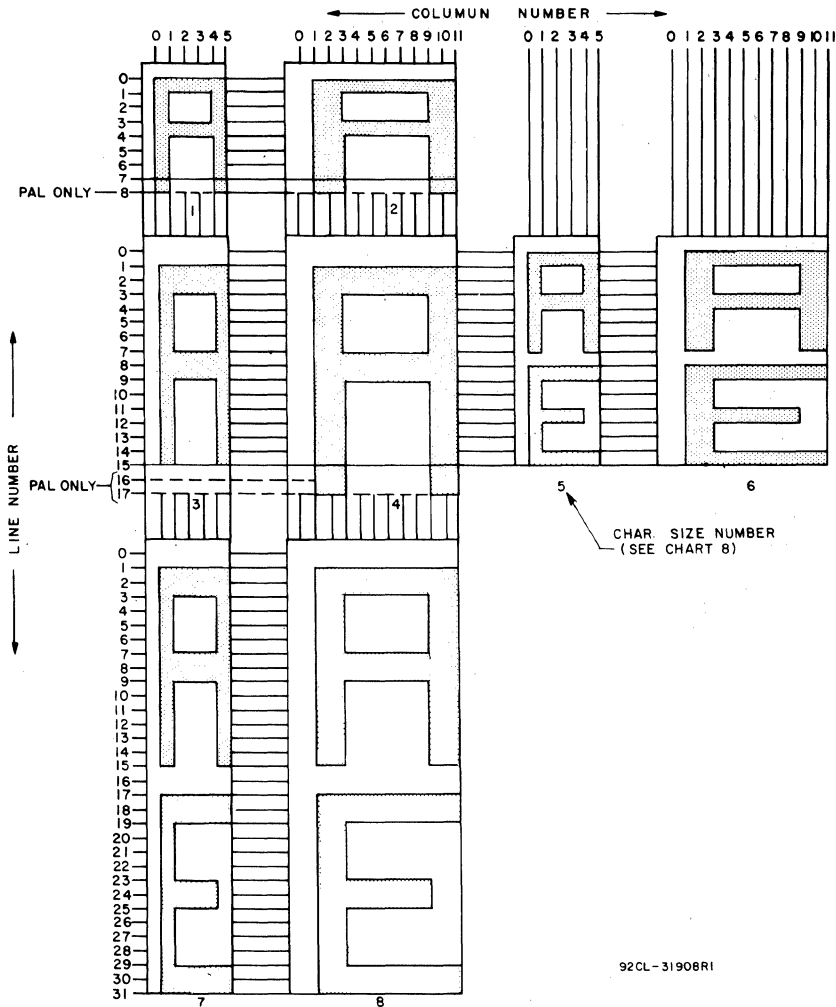


Fig. 5 - Character-display matrix size.

In text-overlay applications in which graphics or text are electronically superimposed on an existing video display, the VIS provides external horizontal and vertical sync inputs. The external inputs become the main synchronizing signals, and the VIS is locked to these external frequencies. The dot crystal is replaced with a simple gated LC oscillator circuit, and the VIS video signal is combined with the existing external video source, Fig. 9.

Other applications of the VIS include video games, low-power portable displays, low-cost industrial displays, and automotive displays and test equipment. In some of these areas, the classical advantages of the CMOS technology, such as high noise-immunity and wide operating temperature and voltage ranges, could be important considerations.

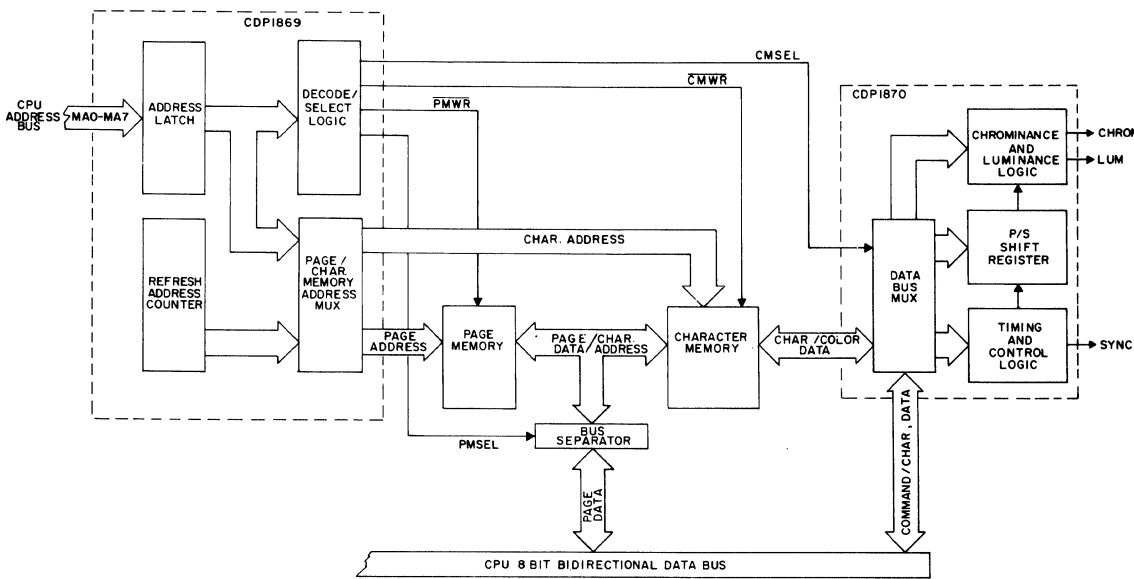
CONCLUSION

The high level of integration and streamlined architecture of the VIS chip set will allow the system designer to easily implement a CRT controller design—a design that previously required dozens of IC's and an intimate knowledge

of the video controller field. Areas of application that would have benefited from the use of a CRT display, but in which that display technology was not considered because of high cost or design difficulties, may now be re-examined.

The advanced features of the VIS (summarized in Table IV) are intended to simplify the design process from a complete system point of view. Once the design is implemented, changes and new features are easily added to the system through software reprogramming. At the same time, the system-software development burden is eased by a unique VIS interpretive language containing easy to use screen-oriented instructions. The VIS Interpreter is a user-extensible language requiring 3K bytes of memory, and is compatible with RCA Development System Assembly programs (ASM4 and ASM8).

These features, along with the availability of mature RCA CMOS CDP1800-series product-line support, and the fact that the VIS is already designed into many RCA system products, should certainly be part of any CRT-controller-system design considerations.



92CL - 34250

Fig. 6 - VIS screen-refresh memory logic.

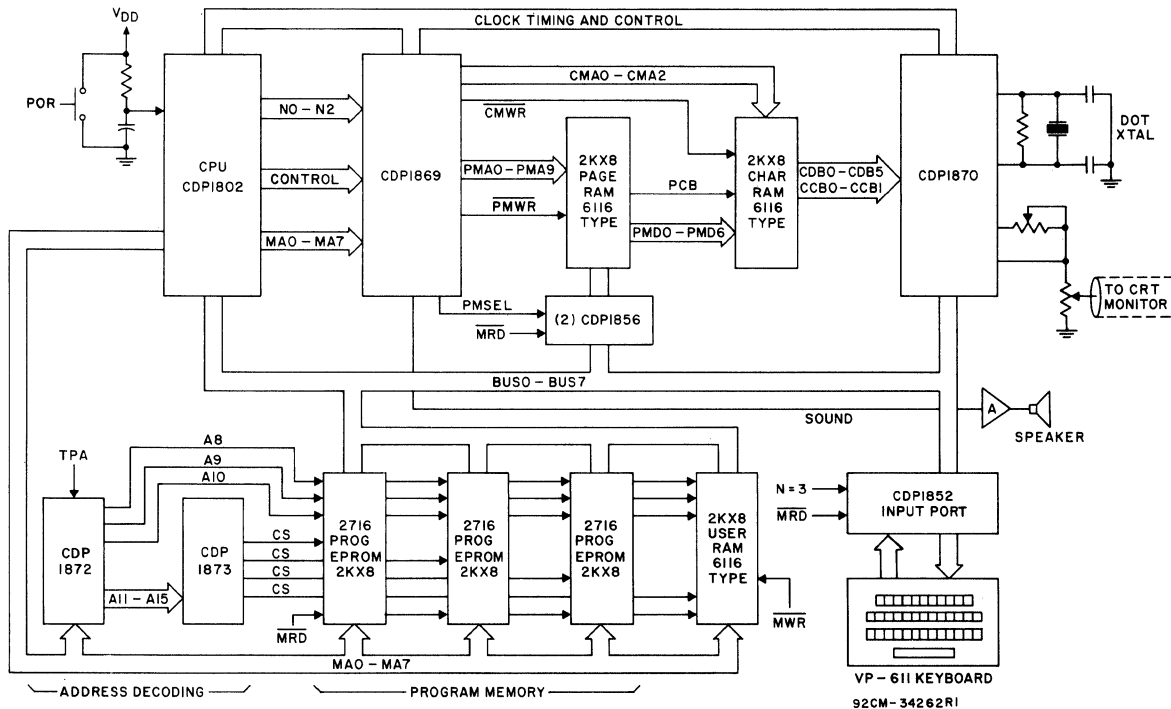
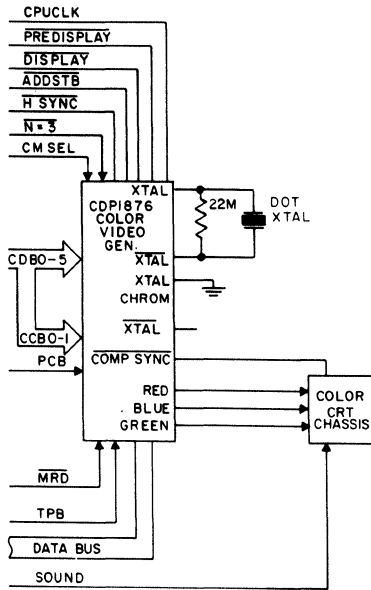


Fig. 7 - Block diagram of video terminal employing the VIS.

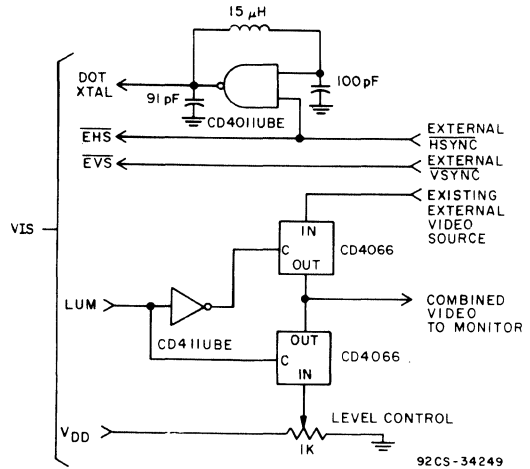


92CS-34248

Fig. 8 - RGB monitor interface.

BIBLIOGRAPHY

1. "COS/MOS Video Interface System," RCA Solid State File No. 1197.
2. "CDP1800-Based Video Terminal Using the RCA Video Interface System, VIS," R. M. Vaccarella, RCA Solid State Application Note ICAN-7032.



92CS-34249

Fig. 9 - Video overlay circuit.

Table IV - Summary of VIS Features

Standard CMOS Advantages -	Low power, high noise immunity, etc.
Asynchronous Operation -	Independent screen-refresh operation
Programmable Display Formats -	More than adequate for industrial/commercial display applications and low-end video terminals
Low System Chip Count -	On-board oscillator, sync outputs, etc.
Color and Sound Capability -	Eight dot, eight background colors, RGB-color option, tone and white-noise generators
NTSC and PAL Compatible -	U.S./European market
ROM or RAM Character Memory -	For nonstandard character sets
Teletext Compatible -	V and H inputs available for picture overlays
Simple CDP1800-Series Interfacing -	CDP1802, CDP1804, CDP1805, CDP1806
System Support -	VIS Microboard prototyping board
Software Support -	VIS Interpreter language package

Understanding The CDP1851 Programmable I/O

by G. Johnson

Overview

The CDP1851 is a general-purpose programmable I/O device, having 20 I/O pins which may be used in several different modes of operation. (See Table II.)

The I/O lines are grouped into two sections, A and B, each having 10 lines; 8 data and 2 handshaking lines (ready and strobe).

In essence, the CPU programs the PIO by asserting the proper address (if memory mapped) or the proper N-line code (if I/O mapped) on the PIO control lines, and outputs a sequence of control bytes on the data bus to the control register of the PIO. The control bytes contain information to define port mode, interrupt enable/disable, I/O bit assignment, bit masking, etc. (see Codes A-P, Table I).

The CPU transfers data bytes to and from each port by asserting codes S, T, U or V, given in Table I. Modes may be combined so that their functional definition can be tailored to almost any I/O requirement.

Modes of Operation

a. Normal Input/Output Mode

Ports A and B can be separately programmed to be 8-bit **input** or **output** parts with handshaking control lines, ready and strobe.

b. BI-Directional Mode

Port A can be programmed to be a **bi-directional** port. This configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data. Handshaking signals are provided to maintain proper bus flow discipline. The handshaking lines for Port A are used in the normal manner for input control. The handshaking lines for Port B are used by Port A for output control; consequently, Port B I/O lines must be in the bit-programmable mode where handshaking is not used.

c. Bit-programmable Mode

Ports A and B can be separately **bit programmed** so that each individual line can be designated as an input or output line.

An additional feature of the bit-programmable mode is that the four handshaking lines, A RDY, A STROBE, B RDY, and B STROBE can be individually programmed as input or output lines (see Code K, Table I).

In the input, output, and bi-directional modes the STROBE lines give the PIO the trigger signals needed to generate interrupt requests. However, in the bit-programmable mode the handshake lines are not used to carry strobe and ready signals, but carry I/O data if programmed to do so.

Interrupts

Interrupt requests are generated differently depending on the port mode.

a. Input and Output Modes

The falling edge of the strobe pulse from the peripheral device sets the $\overline{\text{INT}}$ line, and the rising edge of TPB, during the requested read or write operation resets the $\overline{\text{INT}}$ signal.

b. BI-directional Mode (Port A only)

Set and reset of interrupt requests are done as explained in the input and output modes. However, since the A $\overline{\text{INT}}$ line is used for both input and output interrupts, the CPU must read the status register to determine what condition caused the interrupt request.

c. Bit-Programmable Mode

Interrupt requests can be generated by programming the PIO to react to specified logic functions (AND, OR, NAND, or NOR) on selected I/O lines. The CPU must issue two control bytes; the first will select the logic condition, and the second will contain masking information indicating which bit(s) of eight the PIO will monitor for the logic condition. The $\overline{\text{INT}}$ signal will exist while the logic condition is present. (See Codes I & J, Table I.)

d. Interrupt Enable/Disable

To enable or disable the $\overline{\text{INT}}$ lines in all modes, the CPU must issue a control byte for each port (see Codes L, M, N & P, Appendix II). A and B interrupt status can be read from bit D0 and D1 of the status register to determine which $\overline{\text{INT}}$ line is causing the request if they are wired together (OR'd).

Timing

a. Input Data Transfer

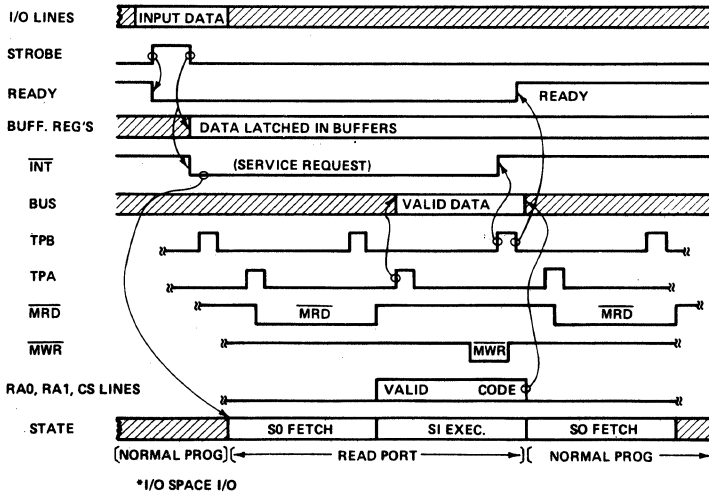
Refer to Input Port Sequential Timing diagram. Assume an I/O mapped I/O system similar to Fig. 1. The peripheral presents data to the I/O port and outputs a strobe pulse to the PIO. The strobe pulse causes three things to happen:

1. The READY signal is reset inhibiting further transmission from the peripheral.
2. The input data is latched in the buffer register.
3. The $\overline{\text{INT}}$ line is set low, signalling the CPU to read the data.

The A and B $\overline{\text{INT}}$ lines of the PIO may be wired to the $\overline{\text{INT}}$ pin on the CPU to signal program interrupts, or they can be wired to separate $\overline{\text{EF}}$ pins where periodic polling of the $\overline{\text{EF}}$ pins is required to check for service requests.

In either case, the program will branch to a subroutine and execute an input instruction (INP6 or INP7, see Codes S & T, Table I) which will assert the proper code

Input Port Sequential Timing*



on the RA0, RA1, and CS pins of the PIO. The PIO will place data onto the system bus so it can be used by the CPU and/or written into memory.

The TPB pulse that occurs during the \overline{MWR} pulse terminates the interrupt request and sets the READY line, indicating to the peripheral that the PIO is ready to accept a new data byte.

b. Output Data Transfer

Refer to Output Port Sequential Timing diagram. Assume an I/O mapped I/O system similar to Example 1. A strobe pulse from a previous output sequence or a dummy strobe causes the \overline{INT} to go low, signalling the CPU to output a data byte to the PIO.

The PIO \overline{INT} line can be wired to the CPU \overline{INT} pin or an \overline{EF} pin as explained in Input Data Transfer, above. The program will branch to a subroutine and execute an output instruction (OUT6 or OUT7) which will assert the proper code on the RA0, RA1, and CS pins of the PIO.

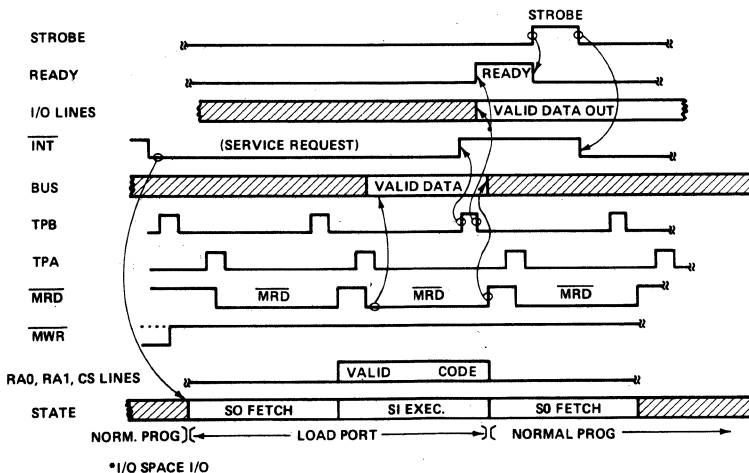
Data will be read from memory and placed on the bus and latched into the port buffers on the trailing edge of the TPB. The READY line is also set at this time. The peripheral will transmit a strobe pulse indicating the reading process is completed. The rising edge of the strobe pulse causes the READY signal to reset, and the falling edge sets the interrupt request, signalling the CPU to output another data byte.

c. Data Transfer, Bit-Programmable Mode

The CPU loads a data byte to the 8-bit port as in the normal output mode. I/O lines programmed as outputs will accept and latch data bits, however, I/O lines programmed as inputs will ignore the loaded data (see Code H, U, and V of Table I).

The CPU reads the 8-bit port as in the normal input mode. I/O lines are non-latching and, therefore, input data must be stable while the CPU reads. All 8 I/O lines are read whether they are programmed as input lines or output lines. Data read from the lines programmed as outputs will be data bits latched during the last output cycle (see Codes H, S, and T, Table I).

Output Port Sequential Timing*



EXAMPLES

Example 1 - Simple Input/Output Mode

Fig. 1 shows an I/O space I/O system involving the 1802 CPU, the 1851 PIO, and two peripheral devices both with handshaking. The INT lines are individually connected to EF lines 1 and 2. Therefore, the CPU is not truly interrupted but must poll the flag lines periodically during the main program.

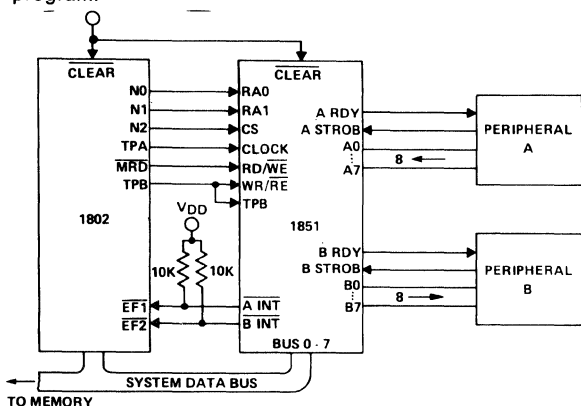
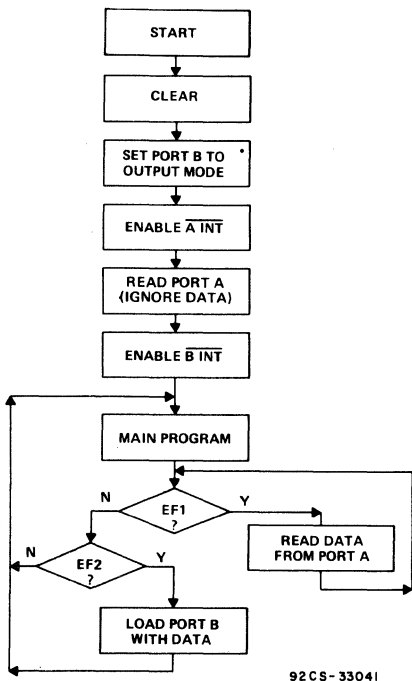


Fig. 1 - Example 1.

Step by Step - Refer to Flow Chart for Example 1.

1. To begin using this system the device must be cleared which automatically programs both Port A and B to the input mode.
2. Port B is set to the output mode by loading the control register with control byte given in Code D.
3. Port A interrupt line is enabled by loading control byte given in Code L, Table I.



92CS-33041

Flow Chart for Example 1.

4. Dummy read Port A to raise the ready line.
5. Port B interrupt line is enabled by loading control byte given in Code M, Table I. Now the main program can begin running.
6. Sometime during or at the end of the main program the EF1 flag is polled. If it is true the program will branch to a subroutine to read Port A. The CPU must output the proper N-line Code to read A. (See Code S, Table I, and Input Port Timing diagram.) When this step is completed the flag is again polled to check for more incoming data.
7. If EF1 is false then EF2 is considered. If EF2 is true then the program will branch to a subroutine to load Port B with data. The CPU must output the proper N-line code and place the 8-bit data word on the bus. (See Code V, Table I and Output Port Timing diagram.) When this step is completed the CPU returns to the main program.
8. If EF2 is false the CPU returns to the main program.

Example 2

Fig. 2 shows an I/O space I/O system involving the 1802 CPU, the 1851 PIO, and two peripheral devices. Peripheral A has a bi-directional bus with handshaking capability for transmit and receive. Peripheral B has a 4-bit receiver and a 4-bit transmitter.

Port A will be programmed for bi-directional mode with interrupt capability.

Port B will be programmed for bit-programmable mode with interrupt capability for output data when a logic NOR (all 0's) occurs on the 4 input lines.

Priorities will be as follows:

1. Port A input data, via interrupt and subroutine.
2. Port A output data, via interrupt and subroutine.
3. Port B output data, via interrupt and subroutine.
4. Port B input data, part of main program.

Port A is used to interface with a bi-directional device, therefore the handshaking lines A READY and A STROBE are used for control of incoming data (from peripheral to CPU). B READY and B STROBE handshaking lines are used for output control (from CPU to peripheral).

Port B is used in the bit-programmable mode having bits B0, B1, B2, and B3 as outputs (data from CPU to peripheral) and Bits B4, B5, B6, and B7 as inputs (data from peripheral to CPU).

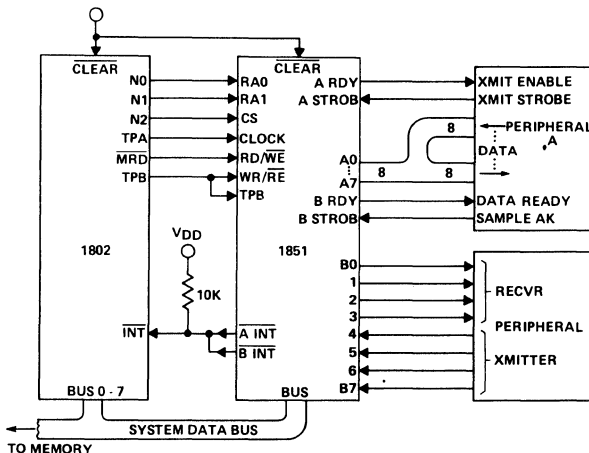
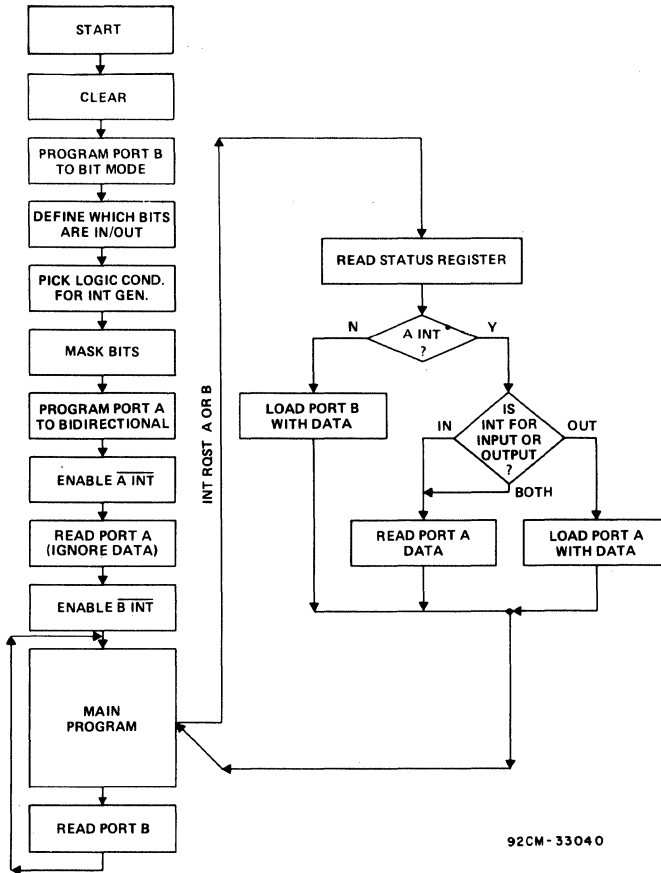


Fig. 2 - Example 2.



Flow Chart for Example 2.

The proper coding of the CPU N-lines will select whether the data will be read in, or written out to the PIO, or whether the status register is to be read, or whether the control register is to be loaded with a control byte. (See Table I).

Step by Step

1. To begin using this system the device must be cleared which automatically programs both Port A and B to the input mode.
2. Since Port A is going to be in the bi-directional mode, Port B **must** be programmed in the bit-programmable mode. Port B is set to the bit mode by loading control byte given in Code G, Table I.
3. Code H control byte must be loaded. This byte determines which bits are input/output.
4. Code I control byte must be loaded which determines the logical condition of bits required to generate an interrupt request from Port B (in this case a NOR condition).
5. Code J control byte must be loaded. This byte tells which of the 8 bits in Port B are monitored and which are masked for interrupt generation. (In this case all of the input lines B4, B5, B6 and B7 will be monitored.)
6. Now Port A is set to bi-directional mode by loading Code E control byte.
7. Port A interrupt line is enabled by loading the control register with Code L control byte.
8. A RDY is raised (input section of Port A handshaking line) by doing a dummy read.

9. Port B interrupt line is enabled by loading the control register with Code M control byte. Now the main program can begin running.
10. Port B will be read periodically as the main program repeats its loop.
11. When an interrupt request is received by the CPU, it will branch to a subroutine, and automatically deactivate the interrupt enable to inhibit further interruptions.
12. The CPU must read the status register by outputting the proper N-line Code R. The status register will contain information describing which interrupt has occurred A and/or B, and also indicated whether \bar{A} INT was caused by an input or an output demand from peripheral A. Note that \bar{A} INT and \bar{B} INT lines are wired (OR'd together) (see Fig. 2).
13. Several decisions will be made based on the information contained in the status register.
14. To read Port A the CPU must output the proper N-line Code S.
15. To load Port A with data the CPU must output the proper N-line code, and place the 8-bit data word on the bus (see Table I).
16. To load Port B (bits B0, B1, B2, and B3) with data the CPU must output the proper N-line code, and place the 4-bit data on the bus (see Table I).
17. Once the interrupt subroutines are completed the CPU returns to the main program and the interrupt enable (CPU) is activated.

Table I

CODE	DESIRED ACTION	CS	RA1	RA0	RD/WE	WR/RE	DATA BUS																			
							D7	D6	D5	D4	D3	D2	D1	D0												
A	SET PORT A TO INPUT MODE	1	0	1	0	1	0	0	X	0	1	X	1	1												
B	SET PORT B TO INPUT MODE	1	0	1	0	1	0	0	X	1	0	X	1	1												
C	SET PORT A TO OUTPUT MODE	1	0	1	0	1	0	1	X	0	1	X	1	1												
D	SET PORT B TO OUTPUT MODE	1	0	1	0	1	0	1	X	1	0	X	1	1												
E	SET PORT A TO BIDIRECTIONAL (PORT B MUST BE IN BIT MODE FIRST)	1	0	1	0	1	1	0	X	0	1	X	1	1												
F	SET PORT A TO BIT - PROG MODE	1	0	1	0	1	1	1	X	0	1	X	1	1												
G	SET PORT B TO BIT - PROG MODE	1	0	1	0	1	1	1	X	1	0	X	1	1												
H	I/O BIT ASSIGNMENT (A OR B)	1	0	1	0	1	(0 = INPUT, 1 = OUTPUT)																			
I	SET LOGICAL CONDITION FOR INTERRUPT GENERATION (BIT - MODE) (D3) 0 = PORT A, 1 = PORT B (D4) 0 = NO MASK, 1 = MASK BYTE FOLLOWS NEXT	1	0	1	0	1	0	D6	D5	D4	D3	1	0	1												
								<table border="1"> <tr> <td>0</td> <td>0</td> <td>NAND</td> </tr> <tr> <td>0</td> <td>1</td> <td>OR</td> </tr> <tr> <td>1</td> <td>0</td> <td>NOR</td> </tr> <tr> <td>1</td> <td>1</td> <td>AND</td> </tr> </table>		0	0	NAND	0	1	OR	1	0	NOR	1	1	AND					
0	0	NAND																								
0	1	OR																								
1	0	NOR																								
1	1	AND																								
J	SET MASKING OF BITS (PRECEDED BY CODE I)	1	0	1	0	1	(0 = MONITORED, 1 = MASKED)																			
K	SET RDY AND /OR STROBE LINES TO I/O LINES (BIT - MODE ONLY) (D1) 0 = PORT A, 1 = PORT B (D2) 0 = NO CHANGE TO RDY LINE FUNCTION, 1 = CHANGE PER BIT D6 (D3) 0 = NO CHANGE PER BIT D7, 1 = CHANGE PER BIT D7 (D4) RDY LINE OUTPUT DATA TO BE LOADED	1	0	1	0	1	D7	D6	D5	D4	D3	D2	D1	0												
														(D5) STROBE LINE OUTPUT DATA TO BE LOADED (D6) RDY LINE USED AS 0 = INPUT LINE 1 = OUTPUT LINE (D7) STROBE LINE USED AS 0 = INPUT LINE 1 = OUTPUT LINE												
L	ENABLE A INT OUTPUT	1	0	1	0	1	1	X	X	X	0	0	0	1												
M	ENABLE B INT OUTPUT	1	0	1	0	1	1	X	X	X	1	0	0	1												
N	DISABLE A INT OUTPUT	1	0	1	0	1	0	X	X	X	0	0	0	1												
P	DISABLE B INT OUTPUT	1	0	1	0	1	0	X	X	X	1	0	0	1												
R	READ STATUS REGISTER (D0) B INT STATUS (1 MEANS SET) (D1) A INT STATUS (1 MEANS SET) (D2) 1 = A INT WAS CAUSED BY B STROBE (D3) 1 = A INT WAS CAUSED BY A STROBE	1	0	1	1	0	D7	D6	D5	D4	D3	D2	D1	D0												
														<table border="1"> <tr> <td>(D4) A RDY INPUT DATA</td> <td rowspan="4">} BIT PROGRAMMABLE MODE</td> </tr> <tr> <td>(D5) A STROBE INPUT DATA</td> </tr> <tr> <td>(D6) B RDY INPUT DATA</td> </tr> <tr> <td>(D7) B RDY INPUT DATA</td> </tr> </table>		(D4) A RDY INPUT DATA	} BIT PROGRAMMABLE MODE	(D5) A STROBE INPUT DATA	(D6) B RDY INPUT DATA	(D7) B RDY INPUT DATA						
(D4) A RDY INPUT DATA	} BIT PROGRAMMABLE MODE																									
(D5) A STROBE INPUT DATA																										
(D6) B RDY INPUT DATA																										
(D7) B RDY INPUT DATA																										
S	READ PORT A	1	1	0	1	0	(INPUT DATA BYTE)																			
T	READ PORT B	1	1	1	1	0	(INPUT DATA BYTE)																			
U	LOAD PORT A	1	1	0	0	1	(OUTPUT DATA BYTE)																			
V	LOAD PORT B	1	1	1	0	1	(OUTPUT DATA BYTE)																			

Table II - CDP1851 Programming Modes

Mode	(8) Port A Data Pins	(2) Port A Handshaking Pins	(8) Port B Data Pins	(2) Port B Handshaking Pins
Input	Accept Input data	Ready Strobe	Accept Input data	Ready Strobe
Output	Output data	Ready Strobe	Output data	Ready Strobe
Bi-directional (Port A only)	Transfer input/output data	Input Handshaking for Port A	Must be previously set to bit-programmable mode	Output Handshaking for Port A
Bit-Programmable	Programmed individually as inputs or outputs	Programmed individually as inputs or outputs	Programmed individually as inputs or outputs	Programmed individually as inputs or outputs

VIS—A Commercially Competitive CRT Controller Chip Set

by R. Vaccarella

The CRT display has become very popular in the past few years for use in data terminals, personal computers, video games, graphics displays, and industrial instrumentation and display. In response to the increased demand for CRT controller electronics, much of the discrete control logic has become integrated into single-chip devices. These new devices interface easily to the popular microprocessors and provide increased 'intelligence' and simple operation. Since the majority of the CRT controllers available today are complex LSI circuits, the data sheets which describe their use are often difficult to interpret. Selecting the proper controller for a particular application may involve considerable time and expense for evaluation and prototyping.

The industry CRT controller comparison chart (Table I) points out the variety of features offered. As a starting point in choosing a CRT controller, it may be helpful to relate the desired application to the available devices by using the CRT Controller Selection Guide (Table II) which shows the relative merit of each device for some common applications, based on cost, capability, and overall system chip-count. After the initial selections are made the comparison chart and individual data sheets could be used to further narrow the choices.

Using the above method, it is evident that the RCA CDP1869, CDP1870/76 two-chip Video Interface System provides an

economical solution to a variety of CRT display applications. The flexibility and features offered by VIS are especially competitive since most are on-chip functions, contributing to a simplified application system with a very low chip-count.

Summary of VIS Competitive Features

- Standard CMOS advantages - Low power, high noise immunity, etc.
- Asynchronous operation - Independent screen-refresh operation
- Programmable display formats - More than adequate for industrial/commercial display applications and low-end video terminals
- Low system chip count - On-board oscillators, sync outputs, etc.
- Color and sound capability - 8 dot/8 background colors, RGB-color option, tone and white-noise generators
- NTSC and PAL compatible - U.S./European market
- ROM or RAM character memory - For non-standard character sets
- Teletext compatible - V and H inputs available for picture overlays
- Simple CDP1800-series interfacing - CDP1802, CDP1804, CDP1805
- Low system cost - Competitive system may require significant amount of support devices

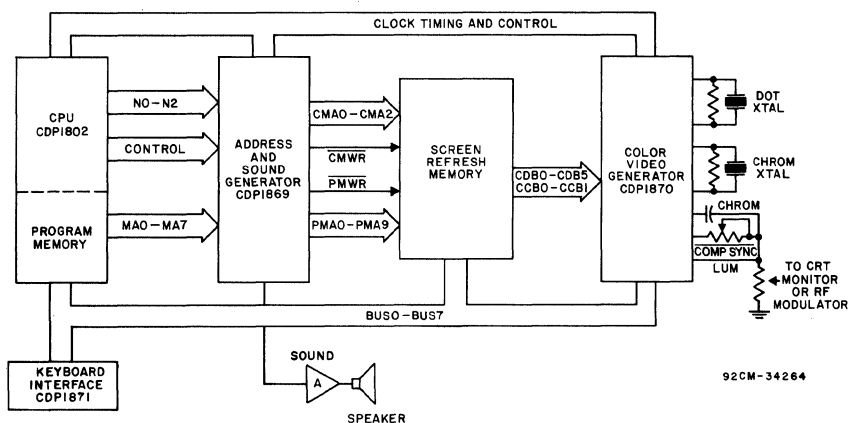


Fig. 1 - System diagram of a typical video system employing the VIS.

ICAN-7067

Table I - Industry CRT Controller Comparison

MANUFACTURER	TYPE NO.	TECHNOLOGY/ PACKAGE	μP/μC BUS COMPATIBILITY	COLOR CAPABILITY	LIGHT- PEN	SYNC OUTPUTS	VIDEO OUTPUTS	ASYNCHRONOUS SYSTEM OPERATION	ON-CHIP OSC	EXTERNAL PAGE MEMORY	EXTERNAL CHAR. MEMORY
RCA	CDP1889 CDP1870/ CDP1876	CMOS 40-PIN DIP +4-10.5 V	CDP1800 SERIES	YES	NO	HORZ AND COMPOSITE	LUMINANCE CHROMINANCE	YES	YES SYNC-DOT AND COLOR	2K x 8 RAM	256 CHARACTERS RAM/ROM
INTEL	18275	NMOS 40-PIN DIP +5 V	MCS80/85	NO	YES	NO	NO	NO DMA FROM PROCESSOR	NO	μP LIMITED- USES SYSTEM RAM	128 CHARACTERS ROM
INTEL	18276	NMOS 40-PIN DIP +5 V	MCS80/85	NO	NO	NO	NO	YES DUAL ROW- BUFFER OPERATION	NO	μP LIMITED- USES SYSTEM RAM	128 CHARACTERS ROM
NATIONAL	DP8580	JL 40-PIN DIP +5 V	MCS80/85	NO	NO	HORZ AND VERT	NO	YES	YES	4K x 8 RAM	256 CHARACTERS ROM
MOTOROLA	MC8845	NMOS 40-PIN DIP +5 V	MC8800 8500	NO	YES	HORZ AND VERT	NO	YES	NO	16K x 8 RAM	256 CHARACTERS ROM
SYNERTEK HITACHI AMI COMMODORE SEMICONDUCTOR	SYP8545 HD48505 S88045 MP8545										
MOTOROLA AMI FAIRCHILD	MC6847 S88047 F6847	NMOS 40-PIN DIP +5 V	MC8800 8500	YES	NO	HORZ VERT COMPOSITE SYNC/LUM	LUMINANCE B-Y R-Y	YES	YES DOT (R-C OSC)	6K x 8 RAM	512 CHARACTERS RAM/ROM INTERNAL MASK- ROM (64 CHARS)
SMC TI	CRT5027 TMS9927	NMOS 40-PIN DIP +5 V, +12 V	GENERAL PURPOSE TMS9900	NO	NO	HORZ VERT COMPOSITE	NO	YES	NO	8K x 8 RAM	256 CHARACTERS ROM
SMC	CRT9007	NMOS 40-PIN DIP +5 V	GENERAL PURPOSE	NO	YES	HORZ VERT COMPOSITE	NO	YES	NO	16K x 8 RAM	256 CHARACTERS ROM
SMC THOMSON CSF	CRT96364 96364	NMOS 40-PIN DIP +5 V	GENERAL PURPOSE (STAND-ALONE) CONTROLLER)	NO	NO	COMPOSITE	NO	YES	YES SYNC GEN	2K x 6 RAM (EXPANDABLE WITH EXT. HARDWARE)	64 CHARACTERS ROM
TI	TMS9918	NMOS 40-PIN DIP +5 V	GENERAL PURPOSE TMS9900	YES 2-TEXT 15-GRAPHIC 8-GRAY- SCALE	NO	COMPOSITE COLOR VIDEO SYNC BURST AND BLANKING		YES	YES SYNC- DOT COLOR	16K x 8 DYNAMIC RAM	256 CHARACTERS DYNAMIC RAM

PROGRAMMABLE FEATURES					EXTERNAL PARTS REQUIRED	COMMENTS
LINES/CHAR	CHARS/ROWS	ROWS/FRAME	CURSOR	OTHER		
8, 9, 16	20, 40	12, 24	N/A	HOME ADDRESS (SCROLL) 8 BK G COLORS/GRAY SCALE 8 DOT COLORS/GRAY SCALE TONE/WHITE NOISE GEN	DATA BUS BUFFER/SEPARATOR (8-BIT) VIDEO-SYNC MIXER CIRCUIT $\mu P/\mu C$	TWO CHIP SET NTSC/PAL COMPATIBLE, NON-INTERLACED SYNC EXTERNAL V AND H INPUTS (TELETEXT, OVERLAYS) RGB BOND-OUT OPTION (CDP1878)
UP TO 16	UP TO 80	UP TO 64	YES LOCATION BLINK RATE	HIGHLIGHT VIDEO REVERSE VIDEO UNDERLINE 11 NON-MEMORY GRAPHICS SPECIAL SCREEN CONTROL— CODES (END-OF-LINE BLANK, ETC.)	DMA CONTROLLER CHIP—18275 (40 PIN) DOT/CHAR CLOCK GEN. CIRCUIT SYNC TIMING GEN. CIRCUIT P/S SHIFT REG. (VIDEO OUTPUT) VIDEO-SYNC MIXER CIRCUIT $\mu P/\mu C$	OBTAINS DISPLAY DATA FROM μC SYSTEM RAM VIA DMA TECHNIQUE NON-INTERLACED SYNC
UP TO 16	UP TO 80	UP TO 64	YES LOCATION BLINK RATE	HIGHLIGHT VIDEO REVERSE VIDEO UNDERLINE 11 NON-MEMORY GRAPHICS SPECIAL SCREEN CONTROL— CODES (END-OF-LINE BLANK, ETC.)	DOT/CHAR CLOCK GEN. CIRCUIT SYNC TIMING GEN. CIRCUIT P/S SHIFT REG. (VIDEO OUTPUT) VIDEO-SYNC MIXER CIRCUIT $\mu P/\mu C$	SIMILAR TO 18275, BUT DOES NOT USE DMA TECHNIQUE LOWER COST OBTAINS DISPLAY DATA FROM $\mu P/\mu C$ SYSTEM RAM VIA ROW-BUFFER TECHNIQUE NON-INTERLACED SYNC
UP TO 16 ALSO UP TO 16 HORIZ DOTS/CHAR	5 TO 110	UP TO 64	YES LOCATION SIZE	HOME ADDRESS (SCROLL) HORIZ AND VERT SYNC	ADDRESS MUX (12-BIT) DATA BUS BUFFER/SEPARATOR (8-BIT) BUS LATCH (8-BIT) P/S SHIFT REG (VIDEO OUTPUT) VIDEO-SYNC MIXER CIRCUIT $\mu P/\mu C$	ALL PROGRAMMABLE FEATURES EXCEPT CURSOR LOCATION AND HOME ADDRESS ARE MASK-PROGRAMMABLE ONLY NON-INTERLACED SYNC
UP TO 32	UP TO 256 DOT RATE LIMITED	UP TO 128	YES LOCATION SIZE BLINK RATE	HORIZ AND VERT. SYNC H AND V DISPLAY WINDOW HOME ADDRESS (SCROLL) FIELD SYNC (NON-INTERLACED, INTERLACED-SYNC, INTER- LACED-SYNC AND VIDEO)	ADDRESS MUX (14-BIT) DATA BUS BUFFER/SEPARATOR (8-BIT) BUS LATCH (8-BIT) DOT/CHAR CLOCK GEN. CIRCUIT P/S SHIFT REG VIDEO-SYNC MIXER CIRCUIT $\mu P/\mu C$	NON-INTERLACED AND INTERLACED SYNC AND VIDEO 5Y545—TRANSPARENT ADDRESSING, ROW, COL ADDRESSING
12 NOT PROGRAM- MABLE	32 NOT PROGRAM- MABLE	16 NOT PROGRAM- MABLE	N/A	ALPHA MODE 4 DOT COLORS 4 BK G COLORS SEMIGRAPHICS MODE UP TO 8 COLORS GRAPHICS MODE UP TO 4 COLORS	ADDRESS MUX (12-BIT) DATA BUS BUFFER/SEPARATOR (8-BIT) PIA (6820) 4-BIT COUNTER (FOR EXT. CHAR ROM) VIDEO-SYNC MIXER CIRCUIT $\mu P/\mu C$	MC6847 AND 58047 ARE FUNCTIONALLY THE SAME, BUT HAVE DIFFERENT PIN-OUTS INTERLACE OPTION AVAILABLE (58047Y) CERTAIN MODES MAY BE CHANGED ON A CHARACTER BY CHARACTER BASIS (MINOR-MODE SWITCHING) HIGH DENSITY COLOR GRAPHICS CAPABILITY (256 x 192 ELEMENTS)
UP TO 16	20 TO 132 (8 PREDEFINED COMBINA- TIONS)	UP TO 64	YES LOCATION	HORIZ AND VERT SYNC HORIZ AND VERT DISPLAY— WINDOW LAST DISPLAYED DATA— ROW (SCROLL) INTERLACE/NON-INTERLACED	ADDRESS MUX (13-BIT) DATA BUS BUFFER/SEPARATOR (8-BIT) DOT/CHAR CLOCK GEN. CIRCUIT P/S SHIFT REG (VIDEO OUTPUT) VIDEO-SYNC MIXER CIRCUIT $\mu P/\mu C$	PROGRAMMABLE FEATURES ARE PROCESSOR, MASK, OR PROM PROGRAMMABLE SELF-LOAD MODE—(USES PROM TO LOAD REGISTERS) BALANCED-BEAM CURRENT OPTION (5037) LINE-LOCK OPTION (5027) PRE-PROGRAMMED OPTION OF THE 5037 (5047) COMPATIBLE CHARACTER GEN. AVAILABLE (7004) COMPATIBLE CHAR GEN/WITH GRAPHICS AVAILABLE (8002)
UP TO 16	8 - 240	2 - 256	YES LOCATION	HORIZ AND VERT SYNC DMA BURST MODE INTERLACE MODES (2) HORIZ SPLIT SCREEN HOME REGISTER (SCROLL) ROW-TABLE MODE INTERRUPT ENABLE	ADDRESS MUX (14-BIT) DATA BUS BUFFER/SEPARATOR 8-BIT) DOT/CHAR CLOCK GEN. CIRCUIT P/S SHIFT REG (VIDEO OUTPUT) VIDEO-SYNC MIXER CIRCUIT $\mu P/\mu C$	NON-INTERLACED AND INTERLACED SYNC AND VIDEO VITAL SCREEN PARAMETERS MASK—ROM VERSION AVAILABLE DMA OR ROW-BUFFER OPERATION COMPATIBLE ROW-BUFFER AVAILABLE (8006) COMPATIBLE CHAR GEN./WITH GRAPHICS AVAILABLE (8002)
8 NOT PROGRAM- MABLE	16 NOT PROGRAM- MABLE	64 NOT PROGRAM- MABLE	YES LOCATION	PAGE ERASE AND HOME END-OF-LINE ERASE/RETURN LINE FEED LINE ERASE CURSOR LEFT/RIGHT NORMAL CHAR/ADVANCE CURSOR	DATA GATE/BUFFER (4-BIT) CHARACTER DATA LATCH (4-BIT) DOT/CHAR CLOCK GEN. CIRCUIT P/S SHIFT REG (VIDEO OUTPUT) VIDEO-SYNC MIXER CIRCUIT COMMAND/DECODE ROM/PROM $\mu P/\mu C$ (OPTIONAL)	SCREEN FORMAT NOT PROGRAMMABLE AUTOMATIC SCROLL CAPABILITY STAND-ALONE SYSTEM CAPABILITY COMPATIBLE CHARACTER GEN. AVAILABLE (7004) COMPATIBLE CHAR GEN. WITH GRAPHICS AVAILABLE (8002) 50-Hz VERSION AVAILABLE (98364A)
8 NOT PROGRAM- MABLE	40 NOT PROGRAM- MABLE	24 NOT PROGRAM- MABLE	N/A	MODES —PATTERN GRAPHICS 32 BLOCKS x 24 LINES —MULTICOLOR GRAPHICS 64 BLOCKS x 48 LINES —TEXT GRAPHICS 40 CHARS x 24 LINES 36 VIDEO PLANES (SPRITES) EXT. VIDEO MIXING COLOR, PATTERN, BASE— ADDRESS REGISTERS	CONTROLLER SELECT LOGIC $\mu P/\mu C$	AUTO DYNAMIC RAM REFRESH 36 VIDEO PLANES (3-D EFFECT) BIT-MAPPED GRAPHICS EXTERNAL VIDEO-MIXING INPUT EXTERNAL SYNC INPUT EXTERNAL COLOR PHASE-LOCKING (INTERLACED AND NON-INTERLACED EXT. SOURCES) SEPARATE PATTERN AND COLOR TABLES HIGH RESOLUTION COLOR GRAPHICS CAPABILITY

Table II - CRT Controller Selection Guide

Display Application	Motorola Motorola									
	RCA	Intel	Intel	National	AMI	AMI	TI, SMC	SMC	SMC	TI
	CDP1869, 70/76	I8275	I8276	DP8350	MC6845	MC6847	TMS9927, CRT5027	CRT9007	CRT 96364	TMS9918
Low-Resolution Dumb Terminal	F	F	F	P	P	P	F	P	E	P
Low-Resolution Smart Terminal	G	G	G	F	G	G	E	G	P	G
High-Resolution Smart Terminal	P	G	E	G	G	P	G	G	P	P
High-Resolution Video Games	F	P	P	P	P	G	P	F	P	E
Broadcast TV Text Overlays	E	P	P	P	P	F	F	P	P	G
Low-Cost Industrial Displays	E	F	F	P	F	F	G	P	G	P
Low-Power Portable Displays (External CRT)	G	P	P	P	P	P	P	P	P	P
Automotive Test Equipment	G	F	F	F	F	F	F	F	P	F

E=Excellent G=Good F=Fair P=Poor

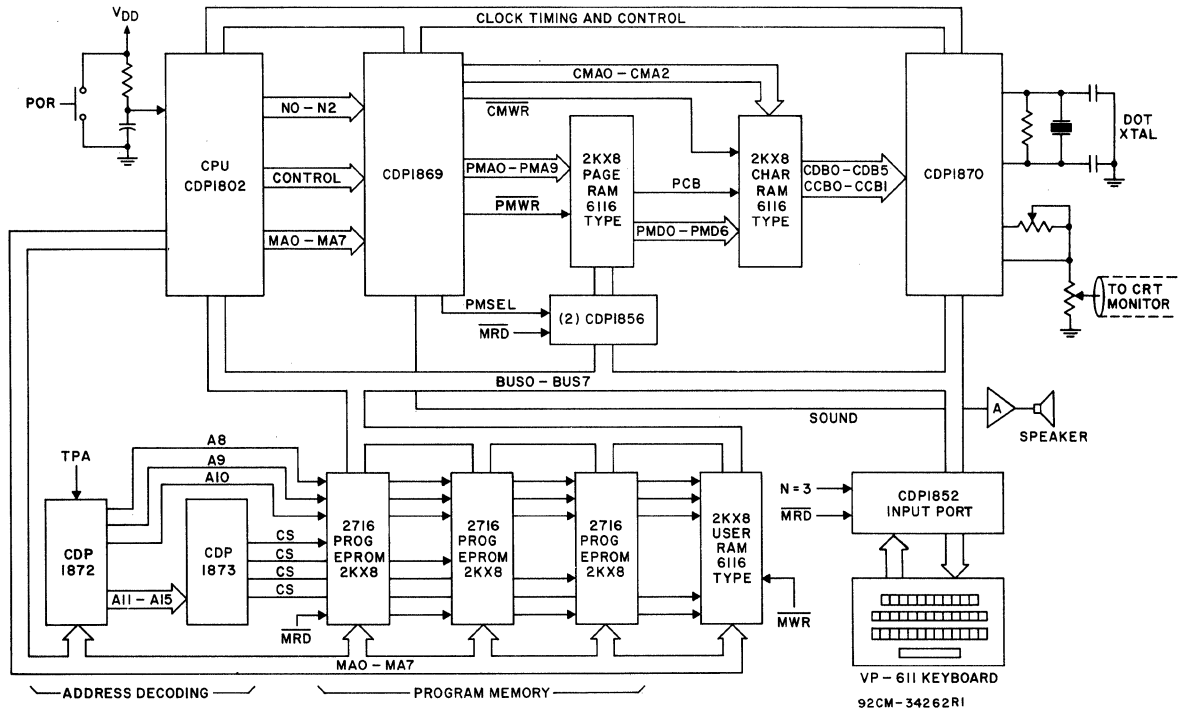


Fig. 2 - Block diagram of low-cost industrial display system employing the VIS.

Other Related Technical Publications

Product Descriptions

MB-601	Microboard Computer, CDP18S601
MB-602	Microboard Computer, CDP18S602
MB-603	Microboard Computer, CDP18S603
MB-604B	Microboard Computer, CDP18S604B
MB-620	Microboard 4-Kilobyte RAM, CDP18S620
MB-621	Microboard 16-Kilobyte RAM, CDP18S621
MB-622	Microboard 8-Kilobyte Battery-Backup RAM, CDP18S622
MB-623	Microboard 8-Kilobyte RAM, CDP18S623
MB-623A	Microboard 8-Kilobyte RAM CDP18S623A
MB-624	Microboard 4-Kilobyte Battery-Backup RAM, CDP18S624
MB-625	Microboard 8/16/32-Kilobyte ROM/PROM, CDP18S625
MB-626	Microboard 32/64-Kilobyte EPROM/ROM/RAM
MB-627	Microboard 4-Kilobyte CMOS EPROM CDP18S627
MB-629	Microboard 32-Kilobyte RAM
MB-640	Microboard Control and Display Module, CDP18S640
MB-640V1	Microboard Control and Display Module, CDP18S640V1
MB-641	Microboard UART Interface, CDP18S641
MB-642	Microboard D/A Converter, CDP18S642
MB-643	Microboard A/D Converter, CDP18S643
MB-644	Microboard A/D and D/A Converters, CDP18S644 and CDP18S654
MB-647	Microboard D/A Converters, CDP18S647 and CDP18S657
MB-648	Microboard A/D Converters, CDP18S648 and CDP18S658
MB-659	Microboard Breadboard, CDP18S659
MB-660	Microboard Combination Memory and I/O Module, CDP18S660
MB-661	Microboard Video-Audio-Keyboard Interface, CDP18S661V1 and CDP18S661V3
MB-661B	Microboard Video/Audio/Keyboard Interface CDP18S661B
MB-670	Microboard 22-Card Chassis with Integral Power Supply CDP18S670
MB-675	Microboard 5-Card Chassis, CDP18S675 and CDP18S676 (with case)
MB-691	Microboard Prototyping System, CDP18S691 and CDP18S691V3
MB-692	Microboard Prototyping System, CDP18S692
MPM-920B	CDP1802 Microprocessor Instruction Summary
PD6	Fixed-Point Arithmetic Subroutine
PD7	Floating-Point Arithmetic Subroutine
PD8	COSMAC Development System IV CDP18S008
PD10	COSMAC Dictionary
PD12	COSMAC Microterminal
PD13	Microboard Computer Development Systems (MCDS) CDP18S693 and CDP18S694
PD14	Color Microboard Computer Development System CDP18S695
PD16C	COSMAC Development Systems
PD17B	COSMAC Floppy Disk System II
PD18C	COSMAC Micromonitor
PD19	COSMAC UART Interface Module
PD20	COSMAC Byte I/O Module
PD22B	PROM Programmer
PD24	EK/Assembler-Editor
PD30	ROM Purchase Policy and Programming Instructions
PD31	Micromonitor Operating System (MOPS)
PD34	BASIC1 Compiler/Interpreter (CDP18S834)
PD37A	Disk Operating System Upgrade Package (CDP18S837)
PD39	PLM 1800 High-Level Language Compiler (CDP18S839)
PD40	BASIC2 High-Level-Language Interpreter CDP18S840
PD44	Micro Concurrent PASCAL Cross-Compiler CDP18S844 and Interpreter/Kernel CDP18S852 and CDP18S853

Product Guides

CMB-250B	COSMAC Microboard Computer Systems
MPG-180D	COSMAC Microprocessor Product Guide
MPL-200	Microsystems Product Guide and Price List
SPG-201L	Solid-State Product Guide

Technical Manuals

MPM-201C	User Manual for the RCA-CDP1802 COSMAC Microprocessor
MPM-206A	Fixed-Point Binary Arithmetic Subroutines for RCA COSMAC Microprocessors
MPM-207	Floating-Point Arithmetic Subroutines for RCA COSMAC Microprocessors
MPM-212	Instruction Manual for RCA COSMAC Microterminal
MPM-216A	Operator Manual for the RCA COSMAC Development System II CDP18S005
MPM-217A	RCA COSMAC Floppy Disk System II CDP18S805 Instruction Manual
MPM-218A	Instruction Manual for the RCA COSMAC Micromonitor CDP18S030
MPM-222A	Operator's Manual for PROM Programmer CDP18S480
MPM-223A	Instruction Guide for the COSMAC Macro Assembler (CMAC)
MPM-224	Instruction Manual for the RCA COSMAC Evaluation Kit CDP18S020 and the EK/Assembler-Editor Design Kit CDP18S024
MPM-231A	Micromonitor Operating System (MOPS) CDP18S831 User's Guide
MPM-232	Operator Manual for the RCA COSMAC DOS Development System (CDS III) CDP18S007
MPM-233	Hardware Reference Manual for the RCA COSMAC DOS Development System (CDS III) CDP18S007
MPM-234	Use of BASIC1 Compiler/Interpreter CDP18S834 with RCA COSMAC DOS Development System (CDS III)
MPM-235A	Operator Manual for the RCA COSMAC Development System IV CDP18S008
MPM-236	Hardware Reference Manual for the RCA COSMAC Development System IV CDP18S008
MPM-239A	User Manual for the RCA COSMAC PLM 1800 High-Level-Language Compiler
MPM-291	User Manual for the RCA COSMAC Microboard Prototyping System CDP18S691 and Control and Display Module CDP18S640
MPM-292	User Manual for the RCA COSMAC Microboard Prototyping System CDP18S692 and Control and Display Module CDP18S640V1
MPM-293	User Manual for the RCA COSMAC Microboard Computer Development System (MCDS) CDP18S693 and CDP18S694

DATABOOK

SSD-260	Memories, Microprocessors, and Support Systems
---------	--

**RCA Sales Offices,
Authorized Distributors and
Manufacturers' Representatives**

RCA Sales Offices

Argentina	Ramiro E. Podetti Reps. P.O. Box 4622 Buenos Aires 1000 Tel: 393-4029	U.S.		Michigan	RCA 30400 Telegraph Road, Suite 440, Birmingham, MI 48010 Tel: (313) 644-1151
Belgium	RCA S.A. Mercure Centre, Rue de la Fusée 100, 1130 Bruxelles Tel: 02/720.89.80	Alabama	RCA Suite 133 303 Williams Avenue Huntsville, AL 35801 Tel: (205) 533-5200	Minnesota	RCA 6750 France Avenue, So., Suite 122, Minneapolis, MN 55435 Tel: (612) 929-0676
Brazil	RCA Solid State Limitada Av. Brig Faria Lima 1476 7th Floor, Sao Paulo 01452 Tel: 210-4033	Arizona	RCA 6900 E. Camelback Road, Suite 460, Scottsdale, AZ 85251 Tel: (602) 947-7235	New Jersey	RCA 1998 Springdale Road, Cherry Hill, NJ 08003 Tel: (609) 338-5042
Canada	RCA Inc. 6303 30th Street, SE Calgary, Alberta T2C 1R4 Tel: (403) 279-3384 RCA Inc. 21001 No. Service Road, Trans-Canada Highway, St. Anne de Bellevue, Quebec H9X 3L3 Tel: (514) 457-2185/2189 RCA Inc. 1 Vulcan Street, Rexdale, Ontario M9W 1L3 Tel: (416) 247-5491	California	RCA 4546 El Camino Real, Los Altos, CA 94022 Tel: (415) 948-8996 RCA 8333 Clairemont Mesa Blvd Suite 105, San Diego, CA 92111 Tel: (714) 279-0420 RCA 4827 No. Sepulveda Blvd, Suite 420, Sherman Oaks, CA 91403 Tel: (213) 468-4200 RCA 17731 Irvine Blvd, Suite 104 Magnolia Plaza, Tustin, CA 92680 Tel: (714) 832-5302	New York	RCA 160 Perinton Hill Office Park Fairport, NY 14450 Tel: (716) 223-5240 RCA One Huntington Quadrangle Suite 2C14, Huntington Station, LI, NY 11746 Tel: (516) 293-0180
France	RCA S.A. 32, Rue Fessart 92100 Boulogne Tel: (01) 603 87 87	Colorado	RCA Corp. 6767 So. Spruce Street Englewood, CO 80112 Tel: (303) 740-8441	Ohio	RCA 29525 Chagrin Blvd., Pepper Pike, OH 44122 Tel: (216) 831-0030
Hong Kong	RCA International Ltd. P.O. Box 112, Hong Kong Tel: 852-5-234-181	Florida	RCA P.O. Box 12247, Lake Park, FL 33403 Tel: (305) 626-6350 RCA 1850 Lee Road, Suite 135 Winter Park, FL 32789 Tel: (305) 647-7100	Texas	RCA Center 4230 LBJ at Midway Road Town No. Plaza, Suite 121 Dallas, TX 75234 Tel: (214) 661-3515
Italy	RCA SpA Piazza San Marco 1, 20121 Milano Tel: (02) 65.97.048-051	Illinois	RCA 2700 River Road, Des Plaines IL 60018 Tel: (312) 391-4380	Virginia	RCA 1901 N. Moore St., Arlington, VA 22209 Tel: (703) 558-4161
Mexico	RCA S.A. de C.V./ Solid State Div., Avenida Cuiclahuac 2519, Apartado Postal 17-570, Mexico 16, D.F. Tel: (905) 399-7228	Indiana	RCA 4410 Executive Blvd, Suite 13A Fort Wayne, IN 46808 Tel: (219) 484-6460 RCA 9240 N. Meridian Street, Suite 102, Indianapolis, IN 46260 Tel: (317) 267-6375	West Germany	RCA GmbH Pfungstrosenstrasse 29, 8000 Munchen 70 Tel: 08971/43047-049 RCA GmbH Justus-von-Liebig-Ring 10 2085 Quickborn Tel: 04106/2001 RCA GmbH Zeppelinstrasse 35, 7302 Ostfildern 4 (Kemnat) Tel: 071145/4001-04
Singapore	RCA Corporation Solid State Division, 2315 Inter- national Plaza, 10 Anson Road, Singapore 0207 Tel: 2224156/2224157	Kansas	RCA 8900 Indian Creek Parkway, Overland Park, KS 66210 Tel: (913) 642-7656		
Sweden	RCA International LTD Box 3047, Hagalundsgatan 8 171 03 Solna 3 Tel: 08/83 42 25	Massachusetts	RCA 20 William Street, Wellesley, MA 02181 Tel: (617) 237-7970		
Taiwan	RCA Corporation Solid State Division, 7th Floor, 97 Nanking East Road, Section 2 Taipei Tel: 571-9171-5				
U.K.	RCA LTD Lincoln Way, Windmill Road Sunbury-on-Thames Middlesex TW 16 7HW Tel: 093 27 85511				

RCA Authorized Distributors

Argentina	<p>Eneka S.A.I.C.F.I. Tucuman 299, 1049 Buenos Aires</p> <p>Radiocom S.A. Conesa 1003, 1426 Buenos Aires</p> <p>Tecnos S.R.L. Independencia 1861 1225 Buenos Aires</p> <p>Galli Hnos, S.A.C.I. e Inm. Canada Entre Rios 628 Buenos Aires</p> <p>L.A.D.E., S.R.L. Sequrola 1879 Buenos Aires 1407</p>	<p>Teleradio Eletronica Ltda. Rua Vergueiro, 3134 CEP-04102 Sao Paulo/SP Tel: (011) 544-1245</p> <p>V.A.M. Importacoes, Vendas E Representacoes Ltda. Rua Correa Dutra, 126 - 4 Andar 4050 Jean Talon Street, West Montreal, Quebec H4P 1W1 Tel: (021) 225-5182</p> <p>Cesco Electronics, Ltd. 4050 Jean Talon Street, West Montreal, Quebec H4P 1W1 Tel: (514) 735-5511</p> <p>Cesco Electronics, Ltd. 909 Blvd, Charest Quest Quebec City, Quebec G1K 6W8 Tel: (418) 524-4641</p> <p>Electro Sonic, Inc. 1100 Gordon Baker Road Willowdale, Ontario M2H 3B3 Tel: (416) 494-1666</p> <p>Hamilton Avnet (Canada) Ltd 210 Colonnade Street Nepean, Ontario K2E 7L5 Tel: (613) 226-1700</p> <p>Hamilton Avnet Elec. 2816 21st St. N.E., Calgary Alberta, T2E 6Z2 Tel: (403) 230-3586</p> <p>Hamilton Avnet (Canada) Ltd 6845 Aexwood Drive Units 3-4-5 Mississauga, Ontario L4V 1M5 Tel: (416) 677-7432</p> <p>Hamilton Avnet (Canada) Ltd 2670 Sabourin Street, St. Laurent, Quebec H4S 1M2 Tel: (514) 331-6443</p> <p>L. A. Varah, Ltd 1832 King Edward Street Winnipeg, Manitoba R2R 0N1 Tel: (204) 633-6190</p> <p>L. A. Varah, Ltd 2077 Alberta Street, Vancouver, B.C. V5Y 1C4 Tel: (604) 873-3211</p> <p>L. A. Varah, Ltd. 4742 14th Street, NE Calgary, Alberta T2E 6L7 Tel: (403) 276-8818</p> <p>L. A. Varah, Ltd 505 Kenara Avenue, Hamilton, Ontario L8E 3P2 Tel: (416) 561-9311</p> <p>R.A.E. Industrial Electronics, Ltd 3455 Gardner Court, Burnaby, B.C. V5G 4J7 Tel: (604) 291-8866</p> <p>Raylex Ltda. Av Providencia 1244, 3Er Piso "D", Casilla 13373, Santiago</p> <p>Amplitel Ltda. Pedro Leon Ugalde 1464 Santiago</p> <p>Industria de Radio y Television S.A. (IRT) Vic. MacKenna 3333 Casilla 170-D, Santiago</p>	Colombia	<p>Miguel Antonio Pena Pena Y Cia. S. En C. Carrera 12 #1906 Bogota</p> <p>Electronica Moderna Carrera 9A, NRO 19-52 Apartado Aereo 5361, Bogota</p> <p>Jose E Marulanda Montoya Carrera 10, NRO 15-39 Of. 701 Apartado Aereo 3697, Bogota</p>
Australia	<p>AWA Microelectronics 348 Victoria Road Rydalmere N.S.W. 2116</p> <p>Amtron Tyree 176 Botany Street, Waterloo, N.S.W. 2017</p>		Costa Rica	<p>Electro-Impex, S.A. Avenida 10, Calles 10 Y 12 San Jose</p> <p>Gallito Tecnico, S.A. Av. 2 Calles 4 Y 6, San Jose</p> <p>J. G. Valdeperas, S.A. Calle 1, Avenidas 1-3, Apartado Postal 3923 San Jose</p>
Austria	<p>Bacher Elektronische Gerate GmbH Rotenmuhlgasse 26, A-1120 Vienna Tel: 0222/8356460</p>		Denmark	<p>Tage Olsen A/S P.O. Box 225 DK - 2750 Ballerup Tel: 02/65 81 11</p>
Bahamas	<p>Home Furniture Company Ltd. Post Office Box 331, Nassau</p>		Dominican Republic	<p>Humberto Garcia, C. por A. El Conde 366, Santo Domingo MAIL ADDRESS: Apartado de Correos 771, Santo Domingo</p>
Barbados	<p>Da Costa and Musson Ltda. Carlisle House Hincks Street P.O. Box 103 Bridgetown Tel: 608-50</p>		Ecuador	<p>Elecom, S.A. Padre Solano 202-OF. 8, P.O. Box 9611, Guayaquil</p>
Belgium	<p>Inelco (Belgium) S.A. avenue des Croix de Guerre 94 1120 Bruxelles Tel: 02/216.01.60</p>		Egypt	<p>Sakrcro Enterprises P.O. Box 1133, 37 Kasr El Nil Street, Apt. 5 Cairo Tel: 744440</p>
Bermuda	<p>DeFontes TV Centre Steed Building Parliament Street Hamilton</p>		El Salvador	<p>Radio Electrica, S.A. 4A Avenida Sur No. 228 San Salvador</p> <p>Radio Parts, S.A. 2A. C. O. No. 319 Postal la Dalia, San Salvador — MAIL ADDRESS: P.O. Box 1262 San Salvador, El Salvador</p>
Brazil	<p>Commercial Bezerra Ltda. Rua Costa Azevedo, 139, CEP- 69.000 Manaus/AM Tel: (092) 232-5363</p> <p>Organizacao Distribuidora E Representacoes Ltda. Rua Vigarrio Tenorio, 105-Conj. 102/402, CEP-50.000 Recife/PE Tel: (081) 224-2229</p> <p>Panamericana Comercial Importadora Ltda. Av. Rio Branco 307 Rua Aurora, 263, CEP-01209 Sao Paulo/SP Tel: (011) 222-3211</p> <p>Saturno Brasileiro Importacao Exportacao Ltda. Chile Rua Sacadura Cabral, 120, Sala 509, CEP-20.000 Rio de Janeiro/RJ Tel: (021) 243-4744</p> <p>Comercial Radio Lux Ltda. Av. Alberto Bins, 533 CEP-90.000 Porto Alegre/RS Tel: (051) 221-6055</p>		Finland	<p>Telercas OY P.O. Box 33 SF - 04201 Kerava Tel: 0/248.055</p>
			France	<p>Almex S.A. 48, rue de l'Aubepine, F - 92160 - Antony Tel: (01) 666 21 12</p> <p>Radio Equipments Antares S.A. 9, rue Ernest Cognacq, F - 92301 - Levallois Perret Tel: (01) 758 11 11</p> <p>Tekelec Airtronic S.A. Cite des Bruyeres, Rue Carle Vernet, F - 92310 - Sevres Tel: (01) 534.75.35</p>
			Greece	<p>Semicon Co 104 Aeolou Str. TT.131 Athens Tel: 3253626</p>
			Guatemala	<p>Electronica Guatemalteca 13 Calle 5-59, Zona 1 Guatemala City — MAIL ADDRESS: P.O. Box 514 Guatemala City, Guatemala</p>

RCA Authorized Distributors

	Tele-Equipos, S.A. 10A Calle 5-40, Zona 1 Guatemala City — MAIL ADDRESS: Apartado Postal 1798, Guatemala City, Guatemala	Japan	Okura & Company Ltd. 3-6 Ginza Nichome, Chuo-Ku Tokyo 104	Philippines	Philippine Electronic Industries, Inc. 3rd Floor, Rose Industrial Bldg, 11 Pioneer St. Pasig, Metro Manila
Haiti	Societe Haitienne D'Automobiles, S.A. Post Office Box 428, Port- Au-Prince	Korea	Panwest Company, Ltd. Room 312, Sam Duk Building 131, Da-Dong, Chung-Ku Seoul, Republic of Korea C.P.O. Box 3358		Semitronics 216 Ortega Street San Juan, Metro Manila 3134 P.O. Box 445
Holland	Inelco Nederland BV Turfstekestraat 63, N - 1431 GD Aalsmeer Tel: (02977) 2 88 55	Mexico	Tong Jin Trading Co. Room 1003, Bock-Chang Bldg. Sokong-Dong, Chung-Ku Seoul	Portugal	Teletra Sari Rua Rodrigo da Fonseca, 103 Lisbon 1 Tel: 68.60.72-75
	Vekano BV Postbus 6115, N - 5600 HC Eindhoven Tel: (40) 81 09 75		Electronica Remberg, S.A. de C.V. Republica del Salvador No. 30-102, Mexico City 1, D.F.	Puerto Rico	Kelvinator Sales of Puerto Rico, Inc. P.O. Box BG, Rio Piedras, Puerto Rico 00928
Honduras	Francisco J. Yones 3A Avenida S.O. 5, San Pedro Sula, Honduras, Central America		Mantenimiento E Instalaciones Internacionales, S.A. Calle 15 No. 79, Col. San Pedro de Los Pinos, Mexico 18, D.F.	Singapore	Edware Eu & Co., Ltd 1 Orchard Road Singapore
Hong Kong	Gibb Livingston & Co., Ltd. 77 Leighton Road Leighton Centre P.O. Box 55		Mexicana de Bulbos, S.A. Michoacan No. 30, Mexico 11, D.F.	South Africa	Allied Electronic Components (PTY) Ltd. P.O. Box 6387 Dunswart 1508 Tel: (011) 528.661
	Chinam Associates Suite 602-3 Ritz Building 625 Nathan Road Kowloon		Sprint S.A. San Juan de Letran #55 Pasaje Lopez Mexico 1, D.F.	Spain	Electrica Comercial Colominas S.A. Division Novoelectric Valencia 109-111, Barcelona 11 Tel: (03) 253.20.07
	Hong Kong Electronic Components Co. Flat A Yun Kai Bldg. 1/F1 466-472 Nathan Road Kowloon		Deksa, S.A. Av. Uno No. 129 Mexico 13, D.F.		Sisteco S.A. Corcega 167, Barcelona 36 Tel: (03) 321.73.47/92 (03) 322.42.05/52
Iceland	Georg Amundason P.O. Box 698, Reykjavik Tel: 81180	Nepal	Enrique Devesa Ramos San Juan de Letran #55 Local E, Mexico, D.F.		Kontron S.A. Salvaterra 4, Madrid 34 Tel: 1/729.11.55
India	Photophone (Comel) 179/5 Second Cross Road Lower Palace Orchards Bangalore 3		Raytel, S.A. Sullivan 47 Y 49, Mexico 4, D.F.		
Indonesia	NVPD Soedarmo Corp. Samudera Indonesia Building JL Letten S. Parman KAV. 35/Sliipi Jakarta Barat	Netherlands Antilles	Continental Commercial Distributors Durbar Marg Kathmandu	Sri Lanka	Ceylon SVC & Sup Co c/o P.A. Silva P.O. Box 89 Colombo
	Italy	New Zealand	El Louvre, S.A. Post Office Box 138, Curacao	Surinam	Kirpalani's Ltd. 17-27 Maagdenstreet, P.O. Box 251, Paramaribo
	Eledra 3S SpA Viale Elvezia 18, I - 20154, Milano Tel: (02) 349751	Nicaragua	AWA NZ Ltd. 36-44 Adelaide Road P.O. Box 830 Wellington 2		Surinam Electronics Keizerstreet 206, Paramaribo- MAIL ADDRESS: P.O. Box 412, Paramaribo, Surinam
	IDAC Elettronica SpA Via Turazza 32, I - 35100 Padova Tel: (049) 66.02.22	Norway	Comercial F. A. Mendieta, S.A. Apartado Postal No. 1956 C.S.T. 5C. Al Sur 2 1/2C Abajo a Radio Sandino, Managua	Sweden	Ferner Electronics AB Snormakarvagen 35, P.O. Box 125, S-161 26 Bromma Stockholm Tel: 08/80 25 40
	LASI Elettronica SpA Viale Lombardia 6, I - 20092 Cinisello Balsamo (MI) Tel: (02) 61.20.441-5	Panama	National Elektro A/S Ulvenveien 75, P.O. Box 53 Okern, Oslo 5 Tel: (472) 64 49 70		Lagercrantz Elektronik AB Kanalvagen 5, P.O. Box 48 S-194 21 Upplands Vasby Tel: 0760-861 20
	Silverstar Ltd. Via dei Gracchi 20, I - 20146 Milano Tel: (02) 49.96		Sistelcom, S.A. Ave. Mexico y Ave. Ecuador Edificio Albertina No. 3 Panama 5	Switzerland	Baerlocher AG Forrlibuckstrasse 110 8005 Zurich Tel: (01) 42.99.00
	DEDO Elettronica SpA Strada Statale 16 Km 403-550 64019 Tortoreto Lido (Te) Tel: 861/78.134	Peru	Tropelco, S.A. Via Espana 20-18, Panama 7- MAIL ADDRESS: P.O. Box 8465, Panama 7, Rep. of Panama	Taiwan	Hwa Sheng Electric Co., Ltd. 5th Floor Pong Lai Building 245 Min Chuan East Road Taipei
			Arven S.A. PSJ Adan Mejia 103, OF. 33 Lima 11		Delta Engineering Ltd. No. 42 Hsu Chang Street 8th Floor, Taipei
			Deltron S.A. Apartado Postal 1574 Lima		

RCA Authorized Distributors

Thailand	Anglo Thai Engineering Ltd. 2160 Klongton-Bangkapi Hwy Hua Mark, Bangkok	Avnet Electronics 350 McCormick Avenue Costa Mesa, CA 92626 Tel: (714) 754-6051	Colorado	Hamilton Avnet Electronics 8765 E. Orchard Road, Suite 708, Englewood, CO 80111 Tel: (303) 740-1000
Trinidad	Kirpalani's Limited Kirpalani's Komplex Churchill Roosevelt Highway San Juan, Port-of-Spain	Electronic Supply Corp. 2486 Third Street Riverside, CA 92507 Tel: (714) 683-7300		Arrow Electronics Inc. 5465 E. Evans Place at Hudson Denver, CO 80222 Tel: (303) 758-2100
Turkey	Teknim Company Ltd. Riza Sah Pehlevi Caddesi 7 Kavaklidere Ankara Tel: 27.58.00	Elmo Semiconductor Corp. 915 North Citrus Avenue Los Angeles, CA 90038 Tel: (213) 465-2163		Kierulff Electronics, Inc. 10890 East 47th Avenue Denver, CO 80239 Tel: (303) 371-6500
U.K.	Crellon Electronics Ltd. 380 Bath Road, Slough, Berks, SL1 6JE Tel: Burnham (06286) 4434	Hamilton Avnet Electronics 3170 Pullman Street Costa Mesa, CA 92626 Tel: (714) 522-8200		Wyle Distribution Group 451 East 124th Avenue Thornton, CO 80241 Tel: (303) 457-9953
	I.T.T. Electronic Services Edinburgh Way, Harlow Essex, CM20 2DE Tel: Harlow (0279) 26777	Hamilton Avnet Electronics 1175 Bordeaux Drive Sunnyvale, CA 94086 Tel: (408) 743-3300	Connecticut	Arrow Electronics, Inc. 12 Beaumont Road Wallingford, CT 06492 Tel: (203) 265-7741
	Jermyn Distribution Vestry Industrial Estate Sevenoaks, Kent Tel: Sevenoaks (0732) 450144	Hamilton Avnet 4545 Viewridge Avenue San Diego, CA 92123 Tel: (714) 571-7510		Hamilton Avnet Electronics Commerce Drive, Commerce Park, Danbury, CT 06810 Tel: (203) 797-1100
	Macro Marketing Ltd. Burnham Lane, Slough, Berkshire SL1 6LN Tel: Burnham (06286) 4422	Hamilton Electro Sales 10912 W. Washington Blvd Culver City, CA 90230 Tel: (213) 558-2020		Schweber Electronics Corp. Finance Drive, Commerce Industrial Park, Danbury, CT 06810 Tel: (203) 792-3500
	Semicomps Northern Ltd. East Bowmont Street, Kelso, Roxburghshire, Scotland Tel: Kelso (05732) 2366	Kierulff Electronics, Inc. 2585 Commerce Way Los Angeles, CA 90040 Tel: (213) 725-0325	Florida	Arrow Electronics, Inc. 1001 NW 62nd Street, Suite 108, Ft. Lauderdale, FL 33309 Tel: (305) 776-7790
	VSI Electronics (U.K.) Ltd. Roydonbury Industrial Park Horsecroft Road, Harlow Essex CM19 5 BY Tel: Harlow (0279) 29666	Kierulff Electronics, Inc. 3969 E. Bayshore Road Palo Alto, CA 94303 Tel: (415) 968-6292		Arrow Electronics, Inc. 50 Woodlake Dr., West-Bldg.B Palm Bay, FL 32905 Tel: (305) 725-1480
	ACCESS Electronic Components Ltd. Austin House, Bridge Street Hitchin, Hertfordshire SG5 2DE Tel: Hitchin (0462) 31 221	Kierulff Electronics, Inc. 8797 Balboa Avenue San Diego, CA 92123 Tel: (714) 278-2112		Hamilton Avnet Electronics 6801 NW 15th Way Ft. Lauderdale, FL 33068 Tel: (305) 971-2900
Uruguay	American Products S.A. (APSA) Av. Italia 4230 Montevideo Tel: 594210	Kierulff Electronics, Inc. 14101 Franklin Avenue Tustin, CA 92680 Tel: (714) 731-5311		Hamilton Avnet Electronics 3197 Tech Drive, No. St. Petersburg, FL 33702 Tel: (813) 576-3930
U.S.		Marshall Industries 9674 Telstar Avenue El Monte, CA 91731 Tel: (213) 686-0141		Schweber Electronics Corp. 2830 North 28th Terrace Hollywood, FL 33020 Tel: (305) 927-0511
Alabama	Hamilton Avnet Electronics 4692 Commercial Drive, NW Huntsville, AL 35805 Tel: (205) 837-7210	Marshall Industries 17321 Murphy Avenue Irvine, CA 92714 Tel: (714) 556-6400	Georgia	Arrow Electronics, Inc. 2979 Pacific Drive Norcross, GA 30071 Tel: (404) 449-8252
Arizona	Hamilton Avnet Electronics 505 South Madison Drive Tempe, AZ 85281 Tel: (602) 894-9600	RPS Electronics, Inc. 6230 Descanso Avenue Buena Park, CA 90620 Tel: (714) 521-5230		Hamilton Avnet Electronics 5825 Peach Tree Corners Norcross, GA 30071 Tel: (404) 449-9170
	Kierulff Electronics, Inc. 4134 East Wood Street Phoenix, AZ 85040 Tel: (602) 243-4101	Schweber Electronics Corp. 17811 Gillette Avenue Irvine, CA 92714 Tel: (714) 556-3880		Schweber Electronics, Inc. 4126 Pleasantdale Road Atlanta, GA 30340 Tel: (404) 449-9170
	Sterling Electronics, Inc. 2001 East University Drive, Phoenix, AZ 85034 Tel: (602) 258-4531	Schweber Electronics Corp. 3110 Patrick Henry Drive Santa Clara, CA 95050 Tel: (408) 496-0200	Illinois	Arrow Electronics, Inc. 492 Lunt Avenue Schaumburg, IL 60193 Tel: (312) 893-9420
	Wyle Distribution Group 8155 North 24th Avenue Phoenix, AZ 85022 Tel: (602) 249-2232	Wyle Distribution Group 124 Maryland Avenue El Segundo, CA 90245 Tel: (213) 322-8100		Hamilton Avnet Electronics 1130 Thorndale Avenue Bensenville, IL 60166 Tel: (312) 678-6310
California	Arrow Electronics, Inc. 9511 Ridge Haven Court San Diego, CA 92123 Tel: (714) 565-4800	Wyle Distribution Group 9529 Chesapeake Drive San Diego, CA 92123 Tel: (714) 565-9171		Newark Electronics 500 North Pulaski Road Chicago, IL 60624 Tel: (312) 638-4411
	Arrow Electronics, Inc. 521 Weddell Sunnyvale, CA 94086 Tel: (408) 739-3011	Wyle Distribution Group 3000 Bowers Avenue Santa Clara, CA 95052 Tel: (408) 727-2500		

RCA Authorized Distributors

- Schweber Electronics Corp.**
1275 Brummel Avenue
Elk Grove Village, IL 60007
Tel: (312) 364-3750
- Indiana**
- Arrow Electronics, Inc.**
2718 Rand Road
Indianapolis, IN 46241
Tel: (317) 243-9353
- Graham Electronics Supply, Inc.**
133 S. Pennsylvania Street
Indianapolis, IN 46204
Tel: (317) 634-8202
- Hamilton Avnet Electronics, Inc.**
485 Gradle Drive
Carmel, IN 46032
Tel: (317) 844-9333
- Iowa**
- Deeco, Inc.**
2500 16th Avenue, SW
Cedar Rapids, IA 52406
Tel: (319) 365-7551
- Kansas**
- Hamilton Avnet Electronics**
9219 Quivira Road
Overland Park, KS 66215
Tel: (913) 888-8900
- Louisiana**
- Sterling Electronics, Inc.**
4613 Fairfield Road
Metairie, LA 70002
Tel: (504) 887-7610
- Maryland**
- Arrow Electronics, Inc.**
4801 Benson Avenue
Baltimore, MD 21227
Tel: (301) 247-5200
- Hamilton Avnet Electronics**
6822 Oakhill Lane
Columbia, MD 21045
Tel: (301) 995-3500
- Pyttronic Industries, Inc.**
Baltimore/Washington
Industrial Pk, 8220 Wellmoor
Court, Savage, MD 20863
Tel: (301) 792-0780
- Schweber Electronics Corp.**
9218 Gaither Road
Gaithersburg, MD 20760
Tel: (301) 840-5900
- Zebra Electronics, Inc.**
2400 York Road
Timonium, MD 21093
Tel: (301) 252-6576
- Massachusetts**
- Arrow Electronics, Inc.**
Arrow Drive
Woburn, MA 01801
Tel: (617) 933-8130
- Hamilton Avnet Electronics**
50 Tower Office Park
Woburn, MA 01801
Tel: (617) 935-9700
- Kierulff Electronics, Inc.**
13 Fortune Drive
Billerica, MA 01821
Tel: (617) 667-8331
- Marshall Industries**
One Wilshire Road
Burlington, MA 01803
Tel: (617) 272-8200
- A. W. Mayer Co.**
38 Border Street
West Newton, MA 02165
Tel: (617) 965-1111
- Schweber Electronics Corp.**
25 Wiggins Avenue
Bedford, MA 01730
Tel: (617) 275-5100
- Sterling Electronics, Inc.**
411 Waverly Oaks Road
Waltham, MA 02154
Tel: (617) 894-6200
- Michigan**
- Arrow Electronics, Inc.**
3810 Varsity Drive
Ann Arbor, MI 48104
Tel: (313) 971-8220
- Hamilton Avnet Electronics**
2215 29th Street
Grand Rapids, MI 49503
Tel: (616) 243-8805
- Hamilton Avnet Electronics**
32487 Schoolcraft Road
Livonia, MI 48150
Tel: (313) 522-4700
- RS Electronics, Inc.**
34443 Schoolcraft Road
Livonia, MI 48150
Tel: (313) 525-1155
- Schweber Electronics Corp.**
33540 Schoolcraft Road
Livonia, MI 48150
Tel: (313) 525-8100
- Minnesota**
- Arrow Electronics, Inc.**
5230 West 73rd Street
Edina, MN 55435
Tel: (612) 830-1800
- Hamilton Avnet Electronics**
10300 Bren Road, East
Minnetonka, MN 55343
Tel: (612) 932-0600
- Kierulff Electronics, Inc.**
5280 West 74th Street
Edina, MN 55435
Tel: (612) 835-4388
- Schweber Electronics Corp.**
7422 Washington Avenue, So.
Eden Prairie, MN 55344
Tel: (612) 941-5280
- Missouri**
- Arrow Electronics, Inc.**
2380 Schultz Road
St. Louis, MO 63141
Tel: (314) 567-6888
- Hamilton Avnet Electronics**
13743 Shoreline Court East
Earth City, MO 63045
Tel: (314) 344-1200
- New Hampshire**
- Arrow Electronics, Inc.**
1 Perimeter Drive
Manchester, NH 03103
Tel: (603) 668-6968
- New Jersey**
- Arrow Electronics, Inc.**
Pleasant Valley Avenue
Moorestown, NJ 08057
Tel: (609) 235-1900
- Arrow Electronics, Inc.**
285 Midland Avenue
Saddlebrook, NJ 07662
Tel: (201) 797-5800
- Hamilton Avnet Electronics**
10 Industrial Road
Fairfield, NJ 07006
Tel: (201) 575-3390
- Hamilton Avnet Electronics**
1 Keystone Avenue
Cherry Hill, NJ 08003
Tel: (609) 424-0100
- Kierulff Electronics, Inc.**
3 Edison Place
Fairfield, NJ 07006
Tel: (201) 575-6750
- Marshall Industries**
1111 Paulinus Avenue
Clifton, NJ 07015
Tel: (201) 340-1900
- Schweber Electronics Corp.**
18 Madison Road
Fairfield, NJ 07006
Tel: (201) 227-7880
- New Mexico**
- Arrow Electronics, Inc.**
2460 Alamo, SE
Albuquerque, NM 87106
Tel: (505) 243-4566
- Hamilton Avnet Electronics**
2524 Baylor SE
Albuquerque, NM 87106
Tel: (505) 765-1500
- Sterling Electronics, Inc.**
3540 Pan American
Freeway, N.E.
Albuquerque, NM 87107
Tel: (505) 884-1900
- New York**
- Arrow Electronics, Inc.**
900 Broad Hollow Road
Route 110, Farmingdale, LI,
NY 11735
Tel: (516) 694-6800
- Arrow Electronics, Inc.**
7705 Maltage Drive
Liverpool, NY 13008
Tel: (315) 652-1000
- Arrow Electronics, Inc.**
3000 South Winton Road
Rochester, NY 14623
Tel: (716) 275-0300
- Hamilton Avnet Electronics**
5 Hub Drive
Melville, NY 11746
Tel: (516) 454-6000
- Hamilton Avnet Electronics**
333 Metro Park
Rochester, NY 14623
Tel: (716) 475-9130
- Hamilton Avnet Electronics**
16 Corporate Circle
East Syracuse, NY 13057
Tel: (315) 437-2641
- Milgray Electronics, Inc.**
191 Hanse Avenue
Freeport, LI, NY 11520
Tel: (516) 546-6000
- Rochester Radio Supply Co.**
140 W. Main Street
Rochester, NY 14614
Tel: (716) 454-7800
- Schweber Electronics Corp.**
2 Town Line Circle
Rochester, NY 14623
Tel: (716) 424-2222
- Schweber Electronics Corp.**
Jericho Turnpike
Westbury, LI, NY 11590
Tel: (516) 334-7474
- Summit Distributors, Inc.**
916 Main Street
Buffalo, NY 14202
Tel: (716) 884-3450
- North Carolina**
- Arrow Electronics, Inc.**
938 Burke Street,
Winston-Salem, NC 27101
Tel: (919) 725-8711
- Hamilton Avnet Electronics**
2803 Industrial Drive
Raleigh, NC 27609
Tel: (919) 829-8030

RCA Authorized Distributors

	<p>Hammond Electronics of Carolina 2923 Pacific Avenue Greensboro, NC 27406 Tel: (919) 275-6391</p>	Texas	<p>Arrow Electronics, Inc. 13715 Gamma Road Dallas, TX 75240 Tel: (214) 386-7500</p>		<p>Wyle Distribution Group 1750 132nd Avenue, N.E. Bellevue, WA 98005 Tel: (206) 453-8300</p>
Ohio	<p>Arrow Electronics, Inc. 7620 McEwen Road Centerville, OH 45459 Tel: (513) 435-5563</p> <p>Arrow Electronics, Inc. 6238 Cochran Road Solon, OH 44139 Tel: (216) 248-3990</p> <p>Hamilton Avnet Electronics 4588 Emery Industrial Parkway Cleveland, OH 44128 Tel: (216) 831-3500</p> <p>Hamilton Avnet Electronics 954 Senate Drive Dayton, OH 45459 Tel: (513) 433-0610</p> <p>Hughes Peters, Inc. 481 East 11th Avenue Columbus, OH 43211 Tel: (614) 294-5351</p> <p>Schweber Electronics Corp. 23880 Commerce Park Road Beachwood, OH 44122 Tel: (216) 464-2970</p> <p>The Stotts Friedman Co. 2600 East River Road Dayton, OH 45439 Tel: (513) 298-5555</p>		<p>Arrow Electronics, Inc. 10700 Corporate Drive #100 Stafford, TX 77477 Tel: (713) 491-4100</p> <p>Hamilton Avnet Electronics 2401 Rutland Drive Austin, TX 78758 Tel: (512) 837-8911</p> <p>Hamilton Avnet Electronics 2111 West Walnut Hill Lane Irving, TX 75060 Tel: (214) 659-4111</p> <p>Hamilton Avnet Electronics 8750 Westpark Houston, TX 77063 Tel: (713) 975-3515</p> <p>Schweber Electronics Corp. 4202 Beltway, Dallas, TX 75234 Tel: (214) 661-5010</p> <p>Schweber Electronics Corp. 10625 Richmond Ste. 100 Houston, TX 77042 Tel: (713) 784-3600</p> <p>Sterling Electronics, Inc. 2335A Kramer Lane, Suite A Austin, TX 78758 Tel: (512) 836-1341</p>	Wisconsin	<p>Arrow Electronics, Inc. 430 West Rawson Avenue Oak Creek, WI 53154 Tel: (414) 764-6600</p> <p>Hamilton Avnet Electronics 2975 South Moorland Road New Berlin, WI 53151 Tel: (414) 784-4510</p> <p>Taylor Electric Company 1000 W. Donges Bay Road Mequon, WI 53092 Tel: (414) 241-4321</p>
			<p>Sterling Electronics, Inc. 11090 Stemmons Freeway (Stemmons at Southwell) Dallas, TX 75229 Tel: (214) 243-1600</p> <p>Sterling Electronics, Inc. 4201 Southwest Freeway Houston, TX 77027 Tel: (713) 627-9800</p> <p>Trevino Electronics, Inc. 2874 Walnut Hill Lane Dallas, TX 75229 Tel: (214) 358-2418</p>	Venezuela	<p>Dinaradio, C.A. Calle Madrid Entre Nueva York y Carolina, Quinta Tana Las Mercedes 107, Caracas</p> <p>Tele-Cuba, S.A. Av. Este O, No. 164, Ferrenquin a La Cruz, La Candelaria, Caracas</p> <p>P. Benavides, P., S.R.L. Avilanes a Rio Edificio Rio Caribe, Local 9 La Candelaria, Caracas</p>
Oklahoma	<p>Radio, Inc. 1000 S. Main Street Tulsa, OK 74119 Tel: (918) 587-9123</p>			West Indies	<p>Da Costa and Musson Ltd. Post Office Box 103, General Post Office, Barbados</p>
Oregon	<p>Hamilton Avnet Electronics 6024 SW Jean Road, Bldg B, Suite J, Lake Oswego, OR 97034 Tel: (503) 635-8159</p>			West Germany	<p>Alfred Neye Enatechnik GmbH Schillerstrasse 14, 2085 Quickborn Tel: 04106/6121</p> <p>ECS Hilmar Frehsdorf Electronic Components Service Carl-Zeiss Strasse 3 2085 Quickborn Tel: 04106/71058-59</p> <p>BECK GmbH & Co. Elektronik Bauelemente KG Eltersdorfer Strasse 7, 8500 Nurnberg 15 Tel: 0911/34961-66</p> <p>Elkose GmbH Bahnhofstrasse 44, 7141 Moglingen Tel: 07141/4871</p> <p>Sasco GmbH Hermann-Oberth-Strasse 16 8011 Putzbrunn bei Munchen Tel: 089/46111</p> <p>Spoerle Electronic KG Otto-Hahn-Strasse 13, 6072 Dreieich bei Frankfurt Tel: 06103/3041</p>
Pennsylvania	<p>Arrow Electronics, Inc. 650 Seco Road Monroeville, PA 15146 Tel: (412) 856-7000</p> <p>Herbach & Rademan, Inc. 401 East Erie Avenue Philadelphia, PA 19134 Tel: (215) 426-1700</p> <p>Schweber Electronics Corp. 101 Rock Road Horsham, PA 19044 Tel: (215) 441-0600</p>	Utah	<p>Hamilton Avnet Electronics 1585 West 2100 South Salt Lake City, UT 84119 Tel: (801) 972-2800</p>		<p>Alfred Neye Enatechnik GmbH Schillerstrasse 14, 2085 Quickborn Tel: 04106/6121</p> <p>ECS Hilmar Frehsdorf Electronic Components Service Carl-Zeiss Strasse 3 2085 Quickborn Tel: 04106/71058-59</p> <p>BECK GmbH & Co. Elektronik Bauelemente KG Eltersdorfer Strasse 7, 8500 Nurnberg 15 Tel: 0911/34961-66</p> <p>Elkose GmbH Bahnhofstrasse 44, 7141 Moglingen Tel: 07141/4871</p> <p>Sasco GmbH Hermann-Oberth-Strasse 16 8011 Putzbrunn bei Munchen Tel: 089/46111</p> <p>Spoerle Electronic KG Otto-Hahn-Strasse 13, 6072 Dreieich bei Frankfurt Tel: 06103/3041</p>
		Washington	<p>Hamilton Avnet Electronics 14212 N.E. 21st Street Bellevue, WA 98005 Tel: (206) 746-8750</p> <p>Robert E. Priebe Company 2211 5th Avenue Seattle, WA 98121 Tel: (206) 682-8242</p>		<p>Avtotehna P.O. Box 593, Titova 36-XI Ljubljana 61000 Tel: 317 044</p>

RCA Manufacturers' Representatives

U.S.

Arizona	<p>Thom Luke Sales, Inc. 2940 North 67th Place, Suite B Scottsdale, AZ 85251 Tel: (602) 941-1901</p>	Florida	<p>G. F. Bohman Associates 130 North Park Avenue Apopka, FL 32703 Tel: (305) 886-1882</p>	Washington	<p>Western Technical Sales, Inc. P.O. Box 3923, Bellevue, WA 98009 Tel: (206) 641-3900</p>
---------	---	---------	--	------------	---

