MOTOROLA
Semiconductor Products Inc.

AN-850
Application Note

MULTI-PROCESSOR CONTROLLER USING THE MC6809E AND THE MC68120

Prepared by
David L. Ruhberg
and
Michael C. Wood
Microprocessor Applications Engineering
Austin, Texas

As the demand for system performance increases, the design engineer is faced with the task of providing additional throughput. To obtain the increased performance, system flexibility should provide for additional expansion without the need for total redesign of the existing system. Two alternatives are available to the designer in developing any microprocessor system: single processor and multi-processor. This application note investigates both alternatives and describes a basic multi-processor system using Motorola's MC6809E and MC68120.

The single processor system is the more common approach in use, since one microprocessing unit (MPU) typically has been able to handle the system performance requirements. Hardware and software are both simpler with only one MPU on the bus; however, as system performance requirements continue to increase, the design engineer is faced with the job of either upgrading the system or redesigning a complete system. The characteristics of a single processor system should be reviewed before jumping into another single processor system redesign. Basically, the total growth of the single processor system is limited to the throughput rate of the MPU, so all future tasks and expansions must be taken into account at design time to avoid another complete system redesign. An MPU capable of handling all of the anticipated expansion must be selected. Thus, the MPU will not perform anywhere near its rated peak efficiency until the system is expanded. In any area where rapid system expansion is anticipated, the single processor system is a temporary solution at best.

The multi-processor configuration can eliminate the expansion problems which are present in a single MPU design. An interface containing a bus arbitrator and data transfer area common to both MPU buses could keep the buses separate and also allow the two systems to communicate. Thus, the simplicity of single bus systems can be maintained while obtaining the expansion capabilities of the multi-processor system. By adding more of these interfaces, the system expansion occurs by simply adding peripherals to an MPU bus. Two features utilized by the Motorola MC68120 Intelligent Peripheral Controller (IPC) provide the bus arbitrator and data transfer area for a multiple MPU system just described. These features are six semaphore registers and 128 bytes of dual-ported RAM. With the MC6809E MPU operating the system bus (master) and the MC68120 containing the system bus interface, as well as the CPU controlling the local bus (slave), the system now has the best features of both the single and multi-processor approaches.

TRADITIONAL MPU MULTI-PROCESSING

One of the most common multi-processor schemes has been a bi-phase technique in which both processors operate from opposite phases of a system clock (see Figure 1). The memory and peripherals are accessed during each MPU clock high time. This scheme has the benefit of lower costs due to the presence of only one bus; however, some of the cost savings may be consumed in circuitry required to synchronize the clocks and in buffers required to prevent bus contention. In order to debug the bi-phase system, most of the hardware and software in both of the MPU systems must be working. Also, care must be taken when all resources are available to both processors, as in this bi-phase configuration, to avoid inadvertently clearing status flags or making changes in RAM. The major drawback to this system is that the system is limited to two MPUs.

The multiple bus configuration can simplify or eliminate most of the constraints and limitations of the bi-phase approach (see Figure 2) provided a simple bus arbitration scheme is available. The debugging of this type of system is simplified since one bus can operate independent of the other, except when the buses need to communicate with each

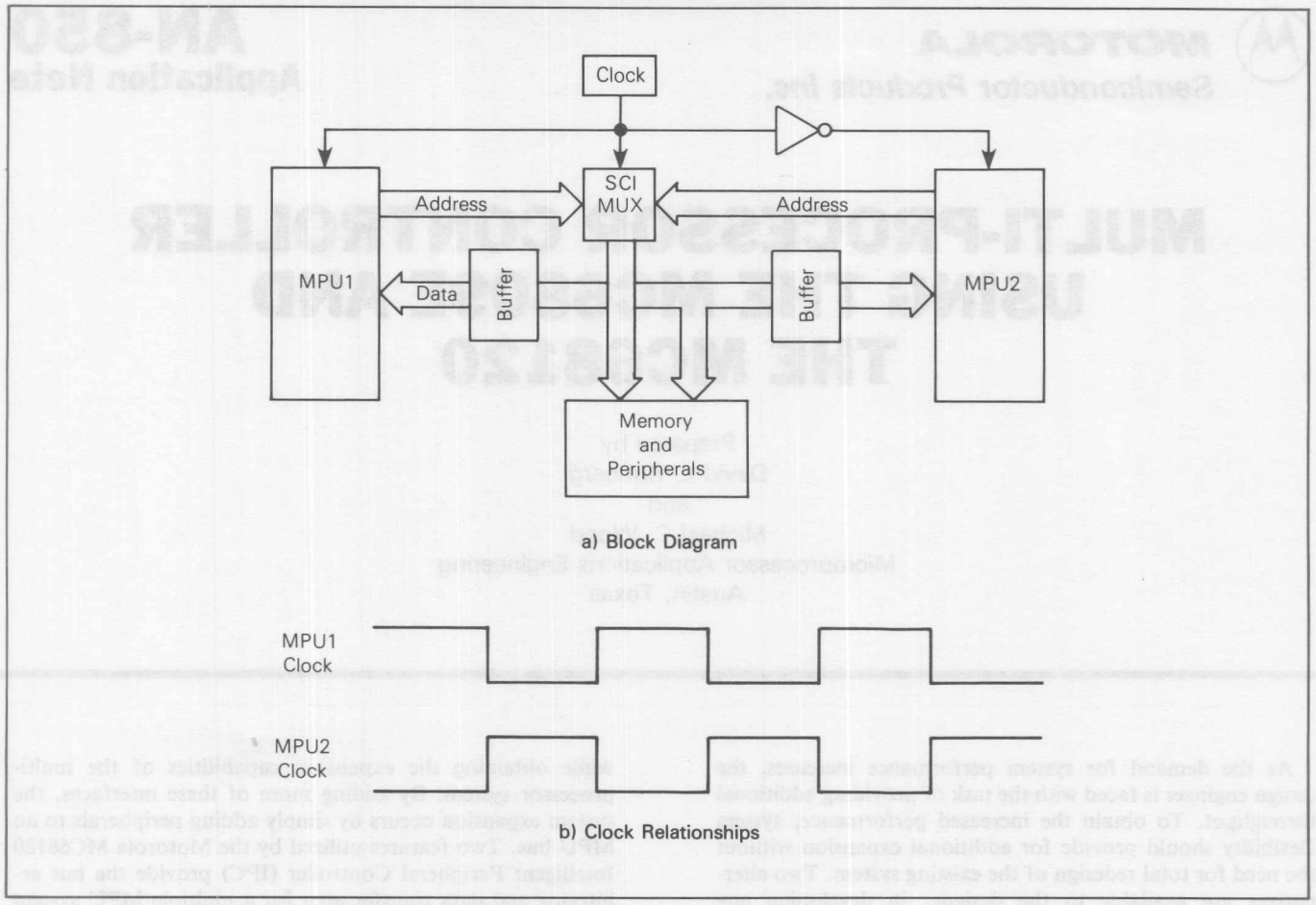

FIGURE 1 — Multi-Processor Bi-Phase Technique

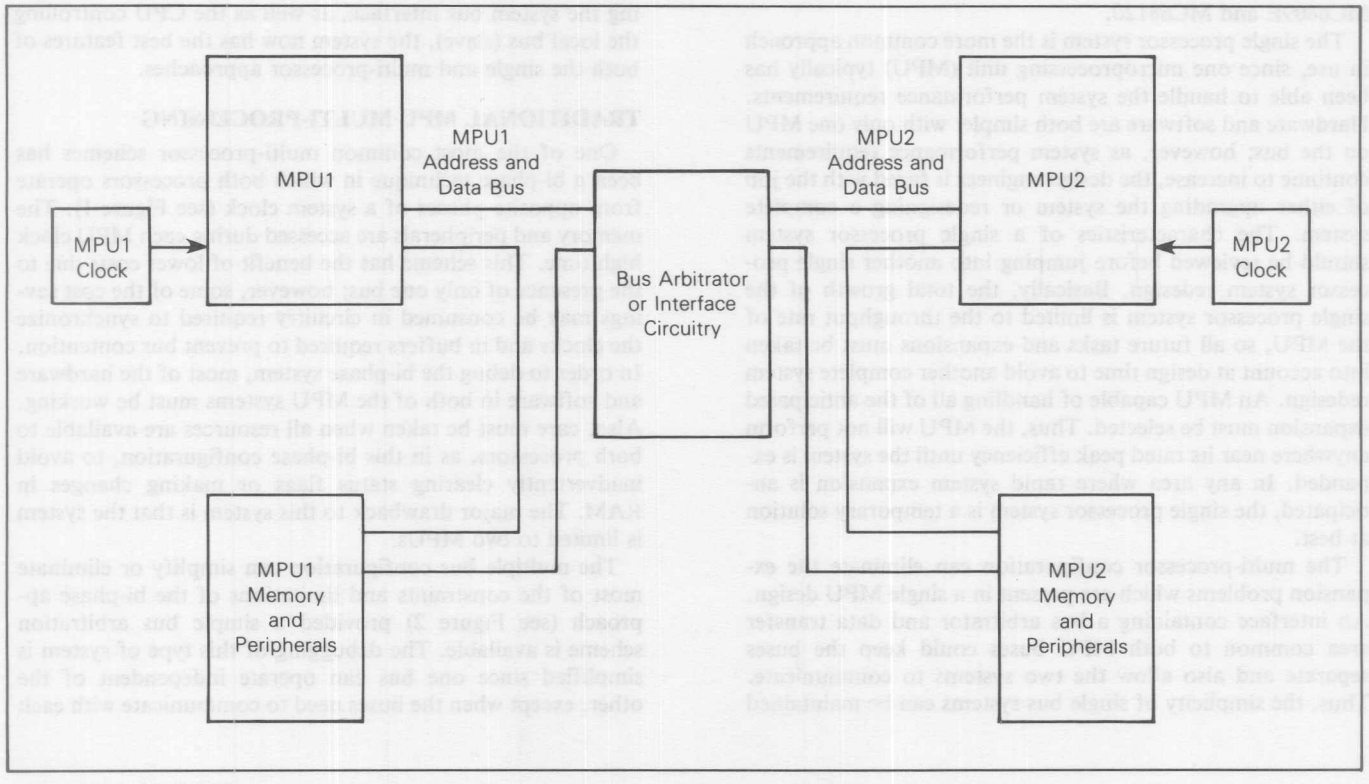

FIGURE 2 — Multi-Processor Multiple Bus Technique (Asynchronous Clocks)

other. This configuration also physically eliminates any chance of one processor accidently clearing any flags in the nonshared resources of the other system. There is no need to determine if the other processor is using the bus for more than one cycle (read-modify-write) since each processor has its own bus, thus eliminating any chance of bus contention. The bi-phase approach is limited to two processors, whereas this system is limited only by the throughput of the system (master) processor.

DESCRIPTION OF THE BASIC SYSTEM

Using the multiple bus scheme, the MC6809E-MC68120 multi-processor pair can be used in many different applications. One particular application could be a system in which the multi-processor pair is responsible for holding the pressure and temperature in a given system within certain limits (see Figure 3). To simplify matters, the application discussed here concentrates only on the MC6809E and MC68120 interface.

HARDWARE

The MC6809E MPU is one of the most advanced 8-bit microprocessor units on the market today. The M6809E (see Figure 4) contains two 16-bit index registers, two 16-bit indexable stack pointers, two 8-bit accumulators (which can be concatenated to form one 16-bit accumulator), and a direct page register that allows the direct addressing mode to be used throughout memory.

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The M6809E has one of the most complete sets of addressing modes available on any microprocessor today. For example, the M6809E contains 59 basic instructions; however, due to these addressing modes, the M6809E will recognize 1464 different variations of the basic instructions. It features an external clock input which facilitates synchronizing the processor to an overall multi-processor system. Other hardware features include three-state control (TSC) inputs for control of internal bus buffers and the advanced valid memory address (AVMA) allows efficient use of common resources in a multi-processor system. Two outputs which facilitate multi-processor configurations are the last instruction cycle (LIC) output and the BUSY output. The LIC output indicates when an opcode fetch will occur. The BUSY output is a status line that indicates the need to hold off the bus transfer for the next bus cycle. The M6809E also contains three prioritized interrupts ($\overline{\text{NMI}}$, $\overline{\text{IRQ}}$, $\overline{\text{FIRQ}}$) and a SYNC acknowledge output which allows synchronization to an external event. These features make the MC6809E an easy MPU to incorporate into a multi-processor system.

The MC68120 Intelligent Peripheral Controller (IPC) is a general purpose mask-programmable peripheral controller

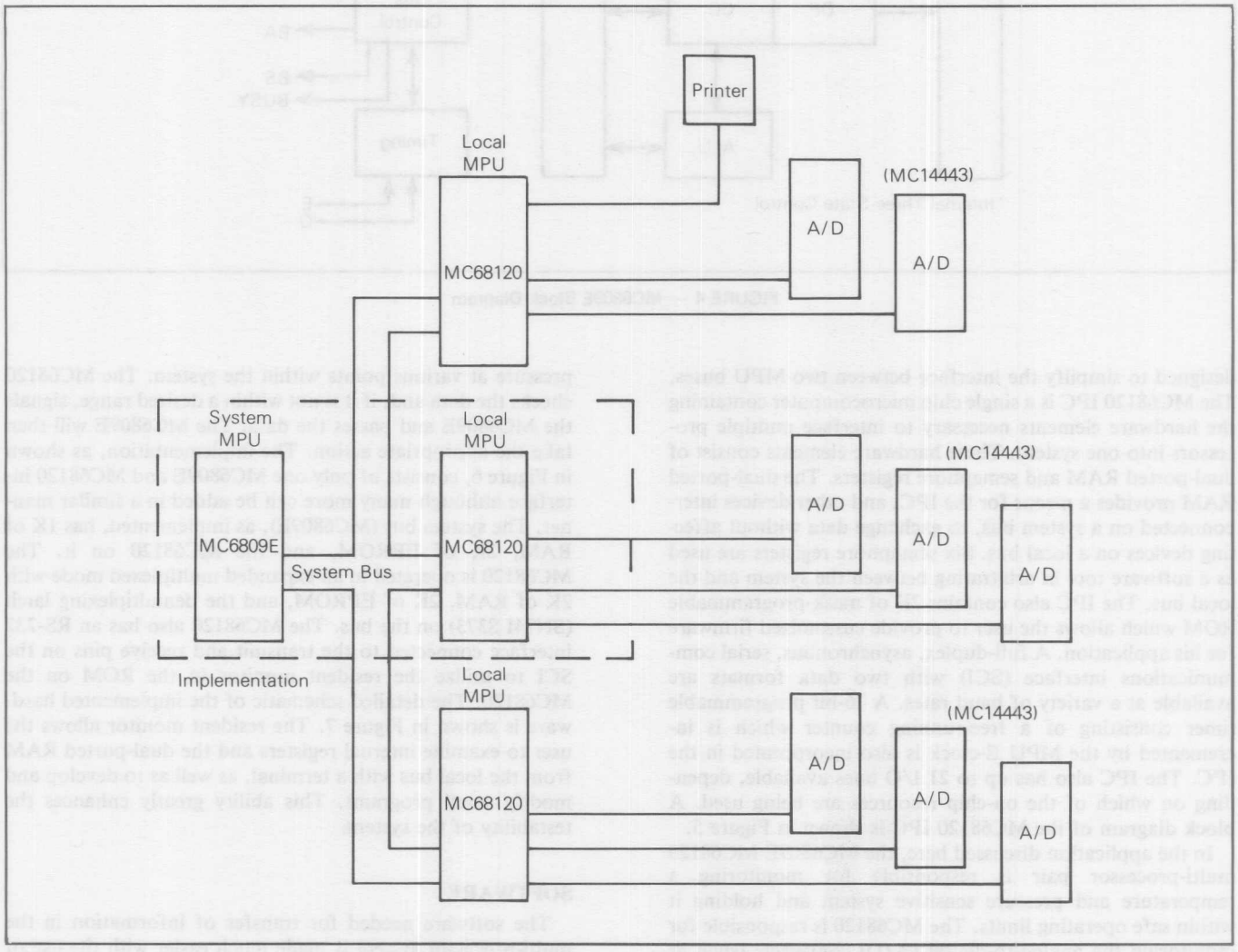

FIGURE 3 — Typical System Configuration

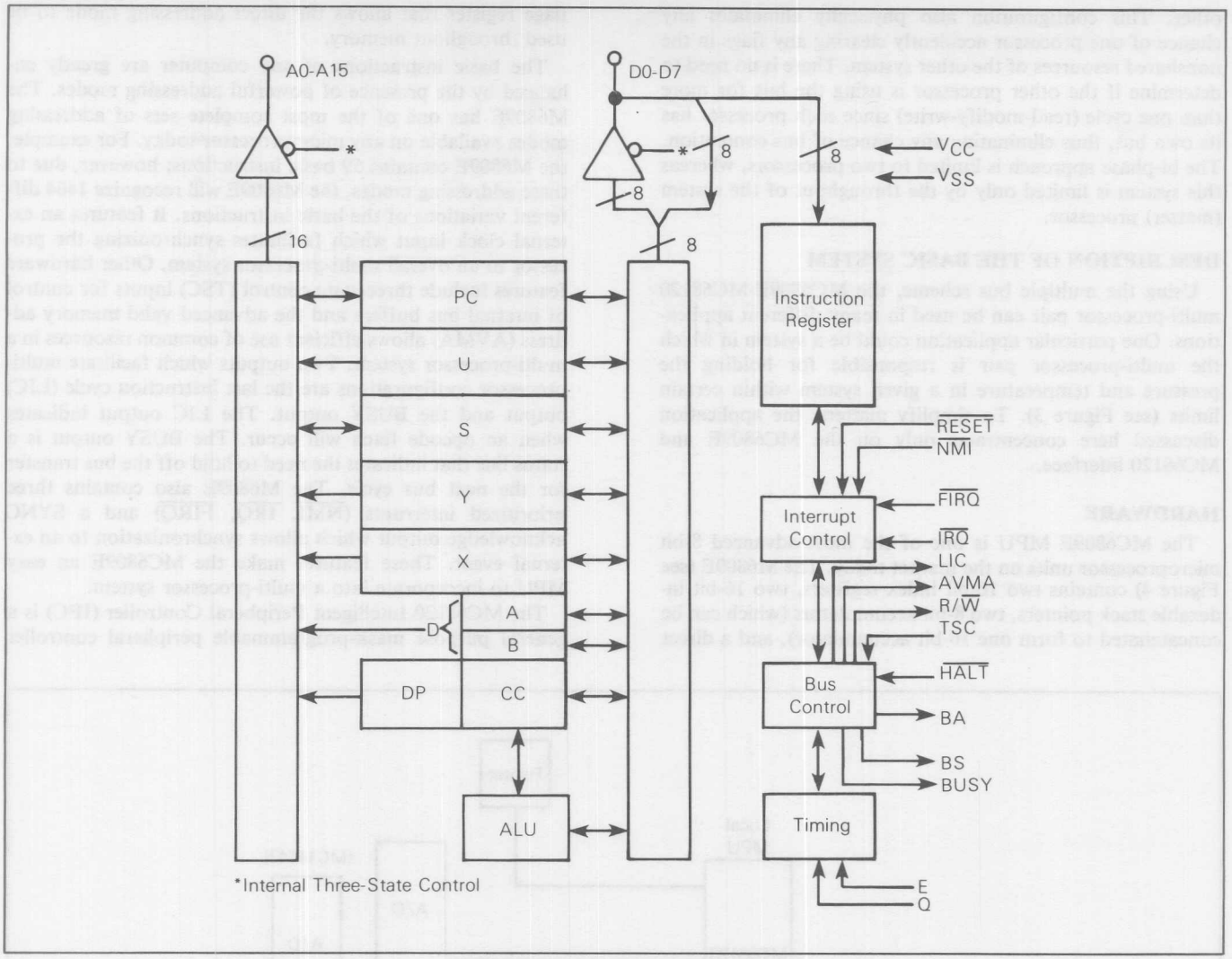

FIGURE 4 — MC6809E Block Diagram

designed to simplify the interface between two MPU buses. The MC68120 IPC is a single chip microcomputer containing the hardware elements necessary to interface multiple processors into one system. These hardware elements consist of dual-ported RAM and semaphore registers. The dual-ported RAM provides a means for the IPC, and other devices interconnected on a system bus, to exchange data without affecting devices on a local bus. Six semaphore registers are used as a software tool in arbitrating between the system and the local bus. The IPC also contains 2K of mask-programmable ROM which allows the user to provide customized firmware for his application. A full-duplex, asynchronous, serial communications interface (SCI) with two data formats are available at a variety of baud rates. A 16-bit programmable timer consisting of a free-running counter which is incremented by the MPU E-clock is also incorporated in the IPC. The IPC also has up to 21 I/O lines available, depending on which of the on-chip resources are being used. A block diagram of the MC68120 IPC is shown in Figure 5.

In the application discussed here, the MC6809E-MC68120 multi-processor pair is responsible for monitoring a temperature and pressure sensitive system and holding it within safe operating limits. The MC68120 is responsible for monitoring the analog-to-digital (A/D) converters (such as Motorola's MC14443) which reflect the temperature and pressure at various points within the system. The MC68120 checks the data and, if it is not within a desired range, signals the MC6809E and passes the data. The MC6809E will then take the appropriate action. The implementation, as shown in Figure 6, consists of only one MC6809E and MC68120 interface although many more can be added in a similar manner. The system bus (MC6809E), as implemented, has 1K of RAM, 2K of EPROM, and the MC68120 on it. The MC68120 is operated in an expanded multiplexed mode with 2K of RAM, 2K of EPROM, and the demultiplexing latch (SN74LS373) on the bus. The MC68120 also has an RS-232 interface connected to the transmit and receive pins on the SCI to utilize the resident monitor in the ROM on the MC68120. The detailed schematic of the implemented hardware is shown in Figure 7. The resident monitor allows the user to examine internal registers and the dual-ported RAM from the local bus with a terminal, as well as to develop and modify small programs. This ability greatly enhances the testability of the system.

SOFTWARE

The software needed for transfer of information in the multi-processor system is made much easier with the use of the semaphore registers and dual-ported RAM, located in the

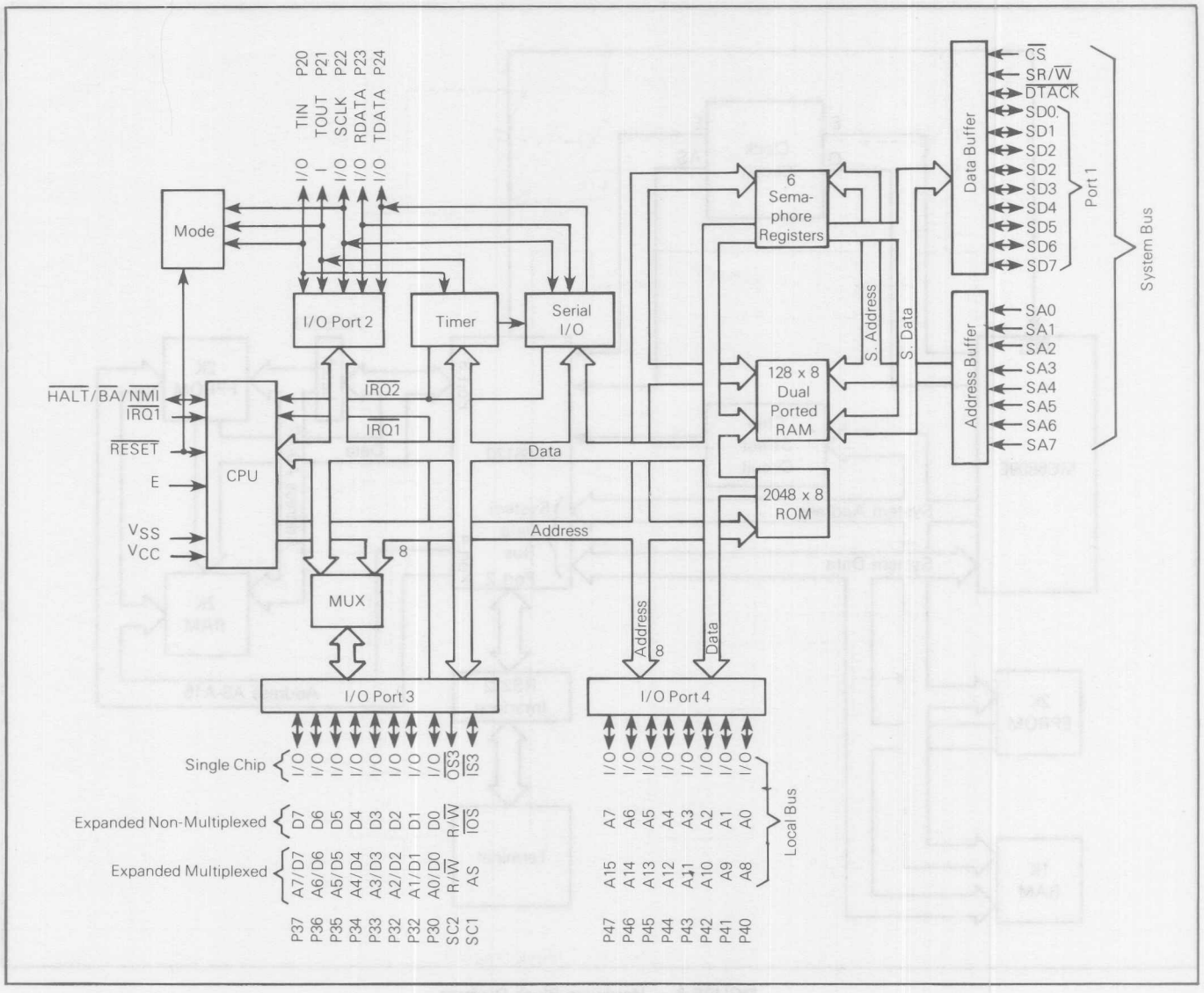

FIGURE 5 — MC68120 Block Diagram

IPC. The dual-ported RAM provides a vehicle for transferring data between a system and local bus while keeping each bus isolated. Semaphore registers are provided as a software tool to arbitrate between shared resources such as the dual-ported RAM or peripheral devices. The semaphore registers may also be used to indicate that a task is in process or has been completed.

Each semaphore register (as shown below) consists of a semaphore bit (SEM, bit 7) and an ownership bit (OWN, bit 6). The remaining six bits (b0-b5) are not used and when read, will read zeroes. The semaphore bits are test and set bits with hardware arbitration during simultaneous accesses. Basically, the semaphore bit is cleared when written and set when read during a single processor access.

b7	b6	b5	b4	b3	b2	b1	b0
SEM	OWN	0	0	0	0	0	0

Semaphore Register

A single processor semaphore bit truth table is shown below. During a write to a semaphore register, the data is disregarded and the semaphore bit is cleared. However, during a read, the data read from the semaphore bit can be interpreted as: 0 — resources are available, 1 — resources are not available. Thus a write to any semaphore register clears the semaphore bit and makes the associated resources "available."

Org. Sem Bit	R/W	Data Read	Resulting Sem Bit
0	R	0	1
1	R	1	1
0	W	—	0
1	W	—	0

Single Processor Semaphore Bit Truth Table

In passing data from the IPC to a system processor through the dual-ported RAM, the semaphore registers can be used to indicate to the system processor that data is ready. The system processor can poll, for example, on semaphore 1 and when data is ready, the IPC CPU will write to semaphore 1, thus clearing the semaphore bit. A simple polling routine for the system processor is shown below. The system processor will always read a 1 in the semaphore bit of semaphore register 1 until semaphore register 1 is written to by the IPC CPU. This will clear the semaphore bit and

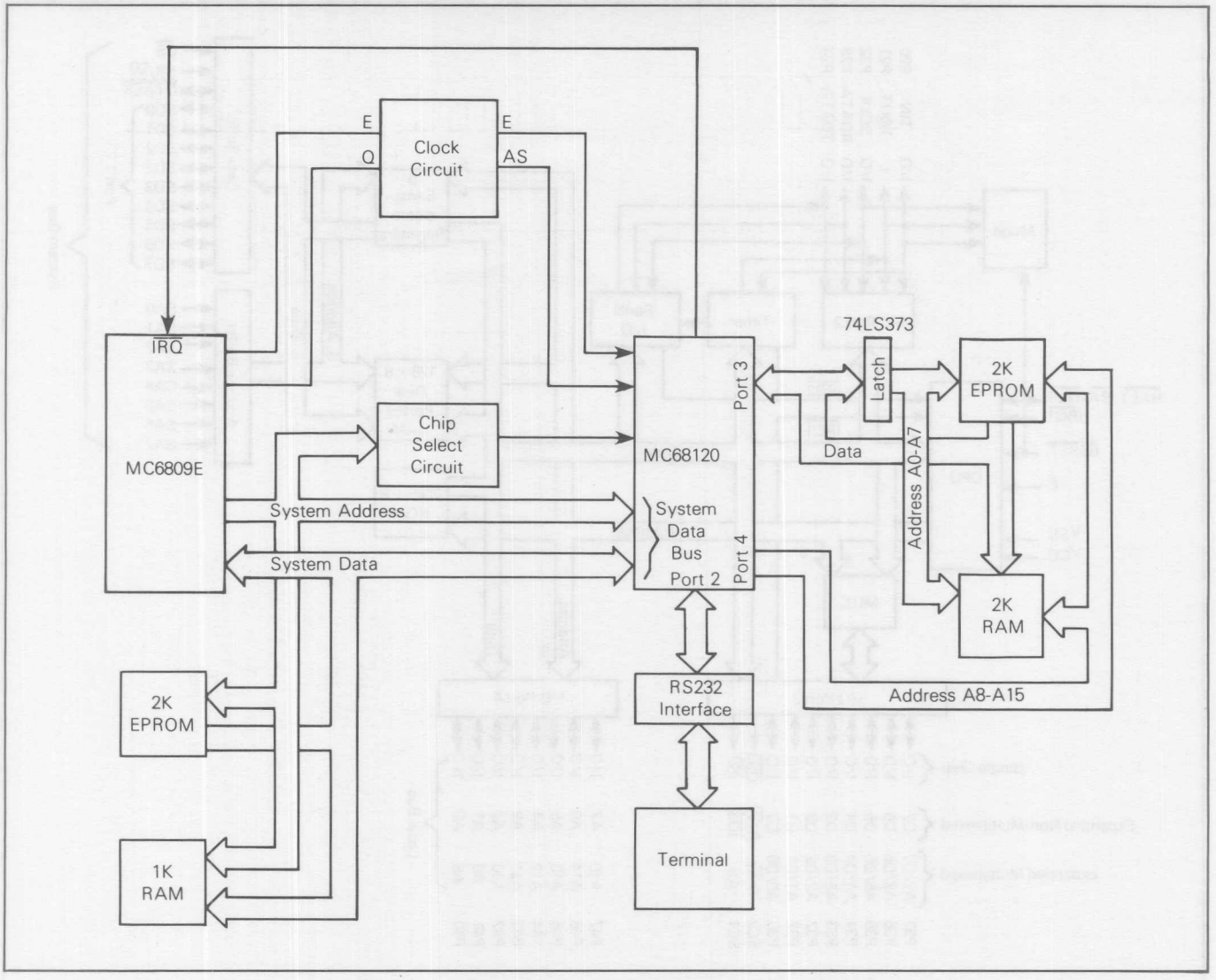

FIGURE 6 — Hardware Block Diagram

cause the system processor to jump to a program and get data.

```
LOOP    LDA     SEMPH1
        ANDA    #$80
        BNE     LOOP
        BSR     GETDATA
```

Polling Routine

It may now be necessary for the IPC CPU to determine if the system processor reads the data from the dual-ported RAM in case more data needs to be sent. Another semaphore register could be dedicated for this purpose or the same semaphore register could be used again. Timing complications could arise when reads and writes of the same semaphore register are occurring from both buses. For example, if the IPC CPU wrote to semaphore 1 to clear the semaphore bit and then polls on semaphore 1, the IPC could set the semaphore bit before the system processor detected it as clear. Therefore, to avoid an inadvertent set, a delay must be incorporated in the program between the read and write of the semaphore to guarantee that the semaphore bit was detected clear by the system processor.

In token-passing applications, the ownership bits can be used to simplify the procedure. The ownership bit is a read-only bit that indicates which processor set the semaphore bit. When the semaphore bit is set, the ownership bit indicates which processor set it. When the semaphore bit is not set, the ownership bit indicates which processor last set the semaphore bit: OWN = 0, the other processor set it; OWN = 1, this processor set SEM. After reset, all semaphores are set and the IPC owns all of them except semaphore 2 which the system processor owns.

As mentioned earlier in the hardware section, this MC6809E-MC68120 system monitors the temperature and pressure in a typical system. Basically, the MC68120 accumulates and monitors the data. The data is transferred to the MC6809E either when the MC6809E requests it, at the end of 12 hours, or if the data is out of the desired range. The CPU on the local bus is responsible for reading the data from the A/D converters every 15 seconds. In this software, it is assumed that the data is formatted in such a way that both the temperature and pressure are available in one byte of data as shown below.

MSB	LSB
TEMP	PRES

One Byte of Data from A/D Converter

The 15 seconds are measured by using the internal timer of the MC68120. The timer sets a flag every 50 ms and after the

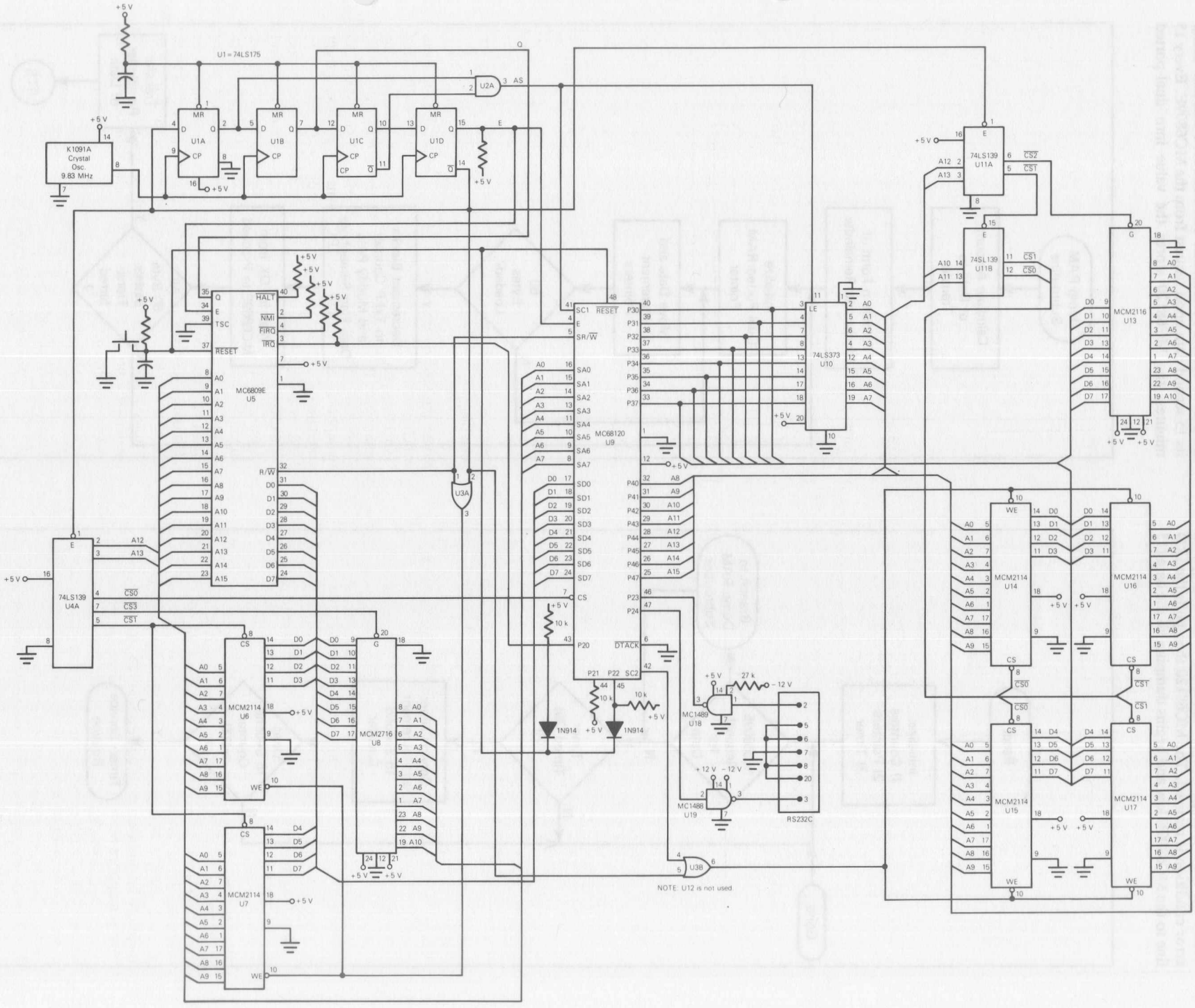

FIGURE 7 — Detailed System Schematic

flag has been set 300 times, the data is read. After the data is read, it is stored in RAM and checked to determine if it is within the desired range. If not within the desired range, an error condition is realized. The MC68120 then pulls the $\overline{IRQ}$ line to the MC6809E low and begins dumping all the data (15 second increments) to the MC6809E through the dual-ported RAM ($B0-EB). The MC68120 can hold up to 8 hours of data in 15 second increments. The MC68120 will also dump its 15 second data upon request from the MC6809E. Every 15 minutes, the MC68120 stores the value into dual-ported

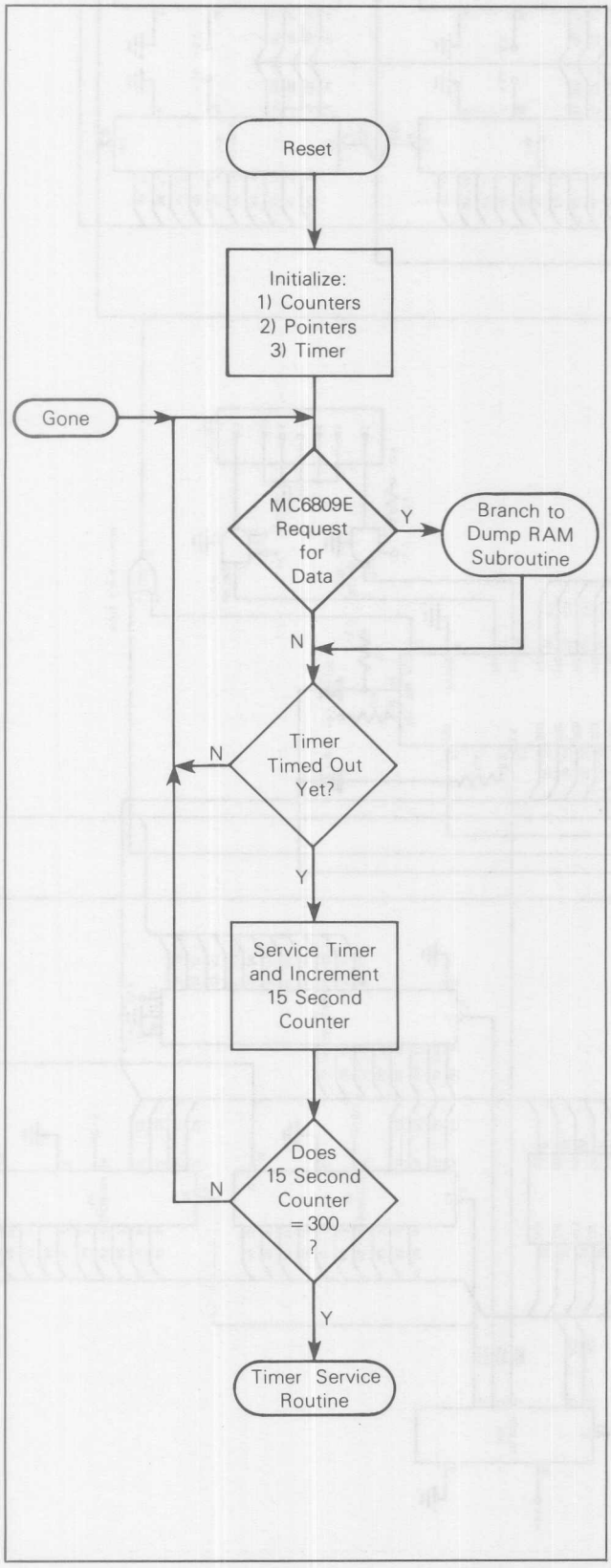

FIGURE 8 — MC68120 Flowchart

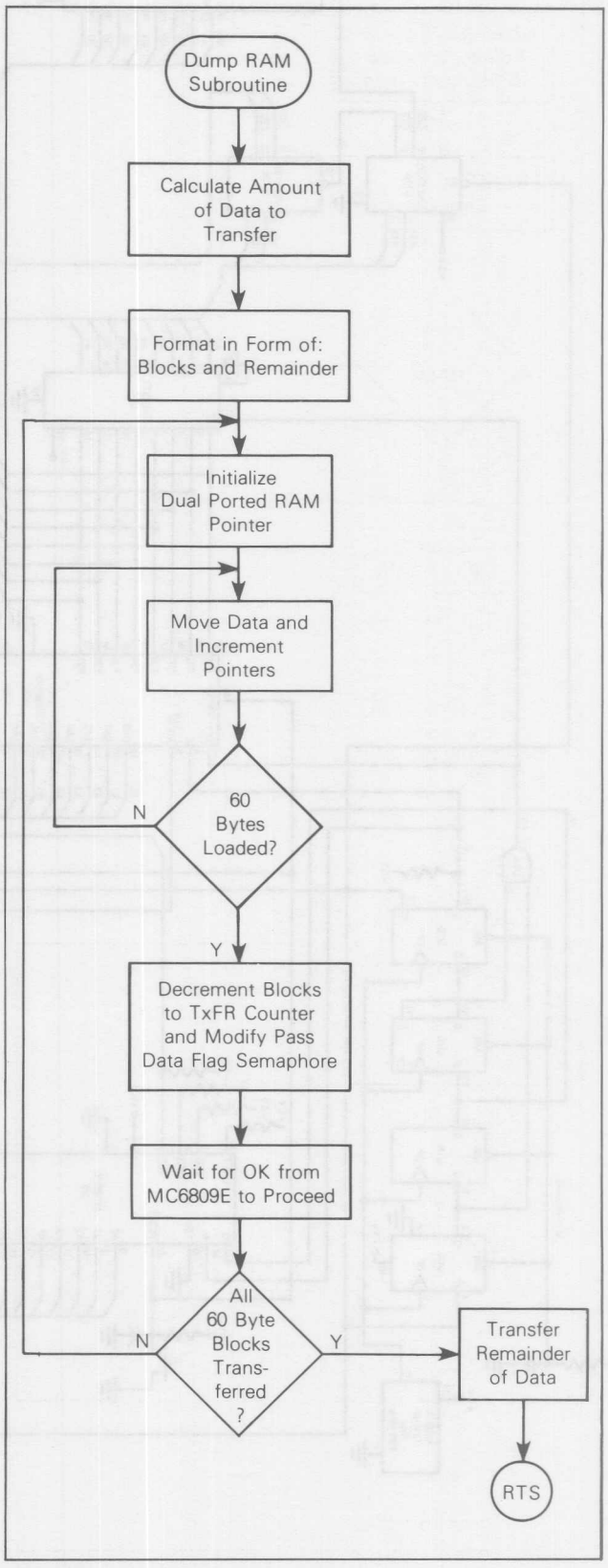

FIGURE 8 — MC68120 Flowchart (Continued)

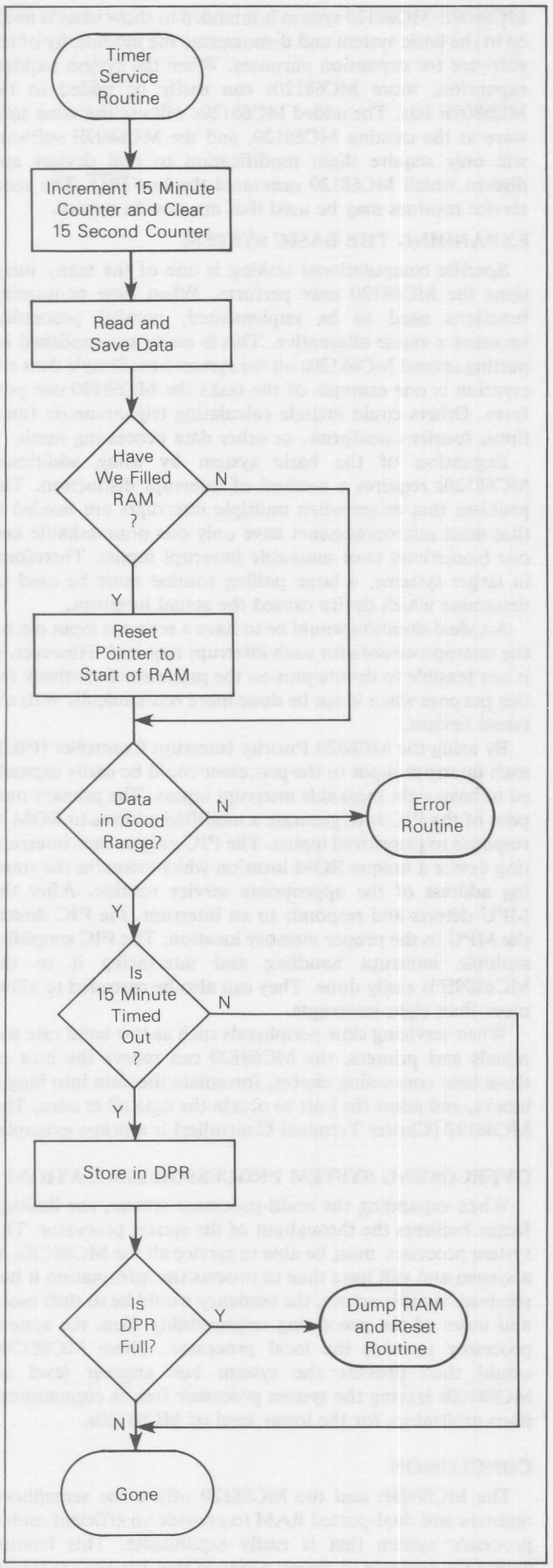

FIGURE 8 — MC68120 Flowchart (Continued)

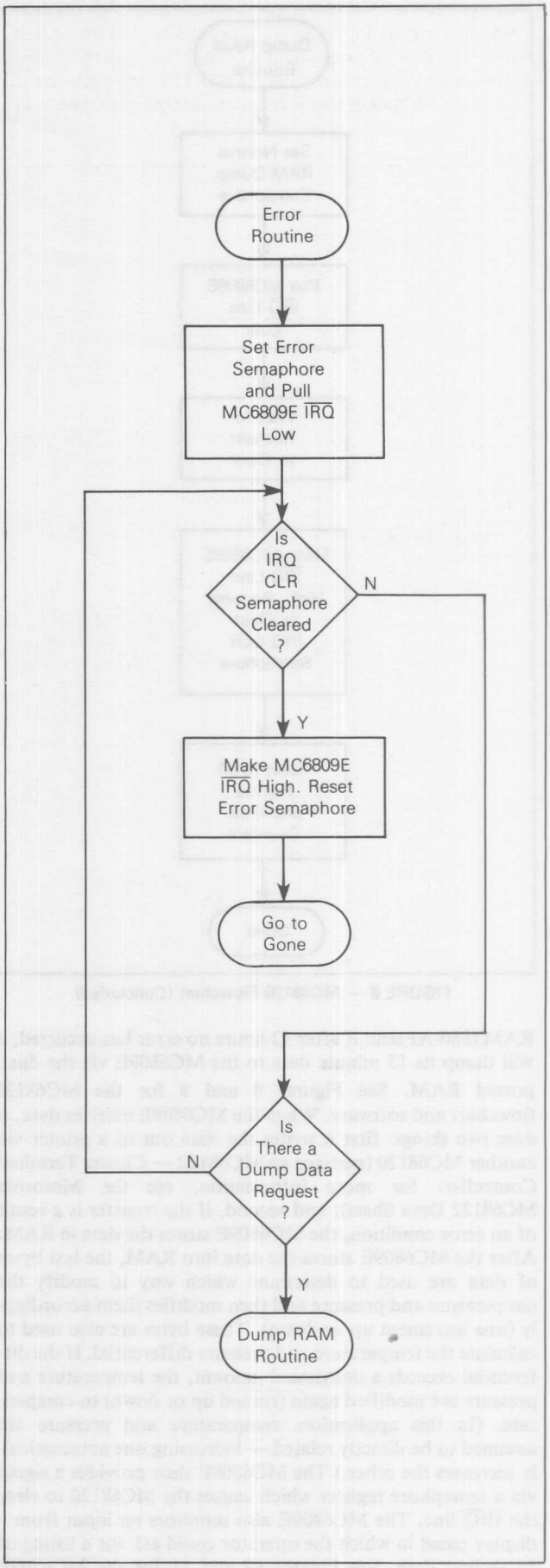

FIGURE 8 — MC68120 Flowchart (Continued)

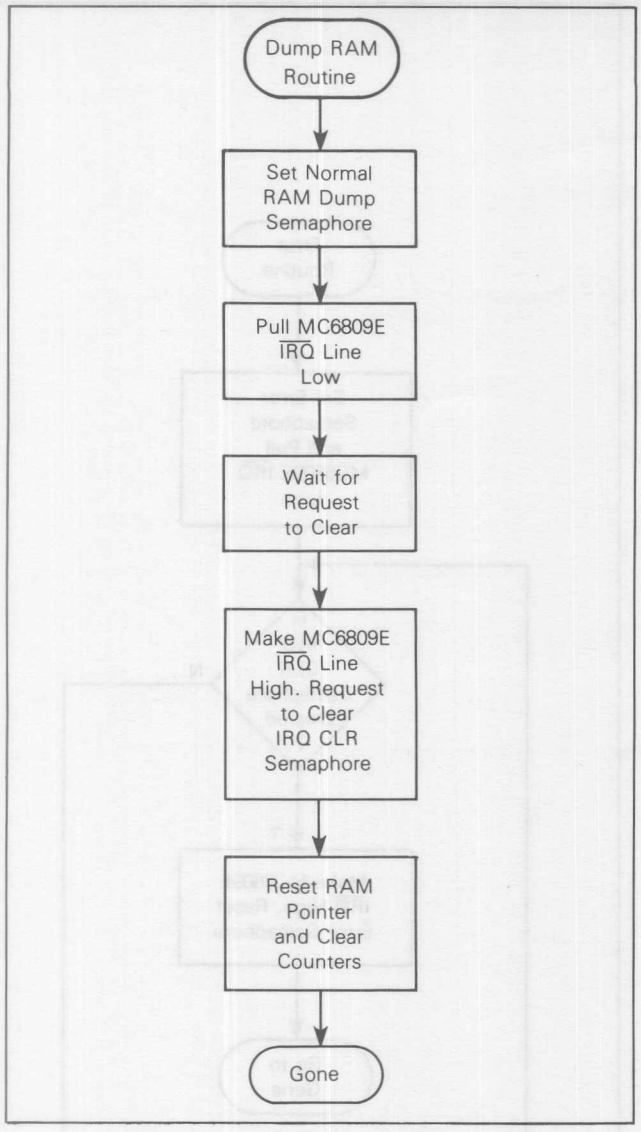

FIGURE 8 — MC68120 Flowchart (Concluded)

RAM ($80-AF)and if after 12 hours no error has occurred, it will dump its 15 minute data to the MC6809E via the dual-ported RAM. See Figures 8 and 9 for the MC68120 flowchart and software. When the MC6809E receives data, it does two things: first it writes the data out to a printer via another MC68120 (perhaps an MC68122 — Cluster Terminal Controller; for more information, see the Motorola MC68122 Data Sheet); and second, if the transfer is a result of an error condition, the MC6809E stores the data in RAM. After the MC6809E stores the data into RAM, the last bytes of data are used to determine which way to modify the temperature and pressure and then modifies them accordingly (one increment up or down). These bytes are also used to calculate the temperature and pressure differential. If the differential exceeds a designated amount, the temperature and pressure are modified again (turned up or down) to compensate. (In this application, temperature and pressure are assumed to be directly related — increasing one automatically increases the other.) The MC6809E then provides a signal via a semaphore register which causes the MC68120 to clear the $\overline{IRQ}$ line. The MC6809E also monitors an input from a display panel in which the operator could ask for a listing of 15 second data. See Figures 10 and 11 for the MC6809E flowchart and software. The implemented portion of the MC6809E-MC68120 system is intended to show what is needed in the basic system and demonstrate the modularity of the software for expansion purposes. When the system requires expansion, more MC68120s can easily be added to the MC6809E bus. The added MC68120s will use the same software as the existing MC68120, and the MC6809E software will only require slight modification to poll devices and discern which MC68120 generated the low $\overline{IRQ}$. The same service routines may be used that are now in service.

EXPANDING THE BASIC SYSTEM

Specific computational tasking is one of the many functions the MC68120 may perform. When time consuming functions need to be implemented, parallel processing becomes a viable alternative. This is easily accomplished by putting several MC68120s on the system bus. Simple data encryption is one example of the tasks the MC68120 can perform. Others could include calculating trigonometric functions, fourier transforms, or other data processing needs.

Expansion of the basic system by using additional MC68120s requires a method of interrupt distinction. The problem that arises when multiple interrupts are needed is that most microprocessors have only one nonmaskable and one (sometimes two) maskable interrupt inputs. Therefore, in larger systems, a large polling routine must be used to determine which device caused the actual interrupt.

An ideal situation would be to have a separate input pin on the microprocessors for each interrupt required. However, it is not feasible to devote pins on the processor exclusively for this purpose when it can be done more economically with external devices.

By using the MC6828 Priority Interrupt Controller (PIC), each interrupt input to the processor could be easily expanded to have eight maskable interrupt inputs. The primary purpose of the PIC is to generate a modified address to ROM in response to prioritized inputs. The PIC assigns each interrupting device a unique ROM location which contains the starting address of the appropriate service routine. After the MPU detects and responds to an interrupt, the PIC directs the MPU to the proper memory location. The PIC simplifies multiple interrupt handling and interfacing it to the MC6809E is easily done. They can also be cascaded to allow more than eight interrupts.

When servicing slow peripherals such as low baud rate terminals and printers, the MC68120 can relieve the host of these time consuming chores, formulate the data into bigger blocks, and allow the host to obtain the data all at once. The MC68122 (Cluster Terminal Controller) is a prime example.

OVERCOMING SYSTEM PROCESSOR LIMITATIONS

When expanding the multi-processor system, the limiting factor becomes the throughput of the system processor. The system processor must be able to service all the MC68120s in a system and still have time to process the information it has received. As this occurs, the tendency would be to shift more and more of the processing responsibility from the system processor towards the local processor. These MC68120s would then provide the system bus another level of MC68120s leaving the system processor free as communications arbitrators for the lower level of MC68120s.

CONCLUSION

The MC6809E and the MC68120 utilize the semaphore registers and dual-ported RAM to provide an efficient multiprocessor system that is easily expandable. This feature allows the engineer to design a system that has the capability of simple expansion and increases its time of usefulness.

```
PAGE  001  MLTPRCA .SA:1

00001                    *
00002                          OPT    ABS,Z01,LLEN=80
00003                    *
00004                    *
00005                    *   THIS PROGRAM IS USED ON THE MC68120 IN A MC6809E
00006                    *     MULTIPROCESSOR CONFIGURATION
00007                    *
00008                    *
00009                    *
00010          0003  A P2DR   EQU    $03      PORT 2 DATA REGISTER
00011          0001  A P2DDR  EQU    $01      PORT 2 DATA DIR. REG.
00012          0008  A TCSR   EQU    $08      TIMER CONTROL & STAT. REG.
00013          0009  A TIMERR EQU    $09      READ TIMER COUNTER REGISTER
00014          000B  A TIMROC EQU    $0B      TIMER OUTPUT COMPARE REG.
00015                    *
00016                    ********************************************************
00017                    * The 1ST 2 Semaphore Registers are cleared by the     *
00018                    *          MC68120 and set by the MC6809E              *
00019                    * The next 2 Semaphore registers are cleared by the    *
00020                    *          MC6809E and cleared the by MC68120          *
00021                    * The last two are used as flags, to pass data between *
00022                    *          the two MPUs                                *
00023                    ********************************************************
00024                    *
00025          0017  A SEMPH1 EQU    $17      SEMPH REG 1 (ERROR SITUATION)
00026          0018  A SEMPH2 EQU    $18      SEMPH REG 2 (NORMAL RAM DUMP)
00027          0019  A SEMPH3 EQU    $19      SEMPH REG 3 (REQUEST FOR DATA)
00028          001A  A SEMPH4 EQU    $1A      SEMPH REG 4 (IRQ CLR)
00029          001B  A SEMPH5 EQU    $1B      SEMPH REG 5 (PASS DATA FLAG)
00030          001C  A SEMPH6 EQU    $1C      SEMPH REG 6 (PASS DATA FLAG)
00031          00F0  A WHNO   EQU    $F0      WHOLE NUMBER OF BLOCKS 60 BYTES
00032                    *                          (1 byte wide)
00033          00F1  A RMDR   EQU    $F1      REMAINDER OF BYTES TO BE
00034                    *                   TRANSFERRED (1 byte wide)
00035          1780  A SECCTR EQU    $1780    15 SECOND COUNTER REGISTER
00036                    *                          (2 bytes)
00037          1782  A MINCTR EQU    $1782    15 MINUTE COUNTER REGISTER
00038                    *                          (1 bytes)
00039          EB00  A TXDCR  EQU    $EB00    TRANSDUCER INPUT LOCATION
00040          1783  A SDPTR  EQU    $1783    START DATA POINTER
00041                    *                       (2 bytes-for RAM)
00042          1785  A EDPTR  EQU    $1785    END DATA POINTER
00043                    *                       (2 bytes-for RAM)
00044          1787  A TXRMSZ EQU    $1787    TRANSMIT RAM SIZE (2 BYTES)
00045          1789  A TEMP   EQU    $1789    TEMPORARY ADDRESS STORAGE
00046                    *                          (2 bytes)
00047          178B  A NRMPTR EQU    $178B    NORM. RAM DUMP POINTER (2 BYTES)
00048          0011  A LOW    EQU    $11      TXDUCER DATA SHOULD BE ABOVE
00049                    *                   THIS VALUE
00050          00CC  A HIGH   EQU    $CC      TXDUCER DATA SHOULD BE BELOW
00051                    *                   THIS VALUE
00052          0000  A IRQLW  EQU    $00      VALUE FOR PORT 2 TO PULL '09
00053                    *                   IRQ LOW
00054          0001  A IRQHG  EQU    $01      VALUE FOR PORT 2 TO PULL '09
00055                    *                   IRQ HIGH
00056          178D  A FROM   EQU    $178D    TEMP RAM FOR DATA ADDRESS(2 bytes)
00057          EB80  A TO     EQU    $EB80    END OF DATA
00058
```

FIGURE 9 — MC68120 Software

```
PAGE  002  MLTPRCA .SA:1

00059                          *
0005A E800                     ORG    $E800
00061                          *
00062                          *   INITIALIZATION ROUTINE
00063                          *
0006A E800 8E 17FF  A          LDS    #$17FF   INIT. STACK
0006A E803 CE 0000  A          LDX    #0       CLEAR COUNTER REGS.
0006A E806 FF 1780  A          STX    SECCTR   15 SECOND COUNTER REG.
0006A E809 FF 1782  A          STX    MINCTR   15 MINUTE COUNTER REG.
0006A E80C CE 1000  A          LDX    #$1000   INIT. START DATA POINTER
0006A E80F FF 1783  A          STX    SDPTR
0006A E812 FF 1785  A          STX    EDPTR    AND END DATA POINTER
00071                          *        TIMER COMES UP INIT. IN DESIRED MODE
0007A E815 96 01    A          LDAA   IRQHG    CONFIGURE AND INIT.
0007A E817 97 03    A          STAA   P2DR     IRQ TO MC6809
0007A E819 86 01    A          LDAA   #$01
0007A E81B 97 01    A          STAA   P2DDR
0007A E81D DC 09    A          LDD    TIMERR   INIT. TIMER FOR 50 MSEC.
0007A E81F C3 EFBF  A          ADDD   #$EFBF
0007A E822 DD 0B    A          STD    TIMROC
0007A E824 CE EB00  A          LDX    #$EB00   START ADDRESS FOR DATA
0008A E827 FF 178D  A          STX    FROM
0008A E82A CE 0080  A          LDX    #$0080   INIT. NORMAL RAM POINTER
0008A E82D FF 178B  A          STX    NRMPTR   TO BEG. OF DUAL PORTED RAM
0008A E830 86 00    A          LDAA   #$00
0008A E832 97 F0    A          STAA   WHNO
00085                          *
00086                          *          CHECKING ON UPDATE DATA SEMAPHORE (#3)
00087                          *             (NO DATA PASSED IN REGISTERS)
0008A E834 96 19    A POLL1    LDAA   SEMPH3   CHECK REQUEST
0008A E836 84 80    A          ANDA   #$80     FOR MORE
0008A E838 26 02 E83C          BNE    CONT1    DATA
0008A E83A 8D 1E E85A          BSR    DMPRAM   GO DUMP IT
0008A E83C 96 08    A CONT1    LDAA   TCSR     CHECK FOR
0008A E83E 84 40    A          ANDA   #$40     TIMER FLAG SET
0008A E840 27 F2 E834          BEQ    POLL1    BRA IF NOT SET
00095                          *
00096                          *     ENTERING THE 50 MSEC TIMEOUT SERVICE LOOP
00097                          *
0008A E842 96 08    A          LDAA   TCSR     DUMMY READ TO CLEAR OCF
0008A E844 DC 09    A          LDD    TIMERR   READ TIMER
00100A E846 C3 EF9C A          ADDD   #$EF9C   REINIT. TIMER - ADJUSTED TO COR-
00101A E849 DD 0B   A          STD    TIMROC   RECT FOR ADDED CYCLES OF ROUTINE
00102A E84B FE 1780 A          LDX    SECCTR   INCREMENT 15 SEC. CTR.
00103A E84E 08                 INX
00104A E84F FF 1780 A          STX    SECCTR
00105A E852 8C 012C A          CPX    #300     CHECK IF 15 SECS. UP?
00106A E855 26 DD E834         BNE    POLL1    BRANCH IF NOT (300 TIMES)
00107A E857 7E E8C9 A          JMP    TMRSRV   GO TO TIMER SERVICE ROUTINE
00108                          *
00109                          *   THIS ROUTINE DUMPS THE RAM (15 SEC. DATA SAMPLES)
00110A E85A FC 1785 A DMPRAM   LDD    EDPTR    CALC. SIZE OF DATA
00111A E85D B3 1783 A          SUBD   SDPTR    TO BE TRANSFERRED
00112A E860 FD 1787 A          STD    TXRMSZ
00113A E863 05                 LSLD            CLEAR SIGN BIT
00114A E864 04                 LSRD
00115                          *    CONFIGURE SIZE IN FORMAT FOR DPR STORAGE
00116A E865 7C 00F0 A COUNT    INC    WHNO     DIVIDING BY 60
```

FIGURE 9 — MC68120 Software (Continued)

```
PAGE  003  MLTPRCA .SA:1

00117A E868 83 003C  A         SUBD    #60
00118A E86B 2A F8 E865         BPL     COUNT
00119A E86D 7A 00F0  A         DEC     WHNO
00120A E870 C3 003C  A         ADDD    #60       DONE
00121A E873 D7 F1    A         STAB    RMDR      SAVE REMAINDER
00122                       *CHECK IF WHOLE NUMBER EQUAL ZERO
00123A E875 96 F0    A         LDAA    WHNO
00124A E877 81 00    A         CMPA    #00
00125A E879 27 2B E8A6         BEQ     LAST
00126                       *     LOADING  60 BYTES OF DATA TO DPR
00127A E87B CC 00B0  A LOOP    LDD     #$00B0    INIT. DPR PTR.
00128A E87E FD 1789  A         STD     TEMP
00129A E881 FE 1783  A TLOOP   LDX     SDPTR     GET MEMORY LOC & DATA
00130A E884 E6 00    A         LDAB    0,X
00131A E886 08                 INX               SET SDPTR UP FOR NEXT
00132A E887 FF 1783  A         STX     SDPTR     TIME
00133A E88A FE 1789  A         LDX     TEMP      GET DESTINATION
00134A E88D E7 00    A         STAB    0,X       STORE DATA
00135A E88F 08                 INX               SET TEMP UP FOR NEXT
00136A E890 FF 1789  A         STX     TEMP      TIME
00137A E893 8C 00EC  A         CPX     #$00EC
00138A E896 26 E9 E881         BNE     TLOOP     CHECK IF 60 BYTES TX
00139A E898 D7 1B    A         STAB    SEMPH5    SET TX'FER SEMPH.- GIVES '09 "00"
00140A E89A 96 1C    A WAIT1   LDAA    SEMPH6    CHECK IF OK TO PROCEED
00141A E89C 84 80    A         ANDA    #$80
00142A E89E 26 FA E89A         BNE     WAIT1     BRANCH IF NOT OK
00143A E8A0 86 00    A         LDAA    #00       CHECK IF ALL 60 BYTE
00144A E8A2 91 F0    A         CMPA    WHNO      BLOCKS ARE TX'FERRED
00145A E8A4 26 D5 E87B         BNE     LOOP      BRANCH IF NOT
00146                       *  TRANSFER  REMAINDER OF DATA
00147A E8A6 CC 00B0  A LAST    LDD     #$00B0    BEGINNING OF TX'FER
00148A E8A9 FD 1789  A         STD     TEMP      AREA
00149A E8AC FE 1783  A ELOOP   LDX     SDPTR     GET START ADDRESS
00150A E8AF E6 00    A         LDAB    0,X       GET DATA
00151A E8B1 08                 INX               PREPARE FOR NEXT FETCH
00152A E8B2 FF 1783  A         STX     SDPTR     AND SAVE
00153A E8B5 FE 1789  A         LDX     TEMP      GET DESTIN. ADDRESS
00154A E8B8 E7 00    A         STAB    0,X       STORE IN DPR
00155A E8BA 08                 INX               PREPARE FOR NEXT STORE
00156A E8BB FF 1789  A         STX     TEMP      AND SAVE
00157A E8BE FE 1785  A         LDX     EDPTR     CHECK IF DONE
00158A E8C1 BC 1783  A         CPX     SDPTR     CHECK IF DONE
00159A E8C4 26 E6 E8AC         BNE     ELOOP     BRANCH IF NOT TO END LOOP
00160A E8C6 D7 1B    A         STAB    SEMPH5    SET TX'FER SEMPH.- GIVES °09 "00"
00161A E8C8 39                 RTS               GOIN HOME
00162                       *
00163                       *          TIMER SERVICE ROUTINE - ACCESSED EVERY 15 SECON
00164                       *
00165A E8C9 7C 1782  A TMRSRV  INC     MINCTR    INCREMENT 15 MIN. CTR.
00166A E8CC CE 0000  A         LDX     #00       CLEAR 15 SEC. CTR.
00167A E8CF FF 1780  A         STX     SECCTR
00168A E8D2 FE 178D  A         LDX     FROM      READ DATA
00169A E8D5 8C EB80  A         CPX     #TO       DUMMY ROUTINE FOR DATA
00170A E8D8 26 06 E8E0         BNE     AROUND    AQUISITION
00171A E8DA CE EB00  A         LDX     #$EB00
00172A E8DD FF 178D  A         STX     FROM
00173A E8E0 A6 00    A AROUND  LDAA    0,X
00174A E8E2 08                 INX
```

FIGURE 9 — MC68120 Software (Continued)

```
PAGE  004  MLTPRCA .SA:1

00175A E8E3 FF 178D  A         STX     FROM
00176A E8E6 FE 1785  A         LDX     EDPTR     GET NEXT OPEN LOCATION
00177A E8E9 A7 00    A         STAA    0,X       STORE DATA THERE
00178A E8EB 08                 INX               INCREMENT AND CHECK
00179A E8EC 8C 1780  A         CPX     #$1780    DATA POINTER FOR
00180A E8EF 26 03 E8F4         BNE     DOVRN     END OF RAM
00181A E8F1 CE 1000  A         LDX     #$1000
00182A E8F4 FF 1785  A DOVRN   STX     EDPTR     SAVE END DATA POINTER
00183A E8F7 BC 1783  A         CPX     SDPTR     CHECK FOR DATA OVERRUN
00184A E8FA 26 07 E903         BNE     OK
00185A E8FC CE 1000  A         LDX     #$1000    DATA OVERRUN
00186A E8FF 08                 INX               INCREMENT START
00187A E900 FF 1783  A         STX     SDPTR     ADDRESS POINTER
00188A E903 81 11    A OK      CMPA    #LOW      CHECK IF DATA IN
00189A E905 25 21 E928         BLO     ERROR     RANGE
00190A E907 81 CC    A         CMPA    #HIGH
00191A E909 22 1D E928         BHI     ERROR
00192A E90B F6 1782  A         LDAB    MINCTR    DATA GOOD- CHECK IF 15 MIN.
00193A E90E C1 3C    A         CMPB    #60       COUNTER TIMED OUT YET?
00194A E910 26 11 E923         BNE     GONE      BRANCH IF NOT
00195A E912 FE 178B  A         LDX     NRMPTR    IF SO STORE IT IN UPPER
00196A E915 A7 00    A         STAA    0,X
00197A E917 8C 00AF  A         CPX     #$AF      CHECK IF DUAL PORTED RAM
00198A E91A 27 29 E945         BEQ     DPRST     OVERRUN-IF SO DMP & RESET
00199A E91C 08                 INX
00200A E91D FF 178B  A         STX     NRMPTR    UPDATE DATA PTR. FOR NEXT TIME
00201A E920 7F 1782  A         CLR     MINCTR    REINIT. 15 MIN COUNTER TO 0
00202A E923 96 17    A GONE    LDAA    SEMPH1    REGAIN OWNERSHIP OF SEMPH1
00203A E925 7E E834  A         JMP     POLL1     GET OUT OF ROUTINE
00204                  *
00205                  *     ERROR ROUTINE
00206                  *
00207A E928 D7 17    A ERROR   STAB    SEMPH1    SET ERROR SEMAPHORE(1)
00208A E92A C6 00    A         LDAB    #IRQLW    PULL '09 IRQ LOW
00209A E92C D7 03    A         STAB    P2DR
00210A E92E 96 1A    A KPLKNG  LDAA    SEMPH4    CHECK FOR IRQ CLEAR SIGNAL
00211A E930 84 80    A         ANDA    #$80
00212A E932 26 06 E93A         BNE     DMPCHK    BRA IF ISN'T CLEAR
00213A E934 C6 01    A         LDAB    #IRQHG    CLEAR
00214A E936 D7 03    A         STAB    P2DR      '09 IRQ
00215A E938 20 E9 E923         BRA     GONE      GET OUT OF ROUTINE
00216A E93A D6 19    A DMPCHK  LDAB    SEMPH3    CHECK FOR REQUEST
00217A E93C 84 80    A         ANDA    #$80      TO DUMP DATA IN RAM
00218A E93E 26 EE E92E         BNE     KPLKNG    KEEP LOOKING
00219A E940 BD E85A  A         JSR     DMPRAM    DUMP THE RAM
00220A E943 20 E9 E92E         BRA     KPLKNG    WAIT FOR IRQ CLEAR
00221A E945 97 18    A DPRST   STAA    SEMPH2    SET NORMAL DUMP SEMPH.
00222A E947 C6 00    A         LDAB    #IRQLW    PULL '09 IRQ LOW
00223A E949 D7 03    A         STAB    P2DR
00224A E94B 96 1A    A WAIT    LDAA    SEMPH4    CHECK REQUEST TO CLR IRQ
00225A E94D 84 80    A         ANDA    #$80      WAITING ON '09
00226A E94F 26 FA E94B         BNE     WAIT
00227A E951 C6 01    A         LDAB    #IRQHG    CLEAR
00228A E953 D7 03    A         STAB    P2DR      '09 IRQ AND
00229A E955 7F 178B  A         CLR     NRMPTR    RESET NORMAL RAM POINTER
00230A E958 7F 1782  A         CLR     MINCTR    RESET 15 MIN. COUNTER TO 0
00231A E95B 20 C6 E923         BRA     GONE
00232                          END
```

FIGURE 9 — MC68120 Software (Continued)

```
PAGE  005  MLTPRCA .SA:1

TOTAL ERRORS 00000--00000


   E8E0 AROUND 00170 00173*
   E83C CONT1  00090 00092*
   E865 COUNT  00116*00118
   E93A DMPCHK 00212 00216*
   E85A DMPRAM 00091 00110*00219
   E8F4 DOVRN  00180 00182*
   E945 DPRST  00198 00221*
   1785 EDPTR  00042*00070 00110 00157 00176 00182
   E8AC ELOOP  00149*00159
   E928 ERROR  00189 00191 00207*
   178D FROM   00056*00080 00168 00172 00175
   E923 GONE   00194 00202*00215 00231
   00CC HIGH   00050*00190
   0001 IRQHG  00054*00072 00213 00227
   0000 IRQLW  00052*00208 00222
   E92E KPLKNG 00210*00218 00220
   E8A6 LAST   00125 00147*
   E87B LOOP   00127*00145
   0011 LOW    00048*00188
   1782 MINCTR 00037*00067 00165 00192 00201 00230
   178B NRMPTR 00047*00082 00195 00200 00229
   E903 OK     00184 00188*
   0001 P2DDR  00011*00075
   0003 P2DR   00010*00073 00209 00214 00223 00228
   E834 POLL1  00088*00094 00106 00203
   00F1 RMDR   00033*00121
   1783 SDPTR  00040*00069 00111 00129 00132 00149 00152 00158 00183 00187
   1780 SECCTR 00035*00066 00102 00104 00167
   0017 SEMPH1 00025*00202 00207
   0018 SEMPH2 00026*00221
   0019 SEMPH3 00027*00088 00216
   001A SEMPH4 00028*00210 00224
   001B SEMPH5 00029*00139 00160
   001C SEMPH6 00030*00140
   0008 TCSR   00012*00092 00098
   1789 TEMP   00045*00128 00133 00136 00148 00153 00156
   0009 TIMERR 00013*00076 00099
   000B TIMROC 00014*00078 00101
   E881 TLOOP  00129*00138
   E8C9 TMRSRV 00107 00165*
   EB80 TO     00057*00169
   EB00 TXDCR  00039*
   1787 TXRMSZ 00044*00112
   E94B WAIT   00224*00226
   E89A WAIT1  00140*00142
   00F0 WHNO   00031*00084 00116 00119 00123 00144
```

FIGURE 9 — MC68120 Software (Concluded)

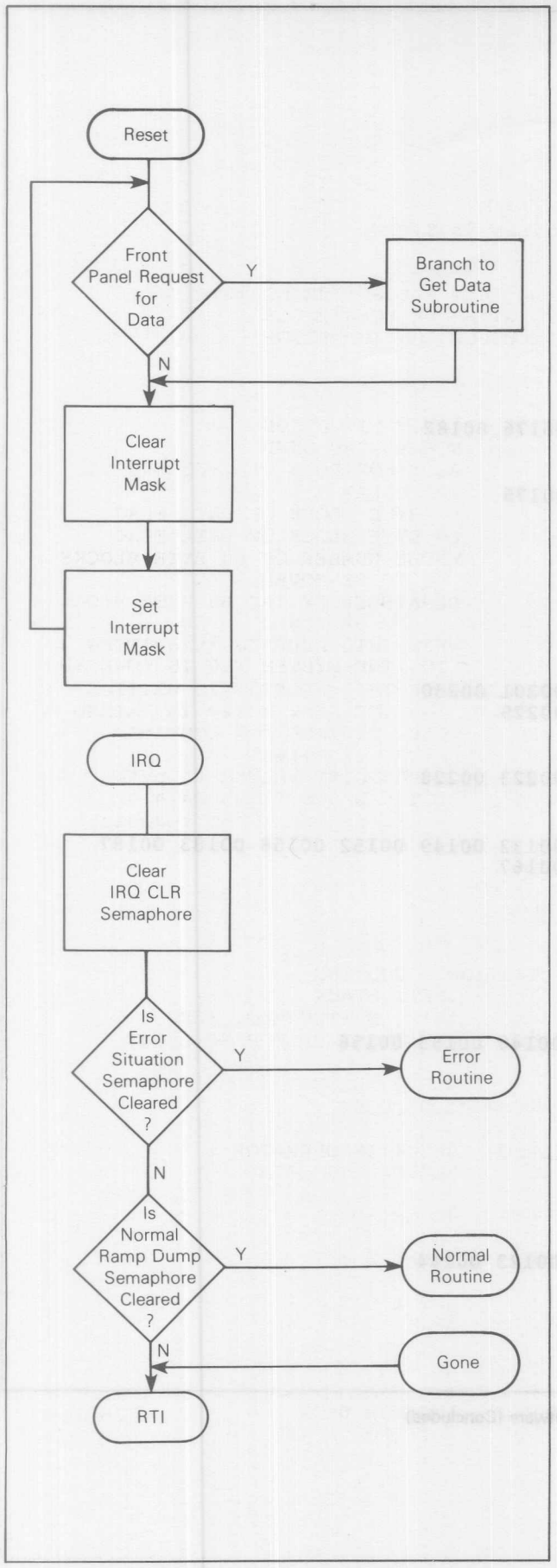

FIGURE 10 — MC6809E Flowchart

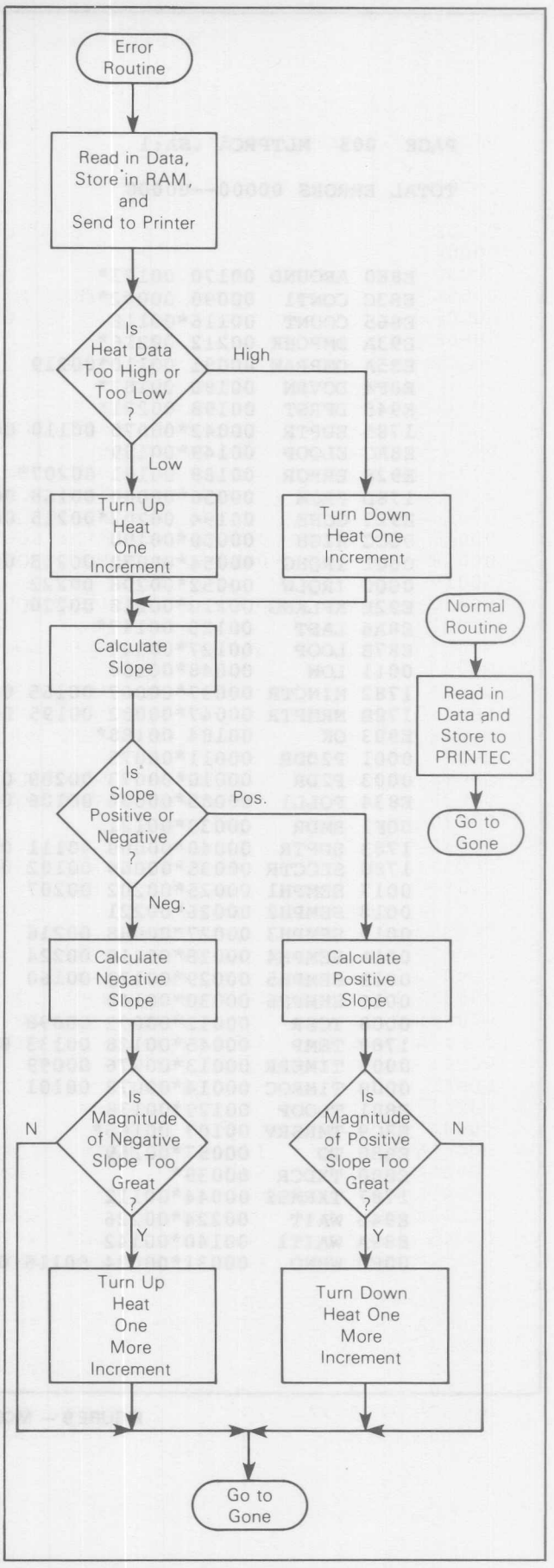

FIGURE 10 — MC6809E Flowchart (Concluded)

```
PAGE  001  MLTPR09A.SA:1

00001                          *
00002                          *
00003                                  OPT    ABS,LLE=85,S,CRE
00004                          *
00005                          *
00006                          *  THIS IS THE CODE THAT ALLOWS THE MC6809E TO
00007                          *       INTERFACE WITH THE MC68120 IN A
00008                          *         MULTIPROCESSOR CONFIGURATION
00009                          *
00010                          *
00011                          *
00012           0017     A SEMPH1 EQU    $17      ERROR SITUATION
00013           0018     A SEMPH2 EQU    $18      NORMAL RAM DUMP
00014           0019     A SEMPH3 EQU    $19      RQST FOR DATA (15 SEC INCR.)
00015           001A     A SEMPH4 EQU    $1A      IRQ CLEAR
00016           001B     A SEMPH5 EQU    $1B      60 BYTE BLOCK OF DATA FLAG
00017           001C     A SEMPH6 EQU    $1C      60 BYTE BLOCK OF DATA FLAG
00018           00F0     A WHNO   EQU    $F0      WHOLE NUMBER OF 60 BYTE BLOCKS
00019                          *                     TO BE MOVED
00020           00F1     A RMDR   EQU    $F1      REMAINDER OF THE 60 BYTE BLOCK
00021                          *                     TO BE MOVED
00022           0100     A LOWER  EQU    $100     WHEN THIS ADDRESS IS WRITTEN
00023                          *                     TO, THE SYSTEM T&P IS LOWERED
00024           0101     A RAISE  EQU    $101     WHEN THIS ADDRESS IS WRITTEN
00025                          *                     TO, THE SYSTEM T&P IS RAISED
00026           12F0     A CMPTMP EQU    $12F0    TEMP. STORAGE FOR COMPARE
00027                          *                     (2 bytes)
00028           12F2     A LAST1  EQU    $12F2    SLOPE POINT VALUES (1 byte)
00029           12F3     A DTAREQ EQU    $12F3    BUTTIN REQUEST FOR DATA
00030                          *                     (1-request;0-no request)
00031                          *
00032                          *
00033                          *
00034A F800                            ORG    $F800
00035                          *
00036                          *
00037                          *          INITIALIZATION   ROUTINE
00038A F800 10CE 13FF    A START  LDS    #$13FF   INIT. STACK
00039A F804 86   00      A        LDA    #$00     INIT. BUTTON REQ. FOR
00040A F806 B7   12F3    A        STA    DTAREQ   DATA (SET UP FOR NO REQ.)
00041                          *
00042                          *    START POLLING ON DUM RAM REQUEST AND WAIT
00043                          *          FOR INTERRUPT REQUEST
00044                          *
00045A F809 B6   12F3    A MAIN   LDA    DTAREQ   CHECK IF OPERATOR
00046A F80C 81   80      A        CMPA   #$80     REQUESTING DATA
00047A F80E 26   02   F812        BNE    OPEN
00048A F810 8D   07   F819        BSR    GETDTA   GO GET DATA
00049                          * LET IN IRQ INPUT
00050A F812 1C   EF      A OPEN   ANDCC  #$EF     CLEAR I BIT
00051A F814 12           A        NOP
00052A F815 1A   10      A        ORCC   #$10     SET I BIT
00053A F817 20   F0   F809        BRA    MAIN     BACK
00054                          *
00055                          *       GET DATA SUBROUTINE
00056                          *
00057A F819 97   19      A GETDTA STA    SEMPH3   ASK FOR DATA
00058A F81B 96   1B      A WAIT1  LDA    SEMPH5   WAIT FOR READY
```

FIGURE 11 — MC6809E Software

```
PAGE  002  MLTPR09A.SA:1

00059A F81D 84   80      A         ANDA   #$80      SEMAPHORE
00060A F81F 26   FA   F81B        BNE    WAIT1     BRANCH IF NOT READY
00061A F821 96   F0      A FCHDTA  LDA    WHNO      READY, READ HOW MUCH DATA
00062A F823 81   00      A         CMPA   #00       TO TRANSFER
00063A F825 27   1C   F843        BEQ    LAST      TX'FER REMAINDER IF WHNO =0
00064A F827 8E   00B0    A         LDX    #$00B0    PREPARE TO TX'FER (READ)
00065A F82A 108E 1000    A         LDY    #$1000    60 BYTE BLOCK
00066                     * LOADS DATA TO IPC PRINTER CONTROLLER AT $E000
00067                     *  THE CONTROLLER IS WAITING FOR THE DATA
00068A F82E EC   84      A MOVED   LDD    0,X       GET 2 BYTES
00069A F830 ED   89 E000 A         STD    >$E000,X  STORE TO PRINTER IPC
00070A F834 30   02      A         LEAX   2,X
00071A F836 ED   A1      A         STD    0,Y++     STORE 2 BYTES
00072A F838 8C   00EC    A         CMPX   #$EC      CHECK IF DONE (60 BYTES)
00073A F83B 26   F1   F82E        BNE    MOVED     GO AGAIN
00074A F83D 97   1C      A         STA    SEMPH6    CLEAR SEMPH6
00075A F83F 0A   F0      A         DEC    WHNO
00076A F841 20   D8   F81B        BRA    WAIT1     WAIT FOR NEXT BLOCK
00077A F843 8E   00B0    A LAST    LDX    #$00B0    INITIALIZE POINTERS FOR LAST
00078A F846 108E 103C    A         LDY    #$103C    TRANSFER ($1000+60=$103C)
00079A F84A 1F   10      A         TFR    X,D       CHECK HOW MUCH TO MOVE
00080A F84C D3   F0      A         ADDD   WHNO
00081A F84E 1083 00B0    A         CMPD   #$00B0    CHECK IF RMDR =0
00082A F852 27   15   F869        BEQ    OUT       AND GET OUT IF SO
00083A F854 FD   12F0    A         STD    CMPTMP    IF NOT GO MOVE BLOCK
00084A F857 7C   12F1    A         INC    CMPTMP+1  ADD 1 TI CMPTMP+1(00F1)
00085A F85A A6   84      A NXTBYT  LDA    0,X       GET NEXT BYTE OF DATA
00086A F85C A7   89 E000 A         STA    $E000,X   STORE TO PRINTER
00087A F860 30   01      A         LEAX   1,X
00088A F862 A7   A0      A         STA    0,Y+      STORE IN RAM
00089A F864 BC   12F0    A         CMPX   CMPTMP    CHECK IF DONE
00090A F867 26   F1   F85A        BNE    NXTBYT    IF NOT GO AGAIN
00091A F869 97   1C      A OUT     STA    SEMPH6    CLEAR SEMPH6
00092A F86B 39                    RTS
00093
00094                       *
00095                       *       IRQ  ROUTINE
00096                       *
00097                     * AH-HA! THE MC68120 WANTS TO TELL ME SOMETHING!!
00098A F86C 96   17      A IRQ     LDA    SEMPH1    CHECK IF ERROR SITUATION
00099A F86E 84   80      A         ANDA   #$80
00100A F870 27   09   F87B        BEQ    ERROR     BRANCH IF SO
00101A F872 96   18      A         LDA    SEMPH2    CHECK FOR NORMAL DATA
00102A F874 84   80      A         ANDA   #$80      DOWNLOAD
00103A F876 27   3B   F8B3        BEQ    NORMAL    BRANCH IF SO
00104A F878 97   1A      A CLRIRQ  STA    SEMPH4    WRITE CLEAR IRQ SEMPH
00105A F87A 3B                    RTI               BACK TO MAIN
00106                       *
00107                     *RECEIVE ERROR DATA ROUTINE
00108                       *
00109A F87B 8D   9C   F819 ERROR   BSR    GETDTA    GET DATA INTO RAM
00110A F87D BE   12F0    A         LDX    CMPTMP    GET ADDRESS OF LATEST DATA
00111A F880 A6   84      A         LDA    0,X       GET DATA AND CHECK
00112A F882 81   CB      A         CMPA   #$CB      IF DATA
00113A F884 25   07   F88D        BLO    CONT      TOO HIGH
00114A F886 86   CC      A         LDA    #$CC      IF SO - TURN DOWN TEMP.
00115A F888 B7   0100    A         STA    LOWER
00116A F88B 20   06   F893        BRA    SLOPE     GO TO SLOPE CHECK
```

FIGURE 11 — MC6809E Software (Continued)

```
PAGE  003  MLTPR09A.SA:1

00117A F88D 12              CONT   NOP
00118A F88E 86   11       A        LDA    #$11
00119A F890 B7   0101     A        STA    RAISE     INCREASE TEMP.
00120                       *
00121                       *  IF SLOPE NEG. - GETTING HOTTER
00122                       *  IF SLOPE POS. - GETTING COLDER
00123                       *
00124A F893 A6   82       A SLOPE  LDA    0,-X      GET LAST
00125A F895 B7   12F2     A        STA    LAST1     DATA AND NEXT
00126A F898 A6   84       A        LDA    0,X       TO LAST DATA
00127A F89A B0   12F2     A        SUBA   LAST1
00128A F89D 2A   0B    F8AA        BPL    MAGNC     BRANCH IF COLDER
00129                       *                              NEG. SLOPE- HOTTER
00130A F89F 84   7F       A        ANDA   #$7F      DELETE NEG SIGN
00131A F8A1 81   10       A MAGNH  CMPA   #$10      COMPARE MAG OF SLOPE
00132A F8A3 25   D3    F878        BLO    CLRIRQ    TO CRIT. SLOPE VALUE
00133A F8A5 B7   0100     A        STA    LOWER     LOWER TEMP. 1 INCR.
00134A F8A8 20   CE    F878        BRA    CLRIRQ
00135A F8AA 81   10       A MAGNC  CMPA   #$10      COMPARE MAG OF SLOPE TO
00136A F8AC 25   CA    F878        BLO    CLRIRQ    CRIT SLOPE VALUE
00137A F8AE B7   0101     A        STA    RAISE     RAISE TEMP. 1 INCR.
00138A F8B1 20   C5    F878        BRA    CLRIRQ
00139A F8B3 8E   0080     A NORMAL LDX    #$0080    PREPARE TO GET DATA
00140A F8B6 EC   84       A        LDD    0,X       GET DATA
00141A F8B8 ED   89 E000  A MOVIT  STD    $E000,X   MOVE TO PRINTER
00142A F8BC 30   02       A        LEAX   2,X
00143A F8BE 8C   00B0     A        CMPX   #$B0      CHECK IF DONE
00144A F8C1 26   F5    F8B8        BNE    MOVIT     KEEPING GOING
00145A F8C3 20   B3    F878        BRA    CLRIRQ    CLEAR IRQ AND OUT
00146                       *
00147A FFF0                 ORG    $FFF0
00148A FFF0       F800    A        FDB    START
00149A FFF2       F800    A        FDB    START
00150A FFF4       F800    A        FDB    START
00151A FFF6       F800    A        FDB    START
00152A FFF8       F86C    A        FDB    IRQ
00153A FFFA       F800    A        FDB    START
00154A FFFC       F800    A        FDB    START
00155A FFFE       F800    A        FDB    START
00156                       END
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000

   F878 CLRIRQ 00104*00132 00134
               00136 00138 00145
   12F0 CMPTMP 00026*00083 00084
               00089 00110
   F88D CONT   00113 00117*
   12F3 DTAREQ 00029*00040 00045
   F87B ERROR  00100 00109*
   F821 FCHDTA 00061*
   F819 GETDTA 00048 00057*00109
   F86C IRQ    00098*00152
   F843 LAST   00063 00077*
   12F2 LAST1  00028*00125 00127
   0100 LOWER  00022*00115 00133
   F8AA MAGNC  00128 00135*
   F8A1 MAGNH  00131*
   F809 MAIN   00045*00053
   F82E MOVED  00068*00073
   F8B8 MOVIT  00141*00144
   F8B3 NORMAL 00103 00139*
   F85A NXTBYT 00085*00090
   F812 OPEN   00047 00050*
   F869 OUT    00082 00091*
   0101 RAISE  0Q024*00119 00137
   00F1 RMDR   00020*
   0017 SEMPH1 00012*00098
   0018 SEMPH2 00013*00101
   0019 SEMPH3 00014*00057
   001A SEMPH4 00015*00104
   001B SEMPH5 00016*00058
   001C SEMPH6 00017*00074 00091
   F893 SLOPE  00116 00124*
   F800 START  00038*00148 00149
               00150 00151 00153
               00154 00155
   F81B WAIT1  00058*00060 00076
   00F0 WHNO   00018*00061 00075
               00080
```

FIGURE 11 — MC6809E Software (Concluded)

MOTOROLA *Semiconductor Products Inc.*

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC.

A14955 PRINTED IN USA 12-81 IMPERIAL LITHO C02127 18,000 AN850