PERQ CPU Technical Reference Manual

June 21, 1994 Version

by

Dr. Tony Duell

Contents

Chapter 1

Introduction

The PERQ 16K 20 bit CPU board contains some 261 integrated circuits varying in complexity from simple TTL gates up to a complete microcode sequencer. The operation of this unit is covered in general terms in the POS user manuals, but these manuals do not explain the operation of some of the more specialised sections of the PCB, for example, the Raster Operation system, and they also do not cover some of the more complicated features of many of the other sections. This manual attempts to remedy that, by explaining the operation of every gate, PAL equation, and state machine on the CPU module at the hardware level.

1.1 Types of PERQ CPU

The PERQ 1 and PERQ 2 series of workstations have been fitted with 3 different types of CPU module, namely the original 4K Writable Control Store board used in the PERQ 1, the enhanced 16K Writable Control Store module introduced with the PERQ 1a, and also used in the PERQ 2 T1 and the PERQ 2 T2, and finally, the 24 bit 16K WCS unit used in the PERQ2 T4. These modules all share many common features, although there are also some significant differences between them.

This Technical Reference Manual covers the PERQ 16K CPU module at component level, and explains the operation of every subsystem of this unit. The manual is also of relevance to the Original 4K module, provided that references to the extra sections of the 16K board (for example, the Multiply/Divide logic and the Extended microcode Sequencer) are ignored. The extra sections of the 24 bit CPU are also covered, in particular, the extended AMUX logic and the 24-bit lookahead carry.

Component references and Signal names used in this manual refer to the PERQ 20 bit 16K module, except for the extra circuits on the 24 bit version, where the component number of this

board is, of course, used. Therefore, although the circuitry on the other CPU modules is similar, as the layout is quite different, care must be taken to translate the component numbers to those of the board under consideration.

1.2 Audience

This manual was originally produced for Service Engineers who are diagnosing and repairing the PERQ CPU to component level, and who need to understand the operation of the module in order to diagnose faults. It will also prove useful to advanced programmers who wish to modify the PERQ's internal microcode and therefore need to understand how this microcode controls the operation of the CPU, and to hardware designers who wish to design custom Input/Output boards to fit into the OIO slot, who need information on the I/O bus operation and timing.

Finally, the manual should prove interesting to those who wish to see how a complete processor system operates, and how the various functional subsystems are interrelated.

1.3 Prerequisites

In order to fully understand this manual, the reader should be familiar with the operation of a microcoded processor system, and more particularly with the operation of the PERQ microcode instructions covered in, for example, the POS G.6 microprogrammer's reference manual, or the ICL publication 'Modifying the Microcode'. It is not, however, essential to have actually written a microprogram.

It is also necessary to understand the operation of digital electronic circuits, including the simple gates and flip-flops, but also including PALs and PROMs, and to understand the basic concepts behind finite state machines. This information can be found in many books on Digital Electronics, and no particular one will be recommended here.

Finally, the reader should have access to data sheets for all the integrated circuits used in the PERQ CPU, but especially to those for the 74S181 ALU and the AMD 2910 microcode sequencer, as these contain much information not contained in the circuit descriptions in this manual.

1.4 Structure of the Remainder of this Manual

The remaining 6 chapters of this technical reference manual cover the individual blocks of the PERQ CPU and their interrelations.

Chapter 2 explains the sections that feed data to the ALU, including the general purpose registers, Arithmetic stack, and the memory data inputs, along with the AMUX and BMUX multiplexers that select between them

Chapter 3 continues the description of the data path, with an explanation of the arithmetic logic unit, the output registers (including the memory address and data registers), and the multiplication and division subsystems

The Control Store is described in Chapter 4, along with the way in which the microcode instructions are decoded, and how they control the other CPU systems

The Control Store Address Sequencer, along with the conditional branch logic and microcode jump address are explained in chapter 5

Chapter 6 returns to the data path, with a description of the 16 bit barrel shifter and it's associated masking logic, together with an explanation of how these sections are used for both programmed shifts and graphics (raster) operations.

Finally, Chapter 7 describes the 2 finite state machines used to control memory cycles and Raster Operations.

Chapter 2

Data Paths 1 - Registers

2.1 Introduction

In explaining the operation of the PERQ CPU, there is a problem: It's impossible to understand the operation of the Data Path system without understanding the microcode word that controls it and the control system that provides that word. It's also impossible to completely understand the microcoded control system unless the data path that it controls, and also that loaded the control store in the first place is fully understood. However, it will be assumed that a microcode word has been provided by the control system and is available at the outputs of the data latches (U55, U60, U65, U110, U43, U41 and U69), and the (conceptually simpler) data path will be explained first. Then, when the operation of that system is understood, the control system will be explained

The format of the microcode word is given in figure 2.1, and the leftmost (most significant fields) control the data path. The leftmost field is the X field which selects the Destination Register, and thus the general purpose register will now be described

2.2 General Purpose (X,Y) registers

The main CPU data path is 20 bits wide, the same width as a memory address (24 bits on the rare T4 CPU board), and the registers are therefore 20 bits wide also. There are 2 copies of each of the 256 registers on the CPU board, one selected by the X field (and hence called the X-register) and fed to the A input of the ALU, while the other is selected by the Y field (and referred to as the Y-register) and fed to the B input of the ALU. Therefore, 2 different registers can be simultaneously selected, and fed to the appropriate inputs of the ALU. Then, when the arithmetic operation is completed, the output of the ALU can be simultaneously written to both copies of the register, one

Field	Width	Use
X	8 bits	X (Destination) Register Address
Y	8 bits	Y (Source) Register Address
A	3 bits	A Multiplexer Control
B	1 bit	B Multiplexer Control
W	1 bit	Enable Write Back to register
H	1 bit	Prevent DMA operations
ALU	4 bits	Select ALU operation
F	2 bits	Select control function
SF	4 bits	Select Special Function
Z	8 bits	Microcode Data Field
C	4 bits	Select Jump Condition
J	4 bits	Select Jump type

Figure 2.1: PERQ Microcode Word Format

in the X-registers, the other in the Y-registers. This duplication of the main registers speeds up CPU operation (as the 2 registers required for an ALU operation can be read in the same cycle), and was common practice in high speed processors, like the DEC PDP11/45.

Although the 2 banks of registers are very similar in design, there are some differences between them, so both will described.

2.2.1 X registers

The 256 20-bit X-registers are physically stored in 4 256*4bit RAM chips at the following locations on the CPU board :

Chip	Bits
U80	0-3
U81	4-7
U82	8-11
U111	12-15
U113	16-19

The registers are loaded from the active low outputs of the R-register which is described in Chapter 3. The outputs from the registers (which are also active low of course) are fed into the AMUX distributed multiplexer that is described below. The first thing to consider is the register control signals, and how the X microcode field addresses the register.

2.2.2 X register control

The chip select lines from the X-register RAMs are tied into a permanently enabled state, namely -CS1 is grounded, and CS2 is pulled up through a 1K resistor R1. This leaves 2 control signals, namely -OE (output enable) and -WE (write enable) to consider

The output enable signal is known as EX' (not Enable X register). It is provided by output 6 of U223 (the AMUX control decoder), and will be described below. Write enable is produced by U130b, a 74S20 4-input NAND gate. This gate combines 2 clock signals (CLK-51R and CLK16), which enable writing after the ALU has completed its operation, with the W signal from the microcode word and the ABORT' signal from the memory cycle control. The W field in the microcode word is used to enable or disable the writing back of the ALU result into the registers. The ABORT' signal disables the write-enable signal if the machine cycle needed data from the main memory and that data was not valid. This signal will be described in more detail in Chapter 7.

2.2.3 X register addressing

The next block to be described is logic between the microcode X field and the address inputs of the X-register RAMs. On the 4K PERQ 1 CPU board, this logic was little more than a buffer, but on the PERQ 1a (16K) CPU board, there is additional gating to provide the Indexed addressing feature of this processor. The logic operates as follows :

Each RAM address line is provided by a 74S51 AND-OR-INVERT gate. This gate performs the following logic function : $XADDR < n >= (X < n > .1) + (Index < n > .XIndexOn)$ where the signals are as follows :

Xaddr<n>	The nth address input to the RAMs
X<n>	The nth bit of the X microcode field
1	a logic one signal provided by the pull-up R5(1k)
Index<n>	The nth bit from the Index register
X Index On	Enable signal for indexed addressing of X

X Index On is produced by the 74S02 NOR gate U35d, which combines the 2 most significant X field signals, X7 and X6. Therefore, Indexing is enabled when one of the lower 64 X-registers is selected, and both these signals are therefore 0 The result of all this logic is that when one of the higher 192 registers is addressed by the microcode, then the address is passed unchanged, but when one of the first 64 registers is selected, the address is logically ORed with the output of the Index register. For reference, the gates that perform this address modification are :

Bit	Gate
0	U74b
1	U75a
2	U75b
3	U96a
4	U96b
5	U74a
6	U94b
7	U94a

2.2.4 Index Register

The Index register is U97, a 74LS273 latch. It is loaded from the outputs of the R register by the LD INDEX signal, and cleared on a processor reset by the INIT' signal. LD INDEX is produced by the Special Function decoder (U149) and will be described in Chapter 4. This register is common to the address modification of both the X and Y registers.

2.3 Y registers

Like the X registers, the Y registers are physically produced from 256*4bit RAMs at the following locations on the CPU board:

Chip	Bits
U62	0-3
U61	4-7
U63	8-11
U64	12-15
U83	16-19

2.3.1 Y register control

The Y register chip enable signals are tied to permanently enable the RAMs, in exactly the same way as the X RAM enable lines described above. The Write Enable signal is also provided by the same signal as the X register Write Enable.

This leaves the Output Enable signal. This signal comes directly from the single-bit B field in the microcode word. Therefore, the outputs of the RAMs are enabled when the B field is 0. This field will be described more fully in the BMUX section below.

2.3.2 Y register addressing

The first stage of the logic to address the Y registers is the same as the X register addressing logic described above, provided the 'X' in the signal is replaced by a 'Y'. The Y Index On signal is produced by the 74S02 NOR gate U95a, while the AND-OR-INVERT gates that actually perform the address modification are as follows :

Bit	Gate
0	U98a
1	U120b
2	U98b
3	U120a
4	U119a
5	U119b
6	U93b
7	U93a

However, the outputs of this circuit are not fed directly to the Y register RAMs, but are instead connected to the Y address multiplexer built from 2 74S158 devices, U76 and U99. This circuit switches the Y RAM address inputs between the Y address logic (when the select input is 0) and the X address logic (when the select input is 1). The select input is connected to the CPU clock signal CLK-4F, so that during the first part of the microcycle the Y register is controlled by the Y microcode field (and thus the register selected by this field is read into the ALU), while in the second part of the microcycle, when the output from the ALU is being written back to the registers, the multiplexer switches over, so that both register banks are controlled by the X field, and so the copy of the destination register (selected by the X field) in both banks is updated simultaneously.

2.4 Distributed Multiplexers

A multiplexer is an electronic multi-way switch that is used to select several different signals. For example, one is needed at the A input to the ALU, to select between memory data, registers, and the other A-line sources. A normal multiplexer is one circuit that selects between these signals - like a multi-way rotary switch. A distributed multiplexer is rather like an interlocked set of push-button switches - each signal has it's own on/off switch (in this case, a 3-state output) which connects the signal to a common bus. External logic ensures that only one output is enabled at a time. In the PERQ CPU, there are 3 main distributed multiplexers, the AMUX and BMUX at the inputs to the ALU, and the JMUX that is used to select microcode jump addresses. The latter will be described in chapter 5, while the other 2 are described below

2.5 AMUX

The AMUX is, as its name implies, a distributed multiplexer used to select the A input to the ALU. It is controlled by the 3-bit A field in the microcode word, and can therefore select between 8 inputs. These 8 inputs will now be described, followed by a discussion of the AMUX control logic.

2.5.1 Input 0 - Shifter

The 16-bit output from the shifter (SHIFT<n>) is gated onto the low 16 AMUX lines via 74S240 inverting buffers, controlled by the ENB SHIFT' signal. The following buffers are used for this signal :

Bits	Buffer
0-7	U140
8-11	U225a-d
12-15	U202e-h

The operation of the shifter is described in chapter 6.

2.5.2 Input 1 - Opcode file

The opcode file is a 8-byte dual port register file built from 4 74LS670 chips, which is automatically loaded from the 16 bit memory data bus. The operation and control signals for this register file are described in chapter 5 in relation to the microcode jump address function, which is the main use of this circuit. For the moment, the only feature that needs to be noted is that the outputs of this register file (OP<n>) can be transferred onto the lowest 8 AMUX lines via the 74S240 buffers U161e-h and U182a-d, when the NEXT OP' signal becomes active. This signal also enables the Byte Program Counter (BPC) via the 74S00 NAND gate U203b, so that the BPC is incremented on such an instruction. The operation of the BPC is described more fully in Chapter 5.

2.5.3 Input 2 - Input Port

During an Input/Output microinstruction, the IOB ENB L signal becomes active when the data on the IOD<n> lines is valid. This signal clocks the data into the 74S534 latches at positions U245 (low 8 bits) and U248 (high 8 bits). The 3-state output of these chips is gated onto the AMUX lines when the EIOB' signal becomes active.

2.5.4 Input 3 - Memory Data

The 16 bit memory data output bus (MDO<n>) is stored in the memory data output register (MDOR) consisting of the 74S534 latches U247 (bits 0-7) and U252 (bits 8-15) by the CLK MDOR signal. This signal is provided first latching the MDO VALID H (Memory Data Output Valid) signal from the memory board, which arrives at the CPU board on pin 141 in the latch U209h to provide the LMDO VALID H signal (Latched Memory Data Output Valid). This signal is then NANDed with the CLK-4E clock signal by the 74S00 U232c to produce the CLOCK MDOR signal.

The outputs of the MDOR are gated onto the AMUX0-15 lines by the EMDO' signal from the AMUX control logic.

2.5.5 Input 4 - Memory Data Extended

Since the Memory data bus is 16 bits wide, while the CPU data path is 20 bits wide, there needs to be a way of transferring memory data into the upper 4 bits of a register. This is performed by the MDX (Memory Data eXtended) logic. The lowest 4 bits of the MDO bus are latched in a 74S534 at location U230 by the CLK MDOR signal, and thence gated onto the AMUX16-19 lines by the ENB MDX' signal, which enables the 3-state outputs of this latch.

2.5.6 Input 5 - Microstate Register

The microstate register is a collection of 20 status bits that are transferred to the AMUX lines via 3-state buffers. These buffers are enabled by the ENB UST' signal from the AMUX control logic, and provide the following signals :

Bit	Buffer	Signal
0	U202d (74S240)	BPC<0> (Byte Program Counter bit 0)
1	U202c (74S240)	BPC<1>
2	U202b (74S240)	BPC<2>
3	U202a (74S240)	BPC<3>
4	U185h (74S240)	OVF (Overflow Flag)
5	U185g (74S240)	EQL (Equal Flag)
6	U185f (74S240)	CRY (Carry Flag)
7	U185e (74S240)	LSS (Less Flag)
8	U254d (74S240)	0
9	U254c (74S240)	STK EMP' (Stack Empty)
10	U254b (74S240)	0
11	U254a (74S240)	0
12	U123h (74S240)	BMUX<16>' (Inverted B Mux output)
13	U123g (74S240)	BMUX<17>'
14	U123f (74S240)	BMUX<18>'
15	U123e (74S240)	BMUX<19>'
16	U145/O0 (16R4)	CCSR0 PAL Arith X,Y diagnostic *
17	U145/O1 (16R4)	CCSR0 PAL Carry bit 15 diagnostic *
18	U145/O2 (16R4)	CCSR0 PAL Arith X,Y diagnostic *
19	U145/O3 (16R4)	CCSR0 PAL Equal diagnostic*

(*) signal undocumented in PERQ systems manuals
and forced to 0 by a bug (?) in the AMUX control logic

2.5.6.1 The CCSR0 PAL

The top 4 signals in the above table are forced low by the AMUX control logic and the 'force 0 buffers' described below. However, they are also driven by a 16R4 PAL, which is inaccessible in a normal PERQ, and, therefore, not documented in the Microcode reference manual. Its function seems to be to enable certain parts of the multiplication logic to be checked. The PAL accepts 5 bits from the output of the ALU (0-3 and 15) and the control signals (Arith X, Arith Y, LC<15>, LB<15>, LA<15>, LA=B) that are used by the multiplication and condition code logic. The latter signals whose names start with 'L' are stored in the 67S380 transparent latch at location U165 which is controlled by the CLK-0R C clock. Therefor, during the second half of the microcycle, when the CLK-0R clock is low, these signals are stable.

Signal	Latched By	Produced from
LC<15>	U165c	C<15> - ALU 16 bit Carry out
LB<15>	U165b	BMUX<15>' - BMUX 15th bit
LA<15>	U165a	AMUX<15>' - AMUX 15th bit
LA=B	U165d	A=B - ALU lower 16 bits all 0 (=) output

Arith X and Arith Y are provided by the ALU control PROM ALD16.1 at location U105, and indicate if an add or subtract operation is being performed by the ALU. This circuit is described in Chapter 3 as part of the ALU control logic.

In order to use this facility, an ALU operation is performed to set the latched signals described in the last paragraph, together with the R register outputs, to the correct values. The special function decoder described in chapter 4 then generates the MUL STEP clock which clocks the latches in the CCSR0 PAL, and stores the result. The outputs of the PAL can then be read via the microstate register onto the AMUX lines.

Signal R<15> seems to act as a toggle between checking the correct operation of the PAL (by feeding the other R lines through it) and checking the Condition logic. The logic functions are as follows

signal	R<15>'=0	R<15>'=1
AMUX<19>	R<3>	!LA=B
AMUX<18>	R<1>	!LA<15>+!ArithY.LB<15>+!ArithX.!LB<15> ArithX.!ArithY.!LA<15>.LB<15>
AMUX<17>	R<2>	LC<15>
AMUX<16>	R<0>	!LA<15>+LB<15>.!ArithY+!LB<15>.!ArithX

2.5.7 Input 6 - X Register

This is probably the simplest AMUX input of all - the EX' output from the AMUX control logic simply enables the 3-state outputs of the X register RAMs, which therefore pass their data onto the AMUX<n> lines.

2.5.8 Input 7 - The Stack

The 16 level arithmetic stack forms the last input to the AMUX. Physically, the stack consists of 5 16*4 bit 27S07 RAMs, at locations (in increasing bit order) U121, U163, U220, U226 and U125. The 3-state data outputs of these RAMs are connected directly to the AMUX lines, and the Data inputs to the R register outputs (the R register is the latch at the output of the ALU and is described in Chapter 3). The RAM addressing and control will now be described.

The RAM address is provided by the 4-bit Stack Pointer – the synchronous 74S169 counter at location U233. The parallel load inputs of this counter are connected to ground, so that the parallel load function acts as a reset. This load input is provided by the STK RESET' signal. The direction of the counter is controlled by the POP' signal, which causes the counter to decrement on a POP instruction, and increment on a PUSH instruction. The counter is clocked by the CPU clock signal CLK-4J and enabled on these instructions by the !P input, which is produced by the 74S00 NAND Gate U237c. This gate combines the STK CLK ENB H (Stack clock enable) with the ABORT' signal, thus preventing stack pointer changes if the cycle is aborted due to a memory cycle not being ready. Finally, the Carry output of the counter provides the STK EMP' input to the microstate register described above.

There are 2 control inputs to the chips that make the data stack, namely CE (Chip Enable) and WE (Write Enable). Both signals are active low. The write enable signal is produced by the 74S20 NAND gate U139a. It combines 2 CPU clock signals, CLK-16 and CLK-4J, which ensure that the RAM is only written to when the data inputs are valid, with the ABORT' signal (to ensure that the stack will not be altered on an aborted cycle), and the STK WE (Stack Write Enable) signal. The Chip Enable signal comes from the 74S08 AND gate U231b, which combines the Write Enable signal with the AMUX control line ENB STK'. Since all the signals involved are active low, this gate enables the RAM if either the AMUX is selecting the stack OR the stack is being written to.

2.5.9 Force 0's

As I mentioned above, the main CPU data path, and hence the AMUX lines are 20 bits wide. However, not all the inputs use all 20 bits, the Opcode file is 8 bits wide (0-7), the MDX input is 4 bits wide (16-19), and so on. When such inputs are selected, it is necessary to fill up the rest of the 20 bit word with logical 0's. Rather than use a separate system for each possible input, there are common buffers for the 0-7, 8-15 and 16-19 bit fields. They are located as follows.

Field	Buffer	Control signal
0-7	U133 (74S240)	AZRO<7:0>'
8-15	U253 (74S240)	AZRO<15:8>'
16-19	U146e-h (74S240)	AZRO<19:16>

In all cases, the inputs to the buffers are connected to ground, while the outputs are connected directly to the AMUX lines. The 3-state control inputs to these buffers come from the AMUX control logic that will be described next

2.5.10 AMUX Control Logic

The control logic for the AMUX distributed multiplexer is composed of 2 circuits. The first is the 74S138 decoder (U229). This decoder is permanently enabled, and is controlled by the 3 A bits (A<0>, A<1>, A<2>) from the microcode word data register. The 8 outputs of this decoder provide the enables for the 8 input systems described above in the standard way.

The second control section is the 74S288 PROM at location U188, and known as AMX16. This PROM provides the control signals for the Force Zero buffers, a signal to indicate that the A field has selected one of the 2 memory data functions (MDO and MDX) and a shift type signal (SH TYPE) to the shifter logic. The PROM is controlled by the A field of the microcode word, and also by the F (Function) 2-bit field. The PROM decodes the fields to produce these signals in the following way :

Address	Signal
A0	F<0>
A1	F<1>
A2	A<0>
A3	A<2>
A4	A<1>

ROM Bit	Signal	Active when
0	Proc Needs MDI	A=3 (MDO) A=4 (MDX)
1	SH TYPE	F=2 (Enable Z field as Shifter Control)
3	AZRO<7:0>	A=4 (MDX)
4	AZRO<15:8>	A=1 (NextOp) A=4 (MDX)
5	AZRO<19:16>	A=0 (Shifter) A=1 (NextOp) A=2 (IOB Input Port) A=3 (MDO) A=5 (Microstate). This causes a bus contention with the CCSR PAL, and may be a bug in the design

2.6 BMUX

The BMUX is the 2-input distributed multiplexer that selects between the Y register and constants at the B side input of the ALU. This multiplexer operates as follows

2.6.1 Input 0 - Y register

The single B bit of the microcode word is directly connected to the output enable pins of the Y register RAMs. Therefore, when the B field is 0, the RAMs have their outputs enabled, and pass their connects onto the BMUX lines.

2.6.2 Input 1 - Constant

The constant control logic is slightly more complicated, because both 8 bit and 16 bit constants are allowed by the hardware. In both cases, the top 4 bits are all 0's, and the lower 8 bits provided by the Y microcode field.

The 74S240 3 state buffer at location U101 connects the Y microcode field lines to the BMUX<7> - BMUX<0> lines when it is enabled. This circuit provides the lower 8 bits of all constants. Similarly U102a-d (another 74S240 3-state buffer), which has its 4 inputs grounded, forces 0's onto the top 4 BMUX lines. Both these buffers are enabled by the B' signal, which is simply the inverted output of the latch that stores the B field.

The BMUX<15>-BMUX<8> lines are driven by different buffers depending on whether a long or short constant is selected. In the case of a short constant, the buffers U103e-h and U102e-h force 0's onto the appropriate BMUX lines. These buffers are enabled by the SHORT CONST' signal, and have their inputs grounded. U123a-d and U124a-d transfer the contents of the Z microcode field (Z0-Z7) onto the BMUX<8>-BMUX<15> lines when a long constant is required. These buffers are enabled by the LONG CONST' signal. All the buffers involved are 74S240's.

The NOP SF' signal from the JPPAL3 at U108 determines the size of the constant, being low for a long constant, and high for a short one. This signal is low during normal long constant cycles (F field is either 0 or 2, SF field is 0), and also during Push Long Constant instructions, when the F field is 1 and the SF field 5. The resulting signal is NANDed with the B field in the 74S00 gate U86c to provide the SHORT CONST' signal. Similarly, the NOP SF signal, produced by inverting the NOP SF' signal in the 74S04 NOT gate U90e, is NANDed with the B field in U86b, another 74S00, to give the LONG CONST' signal.

2.7 The modified AMUX control in the T4 CPU

The AMUX control logic was totally redesigned in the 24 bit T4 CPU. The AMX16 prom was replaced by a 512*8bit PROM designated 'AMUX24' located at U140. The 74S138 decoder remains, and is now at location U215. There are 2 microstate registers (in T4 mode), which selected by the state of the H microcode field. These registers are enabled by outputs on the AMUX24 PROM, and output 5 of the U215 is no longer used for this function. The address inputs of the AMUX24 PROM are assigned as follows:

Addr	Signal
A0	F<0> (Microcode F field)
A1	F<1> (Microcode F field)
A2	A<0> (Microcode A field)
A3	A<2> (Microcode A field)
A4	A<1> (Microcode A field)
A5	H (Microcode H field)
A6	CPU type - Controlled by JP1 (Link to earth) and pulled High by R8. Set to 0 for a T2-compatible CPU, and 1 for a T4 CPU
A7,A8	Connected to ground

The Outputs produced by the ROM are similar to those produced by the AMX16 ROM in the normal 20 bit CPU. The signals are given here.

Bit	Name	Active when	Comments
D0	Ustate1'	A=5&H=1&T4	U166 CE'. Select 2nd Microstate register. U166 places BMUX<19:23> on AMUX<0:7>
D1	Ustate'	A=5&(H=0|T4=0)	Enable main microstate register if A=5 and either it's a T2 or H=0 This is a standard 20-bit like microstate register
D2	AZRO<16:19>	A=0	Shifter
		A=1	NextOp
		A=2	Input Port
		A=3	MDO
		A=5&H=1&T4	Ustate1 register Note: Ustate<16:19> are Not forced to 0. The CCSR0 PAL Can be read in a T4 (in both modes)
D3	AZRO<8:15>	A=1	NextOp
		A=4	MDX
		A=5&H=1&T4	Ustate1 register
D4	AZRO<0:7>	A=4	MDX - Identical to 20 bit CPU
D5	AZRO<20:23>	A=0	Shifter
		A=1	NextOp
		A=2	Input Port
		A=3	MDO
		A=5	Ustate,Ustate 1 (Both microstate registers)
D6	Sh Type	F=2	Identical to 20 bit CPU
D7	Proc Needs MDI	A=3	MDO
		A=4	MDX (Identical to 20 bit CPU)

Bit 8 of the main microstate register is driven by the CPU type signal from JP1. It's 0 if the CPU is set to 20-bit compatible mode, and 1 if the CPU is set to T4 mode

Chapter 3

Data Paths 2 - ALU, Output registers, Multiplication and Division

Having seen how to obtain 2 20 bit words (one form the AMUX, and the other from the BMUX), the means by which they processed by the Arithmetic Logic Unit - the ALU will be described.

3.1 The 20 bit ALU

The PERQ CPU uses the popular 74S181 ALU to combine the AMUX and BMUX signals and produce the ALU Y lines. A complete logic diagram to gate level of this ALU is published in e.g. the Texas Instruments TTL Data Book Volume 1, and therefore little will be said about the operation of the device here. For the moment it is sufficent to note that the device combines 2 4 bit words using either one of 16 logic functions or 1 of 16 arithmetic functions (not all the functions are used in the PERQ).

The distinction is that arithmetic functions have a carry signal between adjacent bits, whereas logic functions do not. The carry circuitry, and hence the choice of arithmetic of logical functions is controlled by the 'M' input to the ALU chip, and the particular function selected by the 4 S inputs. In the PERQ, these signals are driven by the ALU control PROM (ALD16.1), which decodes the ALU field in the microcode word. This logic is described in the ALU control section below.

Since each ALU processes 4 bits, 5 devices are needed for the PERQ's 20 bit processor. They are physically located at the following locations on the CPU board :

Bits	Location
0-3	U133
4-7	U132
8-11	U152
12-15	U153
16-19	U154

The input and output signals for the ALU will now be described.

3.1.1 ALU Data inputs

The inputs to the ALU are simply the AMUX and BMUX lines that were described in the last chapter. The apropriate common line for each bit is directly connected to the A or B ALU input pin. Note that the ALU in the PERQ operates on active-low data and produces an active-low result.

3.1.2 Carry

For an arithmetic operation, like addition, there needs to be a carry signal from each bit to the next higher bit. In fact, the ALU chips used in the PERQ have a carry input signal (which is the carry input to the least significant bit of that ALU) and a carry output signal (the carry output from the most significant bit). The simplest scheme (the so-called 'Ripple Carry), could be implememtned by linking the carry out of one ALU chip to the carry in of the next. Conceptually, the ALU circuit has a logic output from each bit which is fed into the processing circuit for the next bit. Although simple, this system has the disadvantage that the carry into the highest bit is only valid when all the lower bits have stabilised, so the addition time is the sum of the propagation delays for each bit. This therefore means that adders for long word lengths are slow.

In the PERQ, a faster system, called 'lookahead' carry. Each ALU produces 2 signals, the carry generate signal G<n>' and the carry propagate signal P<n>'. Although, in the PERQ, these signals are actually active low, in the description that follows, the signals will be described using a positive logic convention, that is a signal that is active will be given the level 1. In order to explain how these signals are generated and used, consider the following 4-bit additions:

sum	G	P	Comments
0011+0101=1000	0	0	No carry out, carry in will not cause a carry out
1000+0111=1111	0	1	No carry out, carry in propagated to carry out
1000+1100=0100	1	0	Carry out, no matter what carry in is

So, the generate signal G is produced if this addition produces a carry out, independant of the carry in signal, whereas the propagate signal is active if a carry out would be produced had carry in been active. The carry output from this 4-bit stage is therefore given by $Cout = G + P.Cin$. However, this is not a simply a confusing way of describing a carry.

The important thing is that, unlike a carry out signal, the state of the G and P signals depend only on the 4 bits being considered - i.e. for each of the 74181's in the PERQ, the states of the G<n>' and P<n>' signals does not depend on the state of the carry input. Therefore, given an A and B word to process, the 5 ALU chips can calculate the 5 sets of G<n>' and P<n>' simultaneously, without the problem that the (unknown) carry input to that ALU chip could affect the state. Based on these signals, a fairly simple logic ciruit that will be described in more detail below can determine the states of the carry inputs to each of the ALUs. After these signals have stabilised, the ALU outputs may have to change, based on the carry in signals, but the G<n>' and P<n>' signals will not.

The carry input to a given ALU stage is given by :

		Explanation
Cin<n>=	G<n-1>	Stage below generated a carry.
	+P<n-1>.G<n-2>	Stage below propagated a carry generated 2 stages below.
	+P<n-1>.P<n-2>.G<n-3>	
	+etc	

The way this logic is implemented in the PERQ 20 bit CPU will now be described. U142 is a 74S182 Look-ahead carry generator. This chip accepts the G<n>' and P<n>' signals from 4 ALUs along with a carry input, and, using the equation given above, determines the Carry input signals for the 3 higher ALUs (The carry input for the least significant ALU is simply the carry input).

Final propagate and generate signals are also produced for the entire ALU, but these are not used in the PERQ. The 3 carry outputs of U142 are connected to the carry inputs of U132,U152 and U153 respectively to provide a look-ahad carry over the lower 16 bits of the ALU.

The is actually little speed disadvantage in using a ripple carry from the ALU handling bits 12-15 to the most significant ALU. So, as U142 can only control a total of 4 ALUs, this is what was done. The carry input on the most significant ALU (U154) is simply linked to the carry output of U153.

The carry output from bit 15 – signal C<15> is latched in U165c (67S380) by the CLK-0R C clock. It is therefore stable throughout the second half of the cycle before being latched in the carry flip-flop (U209e - 74S374) by the rising edge of CLK-0R B at the end of the microcycle. The output of this flip-flop – old carry' is processed by the ALU control PROM ALD16.1 (at location U105) to provide the carry input for the ALU on the next microcycle. The main carry flag therefore operates over 16 bits.

Carry<19> is the carry output from the most significant ALU bit. It is latched by U165e in a similar manner to the C<15> signal to produce the LC<19>' signal. This signal is then clocked into the extended carry flip-flop U209g by the CLK-0R B signal at the start of the next microcycle. The output of this flip-flop is called OLD CARRY<19>', and is used by the condition code logic.

3.1.3 Carry logic in the 24 bit T4 CPU

The carry logic in the 24 bit version of the PERQ CPU is basically similar to the 20 bit machine. Two 74S182 chips are used, U242 for the lower 4 ALU chips and U243 for the higher 2. Therefore, look-ahead carry is used over all 24 bits in this processor. Link JP2 selectes whether the equivalent of Carry<19> comes from either the carry output of the ALU handling bits 16-19 (U263), when the machine acts like a 20 bit CPU, or the carry output of U263 (which handles bits 20-23), whereupon the processor acts as a 24 bit machine.

3.2 ALU outputs - the R register

The outputs of the ALU (the ALU Y(n) signals) are latched in the 74S373 transparent latches by the LATCH R signal from the clock generation circuitry. This signal is low throughout the second half of a microcycle (while the CLK-0R signal is low), thus latching the ALU outputs. During this half of the cycle, the registers and stack are updated, but the R latch ensures that any changes made to the inputs of the ALU will not affect the outputs. The LATCH R signal goes high shortly after the start of each microcycle, allowing the contents of the latch to be updated, before going low again shortly before the CLK-0R signal does. The chips that constitute the R-latch are as follows :

Bits	Latch
0-7	U191
8-15	U235
16-19	U143a-d

The outputs of U143 are permantently enabled, since the output enable pin is grounded. However, the outputs of the latches for the lower 16 bits can be set to the high impedence state, since these R-lines can also be driven by the Multiplier register or the Victim latch. The output enable pins of U191 and U235 are driven by the 74S10 NAND gate U213b. This gate disables the latch outputs when either RD VICTIM or ENB MDR L are active (low), thus enabling either of these circuits to drive the R-lines.

The outputs of the R register are linked to the data input pins on the RAM chips that make up the X and Y registers. During the second half of the microcycle, the Wr RAM' signal goes active (unless inhibitied by the W microcode field), and the contents of the R register are written to the appropriate X and Y registers. The Register contol logic is described in chapter 2.

3.3 The memory data register

Since the memory address and data words are calculated by the main ALU, this is a logical place to describe the registers that store these words.

The ALU Y<n>' data outputs are stored in the 67S380 inverting transparent latches (U258 for the low byte, and U259 for the high byte) by the LATCH R signal. The outputs of these latches drive the backplane memory data input lines, MDI<n>. These outputs are enabled by the ENB MDI' signal from the 74S10 NAND gate U213c, which combines the following 3 signals : MEM RDING MA/MD, RO/PS', and CLK-4E. These signals have the following significance :

The MEM RDING MA/MD signal is driven by the GMV02 PROM (U204) in the memory state machine. It is active during a memory cycle when the memory card is accepting address or data words. The operation of this state machine is described in chapter 7.

The RO/PS' signal disables the MDI outputs when a raster operation is in progress, since the raster op logic drives the MDI lines itself.

Finally, the CLK-4E signal ensures that the the outputs are only enabled during the second half of a microcycle (when the CLK-0R signal is low). This is the time that the latch is holding the valid memory data.

3.4 The memory address register

The memory address register is similar to the memory data register just described. The 20 bit ALU Y<n>' output word is stored in the 74S534 latches by the rising edge of the CLK MA'R<5> signal. The latches used to perform this function are as follows :

Bits	Latch
0-7	U255
8-15	U257
16-19	U256d-g

During a memory access cycle (that is, when the F microcode field contains 1, and the high bit of the SF field is set), the Load MA output from the 74S138 decoder U149 in the special function decoder is selected. This output goes low shortly after the CLK-4G output goes high at the start of the second half of the microcycle. The Load MA signal is inverted by U234e (a 74S04 NOT gate) to produce the CLK MA'R<5> signal. Therefore, the memory address register is loaded at the start of the second half of memory access microcycles.

The output enable of the memory address register is similar to that for the mmemory data register. The output enable lines of the 74S534s (signal ENB MA') are driven by the 74S00 NAND gate U232d. This gate combines the MEM RDING MA/MD and CLK-4E signals to enable the memory address register during the second half of those microcycles when the memory card is reading the address or data lines. The signals involved are described in the previous section. However, since the Raster Operation logic does not drive the memory address lines, but relies on the ALU to calculate the addresses for such operations, the memory address register is not disabled during such operations, and the RO/PS' signal is not an input to U232d.

3.5 The Output Register and IO Cycles

An input/output (IO) cycle is defined by the F microcode field containing 0 and the SF field containg 17(octal). This condition is detected by NANDing together the SF<0>, SF<1>, SF<2>, SF<3> signals and the F<0>' and F<1>' signals in the 74S30 NAND gate U216. The remaing 2 inputs to this gate are both driven by the DON'T IO ENB signal from the CPU clock circuitry. This signal is low for a short period at the start of each microcycle, thus preventing the output of the gate from going active until the remaining inputs have stabilised. The output of U216 directly drives the IOB ENB L signal on the backplane. This signal also drives the clock inputs to the input

port latches U245 and U248 that are described in raltion to the AMUX in chapter 2. Shortly after the DON'T IO ENB signal goes low at the end of the microcycle, the IOB ENB signal swings high and latches the contents of the IOD data bus in the input port latches, from where it can be read into the ALU via the AMUX.

A output cycle has the additional condition that the most significant Z field bit is high. The IOB ENB L signal is inverted by U185a (a 74S240 NOT gate which is permanently enabled), and then NANDed with the Z<7> microcode data bit and the CLK-4G clock signal by the 74S10 NAND gate U207b. The output of this gate, known as ENB IODO, therfore goes low during the second half of output microcycles. This signal enables the outputs of the IO output latches (U246 for the low byte and U249 for the high byte, both 67S380s) during the second half of such microcycles. These latches store the ALU Y<n>' data outputs when the CLK-0RB signal goes low during the second half of the microcycle, in an identical manner to the memory data register.

The IO address lines (IOA<n>) are produced by buffering the microcode word Z field (Z<n>) by the 74S240 inverting buffers U225e-h and U206e-h. These buffers are permanently enabled since the enable input is connected to ground. The Z field, and hence the IOA lines are valid throughout the microcycle.

3.6 The ALU control PROM

The particular function performed by the ALU is selected by the 27S29 PROM (ALD16.1) at location U105. This PROM combines the ALU microcode field, the carry input (old carry'), the multiplier control bits (MDR<0,1>), the lowest bit of the multiplier register (MDR<0>) and the most significant latched bit of the ALU output from the previous instruction, UW<15>. From these signals it produces the 5 bit ALU control word (ALUF<0-3> and ALU M), the ALU carry input (CN<0>, and the ArtihX,Y signals that are used by the condition code logic. These signals are connected to the PROM as follows :

Address	Signal
A0	ALU<3> (Microcode bit)
A1	ALU<2>
A2	ALU<1>
A3	ALU<0>
A4	OLD CARRY'
A5	UW<15>
A6	MD INSTR <0>
A7	MD INSTR <1>
A8	MDR<0>

Data	Signal
D0	ALU M
D1	ALUF<3>
D2	ALUF<2>
D3	ALUF<1>
D4	ALUF<0>
D5	ARITH X
D6	ARITH Y
D7	CN<0> (Carry Input)

Since there are 4 ALU control bits in the microcode word, there are 16 possible ALU functions. In 2 cases, namely when the ALU field contains 14 (function is A+B) and when the ALU field contains 16 (function is A-B), the state of the PROM outputs depends on the state of the multiplier control signals. For the 14 other functions, the ALU control lines depend only on the state of the microcode ALU field, and these will be described first.

3.6.1 Normal Functions

These 14 functions are best described by the following table :

ALU field	ALU F	ALU M	ALU fn	ArithX	ArithY	CN<0>
0	1111	1	F=A	0	0	0
1	1010	1	F=B	0	0	0
2	0000	1	F=!A	0	0	0
3	0101	1	F=!B	0	0	0
4	1110	1	F=A.B	0	0	0
5	1101	1	F=A.!B	0	0	0
6	0001	1	F=!(A.B)	0	0	0
7	1011	1	F=A+B	0	0	0
10	0111	1	F=A+!B	0	0	0
11	0100	1	F=!(A+B)	0	0	0
12	1001	1	F=A xor B	0	0	0
13	0110	1	F=A xnor B	0	0	0
15	1001	0	F=A add B	0	1	!(Old carry)
17	0110	0	F=A sub B	1	0	!(Old carry)

The Carry signal is forced low except for the A+B+carry (ALU field = 15) and A-B-carry (ALU field = 17) operations. Here, the ALD16.1 PROM inverts the old carry signal and feeds it to the ALU carry input. This inversion compensates for the inversion perfomed by U165c in the carry path. The ArithX and Arith Y signals are used to tell the condition code logic that an add or subtract was the last ALU function performed. ArithX is high for a subtraction, and ArithY for an addition. Both signals are low for all other operations. These signals are also set automatically on the additions and subtractions used in multiply and divide instructions, which will be described next.

3.6.2 Multiply and divide control of the ALU

3.6.2.1 The Add Instruction

The Add instruction is selected when the ALU field contains 14 (Octal). When the MDR<0,1> signals select either no operation, or unsigned divide (i.e. when MDR<1>=0), then the signals are set as follows :

ALU field	ALU F	ALU M	ALU fn	ArithX	ArithY	CN<0>
14	1001	0	F=A add B	0	1	0

However, when a multiply operation is selected (MDR<1>=1) then the function performed by the ALU is controlled by the output from the least significant bit of the multiplier register. The operations performed are :

ALU field	ALU F	ALU M	ALU fn	ArithX	ArithY	CN<0>
MDR(0)=1 (multiplier LSB = 0)						
14	1111	1	F=A	0	0	0
MDR(0)=0 (multiplier LSB = 1)						
14	1001	0	F=A add B	0	1	0

Therefore, the least significant bit of the Multiplier Register (a 16 bit shift register) controls whether the ALU performs an add operation or a simple transfer operation.

3.6.2.2 The subtract instruction

The subtract instruction is selected when the ALU microcode field contains 16 octal. Again, the operation of the ALU is controlled by the MDR<0,1> signals, but this time, the following operations occur.

When the MDR<0> signal is 0, that is, when either no operation or an unsigned multiply is selected, the ALU performs the following operation, independant of the states of any other control signals.

ALU field	ALU F	ALU M	ALU fn	ArithX	ArithY	CN<0>
16	0110	0	F=A sub B	1	0	1

For a signed multiply (MDR<1,0>=11, the ALU operation is controlled by the least significant bit of the multiplier register, in a similar way to the add instruction. Here's what actually happens :

ALU field	ALU F	ALU M	ALU fn	ArithX	ArithY	CN<0>
MDR(0)=1 (Multiplier LSB = 0)						
16	1111	1	F=A	0	0	0
MDR(0)=0 (Multiplier LSB = 1)						
16	0110	0	F=A sub B	1	0	1

This is used for handling the most significant bit of the signed multiply operation. Since the MSB of a signed number can be considered to have a negative value, if this bit is set, then the shifted multiplicand is subtracted from the product.

On an unsigned division cycle, the ALU operation is controlled by the latched Most significant bit of the previous ALU operation. This bit, which is, of course, a sign bit indicating if the previous result was negative, is taken from the microcode control store input latch, U56e, and is called UW<15>, but this reference to the control store has no significance in divide operations, which do not, of course, write to the control store. Anyway, the operations performed by the ALU on such a cycle are :

ALU field	ALU F	ALU M	ALU fn	ArithX	ArithY	CN<0>
UW<15>=1 (Last operation's result was +ve)						
16	0110	0	F=A sub B	1	0	1
UW<15>=0 (Last operation's result was -ve)						
16	1001	0	F=A add B	0	1	0

Therefore the sign of the last ALU operation controls whether the divisor will be added to or subtracted from the shifted remainder register

3.7 The multiplier register

In order to do a shift–and–add multiplication, 2 shift registers are needed, one to shifter the multiplicand or product, and to accumulate said product, and the other to shift the multiplier and look at the least significant bit each time. In the PERQ CPU, the first function is performed by the main CPU barrel shifter that's described in chapter 6, while the second is perfomed by a dedicated shift register – the multiplier register.

This consists of 2 74S299 8-bit universal shift registers are locations U91 (for the low 8 bits) and U89 (for the high 8 bits). The registers are cascaded in the conventional way by connecting QH' on U89 to SR on U91 and QA' on U91 to SL on U89, thus making a 16 bit Universal shift register. The bidirectional data lines of the register are connected to the R-register outputs of the ALU circuit, and all the control lines are commoned between the 2 halves of the register. Now, the manner in which the control lines are provided will be described.

The register's 3-state outputs are enabled by the ENB MDR L signal from the special function decoder circuit desribed in chapter 5. When the R=product special function is selected (F field = 1 and SF field = 4), this signal goes low, and enables the 3 state outputs via the G2* control input on

the 74S299's. It also disables the R register outputs via U213b, a 74S10 NAND gate, as described above.

The shift register is clocked by the MUL STEP signal from output 5 of the 74S138 decoder U149 in the special function decoder. This signal goes low shortly after the start of the MUL STEP microcycle, and then returns high, clocking the multiplier register at the begining of the second half of that cycle. The MUL STEP signal is eabled by the special function decoder on Multiply/divide cycles (F=1, SF=1), and also on Load Multiplier register micrococycles (F=1, SF=2), when the control lines to the multiplier register are forced into the load state.

The Shift register control lines, S1 and S0, are controlled by MDRI<0> and MDRI<1> respectively. These signals are provided by the MULT01 PROM at location U106. The Address and data lines for the PROM are connected as follows:

Address	Signal
A0	R<0>'
A1	R<15>'
A2	LA<15>
A3	LB<15>
A4	MDR<15>
A5	MD INST<1>
A6	MD INST<0>
A7	ARITH Y
A8	LD MDR
A9	ARITH X

Data	Signal
D1	PR<0> (Multiply Sign extension bit)
D2	MDRI<0>
D3	MDRI<1>
D4	PR<15> (Divide Quotient bit)

. As can be seen from this table, this PROM controls also provides 2 bits of the shifter input on multiply and divide cycles, and that function is described later. The control lines for the multiplier register are defined as follows :

MDRI<0>=MD Instr<0> . !(MD Instr<1>) + Ld MDR
MDRI<1>=MD Instr<1> + Ld MDR

These equations produce the following truth table

MDInstr<1>	MDinstr<0>	Ld MDR	MDRI<1>	MDRI<0>	Function	Comments
0	0	0	0	0	Idle	No Op
0	1	0	0	1	Shift Left	Divide
1	X	0	1	0	Shift Right	Mult
X	X	1	1	1	Load	Load Multiplier

So, the LD MDR signal from the Special Function decoder forces the shift register to perform a parallel load operation from the R register. Otherwise, the shift register shifts left on divide, and shifts right on multiply.

On multiply cycles, the least significant bit of the R register (R<0>') is shifted into the register via the SR input of U89. This allows the low bits of the product (calculated by the shfiter and ALU system) to be shifted into the multiplier register, and thus allows this register to store the low 16 bits of the 32 bit product. The complete multiplication algorithm is described below.

On divide cycles, the sign bit of the last result (bit R<15>') is inverted by the 74S240 buffer U151a, which is permanently enabled, and is then loaded into the multiplier register via the SL input of U91. The inverted sign bit is the bit of the quotient that has just been calculated, and therefore the complete quotient is built up in the multiplier register. Again, the complete algorithm is described below.

3.8 ALU to shifter latches

The other shift register required for a multiplication or division is provided by the 16 bit barrel shifter that is described in chapter 6. This shifter can also be used as part of the ALU data path for any shift function. The ALU inputs are appended to themselves to make a 32 bit number, and the shifter extracts a 16-bit field from the middle of this number, thus providing the required shift. This process is illustrated in figure 3.1, and is more completely described in chapter 6, where the masking function required for shifts is also described. The most significant bit of the left copy is never used, since including it in the result field would imply a rotate by 16 bits, with is (of course) the same thing as a rotate of 0. However, this bit is produced in the PERQ CPU.

The ALU R outputs are latched by 4 67S380 inverting latches at the following locations :

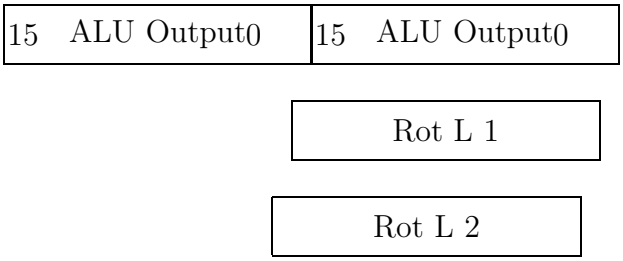

Figure 3.1: Example of Shifter operation

Latch	Bits
U118	24-31
U135	16-23
U117	8-15
U178	0-7

These latches hold the contents of the R register during the first half of the microcycle (that is, when the CLK-4F signal is low), so that the shifter inputs are stable, and the ALU can therefore process the shifted version of the previous ALU result, and are transparent during the second half of the microcycle, therefore updating the shifter inputs with the new ALU result. The latch clock signal is provied by NANDing the CLK-4F signal with the ABORT' signal (so that the latch is not updated if the microcycle is aborted) using the 74S10 gate U207c, and then inverting the result with the 74S04 NOT gate U208c. The outputs of the latches are controlled by the RO/PS signal from the Raster-Op control register, so that the latches are disabled on raster operations, and therefore the Raster-Op hardware can drive the shifter inputs.

Of the 32 inputs to these shifter input latches, all but 2 are directly connected to the apropriate R-register output (R<n> signal). The 2 exceptions are bits 15 and 16. Bit 15 is the least significant bit of a rotate left 1 bit, and is therefore the LSB of the partial remainder on division. Bit 16 is the most significant bit of a rotate right 1 bit, and is is therefore the sign extension bit on multiplication. These bits are provided by the MULT01 PROM, and the methods used for determining them will be described next.

3.8.1 MULT01-modified shifter inputs

Bit 15 of the shifter input is provided by the signal PR<15> from the MULT01 PROM. This signal is defined by the equation :

PR<15> = MDR<15> . !MD INST <1> . MD INST <0>
+R<15> . !(!MD INST <1> . MD INST <0>)

That is to say, bit 15 of the shifter input is simply the R<15> ALU output on all operations except division, when the most significant bit of the multiplier reigster (signal MDR<15>, from QA' of U89) is fed there instead. This allows the next bit of the divident to be shifted into the ALU path.

The logic for bit 16 (PR<0>) is considerably more complicated, since this signal is a sign extension bit on multiplication. I will therefore describe the logic for the 4 possible multiply/divide functions separately, rather than give the overall equation.

3.8.1.1 Sign extension - OFF

In this mode, PR<0> is simply the R<0> bit of the ALU output, and therefore no sign extension occurs.

3.8.1.2 Sign extension - Division

The logic for this mode is identical to the previous case, since the division operation does not require a shift right function, and so no sign extension is needed.

3.8.1.3 Sign Extension - Unsigned Multiply

The logic equation for this mode effectively defines a carry out from the top bit of the ALU, and consists of the following 6 terms, which will be explained separately.

!PR<0>'= (R15' . LA15 . ArithY . !ArithX)
+(!R15' . !LA15 . !ArithY . ArithX)
+(R15'. LB15 . ArithY . !ArithX)
+(LA15 . LB15 . ArithY . !ArithX)
+(!LA15 . LB15 . !ArithY . ArithX)
+(!R15' . LB15 . !ArithY . ArithX)

The first terms to be considered are those used by addition , which are the 1st, 3rd, and 4th lines of the above equation. The first term sets the carry out bit if the output has a 0 in the 15th bit, but the first operand had a 1 there. This can only happen if a carry was propagated into the 15th bit. Therefore a carry out should be generated. The 3rd line is identical, but considers the 15th bit of the second operand. Finally, the 4th line generates a carry if the 15th bit of both operands is set. In effect, this PROM performs a propagate-generate carry function as was described above.

The remaining 3 lines are used on subtraction, which is slightly unusual, because this operation is not used on unsigned multiplication. The second term sets the carry out bit if a subtraction changes a clear 15th bit of the 1st operand into a set 15th bit. Similarly, the 5th term sets the carry out bit if the 1st operand has a clear 15th bit, and the second operand a set 15th bit. Finally the last term sets the carry out signal if both the result and the second operand have set 15th bits. The exact use of this logic is somewhat obscure.

3.8.1.4 Sign Extension - Signed Multiply

In this mode, the MULT01 PROM performs a sign extension function onto the high bit of the shifted word. The function is defined by the following equation :

```
PR<0>'=    (R15' . !LA15)
          +(R15' . !ArithY . !ArithX)
          +(R15' . !LB15 . ArithY)
          +(R15' . LB15 . ArithX)
          +(!LA15 . !LB15 . ArithY . !ArithX)
          +(!LA15 . LB15 . !ArithY . ArithX)
```

These terms define when the sign bit of the result will be clear - that's to say when the result is positive. The first term say that if a positive result (R15' true) is produced from a positive first operand (LA15 false), then the result is genuinely positive, and the sign bit should be clear.

The second term says that a positive first operand will always produce a positive result if the function is neither addition or subtraction - when both ArithX and ArithY are false.

The 3rd and 5th terms are used on addition. Here, the result is truely positive if either a positive number is produced from a positive second operand (the case of a positive 1st operand is covered by term 1), or if both operands are positive.

Finally, the 4th and 6th terms cover subtraction. The sign bit is cleared (the result is positive) if either a positive result occurs on subtracting a negative number, or if a negative number is

subtracted from a positive number. It's fairly easy to see that these cases cover all the ones needed for signed addition and subtraction, and hence signed multiplication.

3.9 The multiplication algorithm

This hardware has been designed to perform a 16bit * 16bit multiplication giving a 32 bit product. The system operates as follows : Initially, the multiplier is loaded into the Multiplier shift register (U89 and U91), and the multiplicand stored in any CPU register.

On each cycle, the least significant bit of the multiplier (signal MDR<0>) controls the ALU via the ALD16.1 PROM. If this bit is set, the ALU adds the multiplicand register (on the BMUX) to the outputs of the shifter (on the AMUX), and feeds the result through the shifter, which is set to shift right by one bit. The shift register is then clocked at the end of the first half of the microcycle, and the multiplier is shifted one bit to the right. Simultaneously, the least significant bit of the ALU output (which would otherwise be lost by the shifter shifting right) is stored in the top bit of the multiplier register, since R<0> is connected to the SR input of U89.

This cycle is repeated a total of 16 times, after which the contents of the R register are stored (via the shifter) in a CPU register. This register contains the top 16 bits of the product, while the lower 16 bits are contained in the multiplier register, from where they can be transferred into a CPU register.

3.10 The Division Algorithm

The PERQ 16K CPU was designed to use a non-restoring division algorithm. This algorithm is related to the normal restoring 'long division' operation performed on binary numbers, and therefore the restoring algorithm will be explained first.

The traditional long division algorithm first compares the left shifted divisor with the dividend. If the left shifted divisor is less than the dividend, it is subtracted, and a 1 shifted into the rightmost bit of the quotient, otherwise no arithmetic operation is performed, and a 0 is shifted into the rightmost bit of the quotient. The divisor is now shifted one place to the right, and the process repeated, until all the quotient bits have been generated.

In practice, this algorithm is modified in 2 ways to make it easier to implement. Firstly, rather than shifting the divisor to the right (and having an ALU capable of processing all the bits of the dividend, which is in general larger than the divisor), it is easier to shift the dividend to the left, and to subtract the divisor from the most significant bits on each cycle. Secondly, it is simpler to always subtract the divisor from the shifted dividend, and to then look at the sign of the result. If

the result is positive, which occurs when the shifted dividend is larger than the divisor, then a 1 is shifted into the quotient, and the next bit is calculated. However, if the result is negative, then a 0 is shifted into the quotient, and the divisor is added back, thus restoring the original shifted dividend. In either case, the dividend is then shifted one bit to the left. The extra add operation needed to restore the orignal dividend is the reason that this method is known as the 'restoring division algorithm'.

Now, let the accumulator register that contains the dividend be called R, and the divisor be called D. After a subtraction that leaves a positive result, R is shifted left one place, and D is the subtracted from it, leaving 2*R-D. However, if the result in R is negative, then D is added back (the restore cycle), the result is shifted left one place, and D is again subtracted, leaving 2*(R+D)-D = 2*R+D. Therefore, the above algorithm can be further optimised by the following process : Start by subtracting the divisor from the shifted dividend, as in the restoring algorithm above. Then, on each subsequent operation, if the previous result was positive, shift a 1 into the least significant bit of the quotient, shift the modified dividend left 1 bit and subtract the divisor. If the result was negative, shift a 0 into the least significant bit of the quotient, shift the modified divisor left 1 bit and add the divisor. Thus, only one arithmetic operation is required per cycle.

In the PERQ, the dividend is initially stored in the multiplier register. On each division step, it is shifted left by one bit, and the most significant bit transferred into the least significant bit of the shifter via the MULT01 PROM described above. The ALU, which is controled by the ALD16.1 PROM then performs either an add or subtract operation depending on the sign of the previous operation - the UW<15> bit. The sign bit is also shifted into the least significant bit of the multiplier register, which thus builds up the quotient one bit at a time. After 16 such cycles, the complete quotient has been generated. An example of the division microcode is given in the POS G.6 microprogramming manual, and will not therefore be reproduced here, although the above description of the algorithm and hardware should help with the understanding of this microprogram.

Chapter 4

Control 1 : Clocks and Control Store

In the last 2 chapter the operation of the main sections of the PERQ 16K CPU data path has been explained. The control section will be described next, and the first part of this is the master clock generator

4.1 CPU Clock Signals

A timing diagram of the main PERQ CPU clock signals is given in figure 4.1. The signals are provided as follows :

4.1.1 Master clocks

The main timing signal, CLK-7R is supplied by the PERQ's RAM board. It enters the CPU on pin J176 and is buffered by the 4 sections of U240 (a 74S37 NAND gate), all of whose inputs are driven by CLK-7R. The outputs of these gates provide the first set of clock signals, CLK-4, as follows :

Gate	Signal
U240d	CLK-4E
U240a	CLK-4F
U240b	CLK-4G
U240c	CLK-4J

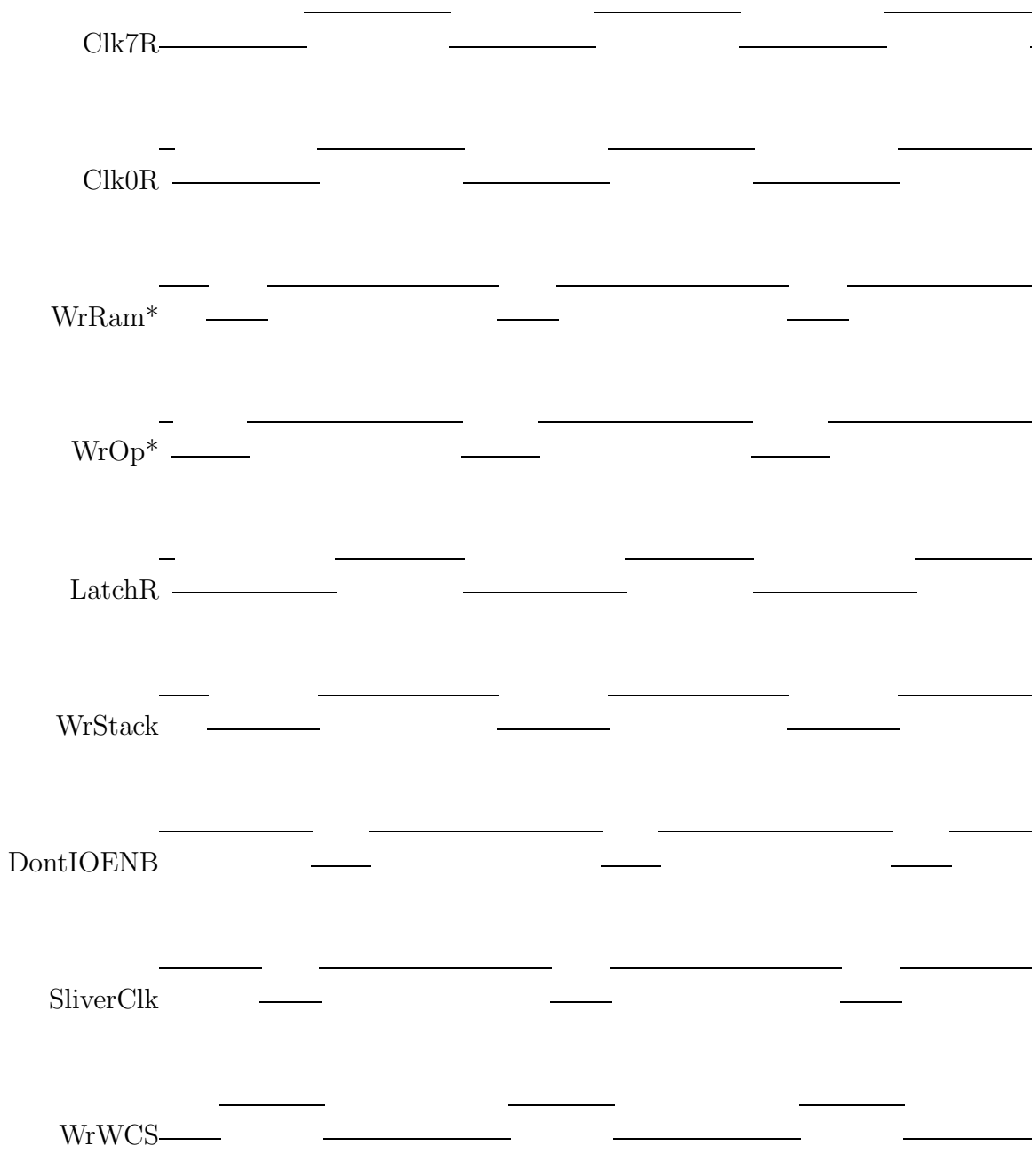

Figure 4.1: PERQ CPU Clock Timing Diagram

The only reason that 4 gates are used to provide these signals is to provide sufficient fan-out to drive all the sections of the CPU that use this clock. Note that CLK-4 is inverted with respect to CLK-7R and CLK-0R, and its transitions occur midway between those of CLK-7R and CLK-0R.

The CLK-4G signal is again buffered by 4 inverters (Since again a large fan-out is needed) to produce the main microcode clock signal, CLK-0R. The gates that perform this function are all 74S04's and are at the following locations:

Gate	Signal
U234d	CLK-0R A
U234c	CLK-0R B
U234f	CLK-0R C
U234b	CLK-0R D

The same CLK-4G is also fed into the 50ns delay line U77. This produces 5 further main clock signals, at 10ns intervals, namely CLK-6, CLK-16, CLK-26, CLK-36 and finally CLK-46. CLK-16 is inverted by the 74S04 NOT gate U90f, to provide the CLK20R signal. Similarly, CLK46 is inverted by the 74S04 NOT gate U234a to produce CLK-51R.

The names of the clock signals indicate the delay in nanoseconds between that signal and the main reference clock, CLK-0R. Unfortunately, the signal name does not indicate whether or not the clock edge occurs before or after the CLK-0R edge, although signals that end with a final letter R are inverted with respect to those that do not.

4.1.2 Derived Clocks

The following signals are produced by combining the master clock signals just described, and are used to time the operation of various parts of the PERQ CPU.

4.1.2.1 LD MIR'

The 2 signals LD MIR' A and LD MIR' B are used to load the microcode pipeline latches (see below) at the start of each microcycle. They are each produced by a 74S10 3-input NAND gate, U215b and U213a respectively. These gates NAND the CLK-4J signal with 2 other signals, ABORT' and WR NOW'. Therefore, the LD MIR' signal has the same timing as CLK-0R, unless either the microcycle is to be aborted (ABORT' from the memory control state machine is low), or the current cycle is the first of the 2 cycles used to load the writable control store, whereupon LD MIR' is forced high. Therefore, on such cycles the microcode pipeline is not reloaded.

4.1.2.2 WR RAM'

This signal is used to write back the contents of the ALU R register into the main CPU registers during the second half of the microcycle. It is produced by NANDing CLK-16 and CLK-51R, together with the W microcode word bit and the ABORT' signal, using the 74S20 NAND gate U130b. This signal is therefore an active low pulse during the second half of the microcycle, provided that the W bit is set (that is, the result should be written back to the CPU register) and that the cycle is not aborted.

4.1.2.3 WR OP'

The 74S04 NOT gate inverts CLK-36 to produce CLK-41. This signal is then NANDed with CLK-4J and LD OP by the 74S10 3-input NAND gate U207a, whose output is connected to the write-enable inputs on the Opcode file memories, U181, U160, U159 and U200. Therefore, when the LD OP signal is high (that is, when the Opcode file is being loaded from memory), the WR OP' signal goes low for a period in the second half of the microcycle, while the memory data output is valid, and writes the data into the Opcode file.

4.1.2.4 LATCH R

The ALU R register is controlled by the 74S02 OR gate U95c. This gate combines the CLK-6 and CLK-4J signals to produce a signal which goes high shortly after the start of the microcycle, and goes low again before the start of the second half of the same cycle. Therefore, the R register is updated during the first half of the microcycle, but the data is latched throughout the entire second half.

4.1.2.5 Stack Write

The 2 clock signals, CLK-4J and CLK-16 are logically NANDed with the STK WE signal and the ABORT' signal by the 74S20 gate U130a. This gate provides an active-low write-enable signal to the stack RAM chips during the period when LATCH R is low, provided that the STK WE signal is high (that is, the microcode instruction specifies a write to stack operation), and that the cycle is not aborted.

4.1.2.6 DON'T IO ENB

The Input/Output gate U216 is disabled by the 74S00 NAND gate U239a while the microcode pipeline registers are being updated. U239a combines the CLK-7R and CLK-26 clocks to produce

a signal which goes low at the start of each microcycle.

4.1.2.7 SLIVER CLK

The half-width pipeline registers (U116, U136, U158 and U177) in the RasterOp circuit store the previous source data word, so that non-aligned graphics operations are possible. These registers are updated at the end of each graphics operation microcycle by a signal called the Sliver Clock. This signal is produced by the 74S10 NAND gate at location U211c. This gate combines the CLK-4J and CLK-46 signals, together with the CLK FIFO OUT signal which enables the sliver clock on graphics cycles. The resulting signal goes low for a short period at the end of each microcycle.

4.1.2.8 The WCS Write-Enable Clock

The control signal to write to the control store is produced by the enable gating of the 74S138 decoder U92. This decoder is enabled by a combination of 3 signals, namely CLK-4E (enabled when high), CLK20R (enabled when low) and WR NOW' (again enabled when low). This last signal is active during the first of the 2 microcycles needed to update the WCS, while the 2 clock signals enable the decoder during the second half of the microcycle, when the WCS data inputs are stable

4.1.3 Overview of an Arithmetic Cycle

The most common type of microcycle is probably an ALU operation on 2 registers, with the result being written back to the destination register. The clock signals for such a cycle operate in the following way.

At the end of the previous cycle, the CLK-4J signal goes low, and forces the LD MIR' signals high, thus clocking the microcode word pipeline registers and the microcode sequencer. The microcode instruction is loaded into the pipeline register.

Since, during the first half of the microcycle, CLK-4F is low, the Y-register address multiplexer U76 and U99 feeds the Y microcode field to the Y-register RAM address inputs. The outputs of the microcode word pipeline register control the ALU and the registers as was described in the last 2 chapters. Since the LATCH R clock is now high, the R register contains the result of the ALU operation.

The CLK-4F signal now goes high and therefore the Y-register address multiplexer applies the X-register (destination) address to the Y-register RAM address inputs. Therefore, the destination register in both the X and Y register banks will be updated. The WR RAM' signal goes low and writes the ALU result (now stored in the R register) to the destination registers.

Throughout the microcycle, the J-inputs to the microcode address sequencer have been stable, and therefore the sequencer has addressed the next instruction in the control store. Therefore when the CLK-4J signal goes low at the end of the microcycle, the next instruction is loaded into the pipeline register.

4.2 CPU Reset

At switch-on, C160 (33 μF) begins to charge via R7 (47 Ω) and R6 (33k). When the voltage across C160 reaches the threshold of the B input of U238a (a 74221 Monostable), U238a triggers since its A input is grounded and its reset input is pulled high via R3 (1k). U238a produces the reset pulses,INIT and INIT', for the CPU logic. The Q output of U238a is buffered and inverted by U254f (a 74S240 inverting buffer, which is permanently enabled), and then fed to the INIT L signal on the backplane via the edge connector pin J72.

The system can be manually reset by pressing the BOOT switch on the front panel. This grounds pin J197 of the CPU card and therefore discharges C160 via R7. When the BOOT switch is released, a reset operation as described above occurs

4.3 The Control Store

4.3.1 The Writable Control Store RAM

The PERQ 16K CPU uses a 16K * 48 bit writable control store containing 48 16K * 1 bit 2167 RAM chips. These RAMs are loaded from the UW register at the output of the ALU, and supply data to microcode pipeline register. They are addressed by one of 2 buffered address buses produced by the microcode address sequencer described in chapter 5. These buses provide identical signals, and the only reason that 2 are used is to provide sufficient fan-out to drive the RAM address inputs. These address lines are active-low. The Control store is written to 16 bits at a time, and the 3 write enable signals for this operation are produced by the Special Function decoder described below.

The Writable control store RAMs are at the following locations on the CPU board. This table also lists the RAM control and address signals, the functions of which are described below.

Bit	RAM	Data In	Write En	CS	Address
0	U59	UW<8>	U WR B’	ENB RAM’ B	UA<n>B’
1	U33	UW<9>	U WR B’	ENB RAM’ B	UA<n>B’
2	U27	UW<10>	U WR B’	ENB RAM’ B	UA<n>B’
3	U52	UW<11>	U WR B’	ENB RAM’ B	UA<n>B’
4	U34	UW<12>	U WR B’	ENB RAM’ B	UA<n>B’
5	U5	UW<13>	U WR B’	ENB RAM’ B	UA<n>B’
6	U26	UW<14>	U WR B’	ENB RAM’ B	UA<n>B’
7	U36	UW<15>	U WR B’	ENB RAM’ B	UA<n>B’
8	U35	UW<8>	U WR A’	ENB RAM’ A	UA<n>B’
9	U6	UW<9>	U WR A’	ENB RAM’ A	UA<n>B’
10	U29	UW<10>	U WR A’	ENB RAM’ A	UA<n>B’
11	U28	UW<11>	U WR A’	ENB RAM’ A	UA<n>B’
12	U32	UW<12>	U WR A’	ENB RAM’ A	UA<n>B’
13	U4	UW<13>	U WR A’	ENB RAM’ A	UA<n>B’
14	U1	UW<14>	U WR A’	ENB RAM’ A	UA<n>B’
15	U9	UW<15>	U WR A’	ENB RAM’ A	UA<n>B’
16	U44	UW<0>	U WR B’	ENB RAM’ B	UA<n>A’
17	U22	UW<1>	U WR B’	ENB RAM’ B	UA<n>A’
18	U18	UW<2>	U WR B’	ENB RAM’ B	UA<n>A’
19	U16	UW<3>	U WR B’	ENB RAM’ B	UA<n>A’
20	U21	UW<4>	U WR B’	ENB RAM’ B	UA<n>A’
21	U24	UW<5>	U WR B’	ENB RAM’ B	UA<n>A’
22	U15	UW<2>	U WR A’	ENB RAM’ A	UA<n>A’
23	U42	UW<4>	U WR A’	ENB RAM’ A	UA<n>A’
24	U49	UW<0>	U WR A’	ENB RAM’ A	UA<n>A’
25	U48	UW<1>	U WR A’	ENB RAM’ A	UA<n>A’
26	U20	UW<6>	U WR B’	ENB RAM’ B	UA<n>A’
27	U11	UW<3>	U WR A’	ENB RAM’ A	UA<n>A’
28	U46	UW<7>	U WR B’	ENB RAM’ B	UA<n>A’
29	U50	UW<5>	U WR A’	ENB RAM’ A	UA<n>A’
30	U19	UW<6>	U WR A’	ENB RAM’ A	UA<n>A’
31	U39	UW<7>	U WR A’	ENB RAM’ A	UA<n>A’

32	U48	UW<0>	U WR C'	ENB RAM' C	UA<n>A'
33	U47	UW<1>	U WR C'	ENB RAM' C	UA<n>A'
34	U13	UW<2>	U WR C'	ENB RAM' C	UA<n>A'
35	U37	UW<3>	U WR C'	ENB RAM' C	UA<n>A'
36	U12	UW<4>	U WR C'	ENB RAM' C	UA<n>A'
37	U25	UW<5>	U WR C'	ENB RAM' C	UA<n>A'
38	U45	UW<6>	U WR C'	ENB RAM' C	UA<n>A'
39	U38	UW<7>	U WR C'	ENB RAM' C	UA<n>A'
40	U31	UW<8>	U WR C'	ENB RAM' C	UA<n>B'
41	U3	UW<9>	U WR C'	ENB RAM' C	UA<n>B'
42	U54	UW<10>	U WR C'	ENB RAM' C	UA<n>B'
43	U53	UW<11>	U WR C'	ENB RAM' C	UA<n>B'
44	U57	UW<12>	U WR C'	ENB RAM' C	UA<n>B'
45	U30	UW<13>	U WR C'	ENB RAM' C	UA<n>B'
46	U2	UW<14>	U WR C'	ENB RAM' C	UA<n>B'
47	U58	UW<15>	U WR C'	ENB RAM' C	UA<n>B'

Since the bits are somewhat scrambled, the following table may help with working out the values to load into the control store.

ALU Bit	Load Low	Load Mid	Load High
0	24(ALU2)	16(SF0)	32(Y0)
1	25(ALU3)	17(SF1)	33(Y1)
2	22(ALU0)	18(SF2)	34(Y2)
3	27(W)	19(SF3)	35(Y3)
4	23(ALU1)	20(F0)	36(Y4)
5	29(A0)	21(F1)	37(Y5)
6	30(A1)	26(H)	38(Y6)
7	31(A2)	28(B)	39(Y7)
8	8(Z0)	0(J0)	40(X0)
9	9(Z1)	1(J1)	41(X1)
10	10(Z2)	2(J2)	42(X2)
11	11(Z3)	3(J3)	43(X3)
12	12(Z4)	4(Cond0)	44(X4)
13	13(Z5)	5(Cond1)	45(X5)
14	14(Z6)	6(Cond2)	46(X6)
15	15(Z7)	7(Cond3)	47(X7)

4.3.2 The Microcode Pipeline Register

The 48 bit output of the control store is clocked into the microcode pipeline register by the rising edge of the LD MIR' signal. This register consists of 48 D-type flipflops at the following locations of the CPU board :

Bit	Signal	Latch	Type
0	J0	U69d	74S175
1	J1	U69c	74S175
2	J2	U69b	74S175
3	J3	U69a	74S175

4	Cond0	U41d	74S374
5	Cond1	U41c	74S374
6	Cond2	U41b	74S374
7	Cond3	U41a	74S374
8	Z0	U41h	74S374
9	Z1	U41g	74S374
10	Z2	U41f	74S374
11	Z3	U41e	74S374
12	Z4	U43d	74S374
13	Z5	U43c	74S374
14	Z6	U43b	74S374
15	Z7	U43a	74S374
16	SF0	U43h	74S374
17	SF1	U43g	74S374
18	SF2	U43f	74S374
19	SF3	U43e	74S374
20	F0	U110d	74S175
21	F1	U110c	74S175
22	ALU0	U65g	74S374
23	ALU1	U65h	74S374
24	ALU2	U65b	74S374
25	ALU3	U65a	74S374
26	H	U110b	74S175
27	W	U65c	74S374
28	B	U110a	74S175
29	A0	U65d	74S374
30	A1	U65e	74S374
31	A2	U65f	74S374
32	Y0	U60g	74F374
33	Y1	U60e	74F374
34	Y2	U60h	74F374
35	Y3	U60f	74F374
36	Y4	U60c	74F374
37	Y5	U60a	74F374
38	Y6	U60d	74F374
39	Y7	U60b	74F374

40	X0	U55f	74F374
41	X1	U55h	74F374
42	X2	U55e	74F374
43	X3	U55g	74F374
44	X4	U55b	74F374
45	X5	U55d	74F374
46	X6	U55a	74F374
47	X7	U55c	74F374

The output-enable pins of the 74S374s and 74F374s are grounded so that the outputs of these latches are permanently enabled. Similarly, the reset input of U110 is pulled high via R3 (a 1k resistor), so that this latch is never reset. However, the reset input of U69, which provides the J field inputs is driven by the INIT' signal, so that this latch is cleared when the processor is reset. This forces the J inputs on the microcode sequencer to logical 0, so that the first operation performed by the sequencer is a Jump0. Therefore the microcode starts executing at location 0 after a CPU reset.

Both the normal and inverting outputs of U110 are used. This latch stores the F, H and B microcode fields, and the availability of the inverted versions of these signals simplifies the microcode decoding circuitry. The inverted signals have a ' suffixed to their names.

The outputs of the microcode pipeline register are fed to the microcode decode circuit that is described below.

4.3.3 The UW register

The writable control store is loaded from the ALU via the UW register, which consists of a pair of 74S374 latches at locations U71 and U56. These latches are clocked by the rising edge of the LD MIR' B signal at the start of each microcycle, and store the contents of the R-register, and hence the ALU result, of the previous microcycle. U71 stores the low 8 bits, while U56 stores the high 8 bits. The sequence of operations necessary to load 16 bits of the writable control store will be described next.

4.3.4 Loading the Writable Control Store

The only CPU subsystem that can address the control store is the microcode sequencer described in chapter 5. In order to load an instruction into the control store, it is therefore necessary to cause the sequencer to output the address of the location to be loaded, and then to continue with the next instruction of the original microprogram. This occurs in the following way.

When an instruction to load the control store is executed (that is to say an instruction with the F field containing 0 or 2, and the SF field 14, 15 or 16), the special function decoder asserts the WRCS (WRite Control Store) signal, which is active high. Since the DON'T WRCS' signal is also high at this point, the 74S00 NAND gate, U212d, which combines these signals, asserts the active low WR NOW' signal. This signal disables the LD MIR' gates U215b and U213a (both 74S10 3-input NANDs) described above and prevents the LD MIR' signal from going low during the second half of this microcycle.

An instruction that loads the writable control store must have the Condition field set to 0 (normally called TRUE), and the Jump (J) field set to 7 (which is known as GOTO S). When the condition field is 0, the lower condition multiplexer chip, U189 (a 74S251) feeds the WR NOW' signal to the CONDITION' input on the microcode sequencer. The condition logic is fully described in chapter 5. Since the WR NOW' signal is low during such a microcycle, the CONDITION' signal is high (the multiplexer chip inverts the signal), and the microcode sequencer interprets this as a false condition.

Therefore, since the J inputs to the microcode sequencer have the states 0111, the microcode sequencer outputs the contents of its internal S register onto the control store address lines. This S register has been previously loaded with the control store address that is to be written to. The required location in the writable control store is now being addressed.

The special function decoder asserts the WRCS<0> and WRCS<1> signals according to whether the low, middle, or high 16 bit word of the control store is being addressed. These signals are decoded by U92, a 74S138 3 to 8 line decoder. Since the WRCS signal is high, and the WR NOW' signal low, during the Wr WCS time in the second half of the microcycle, as defined by the CPU clock timing diagram above the decoder is enabled and asserts one of the U WR A', U WR B' or U WR C' signals. These signals are connected to the write-enable inputs of the control store RAMS, and therefore the contents of the UW register are written to the selected 16 bit word of the control store. The signal asserted is given by the following table:

Word	WRCS<1>	WRCS<0>	Decoder input	Signal Asserted
Low	0	0	100	U WR A'
Mid	0	1	101	U WR B'
High	1	0	110	U WR C'

Since the LD MIR' signals did not go low during the second half of that microcycle, the microcode word pipeline register cannot be clocked by the rising edge at the start of the next clock cycle, and therefore no new microcode word is loaded. The clock input of microcode sequencer is also driven

by the LD MIR' A signal, and therefore, the sequencer is not clocked at this time. So, the microcode program counter inside the sequencer is not updated.

However, U209b, a 74S374 D-type flip-flop is clocked by the rising edge of CLK-0R B at the start of the next microcycle. The D input of this flip-flop is driven by the WR NOW' signal, and the flip-flop is therefore set to 0, thus making the DON'T WRCS' signal also low (as it is driven by the output of U209b) and thus U212d makes the WR NOW' signal high again.

During this cycle, U92 is inhibited (since WR NOW' is high), and the CONDITION' input to the microcode sequencer is low (driven via the condition multiplexer U189), and so the sequencer outputs the contents of the microcode program counter at the end of the previous instruction. This is, of course, the address of the next microinstruction to execute.

During the second half of this microcycle, the LD MIR' signal goes low in the usual way (since the WR NOW' signal is high), and at the end of the microcycle, the next microinstruction to execute is loaded into the microcode word pipeline register in the usual way. Also, the microcode program counter is incremented by the sequencer, just like at the end of any other instruction, and the next instruction is executed.

4.3.5 The Microcode Boot ROM

When the PERQ is switched on, the writable control store contains random data, and therefore the CPU would not have a valid microprogram to execute. This problem is avoided by the 512-word microcode boot rom. This ROM consists of 6 27S29 512 byte PROMs at the following locations on the CPU board

Name	Location	Addressed by
TCA01	U10	UA<n>B'
TCB00	U23	UA<n>A'
TCC00	U17	UA<n>A'
TCD00	U14	UA<n>A'
TCE00	U8	UA<n>B'
TCF00	U7	UA<n>B'

As the above table shows, the ROMs are addressed by one of the 2 buffered address outputs from the microcode sequencer. However, the address lines are scrambled between the sequencer and the ROMs, and the following table indicates which address line goes where:

Sequencer Address Bit	ROM Address Bit
0	1
1	7
2	4
3	2
4	6
5	0
6	5
7	8
8	3

The data lines are also scrambled between the ROMs and the microcode word pipeline register, and the next table indicates how those signals are connected.

Data Bit	TCA01 (U10)	TCB00 (U23)	TCC00 (U17)	TCD00 (U14)	TCE00 (U8)	TCF00 (U7)
0	U<6>	U<12>	U<20>	U<25>	U<37>	U<46>
1	U<7>	U<13>	U<17>	U<24>	U<39>	U<44>
2	U<5>	U<15>	U<16>	U<27>	U<36>	U<47>
3	U<4>	U<14>	U<19>	U<29>	U<38>	U<45>
4	U<3>	U<11>	U<18>	U<23>	U<32>	U<43>
5	U<2>	U<10>	U<21>	U<22>	U<34>	U<41>
6	U<1>	U<9>	U<26>	U<31>	U<33>	U<42>
7	U<0>	U<8>	U<28>	U<30>	U<35>	U<40>

The enable input of all 6 boot ROM chips are connected together and driven by the E BOOT ROM' signal which is derived from the ROM enable latch that is described in the next section.

4.3.6 The ROM Enable Latch

The ROM Enable latch is the 74S74 D-type flip-flop U236b. The set and D inputs of this flip-flop are pulled high by R3 (a 1k resistor), so this flip-flop is set by a rising edge on the clock input and cleared by a low level on the reset input. These signals are produced by the following circuitry.

The main CPU reset signal, INIT, is NANDed with the CLK-0R C clock by the 74S00 NAND gate U237d. The output of this gate drives the clock input of the ROM Enable latch. Therefore, during the CPU reset, when INIT is high, a falling edge of the CLK-0R clock will set the ROM Enable latch. The reset input of the ROM Enable latch is driven by the CONT' signal from the opcode file input control circuit. When the first Reload Op instruction after a reset is executed, the CONT' signal goes low, and the ROM Enable latch is cleared.

The output of the ROM Enable latch, the so-called BOOT signal, is NANDed with UA<11>A' by U131c (a 74S00 NAND gate). The output of this gate is low to enable the ROM, and this occurs when the ROM Enable latch is set and the UA<11>A' signal is also high, that is to say when the microcode address line 11 is a logical 0. Therefore the boot ROM overlays the lower half of each 4K page of the writable control store.

The output of U131c is connected to one input of each of the 3 RAM enable gates U66b, U66c and U66d (all 74S00 NAND gates). The outputs of these gates are known as the ENB RAM' A, ENB RAM' C and ENB RAM' B respectively, and are linked to the chip enable inputs of the writable control store RAMs. Therefore, when the ROM is enabled, the outputs of these gates go high and disable the control store RAM.

The output of U131c is inverted by the 74S240 NOT gate U254g, and then fed to one input of the 74S00 NAND gate U66a. The output of this gate is the E BOOT ROM' signal that drives the chip select inputs of the 6 microcode boot ROMs. Therefore, when the output of U131c goes low, the E BOOT ROM' signal goes low also, and enables the ROMs.

The remaining inputs of the 4 sections of U66 are all linked together and pulled high by the 1k resistor R2. Therefore, the NAND gates in U66 operate as inverters, as the above description indicates. However, these inputs can be forced to a logic 0 state by grounding the DIS USTORE L pin on the CPU diagnostic connector, which therefore disables both the writable control store RAMs and the boot ROM. This allows the CPU to be controlled by an external control store.

4.3.7 The CPU diagnostic connector

The 3 edge connectors on the front edge of the CPU board provide external access to the control store address and data lines. A logic analyser plugged into these connectors may record the microcode program flow and the instructions that were executed. Alternatively, by grounding the DIS USTORE L pin, the internal CPU control store may be disabled, and an external control store, connected to the diagnostic connectors, may provide the microcode program.

The signals present on the diagnostic connectors are given in the following tables. The pin numbering is, however, somewhat unconventional, with pin 1 being the top pin on the component side, and the pins are then numbered consecutively going down the component side, followed by the pins on the track side, with the numbers again increasing downwards.

JA (Top Connector) – Microcode Data 1

1	U<0>
3	U<1>
5	U<2>
7	U<3>
9	U<4>
11	U<5>
13	U<6>
15	U<7>
17	U<8>
19	U<9>
21	U<10>
23	U<11>
25	U<12>
27	U<13>
29	U<14>
31	U<15>
33	U<16>
35	U<17>
37	U<18>
39	U<19>
41	U<20>
43	U<21>
45	U<22>
47	U<23>
49	U<24>

JB (Middle Connector) – Microcode Data 2

2	U<25>
4	U<26>
6	U<27>
8	U<28>
10	U<29>
12	U<30>
14	U<31>
16	U<32>
18	U<33>
20	U<34>
22	U<35>
24	U<36>
26	U<37>
28	U<38>
30	U<39>
32	U<40>
34	U<41>
36	U<42>
38	U<43>
40	U<44>
42	U<45>
44	U<46>
46	U<47>
47	DIS USTORE L

JC (Bottom Connector)	Microcode Address
1	UUA<0>
3	UUA<1>
5	UUA<2>
7	UUA<3>
9	UUA<4>
11	UUA<5>
13	UUA<6>
15	UUA<7>
17	UUA<8>
19	UUA<9>
21	UUA<10>
23	UUA<11>
25	LD MIR' A

4.4 The microcode word decoder

The microcode word is divided into 12 fields, as described in chapter 2. The way each field is used by the PERQ CPU is described in this section.

4.4.1 The X field

The X field specifies the destination register for the ALU operation. It is fed to the X-register addressing gates U74, U75, U94 and U96 only, as described in chapter 2.

4.4.2 The Y field

The Y field is used to specify the source register in the ALU operation. It is therefore fed to the Y register address gates U93, U98, U119 and U120 which are fully described in chapter 2. The Y field is also used to provide the lower 8 bits of a constant operand, and for that reason it is fed onto the BMUX lines via the 74S240 3-state buffer U101. This system is also described in chapter 2. Finally, the Y field provides the top 6 bits of a microcode leap address. It is gated onto the JMUX lines by the 74S240 3-state buffer U122. The JMUX and associated control logic is described in chapter 5

4.4.3 The A field

The A field selects the AMUX input. It is fed to the 2 AMUX control chips (the 74S138 decoder U229 and the AMX16 PROM (U188)). This logic is described in chapter 2.

4.4.4 The B field

The microcode pipeline latch for the single bit B field provides both the normal and inverted outputs. These signals are used to control the BMUX, as described in chapter 2.

4.4.5 The W field

The single bit W field is used to enable the writing back of the ALU output to the CPU register RAM. The output of the appropriate microcode pipeline latch is connected to one input of U130b (a 74S20 NAND gate), which provides the active-low WR RAM' signal. Therefore, when this bit is low, the WR RAM' signal is inhibited by being forced high.

4.4.6 The H field

Again, both the normal and inverted outputs of the H field microcode pipeline latch are available. They are used as follows.

The I/O MEM RQST signal from the EIO board is latched at the start of each microcycle by the 74S273 latch U227a. This latch is clocked by the rising edge of the CLK-0R D clock, and cleared by the CPU reset signal INIT'. The output of this latch, MEM RQST L, indicates when the EIO card wishes to perform a DMA transfer to the main system RAM. This signal is NANDed with the inverted H field (the H' signal) by the 74S00 NAND gate U237a. The output of this gate is the DMA request signal to the memory control state machine described in chapter 7. Therefore, when the H field contains 1, the H' signal is low, and the DMA request signal is disabled.

The H field is also used to distinguish between certain microcode jumps which have the same value in the J field, for example the 'Vector' or 'Dispatch' instructions. For this reason, the H signal is connected to one input of the 12L6 PAL U108 (JPPAL3) which controls the JMUX circuit, and one address input of the extended microcode sequencer control PROM U67 (USQ01). The operation of the microcode sequencer and the JMUX are described in chapter 5

Finally, on the 24 bit T4 CPU, the H field is used to select between the 2 microstate registers. It is therefore connected to an address input of the AMUX24 PROM that controls the AMUX in that version of the CPU. This extended AMUX is described in chapter 2.

4.4.7 The ALU field

The 4 bit ALU field, which selects the ALU function, is simply fed to the ALU control PROM U105 (ALD16.1). The operation of the ALU control logic is described in chapter 3.

4.4.8 The F field

The microcode pipeline latch that stores the F field provides both the normal and inverted outputs. These signals are used to control the following CPU systems.

The active high signals F<0> and F<1> are fed to the AMUX control PROM U188 (AMX16). When the F field contain 2, this PROM asserts the SH TYPE signal thus enabling the Z microcode field to control the shifter. The shifter and its associated control logic is described in chapter 6

The active high signals are also connected to address inputs on the special function decoder PROMs U150 (SFA16.3) and U193 (SFB16.3), since the use of the SF field depends on the value of the F field. The special function decoder is described below.

The signals F<0> and F<1>' are combined with the SF3 signal by the 74S10 NAND gate U215a. The output of this gate therefore goes low when the F field contains 1 and the SF field contains 1x, that is when the microcode instruction specifies a memory cycle. The output of U215a is connected to the memory control state machine described in chapter 7.

The same pair of signals, F<0> and F<1>' are combined by the 74S08 AND gate U231d. The output of this gate, the MEM FUNC signal, is NAND with the CPU clock signal CLK-6 by the 74S00 NAND gate U232a. This gate was originally used to clock the memory address register, but this function is now performed by the LOAD MA signal from the special function decoder.

When the F field contains 3, the microcode instruction contains a long jump. This is detected by the 74S00 NAND gate U131d, which NANDs together the F<0> and F<1> signals. The active low output of U131d, the LONG JUMP' signal enables the 74S240 3-state buffer U151e-h, which then gates the SF field onto the JMUX<8> to JMUX<11> lines. The operation of the JMUX is described in chapter 5

Finally, an Input/output instruction is defined by the F field containing 0 and the SF field containing 17. This condition is detected by the 74S30 NAND gate U216, which combines the F<0>' and F<1>' signals, the SF<0>, SF<1>, SF<2> and SF<3> signals, and the DON'T IO ENB' signal from the CPU clock circuit. When such a microcode instruction is executed, the output of U216 goes low, thus asserting the IO ENB L signal on the system backplane.

4.4.9 The SF field

The 4 bit SF field is used by several parts of the PERQ CPU logic.

Firstly, it is fed to the special function decoder PROMS U150 (SFA16.3) and U193 (SFB16.3). These PROMs decode the SF field, along with the F field, and assert control signals to various parts of the processor, depending on which special function is being executed. The special function decoder is described below.

The SF field also enables the IO ENB L signal during Input/Output instructions via the 74S30 NAND gate U216, as described in the last section.

Since the JMUX control may also depend on the state of the SF field, (for example, the extended special function defined by F=1 and SF=6 gates the output of the shifter onto the JMUX lines) the 4 SF lines are connected to the JMUX control PAL U108 (JPPAL3). The operation of the JMUX is described in chapter 5.

The SF<3> signal enables U215a which controls the memory control state machine. This gate was described in the last section. The lower 3 SF signals are fed to the control inputs of this state machine, and determine which of the 8 types of memory cycles is to be performed. The memory control state machine is described in chapter 7.

The SF field also provided the top 4 bits of a 12 bit microcode jump address. U151e-h (a 74S240 3-state buffer) gate the SF field onto the JMUX lines when the LONG JUMP' signal is active. The JMUX is completely described in chapter 5

4.4.10 The Z field

The Z field of the microcode word provides an 8-bit data value that is used by several sections of the CPU logic.

The Z field is gated onto the lowest 8 BMUX lines by the 74S240 3-state buffers U123a-d and U124a-d when the B field specifies a constant and it therefore provides the least significant byte of this constant. This circuit is described in chapter 2.

The Z field is also used to specify the shifter operation when the F field contains 2. The Z field is stored in the 67S380 latch U127, which is enabled by the LD SHIFT CMD signal from the special function decoder. The outputs of this latch, which are enabled by the active low output of the SH TYPE latch U147a (a 74S74 D-type flip-flop) then provide the SHIFT CMD signal to the shifter control PROMs. The Shifter and its associated control logic is described in chapter 6.

Since the Z field also provides the low 8 bits of an absolute microcode jump address, it is gated onto the lower 8 JMUX lines via U124e-h and U103a-d. Similarly, certain of the Z field bits are used to provide the constant microcode address bits on a NEXT OP microcode jump. These Z-field bits are gated onto the appropriate JMUX lines via U144e-h. During an interrupt vector jump, the remaining constant bits are provided by the remaining Z-field bits, which are gated onto the JMUX lines via U146a-d. All the buffers used for these purposes are 74S240s. The JMUX and its control electronics are described in chapter 5.

The Z field is buffered by the 74S240 buffer U225, which is permanently enabled, and then fed to the IOA Input/Output address lines on the backplane. This circuit is described in chapter 3. The most significant Z field bit, Z<7> distinguishes an input port address from an output port address. It is therefore connected to U217a, a 74S10 NAND gate, which is used to enable the I/O data output buffers onto the I/O data bus on the backplane. Therefore, these buffers are only enabled when Z<7> is high.

Finally, the lower 7 Z field bits are connected to the inputs of the Raster Operation control register U190 (a 74S273 octal D-type flip-flop). The Z field is loaded into this register on the rising edge of the LD C signal from the special function decoder. The register is cleared by the CPU reset signal INIT' described above. The outputs of this register are used to control the Raster Operation state machine that is described in chapter 7.

4.4.11 The CND field

The 4-bit CND field is used to select the condition that will be used to control the microcode sequencer. The lower 3 bits of this field are connected to the 3 select inputs of a pair of 74S251 8-input multiplexers, U187 and U189. The CND<3> bit is connected the enable input of U189, and it is inverted by the 74S240 inverting buffer U185d, before being connected to the enable input of U187. The 3-state outputs of the 2 multiplexers are linked together and provide the CONDITION' signal. Therefore, U189 selects between the first 8 conditions, and U187 the second 8 conditions. The conditional branch logic is described in chapter 5

4.4.12 The JMP field

The microcode sequencer function is controlled by the 4-bit JMP field, which is therefore connected directly to the 4 control inputs (I0-I3) on the 2910 sequencer U112. It is also connected to the JMUX control PAL U108 (JPPAL3), and the JMUX control PROM JPE16 (at location U104) since which JMUX input should be selected depends on the jump instruction being executed. Finally, the JMP field is connected to 4 of the address inputs on the extended microcode sequencer control PROM U67 (USQ01), so that this sequencer executes the correct jump instruction.

4.5 The Special Function decoder

The special function decoder consists of a pair of 512*8 bit 27S29 PROMS at locations U150 (SFA16.3) and U193 (SFB16.3). Corresponding address lines of the 2 proms are connected together, and driven by the microcode word F and SF fields as indicated by the following table.

Bit	Signal
A0	SF<3>
A1	SF<2>
A2	SF<1>
A3	SF<0>
A4	F<1>
A5	Ground
A6	Ground
A7	Ground
A8	F<0>

Twelve of the 16 data outputs of these PROMs are used to directly provide CPU control signals. These signals are defined as follows.

Bit	Signal	Active when	Comments
SFA16.3			
D0	WRCS	F=0|2, SF=14|15|16	Load Control Store Instructions
D1	Raw Ld Shift Cmd	F=0,SF=1	Shift On R
		F=2	Shift on Z
D2	BPC:=	F=0|2, SF=13	Load BPC
D7	LD MDR	F=1, SF=2	Load Multiplier Register

SFB16.3

D0	STK RESET'	F=0|2, SF=2	Reset Stack pointer
D1	STK WE	F=0|2, SF=3	TOS:=(R)
		F=0—2, SF=4	Push
		F=1, SF=5	Push Long Constant
D2	POP'	NOT(F=0|2, SF=4)	Push
		NOT(F=1, SF=5)	Push Long Constant
D3	STK CLK ENB H	F=0|2,SF=4	Push
		F=0—2,SF=5	Pop
		F=1,SF=5	Push Long Constant
D4	Read Victim	F=1, SF=0	(R):=Victim
D5	WRCS<0>	NOT(F=0|2, SF=14|16)	WCS mid instruction
D6	WRCS<1>	NOT(F=0|2, SF=14|15)	WCS high instruction
D7	ENB MDR	F=1,SF=4	R=multiplier register

The remaining 4 outputs of the SFA16.3 PROM are used to control the 74S138 decoder U149. The 3 ROM outputs D3, D4, D5 are fed to the select inputs of this decoder, while the D6 output is connected to an active-low enable input. The decoder is also enabled by a high level of the CPU clock signal CLK-4G. This clock is high during the second half of a microcycle, and therefore a selected decoder output goes low at this time. The devices that are controlled by this decoder are clocked by the rising edge of the decoder output, which occurs at the end of the microcycle, just after the CLK-4G clock has fallen. This decoder provides the following signals :

Output	Signal	Active when	Comments
0	LD C	F=0|2, SF=6	Ctrl RasterOp=(Z)
1	LD S	F=0|2, SF=7	Src RasterOp=(R)
2	LD D	F=0|2, SF=10	Dst RasterOp=(R)
3	LD W	F=0|2, SF=11	Width RasterOp=(R)
4	RELO OP	F=0|2, SF=12	Load Op
5	MUL STEP	F=1, SF=1	Mul Step
		F=1,SF=2	Load Mult/Div
6	LD INDEX	F=1, SF=3	Load Index Register
7	LOAD MA	F=1, SF¿=10	Memory Functions

The circuits controlled by each of these signals will now be described.

4.5.1 WRCS

This signal enables the Control store write enable decoder U92, as described above

4.5.2 Ld Shift Cmd

The appropriate output from the special function decoder ROM is NANDed with the CLK-51R signal by the 74S00 NAND gate U131b, and then NORed with the CLK-4E clock signal by the 74S02 NOR gate U95b. The output of U95b, the LD SHIFT CMD signal is used to enable the 2 shifter command latches U127 and U128 and also to clock the SH TYPE flip-flop U147a. The operation of the shifter is described in chapter 6

4.5.3 BPC:=

This signal is connected to the active-low Load input of the byte program counter U201 (a 74S163 counter). When a BPC:=(R) instruction is executed, this signal goes low, and loads the bottom 4 bits of the shifter input latch U178 (a 67S380) into the byte program counter. This circuit is described in chapter 5

4.5.4 LD MDR

This signal is connected to the multiplier register control PROM U106 (MULT01), and it therefore causes the Multiplier register to be loaded as described in chapter 3.

4.5.5 Stk Reset'

This signal drives the active low Load input of the Stack pointer (U233 - a 74S169 counter). Since the parallel load data inputs of this counter are grounded, this signal clears the stack pointer to 0, as described in chapter 2. It also increments the DDS display as described below.

4.5.6 Stk WE

This signal enables the 74S20 NAND gate U139a on a Push or TOS:=(R) instruction. U139a provides the write enable signal to the stack RAMs as described in chapter 2.

4.5.7 POP'

This signal is connected to the direction input of the 74S169 stack pointer U233. It is normally low (causing the counter to count down, and thus pop the stack), but goes high (to cause the counter to increment and push the stack) on any push instruction

4.5.8 STK CLK ENB H

This signal is NANDed with the ABORT' signal by the 74S00 NAND gate U237c, and is then connected to the active-low clock enable input of the stack pointer U233. Therefore the stack pointer will change on any push or pop instruction.

4.5.9 Rd Victim

When this signal goes low, the 74S10 NAND gate U213b is inhibited, and therefore the outputs of the ALU R register described in chapter 3 are disabled. This signal also enables the outputs of the victim latch U79 and U88 (both 74S534 octal D-type flip-flops) and gate the outputs of this latch onto the R lines in place of the R register. The signal is also connected to one input of the 74S08 AND gate U73Ac, which clears the hold victim latch as described in chapter 5.

4.5.10 WRCS<0> & WRCS<1>

These signals are connected to the select inputs of the control store write enable decoder U92. They therefore select which 16 bit section of the 48 bit control store word is to be written to, as described above.

4.5.11 ENB MDR

This signal inhibits the 74S10 NAND gate U213b, and hence disables the outputs of the R register. It also enables the 3-state outputs of the multiplier register U89 and U91 described in chapter 3, and thus the multiplier register passes its contents onto the R lines.

4.5.12 LD C, LD S, LD D & LD W

These 4 signals are used to clock one of the 4 74S273 Raster Operation Control registers. These registers, which are described in chapter 7, store the control parameters for the raster operation state machine.

4.5.13 RELO OP

This signal is connected to one input of the 74S00 NAND gate U212c. It sets the LD OP control latch, and causes the opcode file to be loaded with the contents of the memory data bus on the next 4 microcycles. The operation of the opcode file is described in chapter 5

4.5.14 MUL STEP

This signal clocks the multiplier register U89 and U91 described in chapter 3 on either a Load Multiplier or a Multiply Step instruction. It also clocks the CCSR0 PAL (U145) described in chapter 2

4.5.15 LD INDEX

This signal clocks the index register U97 (a 74S273 octal D-type flip-flop) and thus loads it with the lower 8 bits of the ALU result. The index register is cleared by the CPU reset signal INIT'. The outputs of the index register are fed to the main register address gates as described in chapter 2

4.5.16 LOAD MA

This signal is inverted by the 74S04 inverter U234e, and then used to clock the memory address registers U255, U257, and U256 at the start of a memory instruction, thus loading them with the ALU result. The memory address register is described in chapter 3.

4.6 The DDS

The digital diagnostic system display that is mounted on the front of the PERQ's cabinet is controlled by a simple 3-digit counter circuit mounted behind the display. The counter is cleared by pressing the system reset switch, and may be incremented under program control in the following way.

The STK RESET' output from the special function decoder is latched by the 74S74 D-type flip-flop U236a. The set and reset inputs of this flip-flop are tied high, and it is clocked by the rising edge of the CLK-0R C clock at the start of each microcycle. The output of this flipflop is buffered by the 74S240 inverting 3-state buffer U254h, which is permanently enabled, before being fed to the DDS module via pin J196 of the CPU board. Therefore executing a stack reset instruction will cause a clock pulse to be sent to the DDS, and the display will increment.

Chapter 5

Control 2 : The Microcode Sequencer and JMUX

5.1 The microcode address sequencer

The original PERQ 1 CPU board had a 4K control store, and therefore 12 microcode address lines. The microcode program was sequenced by a 2910 chip from AMD, and for compatibility the same device was used in the PERQ 2 16K CPU board, and an extra circuit, to provide the upper 2 address lines, was added.

5.1.1 The AMD 2910 sequencer

The lower 12 microcode address lines are controlled by the 2910 sequencer chip at location U112. The operation of this device is described in the relevant AMD data sheet, and will not be repeated here. This section will explain how the device is connected to the rest of the PERQ CPU, and how it is controlled by the microcode.

The 2910 executes one of 16 different microcode branch functions per clock cycle, selected by the 4 control inputs I0-I3. These lines are directly driven by the microcode word JMP field (JMP<0> - JMP <3>), and therefore this field selects one of the 16 functions given in the following table.

JMP	Mnemonic	Comments
00	Jump0	Forced jump to location 0
01	Call	Conditional Subroutine Call
02	*	Load PC from JMUX
03	Goto	Conditional jump
04	Pushload	Push address and load register
05	CallS	Conditional call from register
06	*	Load PC from JMUX
07	GotoS	Conditional jump from register
10	Repeatloop	Dec register, goto TOS
11	Repeat	Dec register, goto JMUX
12	Return	Return from subroutine
13	JumpPop	Pop stack, Conditional Jump to JMUX
14	LoadS	Load Register
15	Loop	Condition jump to TOS
16	Next	Inc PC
17	3way branch	Conditional counted loop

A full explanation of the operations performed by those instructions is given in the 2910 data sheet.

The output enable of the 2910 is tied low, so that the microcode address outputs are permanently enabled. Similarly, the CCEN* (Condition Code Enable) pin is also tied low, so that the condition input is always significant. The RLD* (Register Load) input is disabled by connecting it to the +3B signal, which is pulled high through R3 (a 1k resistor). The CI (Carry Input) is similarly pulled high, so that the program counter may increment.

The outputs of the 2910 are buffered by 74S240 NOT gates as described below. The data inputs of the 2910 are driven by the JMUX lines. The JMUX, which selects between the possible jump address sources, is also described below.

5.1.2 The extended microcode sequencer

The extra 2 address bits required to address the 16K control store in the PERQ 2 CPU board are provided by a specially designed circuit which contains the main elements of the 2910 sequencer, namely a Bank Register (to act as the extended program counter), an 8-level stack, a multiplexer to switch the address outputs between the various sources, and a control system built round a 512byte PROM. The operation of these various sections will be described next.

5.1.2.1 The Control ROM

The key element in this extended sequencer is the USQ01 control ROM at location U67. This chip is controlled by the microcode work H and JMP fields and the condition code input, and depending on their values, controls the other sections of the sequencer. It also handles decrementing the stack pointer on a return instruction. The address and data signals for this PROM are given in the following table.

Address	Signal	Comments
A0	NSP2	Stack pointer bit 2 from USPAL0
A1	NSP1	
A2	NSP0	
A3	CONDITION'	Condition code input
A4	JMP<3>	Microcode JMP field
A5	JMP<2>	
A6	JMP<1>	
A7	JMP<0>	
A8	H	Microcode H field

Data	Signal	Comments
D0	ZeroBank	Clear Bank register (PC)
D1	LD REG	Load Register
D2	MuxA	Address Multiplexer Select
D3	MuxB	
D4	SP<0>	Stack pointer output
D5	SP<1>	
D6	SP<2>	
D7	Push	Control signal to increment SP

The operation of this prom is described by the following table

		Cond=1					Cond=0				
JMP	H	ZeroBank	LdReg	Push	Mux	SP	ZeroBank	LdReg	Push	Mux	SP
00	X	1	0	0	Bank	Zero	1	0	0	Bank	Zero
01	X	0	0	0	Bank	Same	0	0	1	Jmux	Same

02	0	0	0	0	Bank	Same	0	0	0	Bank	Same
02	1	0	0	0	Jmux	Same	0	0	0	Jmux	Same
03	X	0	0	0	Bank	Same	0	0	0	Jmux	Same
04	X	0	0	0	Bank	Same	0	1	0	Bank	Same
05	X	0	0	1	Reg	Same	0	0	1	Jmux	Same
06	X	0	0	0	Bank	Same	0	0	0	Bank	Same
07	X	0	0	0	Reg	Same	0	0	0	Jmux	Same
10	X	0	0	0	Bank	Same	0	0	0	Bank	Same
11	X	0	0	0	Bank	Same	0	0	0	Bank	Same
12	X	0	0	0	Bank	Same	0	0	0	Stack	Dec
13	0	0	0	0	Bank	Same	0	0	0	Jmux	Same
13	1	0	0	0	Bank	Same	0	0	0	Jmux	Dec
14	X	0	1	0	Bank	Same	0	1	0	Bank	Same
15	X	0	0	0	Bank	Same	0	0	0	Bank	Same
16	X	0	0	0	Bank	Same	0	0	0	Bank	Same
17	X	0	0	0	Bank	Same	0	0	0	Bank	Same

The output signals from this PROM will be described along with the circuits they control below.

5.1.2.2 The Bank Register

The bank register is the analogue of the program counter, although no system is provided to increment it - the only way it can be changed is by a Leap or Return instruction. This register consists of 2 sections of a 74S174 hex D-type flip-flop (U72a and U72b) which are clocked by the rising edge of the LD MIR' A signal at the start of each active microcycle, and the 2 D inputs are driven by the unbuffered microcode address lines UUA<13> (U72a) and UUA<12> (U72b). The outputs of these flip-flops are connected to the '2' inputs of the address multiplexer and also to the JMUX logic described below. U72 is reset by the 74S00 NAND gate U86a, which combines the CLK-4J clock signal with the ZeroBank output from the USQ01 control PROM. When the microcode instruction specifies a Jump0 operation, the ZeroBank signal goes high, and the Bank register is cleared during the second half of the microcycle.

5.1.2.3 The S Register

This circuit is the equivalent of the counter register in the 2910, although there is no mechanism for automatically decrementing it. The register consists of 2 sections of the 16R8 registered PAL USPAL0 (at location U68), which is clocked by the rising edge of the LD MIR'A signal. The output

enable pin is tied low, so that the outputs are permanently enabled, and the relevant PAL equations are :

!Sreg12:=!sreg12 . !Jmux12 + !Sreg12.!LdReg + LdReg.!J12
!Sreg13:=!sreg13 . !Jmux13 + !Sreg13.!LdReg + LdReg.!J13

So, while LdReg is low, each S Register Bit is loaded with itself on the rising edge of the LD MIR' A signal. When an S register load instruction is executed, the USQ01 PROM brings LD REG high, and the S Register bits are loaded with the state of the appropriate JMUX input.

The outputs of the S register are connected to the '1' input of the extended address multiplexer.

5.1.2.4 The stack pointer

The circuit to generate the stack pointer for the extended address stack is rather complex, and involves sections of both the USPAL0 and USQ01 Rom. The basic idea is that the 3-bit stack pointer is stored in 3 of the D-type flip-flops in the USPAL0. This 3-bit register stores the address of the first free location in the Stack RAM. The outputs of these flip-flops are fed through the USQ01 ROM, which can then decrement the stack pointer on a POP, and then fed to the address inputs of the Stack RAM. The outputs of the USQ01 ROM are also fed back to the inputs of the USPAL0, which increments the value on a push and then written back to the register inside the USPAL0. This register is clocked on the rising edge of the LD MIR' A signal at the start of each microcycle. The PAL equations for the USPAL0 to perform the gated increment operation are :

!NSP<0>:=!SP<0>.!Push + SP<0>.Push
!NSP<1>:=!SP<1>.!Push + !SP<1>.!SP<0> + SP<1>.SP<0>.Push
!NSP<2>:=!SP<2>.!Push + !SP<2>.!SP<0> + !SP<2>.!SP<1> +
SP<2>.SP<1>.SP<0>.Push

This circuit is capable of performing 4 different operations, which will now be described.

During a Jump0 instruction, the USQ01 PROM forces all the SP<n> outputs to 0, and since the Push output is low, the USPAL0 writes this 0 into the stack pointer register.

When a value is to be pushed onto the stack, the USQ01 PROM passes the value of the SP register unchanged onto the SP<n> lines. Since the PROM now brings the Push output high, the USPAL0 increments the value on the SP<n> lines before writing it back to the SP register inside the USPAL0.

When a value is popped from the stack, the USQ01 PROM decrements the value of the SP register before sending to the SP<n> lines. Thus the last value pushed onto the stack is addressed. Since the Push output from the USQ01 PROM is low, this decremented stack pointer is written back to the SP register at the end of the microcycle.

During all other operations, the contents of the SP register are unchanged by both the USQ01 PROM and the USPAL0 and the SP register is loaded with the same value at the end of the microcycle.

5.1.2.5 The USPAL0 PAL

This 16R8 PAL at location U68 on the CPU board contains the S register and SP register circuits that have just been described. The signals connected to this PAL are given by the following table.

Pin	Signal	Comments
Clock	LD MIR' A	
I2	SP<2>	Stack Pointer
I3	SP<1>	
I4	SP<0>	
I5	Push	Enable Increment of SP
I6	LD REG	Load S Register
I7	JMUX<13>	JMUX line
I8	JMUX<12>	
O12	NSP<0>	Output of SP register
O13	NSP<1>	
O14	NSP<2>	
O15	S<12>	S register output
O16	S<13>	

5.1.2.6 The Stack

The extended microcode address stack is stored in the 16*4 bit 74S189 RAM at location U87. The lower 3 address inputs of the RAM are driven by the SP<n> outputs from the USQ01 PROM, while the top address input is grounded since only 8 locations are used. The Chip Enable input is also grounded, so that the data outputs are always active. The lower 2 data inputs are driven by the outputs of the Bank Register described above, so that the current bank can be pushed onto the stack. The 2 corresponding inverted data outputs are reinverted by a pair of 74S04 NOT gates (U90a for bit 13 and U90d for bit 12), before being fed into the address multiplexer. When a Push operation is performed, the Push output from the USQ01 PROM goes high. This enables the 74S00 NAND gate U86d, and during the second half of the microcycle, when CLK-4E is high, the output of U86d (which combines Push and CLK-4E) goes low and writes the Bank Register onto the stack.

5.1.2.7 The Address Multiplexer

The 74S153 dual 4-input multiplexer at location U70 selects between the 4 possible sources of the extended microcode address. It is controlled by 2 output bits from the USQ01 PROM, and the enable inputs to both sections are grounded. The multiplexer's inputs come from the following sources :

Input	Source
0	Jmux
1	S Register
2	Bank
3	Stack

The outputs of the multiplexer provide the 2 extended microcode address lines, and are buffered by 74S240 NOT gates, as described below, before driving the control store address inputs.

5.1.3 Operation of the Extended Microcode Sequencer

In this section, a NOP will be defined as : The ZeroBank, LDReg and Push signals are all inactive, the Multiplexer gates the Bank Register onto the address lines, and the Stack pointer is unchanged.

The way the extended microcode sequencer executes the various jump instructions will now be described.

5.1.3.1 Jmp=0 - Jump0

During this instruction, the ZeroBank signal goes active, so the Bank Register is cleared during the second half of the microcycle. Also, the stack pointer is cleared as described above. The Multiplexer gates the cleared Bank Register onto the address outputs.

5.1.3.2 Jmp=1 - Call

If the CONDITION' signal is high, the condition is false, and the sequencer executes a NOP. When the CONDITION' signal is low, the PUSH signal goes active, pushing the Bank Register onto the stack, and the multiplexer gates the JMUX outputs onto the address lines.

5.1.3.3 Jmp=2, H=0 - NextInst

This instruction causes the sequencer to execute a NOP

5.1.3.4 Jmp=2, H=1, Revive Victim

During this instruction, the multiplexer gates the JMUX lines (which contain the contents of the victim register, as described below) onto the address outputs. In all other respects, the sequencer executes a NOP.

5.1.3.5 Jmp=3 - Goto

If the condition is false (the CONDITION' signal is high), the sequencer performs a NOP. If the condition is true, the Multiplexer gates the JMUX lines onto the address lines, but the other parts of the sequencer execute a NOP.

5.1.3.6 Jmp=4 - PushLoad

If the Condition is true, the LdReg signal goes active, and loads the JMUX lines into the S Register. In all other respects, a NOP is executed. A false condition causes the sequencer to perform a NOP.

5.1.3.7 Jmp=5 - CallS

The contents of the Bank Register are pushed onto the stack. If the Condition is false, the Multiplexer gates the S Register onto the address outputs, while if the condition is true, the JMUX lines are gated there instead.

5.1.3.8 Jmp=6 - Vector or Dispatch

In all cases, the extended sequencer performs a NOP.

5.1.3.9 Jmp=7 - GotoS

If the Condition is false, the contents of the S register are gated onto the address lines by the Multiplexer, while a true condition causes the Multiplexer to gate the JMUX lines there instead. The rest of the sequencer behaves as for a NOP.

5.1.3.10 Jmp=10 - Repeatloop

The extended sequencer executes a NOP during this instruction.

5.1.3.11 Jmp=11 - Repeat

This instruction causes the extended microcode sequencer to execute a NOP.

5.1.3.12 Jmp=12 - Return

If the CONDITION' input is high, that is the condition is false, the sequencer executes a NOP. If the condition is true, the Stack Pointer is decremented, and the Multiplexer transfers the contents of the top of the stack to the address outputs.

5.1.3.13 Jmp=13, H=0 - JumpPop

If the condition is false, the extended sequencer executes a NOP, while if it is true, the Multiplexer gates the JMUX lines to the address outputs.

5.1.3.14 Jmp=13, H=1 - LeapPop

If the condition is false, a NOP is executed. When the CONDITION' signal is low, signifying a true condition, the Stack Pointer is Decremented, and the Multiplexer gates the JMUX lines onto the address outputs.

5.1.3.15 Jmp=14 - LoadS

This instruction is identical to a NOP except that the LdReg signal goes active and causes the JMUX lines to be loaded into the S Register.

5.1.3.16 jmp=15 - Loop, Jmp=16 - Next, Jmp=17 - 3way Branch

All these instructions cause the extended address sequencer to execute NOPs

5.2 The Microcode address buffers

The 14 address outputs from the sequencers described in the previous section are buffered by 74S240 inverting 3-state buffers. In order to get sufficient fan-out to drive the 48 Writable control store RAMs and the 6 boot PROMs, each line drives 2 such buffers, and 2 separate address busses (UA<n>A' and UA<n>B') are produced. The address lines are series terminated by 33Ω resistors before driving the address pins of the memory devices. The buffers and resistors used to produce each signal are given in the following table.

Address Line	Raw Address Line	Buffer	Terminator
UA<0>A’	UUA<0>	U65d	RS2e
UA<0>B’	UUA<0>	U78h	RS6a
UA<1>A’	UUA<1>	U65c	RS2d
UA<1>B’	UUA<1>	U78g	RS5c
UA<2>A’	UUA<2>	U65b	RS2c
UA<2>B’	UUA<2>	U78f	RS6c
UA<3>A’	UUA<3>	U65a	RS2a
UA<3>B’	UUA<3>	U78e	RS6d
UA<4>A’	UUA<4>	U73d	RS5e
UA<4>B’	UUA<4>	U73h	RS4b
UA<5>A’	UUA<5>	U73c	RS5d
UA<5>B’	UUA<5>	U73g	RS4c
UA<6>A’	UUA<6>	U73b	RS1c
UA<6>B’	UUA<6>	U73f	RS4a
UA<7>A’	UUA<7>	U73a	RS1d
UA<7>B’	UUA<7>	U73e	RS4e
UA<8>A’	UUA<8>	U65h	RS1b
UA<8>B’	UUA<8>	U78d	RS6e
UA<9>A’	UUA<9>	U65g	RS1a
UA<9>B’	UUA<9>	U78c	RS6b
UA<10>A’	UUA<10>	U65f	RS2b
UA<10>B’	UUA<10>	U78b	RS5a
UA<11>A’	UUA<11>	U65e	RS1e
UA<11>B’	UUA<11>	U78a	RS5b
UA<12>A’	UUA<12>	U51g	RS3a
UA<12>B’	UUA<12>	U51h	RS3b
UA<13>A’	UUA<13>	U51e	RS3e
UA<13>B’	UUA<13>	U51f	RS3d

5.3 The JMUX

The JMUX is the 14-bit distributed multiplexer that selects the microcode jump address source and feeds it to the address inputs on the microcode address sequencer. The operation of this multiplexer

is somewhat more complicated than that of the AMUX and BMUX multiplexers described in chapter 2, since the jump address may be 8 bits wide (the Z field - a short jump), 12 bits wide (the SF and Z fields - a long jump), or 14 bits wide (the Y and Z fields - a leap). The JMUX also allows jump addresses to be produced containing the next opcode from the opcode file (a 'Next Instruction' Branch), the number of the highest priority interrupt currently active (the interrupt vector), or the lowest 4 bits of the shifter output (a dispatch). Finally, the entire 14 bit address may also be provided by the shifter outputs or the victim latch.

The operation of the JMUX control logic will be described first, followed by a description of how each type of jump address is gated onto the JMUX lines.

5.3.1 The JMUX control Logic

The main device used to control the JMUX is the 12L6 PAL JPPAL3 at location U108. This PAL is controlled by the H,F,SF and JMP microcode fields and produces signals to control the main JMUX buffers. The signals connected to this PAL are as follows:

Pin	Signal	Comments
I1	Jmp<3>	Microcode JMP field
I2	Jmp<2>	
I3	Jmp<1>	
I4	Jmp<0>	
I5	SF<3>	Microcode SF field
I6	SF<2>	
I7	SF<1>	
I8	SF<0>	
I9	F<1>	Microcode F field
I11	F<0>	
I19	H	Microcode H field
O13	DON'T PAGE'	Disable page outputs
O14	Disable Z Addr'	Disable Z field-lower 8 Jmux lines
O15	ENB Y'	Enable Y field to upper 6 Jmux lines
O16	Enb Shift to JMUX'	Enable shifter to all 14 JMUX lines
O17	NopSF'	Enable Long Constant
O18	Disable Bank'	Disable bank register outputs

The PAL equations programmed into the JPPAL3 are given in the following table :

		Comments
DON'T PAGE	=!Jmp<3>.Jmp<2>.Jmp<1>.!Jmp<0>	Vector or Dispatch
	+!Jmp<3>.!Jmp<2>.Jmp<1>.!Jmp<0>	Next Instruction or Revive Victim
	+!SF<3>.SF<2>.SF<1>.!SF<0>.!F<1>.F<0>	2910 := Shifter
	+!SF<3>.SF<2>.SF<1>.SF<0>.!F<1>.F<0>	Leap
Disable Z Addr	=!Jmp<3>.Jmp<1>.!Jmp<0>	Dispatch,Vector, Next Instruction or Revive Victim
	+!SF<3>.SF<2>.SF<1>.!SF<0>.!F<1>.F<0>	2910 := Shifter
EnbY	=!SF<3>.SF<2>.SF<1>.SF<0>.!F<1>.F<0>	Leap
Enb Shift to JMUX	=!SF<3>.SF<2>.SF<1>.!SF<0>.!F<1>.F<0>	2910 := Shifter
NopSF	=!SF<3>.!SF<2>.!SF<1>.!SF<0>.!F<0>	Long Const
	+!SF<3>.SF<2>.!SF<1>.!SF<0>.!F<1>.F<0>	Push Long Constant
Disable Bank	=!SF<3>.SF<2>.SF<1>.!F<1>.F<0>	2910 := Shifter Or Leap
	+!Jmp<3>.!Jmp<2>.Jmp<1>.!Jmp<0>.H	Revive Victim

The other main element in the JMUX control circuit is the 32*8bit 74S288 PROM JPE16 at location U104. This PROM is addressed by the microcode Jmp and H fields, and provides the control signals for the buffers used to gate the interrupt vector, OpFile, victim latch and dispatch addresses onto the JMUX lines. The signals handled by this PROM are listed in the following table

Address	Signal	Comments
A0	H	Microcode H field
A1	JMP<0>	Microcode JMP field
A2	JMP<1>	
A3	JMP<2>	
A4	JMP<3>	

Data	Signal	Comments
D1	Enb Next Inst'	Gate Opfile onto JMUX
D2	Revive Victim'	Read Victim Latch
D3	Enb Dispatch'	Gate lower 4 shifter outputs to JMUX
D4	Enb Vector'	Gate Interrupt Vector to JMUX
D5	Enb Z Vec Fill'	Fill Interrupt Vector Address with Z
D6	Enb Z Op Fill	Fill Special Address with Z

These signal enable the relevant buffers during the following microcode jump instructions :

Signal			Comments
Enb Next Inst'	H=0	Jmp=2	Next Instruction
Revive Victim'		Jmp=0	Clear Victim Control on Jump0
	H=1	Jmp=2	Revive Victim
Enb Dispatch	H=1	Jmp=6	Dispatch
Enb Vector	H=0	Jmp=6	Vector
Enb Z Vec Fill	H=0	Jmp=6	Vector
	H=1	Jmp=6	Dispatch
Enb Z Op Fill	H=0	Jmp=2	Next Instruction
	H=0	Jmp=6	Vector
	H=1	Jmp=6	Dispatch

The outputs of this PROM directly drive the enable inputs of the relevant buffers, as described below.

The Disable Z Addr output from the JPPAL3 is inverted by the 74S04 NOT gate U90c before driving the enable input of the buffers U124e-h and U103a-d. Therefore, when the JPPAL3 asserts the Disable Z addr signal, the buffer is disabled. In a similar way, the Disable Bank output is

inverted by the 74S04 NOT gate U90b which drives the enable input of the Bank Register buffer U84e-h. When the Disable Bank output goes low, this buffer is disabled.

A long jump occurs when the microcode word F field contains 3. This condition is detected by the 74S00 NAND gate U131d, which then asserts the LONG JUMP' signal. This signal is NANDed with the DON'T PAGE' output from the JPPAL3, and the resulting signal, ENB PAGE' is connected to the output enable input of the page register U109. Therefore, when either a long jump or a special jump that changes these 4 microcode address lines is executed, the ENB PAGE' signal goes high and disables the outputs of the page register.

The NOP SF' output from the JPPAL3 is not used by the JMUX control logic. When it is high, it enables the short constant logic via the NAND gate U86c. It is inverted by the 74S04 NOT gate U90e, and then fed to the NAND gate U86b. Therefore, when the NOP SF' signal is low, U86b enables the long constant logic. The constant logic is described in chapter 2.

5.3.2 Operation of the JMUX

The operation of the JMUX buffers and the signals that control them will now be described for each type of jump.

5.3.2.1 Short Jump

When a short jump instruction is executed, the Disable Z Addr' output from the JPPAL3 is high, and therefore the ENB Z ADDR' signal, provided by the 74S04 NOT gate U90c is low. This enables the 74S240 3-state inverting buffers U124e-h and U103a-d and gates the microcode word Z field onto the lower 8 JMUX lines.

The 4 microcode address lines UUA<8>-UUA<11> are clocked into the 74S374 page register U109c-f on the rising edge of the LD MIR' A clock signal at the start of each microcycle. Since both the DON'T PAGE' and the LONG JUMP' signals are high, the 74S00 NAND gate U131a brings the ENB PAGE' signal low and gates the contents of the page register onto the JMUX<8>-JMUX<11> lines. Therefore these address lines are unchanged on a short jump.

The Bank Register in the extended microcode sequencer is loaded with the upper 2 microcode address lines on the rising edge of the LD MIR' A signal as described above. The JPPAL3 output Disable Bank is high, and therefore the 74S04 NOT gate U90b brings the ENB BANK' output low and enables the 74S244 3-state buffer U84. This gates the contents of the bank register onto the upper 2 JMUX lines, so the corresponding microcode address bits are also unchanged on a short jump.

5.3.2.2 Long Jump

The lower 8 bits of a long jump address are provided by the microcode word Z field. Since the Disable Z Addr' output from the JPPAL3 is high, the Z field is gated onto the appropriate JMUX lines as described in the last section.

A long jump occurs when the microcode word F field contains 3. This condition is detected by the 74S00 NAND gate U131d, which therefore brings the LONG JUMP' signal low. This signal disables the 74S00 NAND gate U131a, and the output of this gate is therefore high. This signal, ENB PAGE' then disables the outputs of the page register U103. Since the LONG JUMP' signal is connected to the enable input of the 74S240 inverting 3-state buffer U151e-h, this buffer is enabled during a long jump, and gates the microcode word SF field onto the JMUX<8>-JMUX<11> lines. Therefore the SF field provides those 4 microcode address bits.

The top 2 microcode address bits are provided by the bank register as described in the last section

5.3.2.3 Leap

Again, the lower 8 JMUX bits are provided by the microcode word Z field as described for the Short Jump above.

The JPPAL3 makes the DON'T PAGE' output low, which therefore causes the 74S00 NAND gate U131a to force the ENB PAGE' signal high, and thus disable the outputs of the page register U103. Similarly, the JPPAL3 Disable Bank output is low, and the 74S04 NOT gate U90b disables the 74S244 3-state buffer U84, and therefore the Bank Register is not gated onto the top 2 JMUX lines.

The ENB Y' output of the JPPAL3 is also low, and this enables the 74S244 3-state buffer U122, which gates the lower 6 bits of the microcode word Y field onto the upper 6 JMUX lines.

5.3.2.4 Next Instruction

The Next Instruction Jump is a 256-way conditional jump dependant on the contents of the Opcode file. When such an instruction is executed, the JPPAL3 brings the Disable Z addr output low, and thus the ENB Z ADDR signal from the 74S04 NOT gate U90c is high. This condition disables the 74S240 inverting 3-state buffers U124e-h and U103a-d, and the microcode word Z field is not placed on the lower 8 JMUX lines. The DON'T PAGE' output is also low, and the page register is disabled via the 74S00 NAND gate U131a as described for the Long Jump above.

The JPE16 PROM U104 asserts the ENB NEXT INST' output by making it low. This signal therefore enables the 74S240 inverting 3-state buffer U162, which gates the contents of the Opcode file (OP<n>) onto the JMUX<2>-JMUX<9> lines. The JPE16 PROM also brings the ENB Z

OP FILL' signal low, which enables the 74S240 inverting 3-state buffer U144e-h. This buffer gates 4 of the microcode word Z field bits onto the JMUX lines, as given in the following table.

Z	JMUX	Buffer
Z<0>	JMUX<0>	U144h
Z<1>	JMUX<1>	U144g
Z<6>	JMUX<10>	U144f
Z<7>	JMUX<11>	U144e

The upper 2 bits of the JMUX are drive by the bank register via the buffer U84e-h as described above.

5.3.2.5 Dispatch

When a Dispatch instruction is executed, the JPPAL3 disables the Z field address buffers U124e-h and U103a-d, and the outputs of the page register U109, as described above.

The JPE16 PROM asserts the ENB Z OP FILL signal as described in the last section, and the 74S240 inverting 3-state buffer U144e-h gates the same 4 Z field bits onto the JMUX lines. The JPE16 PROM also brings the ENB Z VEC FILL signal low, which drives the enable input on the 74S240 buffer U146a-d. This buffer gates the remaining 4 bits of the microcode word Z field onto 4 more JMUX lines, as given in the following table:

Z	JMUX	Buffer
Z<2>	JMUX<6>	U146a
Z<3>	JMUX<7>	U146b
Z<4>	JMUX<8>	U146c
Z<5>	JMUX<9>	U146d

The JPE16 PROM also enabled the dispatch address buffer U144a-d by bringing the ENB DISPATCH' signal low. This buffer gates the latched versions of the lower 4 shifter outputs (L SHIFT <n>) onto the JMUX<2>-JMUX<5> lines. The latched shifter outputs are provided by the 74S373 transparent latch U143e-h. This latch is controlled by the LATCH R signal described in chapter 4, and therefore stores the appropriate shifter outputs throughout the second half of the microcycle. This latch ensures that the jump address is stable on the rising edge of LD MIR' at the start of the next microcycle.

The top 2 JMUX lines are controlled by the Bank Register as described above

5.3.2.6 Vector

The Vector Jump is used to provide a conditional microcode branch which depends on the highest active interrupt source. The operation of the JMUX and its control logic is identical to the operation during a Dispatch, except that the ENB DISPATCH' output from the JPE16 PROM is not asserted (low), and therefore the latched shifter outputs are not gated onto the JMUX<2>-JMUX<5> lines. Instead, the JPE16 PROM makes the ENB VECTOR' output low, which enables the 74S244 3-state buffer U84a-d. U84a forces a zero onto the JMUX<5> line, while the remaining sections of this buffer gate the outputs of the interrupt priority encoder U251 (74LS148) onto the JMUX<2>-JMUX<4> lines.

5.3.2.7 2910:=Shifter

This instruction allows the microcode jump address to be provided by the lower 14 outputs of the barrel shifter described in chapter 6. When the microcode word F field contains 1 and the SF field contains 6, signifying this instruction, the JPPAL3 brings the Disable Z Address' signal low, which disables the Z address buffers U124e-h and U103a-d via the 74S04 NOT gate U90c. Similarly, the 74S00 NAND gate U131a is inhibited by the DON'T PAGE' output from the JPPAL3 going low, and therefore the outputs of the page register U109 are disabled. The bank register is not gated onto the upper 2 JMUX lines, since the JPPAL3 forces the Disable Bank' signal low which disables the 74S244 3-state buffer U84e-h via the 74S04 NOT gate U90b.

The lower 14 outputs of the shifter are connected to the D inputs of a pair of 74S373 octal D-type transparent latches at locations U141 (lower 8 bits) and U183 (upper 6 bits). These latches are controlled by the LD MIR'A signal described in chapter 4, and so, during the second half of the microcycle, when the LD MIR'A signal is low, the shifter outputs are held in these latches. The outputs of U141 and U183 are directly connected to the 14 JMUX lines. When a 2910:=shifter instruction is executed, the JPPAL3 brings the ENB SHIFTER TO JMUX' signal low, which drives the Output Enable pin of these latches. Therefore, the outputs are enabled, and the latched shifter outputs are gated onto the appropriate JMUX lines.

5.3.2.8 Revive Victim

The Victim latch, described below, stores the address of a microinstruction that attempted to read a byte from the opcode file when it was empty. The contents of the victim latch can be gated onto the JMUX lines to allow the instruction to be executed again, and this occurs when a Revive Victim jump is executed.

When such a jump happens, the JPPAL3 disables the Z address buffers, the page register, and the bank register buffer as described in the last section. The JPE16 ROM brings the REVIVE

VICTIM' output low which is connected to the output enable pins of a pair of 74S374 flip-flops (U109 and U107) which form the victim latch. The outputs of these flip-flops are directly connected to the 14 JMUX lines. Thus, during a Revive Victim instruction, the outputs of these devices are enabled and gate the victim address onto the JMUX lines.

5.4 The Opcode File

The opcode file is a 64 bit register file consisting of 4 74LS670s at locations U160, U181, U159 and U200. It is loaded as 4 16-bit words provided by the memory PCB on successive microcycles during a FETCH4 memory instruction, and it is read as 8 bytes into either the ALU via the AMUX as described in chapter 2, or onto the JMUX lines as described above.

The control logic used to load and read the Opcode File will now be described.

5.4.1 Loading the Opcode File

The 16 bit data words arrive from the memory PCB via the MDO lines and are stored in 2 74S374 octal D-type flip-flops at locations U224 (for the low byte) and U244 (for the high byte). The outputs of these flip-flops, which are permanently enabled, drive the data inputs of the 74LS670 devices that form the opcode file. The MDO VALID H output from the memory board is strobed into the 74S374 D-type flip-flop U209h by the rising edge of the CLK-0R B clock at the start of each microcycle. The output of this flip-flop, LMDO VALID H, is NANDed with the CLK-4E clock signal by the 74S00 NAND gate U232c, and the output of this gate, the CLK MDOR signal, drives the clock inputs of U224 and U244. When the memory card is going to output a valid data word, it sets the MDO VALID H signal, and at the start of the next microcycle, the LMDO VALID H signal goes high. Near the end of that microcycle, the rising edge of the CLK MDOR signal stores the data word in U224 and U244.

The bottom 2 bits of the main timing counter on the memory board are inverted on that PCB, and fed to the CPU board as the TIME<0>' and TIME<1> signals. The value in this counter, the so-called T-state number, is used to determine which word is begin read from the main memory, as described in chapter 7. The TIME<0>' signal is inverted by the 74S04 NOT gate U208d, to produce B TIME <0>' which is connected to the lower write address input to the Opcode File. The TIME<1>' signal is buffered by connecting it to both inputs of the 74S08 AND gate U231c. The output of this gate, B TIME <1>', is connected to the higher write address input of the Opcode File. Since the first word is read out of the memory during the T state t2, this word is written to the zeroth location of the opcode file, as indicated in the following table.

T state	Word	TIME <1,0>'	B TIME<1,0>'	Location
T2	15-0	01	00	0
T3	31-16	00	01	1
T0	47-32	11	10	2
T1	63-48	10	11	3

So, the words fetched from the main memory are written to the appropriate locations of the Opcode File.

The write-enable control signal for the Opcode File is produced by the following circuit. When the microcode instruction specifies a Reload Op operation (i.e. the F field contains 0 or 2 and the SF field contains 12), the special function decoder asserts the RELD OP signal by forcing it low. This signal is connected to one input of the 74S00 NAND gate U212c, and thus the output of this gate is set high. This output drives the LD OP control flip-flop, the 74S374 U209c. Therefore, at the start of the next microcycle, this flip-flop is clocked by the rising edge of the CLK-0R B clock, and thus the LD OP signal, produced by the output of this flip-flop, is set.

The microcode program is written to cause the microcode cycles between the Reload Op Instruction and the next T2 state to be aborted, e.g. by making the A microcode field of the Reload Op instruction specify one of the 2 memory data operations. Therefore, the LD MIR' clock does not go active until the next T2 state, and the microcode pipeline register is not reloaded. Therefore the RELD OP output remains active, and the LD OP flip-flop is held set.

At the next T2 state, the RELD OP signal returns to the inactive (high) state. However, since both the output of the 74S00 NAND gate U210b, and the LD OP signal are high, the 74S00 NAND gate U210c which combines these 2 signals and produces CONT' is enabled, and the CONT' signal is forced low. This signal is applied to the other input of the 74S00 NAND gate U212c, and thus the LD OP flip-flop U209c is held set until the end of the next T1 state.

During the subsequent T1 state, both the B TIME<1>' and B TIME<0> signals are high. These signals are NANDed together by the 74S00 NAND gate U210b, whose output therefore goes low. This output inhibits the 74S00 NAND gate U210c and therefore the CONT' signal goes high. Since the RELD OP output from the special function decoder is now also high, the output of U212c goes low, and on the rising edge of the CLK-0R B clock at the start of the next microcycle, the LD OP flip-flop U209c is cleared.

The output of the LD OP flip-flop (the LD OP signal) is connected to one input of the 74S10 NAND gate U207a. This gate NANDs LD OP with the CLK-4J and the CLK-41 clock signals, and produces a write-enable pulse for the Opcode File chips. The timing of this signal is described in chapter 4. The output of U207a is directly connected to the write-enable inputs of the Opcode File

registers.

Therefore, to load the opcode file, the microcode program starts a FETCH4 memory cycle, and then executes a Reload Op instruction. This sets the LD OP flip-flop, and causes the data fetched from memory to be written into the Opcode File. When all 4 data words have been fetched, the LD OP flip-flop is cleared, and the system returns to the idle state.

A low level on the CONT' signal (which occurs when the Opcode File write logic is active) is also used to clear the Boot ROM enable flip-flop U236b which is described in chapter 4

5.4.2 Reading the Opcode File

The data outputs of the opcode file are gated onto the lower 8 AMUX lines by the 74S240 inverting 3-state buffers U161e-h and U182a-d. These buffers are enabled by the NEXT OP' signal from the AMUX control logic, and this circuit, together with the rest of the AMUX system, is described in chapter 2. The outputs of the Opcode file can also be used to control the microcode jump address during a Next Instruction operation by gating them onto the appropriate JMUX lines as described above. This function is enabled by the ENB NEXT INST' signal from the JPE16 PROM U104.

The read address for the Opcode File is provide by the Byte Program Counter (BPC), which is the 74S163 4-bit binary counter at location U201. The B and C (2^1 and 2^2) outputs of this counter are connected to the read address lines of the 74S670s in the Opcode file. Since there are only 8 bytes in the Opcode File, the most significant output of the BPC (BPC<3>) is low whenever the BPC points to a valid Opcode File Location.

The least significant output of the BPC (BPC<0>) is inverted by the 74S04 NOT gate U208b to produce the BPC<0>' signal. Similarly BPC<3>, the most significant output, is inverted by U192f (74S04) to give BPC<3>'. The read enable signal to the 74S670 Opcode File Chips U181 and U160, ENB OP<0>' is produced by NANDing the BPC<3>' and BPC<0>' with the 74S00 NAND gate U203a. Therefore, these chips, which are loaded with the low byte of the memory data, are enabled when the BPC contains a valid even address. Similarly, the read enable signal for U159 and U200, which store the upper byte of the memory word, is provided by the 74S00 NAND gate U203d, which combines the BPC<3>' and BPC<0> signals. Therefore, these devices are enabled when the BPC contains a valid odd address.

The BPC can be loaded with the latched form of the bottom 4 bits of the R register (SL<0>-SL<3>) by the BPC:= signal from the Special Function decoder as described in chapter 4. The Reset input of the BPC is connected to the +3B line, and thus pulled high through R3 (a 1k resistor). The BPC is clocked by the rising edge of the LD MIR'B signal at the start of every microcycle, and therefore, if the NEXT INST OP signal is high, the BPC is incremented and selects the next byte in the Opcode File.

The NEXT INST OP signal is produced by the 74S00 NAND gate U203b which combines the

active-low signals NEXT OP' and ENB NEXT INST'. One of these signals goes active on either type of instruction that reads a byte from the Opcode File, and therefore after such an instruction, the BPC is incremented and points to the next byte in the Opcode File.

The value in the BPC can be read onto the lowest 3 AMUX lines via the 74S240 inverting 3-state buffer U202a-d, which is part of the microstate register. This register, and its associated control logic, is described in chapter 2.

5.5 The Victim Latch

The Victim latch consists of a pair of 14 bit registers which are loaded with the microcode control store address at the start of each instruction, one of which can be transferred to the JMUX inputs to the microcode sequencer, while the other may be transferred into a normal CPU register If an attempt is made to read the Opcode File via the AMUX and the Opcode file is empty (that is, the BPC<3> bit is 1), then the CPU control electronics locks the address of that microcode instruction in the Victim l=Latch. After the Opcode file has been refilled from the main memory, the contents of the Victim Latch can be transferred to the microcode sequencer via the JMUX, and the original instruction repeated.

The copy of the Victim Latch which can be transferred to the JMUX consists of the pair of 74S374 flip-flops at locations U107 (for the top 8 bits of the microcode address) and U108 (for the lower 6 bits). The D inputs of these flip-flops are directly connected to the UUA<n> microcode address lines, while the outputs are connected to the JMUX lines. The flip-flops are clocked by the LD VICTIM' signal that is described below, and the outputs are gated onto the JMUX lines by the REVIVE VICTIM signal which is connected to the Output Enable pin of U107 amd U108. This signal is described above.

The second copy of the Victim Latch may be transferred onto the R register lines. This consists of 2 74S534 octal inverting D-type flip-flops at locations U88 (for the top 8 bits) and U79 (for the bottom 6 bits of the microcode address). The D inputs of these flip-flops are again connected to the UUA<n> control store address lines, and the chips are both clocked by the LD VICTIM' signal. The 3-state outputs of these devices are connected to the bottom 14 of the R-register outputs. These outputs are enabled by the active-low RD VICTIM output from the special function decoder described in chapter 4. This signal is also connected to one input of the 74S10 NAND gate U213b which drives the output enable inputs of the R-register latches. When the (R):=Victim instruction is executed (that is, when the microcode word F field contains 1, and the SF field contains 0), the RD VICTIM signal goes low, disables the R register outputs via U213b and gates the contents of the Victim Latch onto the R lines and hence into a CPU register.

The way in which the Victim Latch is loaded will now be described. Normally, the P HOLD

VICTIM' signal is high. This signal is NANDed with the CLK-4E CPU clock signal by the 74S00 NAND gate U210d to produce the LD VICTIM' signal that directly drives the clock inputs of the 4 chips that make up the Victim Latch. Therefore, at the start of every microcycle, when the CLK-4E clock goes low, there is a rising edge on the LD VICTIM' signal that clocks the microcode address into the Victim Latch.

When the Opcode File is read via the AMUX into the ALU, the NEXT OP' output from the AMUX control decoder U223 (74S138) goes low. This signal is inverted by the 74S240 NOT gate U206c, and then ANDed with the most significant bit of the Byte Program Counter, BPC<3>, by one side of the 74S51 AND-OR-Invert gate U205b. If an attempt is made to read the opcode file when it is empty, therefore, the output of U205b, the P HOLD VICTIM' signal goes low, which inhibits the 74S00 NAND gate U210d, and therefore the Victim Latch is not clocked at the start of the next microcycle.

The P HOLD VICTIM' signal is also connected to the D input of the 74S374 flip-flop U209d, which is clocked by the main clock signal CLK-0R B. Therefore, at the start of the next microcycle, this flip-flop is set to 0. The output of this flip-flop is inverted by the 74S240 NOT gate U206d, and then ANDed with the output of the 74S08 AND gate U73Ac by the other side of U205b. Therefore, while the output of U73Ac is high, U209d is reloaded with 0.

U73Ac ANDs the 2 active-low signals that read the Victim Latch, namely RD VICTIM and REVIVE VICTIM'. Since both of these signals are normally high, the output of U73Ac is also high, as required by the previous paragraph. When the Victim Latch is read, the output of U73Ac is set to 0, and thus the output of U205b is forced high. At the end of the microcycle, when a rising edge of the CLK-0R B clock signal occurs, U209d is loaded with 1, which therefore causes the output of U205b to remain high, and the loading of the Victim Latch at the start of each microcycle resumes.

5.6 The Interrupt Encoder

The PERQ 16K CPU board can accept interrupt signals from 8 different sources, which are listed in the following table.

Signal	Latch	U251 Input	Comments
PAR INTR L	U250h	0	Memory Parity Error
X INT INTR L	U250g	1	OIO Interrupt # 1
LINE COUNT INTR L	U250f	2	Line Counter overflow
Z80 RDY INT L	U250e	3	EIO CPU data In ready
NET INTR L	U250d	4	EIO Network Interrupt
DISK INTR L	U250c	5	EIO HArd Disk Interrupt
Y INTR L	U250b	6	OIO Interrupt # 2
UPROC INTR L	U250a	7	EIO CPU Data Out ready

Unlike a conventional processor, the PERQ CPU does not automatically respond to interrupts, but instead the microcode program must check for a pending interrupt periodically, and if one occurs, jump to the appropriate service routine.

The 8 interrupt input signals are clocked into the 74S374 Octal D-type flip-flop at location U250 by the rising edge of the CLK-0R D clock at the start of each microcycle. The outputs of U250, which are permanently enabled, are connected to the 8 inputs of the 74LS148 priority encoder U251, as indicated in the table above. The EI expansion input of this encoder is grounded, so that the encoder is always active. The EO output, which indicates if any of the interrupt signals are active, is latched by the 74S374 flip-flop U209a on the rising edge of the next CLK-0R B clock to give the INTR PEND (Interrupt Pending) Input to the Condition Code logic. The 3 data outputs from the priority encoder U251, which carry the number of the most significant active Interrupt, are fed to the interrupt vector buffer U84a-d and then gated onto the JMUX lines during a vector jump instruction as described above.

Five of the 8 interrupt lines are driven by hardware that is fitted to every PERQ system, but the remaining 3, namely X INT INTR L, Y INTR L and NET INTR L, are provided by optional equipment. Therefore these 3 inputs are pulled up on the CPU board by the 1k resistors RS7b, RS7c and RS7i respectively.

5.7 The Condition Flags

The microcode sequencer may be controlled by one of 15 different conditions which are selected by the 4 bit microcode word COND field. This field controls the condition multiplexer U187 and U189 (both 74S251s) which then pass the selected condition signal onto the CONDITION' input of the sequencer. Eight of these condition flags, the so-called Arithmetic Conditions, are provided by a 16R8 PAL (CCPAL0) at location U186, and this circuit will therefore be described first.

Four signals from the data path are latched in the 67S380 inverting transparent latch U165a-d, as listed in this table.

Signal	Latch	Latched Signal	Comments
AMUX<15>'	U165a	LA<15>	Top bit (sign) of AMUX
BMUX<15>'	U165b	LB<15>	Top bit (sign) of BMUX
C<15>	U165c	LC<15>	Carry out from bit 15
A=B	U165d	LA=B	Lower 16 ALU outputs 0

These latches are controlled by the CLK-0R C clock described in chapter 4. During the first half of the microcycle, the outputs of the latches are updated, but the signals are held stable during the second half, when the CLK-0R clock is low. The outputs of U165 are permanently enabled, since the output enable pin is tied low. These signals are then fed to the inputs of the CCPAL0, which is described next.

5.7.1 The Condition Code PAL

The condition code PAL is the 16R8 registered PAL CCPAL0 at location U186, which accepts inputs from the data path, and produces the arithmetic flags. The signals processed by this chip are listed in the next table.

Pin	Signal	Comments
I2	ArithX	ALU Subtraction indicator
I3	ArithY	ALU Addition indicator
I4	R15'	Active-low R-register sign bit
I5	LA=B	Active-low 'Low 16 ALU outputs 0' signal
I6	LC<15>	Active-low Carry Out from bit 15
I7	LB<15>	Active-high BMUX bit 15
I8	LA<15>	Active-high AMUX bit 15

(All outputs are active-high)

O12	OVF	Overflow Flag
O13	CRY	Carry Flag
O14	Leq	Less than or equal Flag
O15	Lss	Less than Flag
O16	Geq	Greater than or Equal Flag
O17	Gtr	Greater than Flag
O18	Neq	Not Equal Flag
O19	EQL	Equal Flag

The outputs of this PAL are always active, since the output enable pin is tied to ground. The CCPAL0 output register is clocked by the LD MIR' B signal. Therefore, at the start of each microcycle, on the rising edge of the LD MIR' clock, the flags are updated based on the previous microcycle's ALU result.

The logic functions used to generate each of the arithmetic flags will now be described.

5.7.1.1 OVF

This flag tests for overflow on addition or subtraction. The Logic Equation used to generate this flag is :

		Comments
!OVF:=	R<15>'.!LA<15>	No Overflow if +ve result, first operand +ve
	+ArithX.ArithY.R<15>'	Never Occurs!
	+!ArithX.R<15>'.!LB<15>	Addition of a +ve, Giving a +ve result
	+!ArithY.R<15>'.LB<15>	Subtraction of a -ve Leaving a +ve Result (The above 2 terms also Cover logical operations)
	+!R<15>'.LA<15>	-ve result from -ve first operand
	+ArithY.!R<15>'.LB<15>	Additon of a -ve and leaving a -ve result
	+!ArithX.!ArithY.!R<15>'	-ve result on any logical Op
	+ArithX.!R<15>'.!LB<15>	Subtration of a +ve leaving a -ve result

In the above cases, the sign of the result is correct, so the overflow flag is not set. In all other cases, a carry out of the most significant bit has caused the sign to be incorrect, so the overflow flag is set.

5.7.1.2 CRY

The carry flag is produced in the obvious way by the following PAL equation :

		Comments
!CRY:=	LC<15>	No Carry if LC<15> high

The Carry output from bit 15 of the ALU is inverted by the latch U165c, and therefore reinverted by the CCPAL0 before being fed to the condition multiplexer.

5.7.1.3 LEQ

The PAL equation that produces the Less than or equal flag is the following:

```
!LEQ:=    ArithX.!ArithY.!R<15>'.LA=B.LB<15>.!LA<15>
          +!ArithX.ArithY.!R<15>'.LA=B.!LB<15>.!LA<15>
          +R<15>'.LA=B.!LA<15>
          +ArithX.ArithY.R<15>'.LA=B
          +!ArithX.R<15>'.LA=B.!LB<15>
          +!ArithY.R<15>'.LA=B.LB<15>
```

The LEQ flag is set if the result is either 0 or negative. Therefore, the state of the LEQ flag is given by logically ORing the Equal flag with the LSS flag. Since the PAL internally operates with inverted signals, the equation for the !LSS flag given below is ANDed with the inverted Equals flag - the LA=B signal, to produce the !LEQ signal.

5.7.1.4 LSS

This flag is defined by the following logic equation:

!LSS:=	ArithX.!ArithY.!R<15>'.LB<15>.!LA<15>	
		Subtraction of a -ve from a +ve, leaving a -ve Overflow, should be +ve
	+!ArithX.ArithY.!R<15>'.!LB<15>.!LA<15>	
		Addition of 2 +ves Giving a -ve Again, Overflow - should be +ve
	+R<15>'.!LA<15>	+ve result from +ve first operand
	+ArithX.ArithY.R<15>'	Never Occurs
	+!ArithX.R<15>'.!LB<15>	No Overflow conditions for
	+!ArithY.R<15>'.LB<15>	Addition or Subtraction (Or a Logical operation) Giving a +ve result

The Less flag is set if the ALU result is negative, and therefore the !LSS signal should be true if the top bit of the previous ALU result was +ve. A complication arrises because an overflow can occur on either addition or subtraction, and the flag must take account of this. Therefore, the equation considers all cases when a positive result should have occurred, irrespective of whether or not the result overflowed.

The resultant flag indicates whether the result would have been negative on a machine with arbitrary word length.

5.7.1.5 GEQ

This flag is the logical inverse of the LSS flag, and indicates when the previous ALU result would have been 0 or positive - that is when the top bit, corrected for a possible overflow, would have been 0. The inverted !GEQ signal, used inside the CCPAL, therefore tests for the sign bit of the result being set. The equation used is :

!GEQ:=	ArithX.!ArithY.R<15>'.!LB<15>.LA<15>	
		Subtraction of a +ve
		from a -ve, leaving a +ve
		Overflow, should be -ve
	+!ArithX.ArithY.R<15>'.LB<15>.LA<15>	
		Addition of 2 -ves
		Giving a +ve
		Again, Overflow - should
		be -ve
	+!R<15>'.LA<15>	-ve result from
		-ve first operand
	+!ArithX.!ArithY.!R<15>'	-ve result on
		Logical Operation
	+ArithX.!R<15>'.!LB<15>	No Overflow conditions for
	+ArithY.!R<15>'.LB<15>	Addition or Subtraction
		Giving a -ve result

5.7.1.6 GTR

The Greater Than flag is set if the greater than or equal flag is set, and the equal flag is not - the ALU result was strictly positive. Therefore, inside the PAL, the !GTR signal is produced by logically ORing the !GEQ signal with the !LA=B not equal signal, using the following equation :

!GTR:=	!LA=B
	+ArithX.!ArithY.R<15>'.!LB<15>.LA<15>
	+!ArithX.ArithY.R<15>'.LB<15>.LA<15>
	+!R<15>'.LA<15>
	+!ArithX.!ArithY.!R<15>'
	+ArithX.!R<15>'.!LB<15>
	+ArithY.!R<15>'.LB<15>

5.7.1.7 NEQ

The Not Equal flag is produced in the obvious way by latching the LA=B inverted equal signal in the CCPAL0 output register. This is done with the following equation:

!NEQ:= !LA=B

5.7.1.8 EQL

The Equal flag is produced by inverting the LA=B inverted equal signal, and storing the result in the PAL's output register. This is performed by the following simple equation:

!EQL:= LA=B

5.7.2 The Condition Multiplexer

The condition multiplexer consists of a pair of 72S251 8-input multiplexers at locations U189 and U187. The select inputs of these chips are directly connected to the lower 3 bits of the condition code field of the microcode pipeline register described in chapter 4 (the CND<0>-CND<2> signals). The 3-state inverting outputs of the 2 multiplexers are linked together, and to the CONDITION' input of both the 2910 microcode address sequencer and the extended microcode address sequencer, as described above. The Enable input of U189 is driven by CND<3>, the most significant bit of the COND field, so that this multiplexer is enabled when one of the first 8 conditions is selected. The CND<3> signal is inverted by the 74S240 NOT gate U185d, the output of which is connected to the enable input of U187, which is thus enable when one of the second 8 conditions is selected. This circuit therefore selects between the 16 condition flags and therefore operates as a 16 input multiplexer.

The way in which each of the condition flags is produced will now be explained.

5.7.2.1 0 - U189(0) - True

This input is driven by the WR NOW' signal from the control store loading circuit. This signal is normally high, so the microcode sequencer receives a True (low) CONDITION' input. However, this signal is forced low during the first microcycle of the control store load operation, and this causes the microcode sequencer to output the contents of the S register onto the control store address

lines. During the second microcycle of such an instruction, the WR NOW' signal is high, and thus the sequencer behaves normally. The operation of the control store load circuitry is described in chapter 4.

5.7.2.2 1 - U189(1) - False

This input is simply connected to ground, so that the sequencer always receives a false condition.

5.7.2.3 2 - U189(2) - Intr Pend

The output of U209a (a 74S374 D-type flip-flop) is connected to this input of the multiplexer. U209a is loaded with the state of the Interrupt encoder EO output at the start of each microcycle, so this condition is true if a CPU interrupt input is active. The interrupt priority encoder is described above.

5.7.2.4 3 - U189(3) - Space

This input is not used, and simply left floating.

5.7.2.5 4 - U189(4) - Opfile Empty

This input is connected to the most significant bit of the Byte Program Counter - the BPC<3> signal. It is therefore active when the value stored in the BPC is greater than or equal to 8, i.e. when all the words stored in the opcode file have been read out. The Opcode file and its control logic are described above.

5.7.2.6 5 - U189(5) - C19

The CARRY<19> carry output from the most significant ALU chip U154 (74S181) is latched in the 67S380 D-type latch U165e. This latch is controlled by the CLK-0R C clock signal, and thus the state of this carry signal is held throughout the second half of the microcycle. The output of U165e is connected to the D input of the 74S374 D-type flip-flop U209g, which is directly clocked by the rising edge of the CLK-0R B clock at the start of the next microcycle. Therefore, the output of U209g (the OLD CARRY<19> signal) holds the state of the final carry out from the ALU on the previous microcycle. The output of U209g then drives the relevant input on the condition code multiplexer. Since U165 is an inverting latch, this condition is true when no carry out occurs.

On the PERQ T4 24 bit CPU, a link (JP2) enables the input of the final carry latch (the equivalent of U165e on the 20 bit CPU) to be switched between the carry out from bit 19 (for T2 compatability) or the carry out from bit 23 (when the CPU operates in 24 bit mode).

5.7.2.7 6 - U189(6) - Odd

The contents of the R register are transferred to the microcode store input latch U56 and U71 (both 74S374s) on the rising edge of the LD MIR'B clock at the start of each microcycle. The operation of this register is described in chapter 4. Since the data path uses active-low data signals, UW<0>, the least significant bit of this register, is therefore set low if the previous ALU result was odd. This bit is stored in U71d, and the output from this flip-flop is inverted by the 74S240 NOT gate U192h, the output of which drives the appropriate condition multiplexer input.

5.7.2.8 7 - U189(7) - Bytesign

This signal is produced in a similar way to the Odd flag just described. UW<7>, bit 7 of the microcode store input latch, indicates the sign of low byte of the previous ALU result. The output of U71e, which stores UW<7>, is inverted by U192g, the output of which drives the relevant condition code multiplexer input.

5.7.2.9 10 - U187(0) - Neq

This input is driven by the O18 output of the CCPAL0 described above.

5.7.2.10 11 - U187(1) - Leq

The O14 output of the CCPAL0 is directly connected to this input of the condition code multiplexer. The operation of the CCPAL0 is described above.

5.7.2.11 12 - U187(2) - Lss

Again, this input is directly driven by one of the CCPAL0 outputs, in this case the O15 output.

5.7.2.12 13 - U187(3) - Ovf

The overflow output O12 from the CCPAL0 described above is connected to this input of the condition multiplexer.

5.7.2.13 14 - U187(4) - Carry

The latched carry output from the CCPAL0 appears on the O13 pin, which drives this input of the condition code multiplexer.

5.7.2.14 15 - U187(5) - Eql

This input is driven by the O19 output from the CCPAL, which carries the latched Equal signal as described above.

5.7.2.15 16 - U187(6) - Gtr

The O17 output from the CCPAL0 directly drives this input.

5.7.2.16 17 - U187(7) - Geq

This last input of the condition code multiplexer is driven by the O16 output from the CCPAL0 described above.

Chapter 6

The Shifter and Combiner

The operation of shifting a binary number left or right by a given number of bits is often performed by a sequential circuit - the shift register, which shifts the number by one bit on each clock cycle. However, the operation of shifting a number is a combinatorial one - the output depends only on the input and on the number of bits by which it is shifted. The combinatorial circuit that performs such a shift is known as a barrel shifter.

A barrel shifter simply consists of a number of multiplexers, one for each bit in the output word, which select the relevant bits from the input word. The select inputs of all the multiplexers are joined together, and driven by the signals that select the number of bits to shift by. The output of each multiplexer provides one bit in the output word. The following table gives a simple example of a barrel shifter - one that will rotate a 4 bit word by 0,1,2,3 bits.

Input		Multiplexer		
	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

The control inputs on the multiplexers effectively select one horizontal line of the table, and the 4 rightmost columns show which bit of the input word is fed to each output.

A barrel shifter, similar to this, is used in the PERQ 16K CPU board. It selects a 16 bit field from a 31 bit input word, and is used for both arithmetic and graphics operations.

6.1 The shifter

The barrel shifter in the PERQ 16K CPU board consists of 11 AM25S10 chips. The shifter is designed as a 2-stage cascade, where the first stage selects a 28 bit word from the 31 bit shifter input word, starting at bit 0,1,2 or 3, and the second stage extracts a 16 bit result from the 28 bit intermediate word, with the control inputs selecting the first nybble to be included.

The first stage consists of 7 25S10 chips at the following locations:

Location	Input Bits	Output
U196	0-6	SA<0>-SD<0>
U176	4-10	SA<1>-SD<1>
U114	8-14	SA<2>-SD<2>
U134	12-18	SA<3>-SD<3>
U175	16-22	SA<4>-SD<4>
U155	20-26	SA<5>-SD<5>
U115	24-30	SA<6>-SD<6>

These shifters are controlled by a pair of control lines, SHIFT CTRL<2> (The least significant bit) and SHIFT CTRL<3>. These signals are provided by the shifter control PROMs described below. The outputs of this stage of the barrel shifter are directly connected to the inputs of the second stage.

The second stage of the shifter consists of 4 25S10 devices which extract a 16 bit word from the 28 bit output of the first stage, aligned on a nybble boundary. Each chip handles one particular bit in each nybble, as indicated in the following table:

Location	Input bits	Output Bits
U195	SA<0>-SA<6>	0,4,8,12
U194	SB<0>-SB<6>	1,5,9,13
U218	SC<0>-SC<6>	2,6,10,14
U217	SD<0>-SD<6>	3,7,11,15

For example, U195 takes the least significant bit of each nybble of the input word, and passes the relevant 4 bits to the least significant bit of each nybble of the 16 bit shifter output word. These shifters are controlled by the SHIFT CTRL<1> and SHIFT CTRL<0> signals from the shifter

control PROMs, which specify the number of bits by which the shifter input should be rotated right.

The outputs of all 11 barrel shifter chips are permanently enabled since the output enable pin is connected to ground. The outputs of the second stage of the shifter are fed to the combiner logic that is described below.

6.2 The shifter input

When the shifter is used as part of the main CPU data path, the shifter inputs are driven by the ALU to Shifter Latch, but when it is used for raster operations, it is driven by the outputs of the half pipeline latches. These 2 circuits will now be described.

6.2.1 The ALU to Shifter Latch

The 31 bit ALU to Shifter Latch consists of 4 67S380 transparent latches which are loaded with the contents of the R register during the second half of the microcycle. The latches that form part of this circuit are listed in the following table

Location	Input bits	Output bits
U178	R<0>-R<7>	SL<0>-SL<7>
U117	R<8>-R<15>	SL<8>-SL<15>
U135	R<0>-R<7>	SL<16>-SL<23>
U118	R<8>-R<14>	SL<24>-SL<30>

Therefore the input to the shifter consists of 2 copies of the ALU result concatenated together, so that when the shifter extracts a 16 bit field from the middle of this 31 bit word, the result is as if the ALU result had been rotated by the appropriate number of bits.

There is, however, one extra feature of the shifter input latch used in the PERQ. The 15th and 16th inputs (the high bit of the low copy of the R register) and the low bit of the high copy of the R register) are modified by the MULT01 PROM in the multiplication/division logic, so that on multiplication, when the ALU result is shifted to the right, the data can be correctly sign-extended, and on division, when the ALU result is shifted to the left, the next bit of the dividend is placed in the new least significant bit. The operation of the MULT01 prom and its associated logic is described in chapter 3.

The CLK-4F CPU clock signal is NANDed with the ABORT' signal and a pull-up signal from the 1k resistor R5 by the 74S10 NAND gate U210c. The output of this gate is inverted by the 74S04 NOT gate U208c, and the output of this inverted is connected to the clock inputs of the 4 67S380s that form the ALU to shifter latch. Therefore, this latch is enabled during the second half of a non-aborted microcycle, and the contents of the R register are held in the ALU to shifter latch throughout the first half of the next microcycle, when the ALU may read the output of the shifter via the AMUX lines.

The output enable pins of these latch chips are all connected together, and driven by the RO/PS signal from the raster operation control register. When the shifter is being used for arithmetic operations, this signal is low, so the outputs are enabled, and the ALU result is transferred into the shifter. During raster operations, the RO/PS line is high, so the ALU to shifter latch is disabled.

6.2.2 The Raster Operation Source FIFO

As is described in the PERQ User Manual, the PERQ CPU includes a system to simplify the updating of a bit-mapped graphics image. This system, the so-called Raster Operation unit, allows the destination image to be combined with the source image in one of 8 different ways, and the 2 bit-maps do not have to start in the same positions within a 16 bit machine word. The shifter is an important part of the Raster Operation system, as it is used to shift the source bitmap's words to align them with the destination. The Raster Operation unit operates on 64 bit words as are stored in the main system memory, and the words of the source image required to update the next word of the destination image are stored in a 16bit * 16 level FIFO queue - the Raster Operation Source FIFO.

This FIFO is constructed from 4 74S225 devices at the following locations :

Location	bits
U157	0-3
U158	4-7
U137	8-11
U138	12-15

The way in which data is loaded into the FIFO and then transferred from the FIFO into the shifter will now be described.

The data is loaded into the FIFO from the main system memory by a FETCH4 cycle as described in chapter 7. The data path for these words operates in the following way.

The incoming memory data is stored in a pair of 74S374 octal D-type flip-flops at U224 and U244 that are also used to load the Opfile memory, and are therefore described in chapter 5. The MDO VALID H output from the memory board, which indicates when valid data is present on the MDO lines, is latched in U209h (74S374) by the rising edge of the CLK-0R B clock, and the output of this flip-flop is NANDed with the CLK-4E CPU clock signal by the 74S00 NAND gate U232c. The output of this gate, CLK MDOR, is used to strobe the memory data into U224 and U244. The outputs of these memory data registers are connected directly to the data inputs on the FIFOs.

The data is clocked in by the logical AND of the CLK-4F and CLK FIFO IN signals which drive the CK A and CK B pins of the 74S255s. The latter signal is produced by the RTI02 PROM at location U171, which is part of the Raster Operation State Machine. This state machine is described in more detail in chapter 7.

There are a pair of miscellaneous control signals on the FIFO chips, namely CLR* (an active-low reset input) and OE* (an active low output enable). These signals will now be described. The LD W output from the Special Function Decoder described in chapter 4 is connected to the CLR* input of these 4 FIFO devices When the microcode instruction loads the Raster Operation Width Register, this signal goes low, and clears the FIFO. The output enable pins of the FIFO chips are grounded, so the outputs are permanently enabled.

The source bit-map words are transferred out of the FIFO to the shifter by the CK IN input on each FIFO chip. This signal is produced by the following circuit The 2 CPU clock signals CLK-16 and CLK-4E are logically ANDed by the 74S08 AND gate U73Ab to produce a clock pulse that is goes high during the second half of the microcycle, and returns low at the end of the microcycle. This clock is combined with the CLK FIFO OUT enable signal from the Raster Operation State machine by the 74S00 NAND gate U203c, and the output of this gate, the UNLD signal, is connected to the output clock (CK IN) pins of the FIFO devices. Therefore, when the CLK FIFO OUT signal is high, a rising edge of the UNLD signal occurs at the end of the microcycle, and thus the next word in the FIFO appears on the data output lines

A given 16 bit word of the Destination bitmap will in general be combined with parts of 2 source bitmap words, as shown in figure 5.1. For this reason, the 31 bit input to the shifter consists of the current FIFO output word concatenated to the previous shifter output word. Which half of the 31 bit word is latched depends on whether the Raster Operation is being performed Left-to-Right, or Right-to-Left, and this is controlled by the state of the SL/SR signal from the Raster Operation control register U190 described below.

For this reason, the 16 bit output of the FIFO is connected to the inputs of the 2 Raster Operation half pipeline latches, which provide the input to the shifter. These latches are each implemented by a pair of 74S373s, and together they provide the 31 bit shifter input as indicated in the following table.

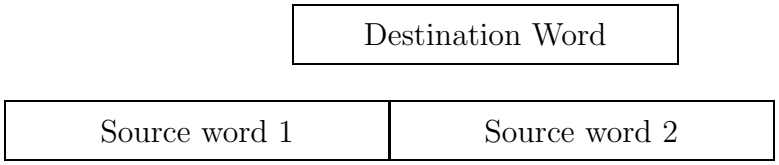

Figure 6.1: Parts of 2 source words are needed to update each destination word

Latch	Input Bits	Shifter Bits
U177	0-7	SL<0>-SL<7>
U136	8-15	SL<8>-SL<15>
U156	0-7	SL<16>-SL<23>
U116	8-14	SL<24>-SL<30>

For any given Raster operation, one of the latches (either U177 and U136, or U156 and U116) is forced into the transparent state, and the other one holds the previous source word, as was described in the previous paragraph. The control logic that achieves this will be described next.

The main timing signal used to control the half pipeline latches is the Sliver clock, which is produced by NANDing the CLK-4J and CLK-46 clocks with the CLK FIFO OUT enable signal using the 74S10 NAND gate U211c. The output of this gate is the Sliver Clock, the relative timing of which is shown on the master CPU timing diagram in chapter 4.

The latch enable signal for the low half pipeline latch (U177 and U136) is provided by the 74S10 NAND gate U211a. This gate combines the sliver clock with the LATCH ON enable signal from the Raster operation control register U190, and the SL/SR signal from the same register. In a similar way, the 74S10 NAND gate U211b controls the high half pipeline latch, but its inputs are the sliver clock, the LATCH ON signal, and the inverted signal, SL/SR'. This last signal is produced by inverting the SL/SR signal with the 74S04 inverter U208f.

This circuit operates as follows: If the LATCH ON signal is low, then the enable inputs to both halves of the pipeline latch are forced high, and the entire pipeline latch is transparent. If the LATCH ON signal is high, and the SL/SR signal is low, then U211a is still inhibited, and the low

half pipeline latch is transparent. U211b, on the other hand, is enabled and inverts the sliver clock and applies it to the enable input of the high half pipeline latch. This register is therefore updated when the sliver clock is low, which occurs during the second half of the microcycle. The operation of U221a and U211b, and of the 2 halves of the pipeline latch, is reversed when the SL/SR signal is high.

The output enable signals for the half pipeline latches are all driven by the RO/PS' signal. This signal is provided by inverting the RO/PS Raster Operation Enable signal from the Raster Operation control register by the 74S240 NOT gate U185c. The outputs of these latches are directly connected to the inputs of the barrel shifter, and thus form a distributed multiplexer with the outputs of the ALU to shifter latch described above.

The Raster operation source FIFO to shifter data path is controlled by the Raster Operation state machine described in chapter 7. During a normal raster operation, the LATCH ON signal is set high by the Raster Operation Control Register U190 and if the SL/SR signal is low, the Raster Operation is being performed left-to-right, and the low half pipeline latch is forced into the transparent state since U211a is inhibited. When the next word of the source bitmap is needed, the state machine asserts the CLK FIFO OUT signal, which enables the Sliver Clock via U211c. Therefore, during the second half of the microcycle, the high half pipeline latch is enabled via U211b, and the current source word transferred into this latch. The CLK FIFO OUT signal also enables the output clock to the FIFO chips via U203c, and therefore at the end of the microcycle, the FIFO is clocked and the next word appears at its outputs. This word passes through the low half pipeline latch, and appears at the shifter inputs. Therefore, the high bits of the shifter input consist of the pervious word, and the low bits the current word. The 2 halves of the shifter input are reversed if the SL/SR signal is high, indicating that the Raster Operation is being performed right-to-left. In either case, the most significant bit of the shifter input represents the leftmost pixel on the PERQ's display.

6.3 The Combiner

The 16 bit output of the barrel shifter described above is directly connected to a combinatorial logic circuit called the combiner. This circuit masks the shifter output before feeding it to the AMUX during an CPU shift operation, and combines the source and destination bitmaps during a raster operation.

This circuit is implemented by 8 7643 1024*4 bit CMB00 PROMs, each of which handles 2 bits of the shifter output, and which are located at the following positions on the CPU board.

Location	Bits Handled
U197	0,1
U220	2,3
U242	4,5
U241	6,7
U199	8,9
U223	10,11
U198	12,13
U222	14,15

The signals connected to U197 are listed in the next table. The other 7 CMB00 PROMs handle identical signals except for the fact that the destination, mask, and shifter output bit numbers are different.

Address	Signal	Comments
A0	ROP FN<0>'	Raster Function Select bit 0
A1	ROP FN<1>'	
A2	ROP FN<2>'	
A3	DON'T MASK	Force Raster Outputs to equal Destination
A4	DST<0>	Low bit of Destination
A5	DST<1>	
A6	MSK<0>	Low bit of Mask
A7	MSK<1>	
A8	SHOUT<0>	Low bit of shifter output (source)
A9	SHOUT<1>	

Data	Signal	Comments
D0	ROP OUT<0>	Low bit of updated bitmap
D1	ROP OUT<1>	
D2	SHIFT<0>	Low bit of shifted result to AMUX
D3	SHIFT<1>	

The ROP FN<n> lines are control inputs from the Raster Operation control registers described below which allow the source and destination bitmaps to be combined by one of 8 different functions during a Raster Operation. The Raster Operation state machine also provides the DON'T

MASK input to the combiner, which when high disables the update of the bitmap, and passes the destination word, DST<n>, unchanged to the ROP OUT <n> outputs. These control inputs have no effect on the SHIFT<n> data outputs to the AMUX.

The 16 bit mask word MSK<n> is produced by the shifter control logic described below. During arithmetic operations, the output of the shifter is logically ANDed with the mask word, so that unwanted bits in the shifter output can be forced to 0. During Raster Operations, the bits of the mask word determine whether the corresponding bit of the destination word is inside the are to be updated (when it should be combined with the relevant bit of the shifter output), or outside it (when it should be passed unchanged). The operation of the combiner for each of these uses will now be described.

6.3.1 CPU shifts

The shifter extracts a contiguous 16 bit word from the 31 bit shifter input word, and when it is used as part of the main CPU data path, this 31 bit word essentially consists of 2 copies of the ALU output R register, as described above. Therefore the shifter performs Rotate operations on the ALU result. A shift operation can, however, be performed by masking the rotated R register with a constant 16 bit word, to force the unwanted bits to 0. Such a 16 bit mask is provided by the shifter control logic described below, and the mask operation itself is performed by the combiner.

The mask word must therefore be logically ANDed with the shifter output, and the combiner proms are therefor programmed to implement the following equation:
SHIFT<n>=SHOUT<n>.MSK<n>

6.3.2 Raster Operations

During a raster operation, the combiner is controlled by the 3 function select inputs ROP FN<n>' from the Raster Operation control registers, and the DON'T MASK signal from the Raster Operation state machine. When the latter signal is high, the destination word is passed unchanged through the combiner - that is ROP OUT<n>=DST<n>
When the DON'T MASK signal is low, the combiner combines the source and destination words to produce the updated bitmap. The mask word from the shifter control logic now specifies whether or not a given bit of the destination word lies inside or outside the region to be updated. When the MSK<n> signal is 0, the corresponding DST<n> bit lies outside the area, and is therefore unchanged, but when the MSK<n> bit is 1, then the corresponding DST<n> bit is updated by combining it with the corresponding shifter output bit. The 8 possible logic functions used for this update are selected by the ROP FN<n> lines, and are listed in the following table:

ROP FN	ROP OUT<n> =	Name
0	DST<n>.!MSK<n>+SHOUT<n>.MSK<n>	Insert
1	DST<n>.!MSK<n>+!SHOUT<n>.MSK<n>	Insert Not
2	DST<n>.!MSK<n>+SHOUT<n>.DST<n>.MSK<n>	And
3	DST<n>.!MSK<n>+!SHOUT<n>.DST<n>.MSK<n>	And Not
4	DST<n>.!MSK<n>+(SHOUT<n>+DST<n>).MSK<n>	Or
5	DST<n>.!MSK<n>+(!SHOUT<n>+DST<n>).MSK<n>	Or Not
6	DST<n>.!MSK<n>+(SHOUT<n> xor DST<n>).MSK<n>	Xor
7	DST<n>.!MSK<n>+(!SHOUT<n> xor DST<n>).MSK<n>	Xnor

6.4 The Raster Operation Destination Pipeline

The 16 data outputs of the CPU memory data input register (U224 and U244) are directly connected to the D inputs of a pair of 74S374 flip-flops at locations U219 (low byte) and U221 (high byte). The clock inputs on these flip-flops are directly drive by the CLK-0R B CPU clock signal, and the outputs, which are permanently enabled, drive the DST<n> inputs on the combiner PROMs described above.

The ROP OUT<n> outputs from the combiner are latched in a further pair of 74S374 flip-flops at U243 (low byte) and U260 (high byte). These flip-flops are clocked by the CLK-0R A clock signal at the start of each microcycle. The outputs of these latches are directly connected to the MDI<n> data input lines to the memory board. The output enable pins of these flip-flops are driven by the 74S00 NAND gate U237b, which combines the CLK-4E CPU clock signal with the RO DATA RDY output from the Raster Operation state machine. Therefore, the updated word is transferred to the main memory during the second half of a microcycle when enabled by the Raster Operation state machine.

The timing of this pipeline is as follows: The first valid memory data word of a FETCH4 cycle appears at the outputs of U224 and U244 during the t2 time. This word is transferred into U219 and U221 at the start of the next microcycle, and therefore appears at their outputs during the t3 time. During this microcycle, the combiner updates the word as necessary, and the modified word is loaded into U243 and U260 at the start of the next microcycle. The modified word is therefore available at the outputs of U243 and U260 during the next t0 time, as is required by the overlapped Store4 cycle to transfer it back to the main memory.

6.5 The shifter and combiner control system

During CPU programmed shifts, the shifter and combiner are controlled by an 8 bit word previously loaded into the shifter control latches from either the ALU outputs or the microcode word Z field. During raster operations, the shifter is controlled by the Source and Destination position bits from the Raster Operation control registers and the combiner is controlled by the Destination and Width control bits, along with 2 control lines from the Raster Operation state machine. The control circuits used for these 2 different functions are largely independent, and will therefore be described separately.

6.5.1 Programmed shift control

When a microcode instruction to load the shift control word is executed (that is, one in which the F field contains 2, or one in which the F field contains 0 and the SF field contains 1), the special function decoder described in chapter 4 makes the LD SHIFT COMMAND signal high. This signal is connected to the enable inputs of a pair of 67S380 transparent latches at locations U127 and U128.

The data inputs of U127 are directly connected to the microcode word Z field, while the ones of U128 are connected to the ALU outputs, ALU Y<n>. Therefore, these latches are loaded with the 2 possible shifter control words. The corresponding 3-state outputs of the 2 latches are connected together, and they thus form a distributed multiplexer to allow either source of the control word to be selected.

The LD SHIFT COMMAND signal is also connected to the clock input of the 74S74 D-type flip-flop U147a, while the D input of this flip-flop is connected to the SH TYPE output from the AMUX16 control PROM described in chapter 2. This signal goes high when the microcode word F field contains 2, and thus the flip-flop is set if a shift-on-Z operation is to be performed, and cleared when a shift-on-R operation is required. The Q output of U147a is connected to the active-low output enable input on U128, while the output enable pin of U127 is driven by the !Q output. Hence, during a shift-on-R operation, U127 is disabled, and U128 is enabled, so the shifter control word is provided by the ALU Y<n> bits. When a shift-on-Z instruction is executed, U128 is disabled and U127 enabled, and the shifter is controlled by the latched microcode word Z field.

The outputs of these latches, the SHIFT CMD<n> signals, are connected to the address inputs of a 27S29 PSF00 control PROM at location U129. This PROM is programmed to provide the 4-bit shifter control word required for each of the 256 possible programmed shifter operations. The signals connected to this PROM are listed in the following table.

Address	Signal
A0	SHIFT CMD<7>
A1	SHIFT CMD<6>
A2	SHIFT CMD<5>
A3	SHIFT CMD<4>
A4	SHIFT CMD<3>
A5	SHFIT CMD<2>
A6	SHIFT CMD<1>
A7	SHIFT CMD<0>
A8	Ground

Data	Signal	Comments
D0	RAW SHIFT CNTRL<3>	
D1	RAW SHIFT CNTRL<2>	LSB of Shifter Control
D2	RAW SHIFT CNTRL<1>	MSB of Shifter Control
D3	RAW SHIFT CNTRL<0>	

The output enable signal of the PSF00 ROM is connected to ground, so the outputs are always active. These outputs are connected to the D inputs of the 74F373 transparent latch U148a-d. This latch is enabled by the CLK-4E CPU clock signal, which ensures that the SHIFT CNTRL<n> can only change during the second half of a microcycle, and in particular, if the shifter is used during the ALU path of an instruction that loads the Shifter Command word, then that ALU operation will use the old Shifter Command. The outputs of U148a-d are enabled by the RO/PS signal, and so are enabled during programmed shift, and disabled during Raster Operations. In fact, they form a distributed multiplexer with the outputs of the RSF00 ROM described below, which provides the shifter control word during raster operations. These outputs, the SHIFT CTRL<n> signals, are directly connected to the control inputs of the barrel shifter chips.

The SHIFT CMD<n> signals are also connected to the D inputs of the 74F373 transparent latch U126. This latch is also enabled by the CLK-4E clock, to prevent the mask control word changing during the first half of the microcycle. The output enable pin of U126 is also driven by the RO/PS signal, and the outputs form a distributed multiplexer with the data outputs of the RSH00 PROM described below. These outputs, the LM SHIFT<n> signals, are connected to the address inputs of the mask control PROMs MSK40 (U179) and MSK50 (U180), and thus determine the mask word to be fed to the combiner.

The mask control PROMs are 512*8 bit 27S29s, and together they provide the 16-bit MSK<n>

word used by the combiner. The highest address line, A8, of these PROMs is connected to the RO/PS line, and thus the first half of each PROM is used for programmed shift masking, and the second half for Raster Operation control. The remaining address inputs are driven by the LM SHIFT<n> signals described in the last paragraph. The connections to the mask PROMs are given in the following table.

Address	Signal
A0	LM SHIFT<7>
A1	LM SHIFT<6>
A2	LM SHIFT<5>
A3	LM SHIFT<4>
A4	LM SHIFT<3>
A5	LM SHIFT<2>
A6	LM SHIFT<1>
A7	LM SHIFT<0>
A8	RO/PS

Data	Signal
MSK40 (U179)	
D0	MSK<7>
D1	MSK<6>
D2	MSK<5>
D3	MSK<4>
D4	MSK<3>
D5	MSK<2>
D6	MSK<1>
D7	MSK<0>

MSK50 (U180)	
D0	MSK<15>
D1	MSK<14>
D2	MSK<13>
D3	MSK<12>
D4	MSK<11>
D5	MSK<10>
D6	MSK<9>
D7	MSK<8>

6.5.1.1 Programmed Shift Commands

The Shift Control and Mask words for each of the 256 values of the shifter command word are given in the following array. For each possible shifter command word, the upper number is the (decimal) value applied to the shifter control lines (that is, the number of bits by which the ALU result is shifted), and the lower number is the (hexadecimal) 16 bit mask word applied to the combiner.

High Nybble ↓	Low Nybble → x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	3FFF	7FFF	FFFF
1x	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	15
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	3FFF	7FFF	FFFE
2x	2	2	2	2	2	2	2	2	2	2	2	2	2	2	0	14
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	3FFF	7FFF	FFFC
3x	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0	13
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	3FFF	7FFF	FFF8
4x	4	4	4	4	4	4	4	4	4	4	4	4	0	0	0	12
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	3FFF	7FFF	FFF0
5x	5	5	5	5	5	5	5	5	5	5	5	0	0	0	0	11
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	3FFF	7FFF	FFE0
6x	6	6	6	6	6	6	6	6	6	6	0	0	0	0	0	10
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	3FFF	7FFF	FFC0
7x	7	7	7	7	7	7	7	7	7	0	0	0	0	0	0	9
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	3FFF	7FFF	FF80
8x	8	8	8	8	8	8	8	8	0	0	0	0	0	0	8	8
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	FFFF	FFFF	FF00
9x	9	9	9	9	9	9	9	0	0	0	0	0	0	1	9	7
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	FFFF	FFFF	FE00
Ax	10	10	10	10	10	10	0	0	0	0	0	0	0	2	10	6
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	FFFF	FFFF	FC00
Bx	11	11	11	11	11	0	0	0	0	0	0	0	0	3	11	5
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	FFFF	FFFF	F800
Cx	12	12	12	12	0	0	0	0	0	0	0	0	0	4	12	4
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	FFFF	FFFF	F000
Dx	13	13	13	0	0	0	0	0	0	0	0	0	0	5	13	3
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	FFFF	FFFF	E000
Ex	14	14	0	0	0	0	0	0	0	0	0	0	0	6	14	2
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	FFFF	FFFF	C000
Fx	15	0	0	0	0	0	0	0	0	0	0	0	0	7	15	1
	0001	0003	0007	000F	001F	003F	007F	00FF	01FF	03FF	07FF	0FFF	1FFF	FFFF	FFFF	8000

The upper left triangle of this array (00-F0,01-E1, ..., 0E-1E) define the bit-field operations. The low nybble specifies the size of the extracted bit field (0 specifies a 1-bit field, up to E, which specifies a 15 bit field), and the upper nybble is the bit-position of the least significant bit of the extracted bit-field. The word is rotated by the appropriate amount, and then anded with a mask word containing the appropriate number of 1's in the least significant bits.

The rightmost column, (0F-FF), specifies left shift operations. The high nybble of the shifter command word determines the number of bits by which the data word will be shifted. The input data word is rotated by the appropriate amount, and the low bits are masked out by combining them with 0s.

Rotates are performed by the shifter command words 8D-FD and 8E-FE. The shifter naturally performs a rotate operation, so the result need not be masked, and therefore the mask word is set to FFFF.

Right Shifts are specified by the shifter command words lying on the trailing diagonal of the array, 0F-F0. The word is again rotated by the appropriate amount by the shifter, and then

ANDed with a word containing the appropriate number of 1's in the least significant locations. These operations, are of course, identical to certain of the bit-field operations described above, and are specified by identical command words.

6.5.2 Raster Operation Shift Control

During Raster Operations, the shifter is controlled by the Source and Destination position control nybbles (SRC BIT<n> and DST BITS<n>) from the Raster Operation control registers. These signals are connected to the address inputs of the 27S29 512*8bit RSF00 PROM at location U166 as indicated in the following table. The lower 4 data outputs of this PROM are connected to the SHIFT CNTL<n> shifter control lines, and form a distributed multiplexer with the outputs of U148a-d described above

Address	Signal	Comments
A0	DST BITS<3>	Destination bit position
A1	DST BITS<2>	
A2	DST BITS<1>	
A3	DST BITS<0>	
A4	SRC BIT<3>	Source bit position
A5	SRC BIT<2>	
A6	SRC BIT<1>	
A7	SRC BIT<0>	
A8	Ground	

Data	Signal	Comments
D0	SHIFT CNTL<3>	Shifter Control
D1	SHIFT CNTL<2>	
D2	SHIFT CNTL<1>	
D3	SHIFT CNTL<0>	
D4	Not Used	
D5	Not Used	
D6	Not Used	
D7	Not Used	

The output enable pin of the RSF00 PROM is driven by the inverted RO/PS signal, RO/PS'. The outputs are therefore enabled during raster operations, and this PROM controls the shifter.

Since the RO/PS signal is high, the outputs of U148a-d are disabled, and therefore the CPU shifter control circuit has no effect on the shifter.

The RO/PS' signal is also connected to both the set and reset inputs of the shifter type latch U147a. Therefore, during Raster Operations, both the Q and !Q outputs of this latch are forced high, and both U127 and U128 are disabled. However, since no part of the Raster Operation control logic drives the SHIFT CMD<n> lines, the reason for this feature is unknown.

The mask word required for a Raster Operation depends on the Destination and Width control nybbles (DST<n> and WID<n>) from the Raster Operation Control Registers, together with a pair of control signals, EL and ER from the Raster Operation state machine. The latter signals indicate if there is an edge of the region to be updated in the current 16 bit word, as indicated by the following table.

ER	EL	
0	0	Word contains Both edges of updated region
0	1	Word contains Left edge
1	0	Word contains Right edge
1	1	Word contains No edges

The position of the edge within the word is determined by the value of the DST<n> and WID<n> nybbles.

These 10 signals are connected to the address pins of a 1024*8 bit 29S86 RSH00 PROM at location U174. The 3-state date outputs of this PROM are connected to the LM SHIFT<n> lines, forming a distributed multiplexer with the outputs of U126, and thus drive the address inputs of the MSK40 and MSK50 PROMs. The signals connected to the RSH00 PROM are listed in the next table.

Address	Signal	Comments
A0	EL	Left Edge
A1	ER	Right Edge
A2	WID BITS<0>	Width control Nybble
A3	WID BITS<1>	
A4	WID BITS<2>	
A5	WID BITS<3>	
A6	DST BITS<0>	Destination Position Nybble
A7	DST BITS<1>	
A8	DST BITS<2>	
A9	DST BITS<3>	

Data	Signal	Comments
D0	LM SHIFT CMD<0>	Mask Control
D1	LM SHIFT CMD<1>	
D2	LM SHIFT CMD<2>	
D3	LM SHIFT CMD<3>	
D4	LM SHIFT CMD<4>	
D5	LM SHIFT CMD<5>	
D6	LM SHIFT CMD<6>	
D7	LM SHIFT CMD<7>	

The data lines of this PROM are directly connected to the address inputs (A0-A7) of the MSK40 and MSK50 PROMs described above. The output enable signal of the RSH00 ROM is driven by the RO/PS' signal, so that the outputs are only enabled during Raster Operations. Since the A8 line of the 2 mask PROMs is driven by the RO/PS signal, the second half of these PROMS is used for Raster Operations, while the first half is used for programmed shifts.

6.5.2.1 Raster Operation Shift Commands

The number of bits by which the source bit-map words must be shifted to align them with the destination is simply the difference between the SRC BIT<n> and DST BIT<n> control nybbles. Therefore, the RSF00 PROM is programmed to act as a 4-bit subtracter, as shown in this table.

SRC<n>→

DST<n> ↓	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12	13
3	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11	12
4	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10	11
5	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9	10
6	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8	9
7	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7	8
8	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6	7
9	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5	6
A	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4	5
B	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3	4
C	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2	3
D	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1	2
E	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1
F	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0

The particular mask word used depends on the state of the EL and ER signals, and the 4 cases will be described separately. For each value of the DST<n> and WID<n> nybbles, the table indicates the value of the LM SHIFT<n> lines, and below it, the mask word sent to the combiner.

The case when both EL and ER are low occurs when one 16 bit word contains both edges of the region to be updated. The DST<n> nybble then specifies the position of the left edge (most significant bit) and the complement of the WID<n> nybble is one less than the number of bits to update. This is illustrated by the following table

WID<n>→

DST<n> ↓	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	00
	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001
1	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	10	11
	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0002
2	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	20	21	22
	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007	0006	0004
3	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	30	31	32	33
	000F	000F	000F	000F	000F	000F	000F	000F	000F	000F	000F	000F	000F	000E	000C	0008
4	45	46	47	48	49	4A	4B	4C	4D	4E	4F	40	41	42	43	44
	001F	001F	001F	001F	001F	001F	001F	001F	001F	001F	001F	001F	001E	001C	0018	0010
5	56	57	58	59	5A	5B	5C	5D	5E	5F	50	51	52	53	54	55
	003F	003F	003F	003F	003F	003F	003F	003F	003F	003F	003F	003E	003C	0038	0030	0020
6	67	68	69	6A	6B	6C	6D	6E	6F	60	61	62	63	64	65	66
	007F	007F	007F	007F	007F	007F	007F	007F	007F	007F	007E	007C	0078	0070	0060	0040
7	78	79	7A	7B	7C	7D	7E	7F	70	71	72	73	74	75	76	77
	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FE	00FC	00F8	00F0	00E0	00C0	0080
8	89	8A	8B	8C	8D	8E	8F	80	81	82	83	84	85	86	87	88
	01FF	01FF	01FF	01FF	01FF	01FF	01FF	01FF	01FE	01FC	01F8	01F0	01E0	01C0	0180	0100
9	9A	9B	9C	9D	9E	9F	90	91	92	93	94	95	96	97	98	99
	03FF	03FF	03FF	03FF	03FF	03FF	03FF	03FE	03FC	03F8	03F0	03E0	03C0	0380	0300	0200
A	AB	AC	AD	AE	AF	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA
	07FF	07FF	07FF	07FF	07FF	07FF	07FE	07FC	07F8	07F0	07E0	07C0	0780	0700	0600	0400
B	BC	BD	BE	BF	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB
	0FFF	0FFF	0FFF	0FFF	0FFF	0FFE	0FFC	0FF8	0FF0	0FE0	0FC0	0F80	0F00	0E00	0C00	0800
C	CD	CE	CF	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC
	1FFF	1FFF	1FFF	1FFF	1FFE	1FFC	1FF8	1FF0	1FE0	1FC0	1F80	1F00	1E00	1C00	1800	1000
D	DE	DF	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD
	3FFF	3FFF	3FFF	3FFE	3FFC	3FF8	3FF0	3FE0	3FC0	3F80	3F00	3E00	3C00	3800	3000	2000
E	EF	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE
	7FFF	7FFF	7FFE	7FFC	7FF8	7FF0	7FE0	7FC0	7F80	7F00	7E00	7C00	7800	7000	6000	4000
F	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0	FF80	FF00	FE00	FC00	F800	F000	E000	C000	8000

When the EL signal is high and the ER signal is low, the current word contains the left edge of the region to update. The DST<n> nybble specifies the location of this edge, and the WID<n> has no effect, as show in this table

WID<n>→

DST<n> ↓	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001	0001
1	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003	0003
2	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007	0007
3	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
	000F	000F	000F	000F	000F	000F	000F	000F	000F	000F	000F	000F	000F	000F	000F	000F
4	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
	001F	001F	001F	001F	001F	001F	001F	001F	001F	001F	001F	001F	001F	001F	001F	001F
5	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
	003F	003F	003F	003F	003F	003F	003F	003F	003F	003F	003F	003F	003F	003F	003F	003F
6	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60
	007F	007F	007F	007F	007F	007F	007F	007F	007F	007F	007F	007F	007F	007F	007F	007F
7	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70	70
	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FF	00FF
8	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80	80
	01FF	01FF	01FF	01FF	01FF	01FF	01FF	01FF	01FF	01FF	01FF	01FF	01FF	01FF	01FF	01FF
9	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90	90
	03FF	03FF	03FF	03FF	03FF	03FF	03FF	03FF	03FF	03FF	03FF	03FF	03FF	03FF	03FF	03FF
A	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0	A0
	07FF	07FF	07FF	07FF	07FF	07FF	07FF	07FF	07FF	07FF	07FF	07FF	07FF	07FF	07FF	07FF
B	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0
	0FFF	0FFF	0FFF	0FFF	0FFF	0FFF	0FFF	0FFF	0FFF	0FFF	0FFF	0FFF	0FFF	0FFF	0FFF	0FFF
C	C0	C0	C0	C0	C0	C0	C0	C0	C0	C0	C0	C0	C0	C0	C0	C0
	1FFF	1FFF	1FFF	1FFF	1FFF	1FFF	1FFF	1FFF	1FFF	1FFF	1FFF	1FFF	1FFF	1FFF	1FFF	1FFF
D	D0	D0	D0	D0	D0	D0	D0	D0	D0	D0	D0	D0	D0	D0	D0	D0
	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
E	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0	E0
	7FFF	7FFF	7FFF	7FFF	7FFF	7FFF	7FFF	7FFF	7FFF	7FFF	7FFF	7FFF	7FFF	7FFF	7FFF	7FFF
F	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF

If the ER signal is high, and the EL signal is not, then the right edge of the region to be updated is contained in the current word. The width of the region (modulo 16) is equal to the complement of the WID<n> nybble plus 1, so the sum of the number of 1s in the corresponding elements of this table and the last is equal to that value.

DST<n> ↓	WID<n>→ 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F0
	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0	FF80	FF00	FE00	FC00	F800	F000	E000	C000	8000	FFFF
1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F0	F1
	FFFC	FFF8	FFF0	FFE0	FFC0	FF80	FF00	FE00	FC00	F800	F000	E000	C000	8000	FFFF	FFFE
2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F0	F1	F2
	FFF8	FFF0	FFE0	FFC0	FF80	FF00	FE00	FC00	F800	F000	E000	C000	8000	FFFF	FFFE	FFFC
3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F0	F1	F2	F3
	FFF0	FFE0	FFC0	FF80	FF00	FE00	FC00	F800	F000	E000	C000	8000	FFFF	FFFE	FFFC	FFF8
4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F0	F1	F2	F3	F4
	FFE0	FFC0	FF80	FF00	FE00	FC00	F800	F000	E000	C000	8000	FFFF	FFFE	FFFC	FFF8	FFF0
5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F0	F1	F2	F3	F4	F5
	FFC0	FF80	FF00	FE00	FC00	F800	F000	E000	C000	8000	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0
6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F0	F1	F2	F3	F4	F5	F6
	FF80	FF00	FE00	FC00	F800	F000	E000	C000	8000	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0
7	F8	F9	FA	FB	FC	FD	FE	FF	F0	F1	F2	F3	F4	F5	F6	F7
	FF00	FE00	FC00	F800	F000	E000	C000	8000	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0	FF80
8	F9	FA	FB	FC	FD	FE	FF	F0	F1	F2	F3	F4	F5	F6	F7	F8
	FE00	FC00	F800	F000	E000	C000	8000	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0	FF80	FF00
9	FA	FB	FC	FD	FE	FF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9
	FC00	F800	F000	E000	C000	8000	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0	FF80	FF00	FE00
A	FB	FC	FD	FE	FF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA
	F800	F000	E000	C000	8000	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0	FF80	FF00	FE00	FC00
B	FC	FD	FE	FF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB
	F000	E000	C000	8000	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0	FF80	FF00	FE00	FC00	F800
C	FD	FE	FF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC
	E000	C000	8000	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0	FF80	FF00	FE00	FC00	F800	F000
D	FE	FF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD
	C000	8000	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0	FF80	FF00	FE00	FC00	F800	F000	E000
E	FF	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE
	8000	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0	FF80	FF00	FE00	FC00	F800	F000	E000	C000
F	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
	FFFF	FFFE	FFFC	FFF8	FFF0	FFE0	FFC0	FF80	FF00	FE00	FC00	F800	F000	E000	C000	8000

Finally, if both EL and ER are high, the current word lies completely inside the region to be updated, so all the bits must be updated. Therefore a constant mask value of FFFF (hex) is used, as given by the following table.

WID<n>→

DST<n> ↓	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
1	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
2	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
3	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
4	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
5	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
6	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
7	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
8	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
9	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
A	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
B	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
C	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
D	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
E	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
F	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0	F0
	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF

6.6 The Raster Operation Control Registers

The Raster Operation system is controlled by 4 control registers which are loaded by certain microcode instructions. The control signals produced by each register will now be described. The operation of the Raster Operation state machine, and the way in which these signals control it, is described in chapter 7.

6.6.1 The Control Register

This register consists of a 74S273 octal D-type flip-flop at location U190. The clock input of this chip is directly driven from the LD C output of the special function decoder described in chapter 4. The D-inputs of the first 7 flip-flops are connected to the lower 7 microcode word Z field lines, and the reset input is connected to the INIT' signal so that the register is cleared when the CPU is reset. The bits stored in this register have the following meanings.

Latch	Z-bit	Output Signal	Comments
U190a	Z<6>	LATCH ON	Enable half pipeline latches
U190b	Z<5>	PHASE<4>	Enable Extra Source Word Fetch
U190c	Z<4>	PHASE<2>	Raster Operation State Machine Control
U190d	Z<3>	PHASE<1>	Raster Operation State Machine Control
U190e	Z<2>	PHASE<0>	Raster Operation State Machine Control
U190f	Z<1>	RO/PS	Enable Raster Operations
U190g	Z<0>	SL/SR	Raster Operation Direction Select

The SL/SR signal is inverted by the 74S04 NOT gate U208f to produce the SL/SR' signal which is used by the half pipeline latch control circuit described above. Similarly, the RO/PS signal is inverted by the permanently enabled 74S240 NOT gate U185c to produce the RO/PS' signal that is used to disable the programmed shift system when Raster Operations are taking place.

6.6.2 The Width Register

The Width register again consists of a 74S273, this time at location U168. Six of the D-inputs are connected to the lower 6 R-register outputs, while R<6>' is inverted by U192c (74S240) and then connected to U168b's D-input. Similarly, R<7> is inverted by U192b (again a 74S240 NOT gate), and connected to the D-input of U192a. The register is loaded by a rising edge of the LD W signal from the special function decoder described in chapter 4, and, since the reset pin is driven by the INIT' signal, the register is cleared when the CPU is reset. The 8 outputs of this register are used to define the width of the updated region (modulo 64), and also to control the multiplication/division hardware described in chapter 3. The following signals are produced by this register.

Latch	R-bit	Output Signal	Comments
U168a	R<7>	MD INSTR<1>	Multiplication control
U168b	R<6>	MD INSTR<0>	
U168c	R<5>'	WID WRD<1>'	Number of extra 16 bit words in width
U168d	R<4>'	WID WRD<0>'	
U168e	R<3>'	WID BITS<3>	Number of extra bits in width
U168f	R<2>'	WID BITS<2>	
U168g	R<1>'	WID BITS<1>	
U168h	R<0>'	WID BITS<0>	

6.6.3 The Destination Register

The destination register is the 74S273 register at location U164. The D-inputs of this device are directly connected to the outputs of the ALU R register (R<n>), while the reset input is connected to the INIT' line. The register is clocked by the rising edge of the LD D output from the special function decoder described in chapter 4. The outputs of this register provide the following signals:

Latch	R-bit	Output Signal	Comments
U164a	R<7>'	ROP FN<1>'	Combiner Function Select
U164b	R<6>'	ROP FN<0>'	
U164c	R<5>'	DST WRD<1>'	Destination 16 bit word position
U164d	R<4>'	DST WRD<0>'	in 64 bit word
U164e	R<3>'	DST BITS<3>	Destination bit position in 16 bit word
U164f	R<2>'	DST BITS<2>	
U164g	R<1>'	DST BITS<1>	
U164h	R<0>'	DST BITS<0>	

6.6.4 The Source Register

The source register is the 74S273 flip-flop at location U167. The signals that drive the inputs of this chip are identical to those that drive the destination register except for the fact that it is clocked by the LD S output from the special function decoder. The outputs of this register provide the following control signals to the Raster Operation Hardware.

Latch	R-bit	Output Signal	Comments
U167a	R<7>'	OFF	Turn Off PERQ 1
U167b	R<6>'	ROP FN<2>'	Combiner Function Select
U167c	R<5>'	SRC WRD<1>'	Source 16 bit word position
U167d	R<4>'	SRC WRD<0>'	in 64 bit word
U167e	R<3>'	SRC BIT<3>	Source bit position in 16 bit word
U167f	R<2>'	SRC BIT<2>	
U167g	R<1>'	SRC BIT<1>	
U167h	R<0>'	SRC BIT<0>	

The output of U167a is logically NANDed with the INIT' signal by the 74S00 NAND gate U232b, and the output of this gate is connected to pin J154 (PWR DOWN) on the CPU backplane connector. This signal is not used on any PERQ2 series machine, but on the PERQ 1, when the output of U167a goes high, and the CPU is not being reset, the PWR DOWN pin goes low and operates a relay in the power supply that turns off the machine.

Chapter 7

The Memory and Raster Operation State Machines

A finite state machine (often shortened to 'state machine') is simply a digital system that is internally in one of a finite number of states. On each clock pulse, the system may change to a different internal state, and the particular state it goes into depends on the current state and on the values of external input signals. The internal state may be decoded and used to control external circuitry.

A state machine is usually implemented by a number of D-type flip-flops (n flip-flops allow a maximum of 2^n states), which are all clocked by the clock pulses. A combinatorial logic circuit then drives the D-inputs on these flip-flops depending on the state of the Q-outputs (that is, the current state), and the external signals. Thus, this combinatorial circuit calculates the new state based on the current state and the inputs, and when the flip-flops are clocked, the internal state is updated. A further combinatorial circuit accepts inputs from the Q-outputs and produces control signals for the rest of the system.

In the PERQ 16K CPU board, such state machines are used to control the memory cycles and the Raster Operation data path. In both cases, the combinatorial logic circuit is provided by specially programmed PROM. These 2 state machines will now be described.

7.1 The Memory Control State Machine

The Memory control state machine follows the general scheme just described, and the only unconventional feature is the way in which the external control inputs are applied to the system.

The 4-bit internal state of this state machine is knows as the BOOKMARK value, and is stored by the 74S273 flip-flops U227e-h. These flip-flops are clocked by the rising edge of the CLK-0R

D signal at the start of each microcycle, while they are reset to 0 by the INIT' signal, which is connected to the CLR input of U227, when the machine is reset. The outputs of these 4 flip-flops form the top 4 address lines to the state machine PROMs, as described below, while the D-inputs are provided by 4 of the data outputs of the BKM16.2 state machine PROM U173. This circuit forms the main memory control state machine.

The remaining 4 flip-flops in U227 are used to hold 4 control signals from the other 2 PCBs in the PERQ system, namely the microcode t-state counter from the memory board and the DMA control signals from the EIO board. These signals are listed in the following table.

Flip-Flop	Input	Output	Comments
U227d	TIME<0>'	L TIME<0>'	T-state counter
U227c	TIME<1>'	L TIME<1>'	T-state counter
U227b	I/O MEM WR	I/O WANTS WR/¿DMA Write	
U227a	I/O MEM RQST	MEM RQST L	DMA Request

The L TIME<n> signals are connected to 2 of the address inputs on the state machine control PROMs, as described below. The DMA control signals form 2 of the external control inputs to the state machine, and that section will be described next.

A memory-control microcode instruction is one in which the F field contains 1 and the SF field has the high bit (SF<3>) set. This condition is detected by the 74S10 NAND gate U215a, which combines the SF<3>, F<1>' and F<0> signals. The output of this gate therefore goes low when such an instruction is executed.

The ROMs used in the Memory control state machine contain 1024 words each, and therefore have 10 address lines. Six of these lines are used for the BOOKMARK and L TIME<n>' signals, leaving 4 for the external control inputs. However, 6 external control signals need to be applied to the state machine, namely the 2 DMA control signals, the memory instruction output from U215a, and the remaining 3 SF signals which indicate which particular memory instruction is to be executed. Therefore, these signals are multiplexed onto the 4 remaining address lines by U214, a 74LS158 quad 2-input multiplexer.

The inverted outputs of U214 are directly connected to the lower 4 address lines of the 2 state machine control PROMs, while the Enable input is connected to ground, so that the data flow through the multiplexer is permanently enabled. The select input is driven by the L TIME<1>' signal described above, so that the 2 sets of inputs are selected alternately for 2 microcycles each. The A-inputs, selected when L TIME<1>' is low, are the CPU memory cycle control signals, while the B-inputs, selected when L TIME<1>' is high, are the DMA control signals. These signals are listed in the following table.

Input	Signal	Comments
A1	SF<0>	Microcode word SF signal
A2	SF<1>	
A3	SF<2>	
A4	Mem Instruction	Output of U215a
B1	IO WANTS WR	Latched DMA Write
B2	GATED MEM RQST	Gated DMA Request
B3	STORE	Output of U215c - low when store instruction
B4	Mem Instruction	As above

The STORE signal is produced by U215c (a 74S10 NAND gate), which combines the 3 lower SF field bits, SF<0>-SF<2>. This signal therefore goes active when a Store memory instruction is executed, which causes this instruction to have a slightly different cycle timing to the other instructions, as described below.

The MEM RQST L DMA signal output from U227a is NANDed with the inverted microcode H field, H', by the 74S00 NAND gate U237a, and the output of this gate is connected to the B2 input of the multiplexer U214. Therefore, when the H field contains 1, H' is low, U237a is inhibited, and no DMA transfers can occur.

The control logic for the state machine is logically provided by a 1024*12bit ROM, consisting of the 1K*8 bit 28S86 PROM BKM16.2 at location U173 and the 1K*4 bit 7643 PROM GMV02 at location U204. Corresponding address lines of these 2 PROMs are connected together, so that the PROMs act as the required 12-bit wide unit. The Chip Select signals of the PROMs are permanently connected to either ground or the +3B signal (pulled high via R3), so that the PROMs are always enabled. The signals connected to the state machine ROM are listed in the following table.

Address	Signal	Comments
A0	CTRL<0>	Multiplexed Control Inputs
A1	CRTL<1>	(Outputs of U214)
A2	CTRL<2>	
A3	CTRL<3>	
A4	L TIME<0>	Latched T-state counter
A5	L TIME<1>	
A6	BOOKMARK<0>	State Latch output
A7	BOOKMARK<1>	
A8	BOOKMARK<2>	
A9	BOOKMARK<3>	

BKM16.2

Data	Signal	Comments
D0	RAS NOW	Memory board Control Strobe
D1	MEM RQST ST<2>	Memory Operation Select
D2	MEM RQST ST<1>	
D3	MEM RQST ST<0>	
D4	NEW BOOKMARK<0>	Input to State Latch
D5	NEW BOOKMARK<1>	
D6	NEW BOOKMARK<2>	
D7	NEW BOOKMARK<3>	

GMV02

Data	Signal	Comments
D0	MDI Valid	High if memory data invalid
D1	Wait	High to abort microcycle
D2	MEM RDING MA/MD	Enable signal to CPU buffers
D3	GRANT	DMA Grant

The NEW BOOKMARK<n> outputs of the BKM16.2 PROM are directly connected to the D-inputs of the state latch U227e-h, and determine the next state of the state machine. The MEM RQST ST<n> signals, which determine the particular type of memory cycle to be performed, are connected to the Memory PCB via J181-J183 on the CPU backplane connector. Similarly, the RAS NOW strobe, which starts a memory cycle, is fed to the Memory PCB via J184.

The GRANT output from the GMV02 PROM is connected to the D-input of the 67S380 inverting transparent latch U165h. The output of this latch is permanently enabled since the Output Enable pin is grounded, and the strobe input is driven by the CLK-0R C CPU clock signal described in chapter 4. Therefore, the output of this latch is stable during the second half of the microcycle. This output is re-inverted by the 74S240 NOT gate U254e, which is also permanently enabled, and the output of this gate, the GRANT DMA signal, is connected to the EIO board via J171.

The MEM RDING MA/MD output is used to disable the address and data buffers during DMA operations. It is connected to one input of the 74S00 NAND gate U232d, which drives the output enable pins of the memory address latches U255,U257 and U257, and also to one input of the 74S10 NAND gate U213c, which controls the output enable pins of the memory data output latches U258 and U259. Therefore, when the MEM RDING MA/MD signal is low, both U232d and U213c are inhibited, and the outputs of the address and data latches are forced to the high impedance state. The operation of these latches is described in more detail in chapter 3.

A CPU microcycle should be aborted if either the memory system is waiting for the correct t-state to start the memory cycle in - that is when the WAIT output is high, or if the processor is reading the memory data via the AMUX (that is, the PROC NEEDS MDI output from the AMUX control PROM is high), and the MDI lines do not contain valid data - the MDI VALID L line is also high. These conditions are detected by the 74S51 AND-OR-INVERT gate U205a, one side of which ANDs the PROC NEEDS MDI with the MDI VALID L signal, while the other side has both inputs driven by the WAIT signal. The output of this gate, the P ABORT signal, is therefore low during any memory cycle that should be aborted. This active-low signal is logically ORed with the HOLD OFF L signal from the EIO DMA controller by the 74S08 AND gate U231a to provide the ABORT' signal that inhibits the execution of the next microinstruction as described in earlier chapters.

7.1.1 CPU Memory Cycles

The CPU can perform 8 different memory cycles, which are selected by the lower 3 bits of the SF microcode word field, as mentioned above. These cycles, the SF value needed to produce one, and the value sent to the memory board via the MEM RQST ST lines are listed in the following table

Cycle	SF	MEM RQST ST
FETCH4R	10	2
STORE4R	11	6
FETCH4	12	1
STORE4	13	5
FETCH2	14	3
STORE2	15	7
FETCH	16	0
STORE	17	4

The timing of the control signals for these memory cycles will now be described.

7.1.1.1 Fetch Cycles

The 4 different Fetch cycles all have very similar timing. When the Memory cycle is started (i.e. when the output of U215a goes low), the memory control state machine asserts the Wait signal until the next t3 state, so that subsequent microcycles are aborted. At the end of the next T3 state, the LD MIR' signal rises as usual, and the next microcode instruction is executed. The RAS NOW and MEM RQST ST signals are asserted during this T3 state, and the next 2 microcycles are aborted if the Processor tries to read the MDI lines via the AMUX since the MDI VALID L signal is High. At the next T2 state, the memory data input lines are valid, and may be read into the data path via the AMUX.

The operation of the Fetch Cycles is described by the 4 following timing charts, which show the relative timing of the MEM RQST ST, RAS, WAIT and MDI VALID L signals.

FETCH4R

Time	F	SF	Bookmark	ras	Mem RQST	Wait	MDI Valid
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	8	0	0	0	1	0
2	1	8	0	0	0	1	0
3	1	8	6	1	2	0	0
0	1	8	6	0	0	0	1
1	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0

FETCH4

Time	F	SF	Bookmark	ras	Mem RQST	Wait	MDI Valid
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	10	0	0	0	1	0
2	1	10	0	0	0	1	0
3	1	10	5	1	1	0	0
0	1	10	5	0	0	0	1
1	0	0	0	0	0	0	1
2	0	0	0	0	0	0	0

FETCH2

Time	F	SF	Bookmark	ras	Mem RQST	Wait	MDI Valid
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	12	0	0	0	1	0
2	1	12	0	0	0	1	0
3	1	12	4	1	3	0	0
0	1	12	4	0	3	0	1
1	0	0	4	0	0	0	1
2	0	0	0	0	0	0	0

FETCH

Time	F	SF	Bookmark	ras	Mem RQST	Wait	MDI Valid
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	14	0	0	0	1	0
2	1	14	0	0	0	1	0
3	1	14	3	1	0	0	0
0	1	14	3	0	0	0	1
1	0	0	3	0	0	0	1
2	0	0	0	0	0	0	0

7.1.1.2 Multiple-Word Store Cycles

Again, the timing of the STORE4R, STORE4 and STORE2 memory cycles is very similar. The WAIT signal is asserted during the microcycles between the memory instruction being executed and the next T3 state, so that these cycles are aborted, and the next microcode instruction is loaded into the pipeline at the start of the next T0 state. This instruction provides the first word of data

to the memory board, as described in the microprogramming manual. The MEM RQST ST code is asserted during the next T1 and T2 states, and finally, the RAS NOW signal becomes high during the subsequent T3 state, thus starting the memory cycle. The following timing diagrams show the relative sequence of these signals.

STORE4R

Time	F	SF	Bookmark	ras	Mem RQST	Wait	MDI Valid
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	9	0	0	0	1	0
2	1	9	0	0	0	1	0
3	1	9	8	0	0	0	0
0	1	9	8	0	0	0	0
1	0	0	8	0	6	0	0
2	0	0	8	0	6	0	0
3	0	0	0	1	0	0	0

STORE4

Time	F	SF	Bookmark	ras	Mem RQST	Wait	MDI Valid
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	11	0	0	0	1	0
2	1	11	0	0	0	1	0
3	1	11	7	0	0	0	0
0	1	11	7	0	0	0	0
1	0	0	7	0	5	0	0
2	0	0	7	0	5	0	0
3	0	0	0	1	0	0	0

STORE2

Time	F	SF	Bookmark	ras	Mem RQST	Wait	MDI Valid
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	13	0	0	0	1	0
2	1	13	0	0	0	1	0
3	1	13	9	0	7	0	0
0	1	13	9	0	7	0	0
1	0	0	9	0	7	0	0
2	0	0	9	0	7	0	0
3	0	0	0	1	0	0	0

7.1.1.3 Single-Word STORE Cycle

The STORE single-word write cycle has a different timing sequence to all other memory cycles. When the appropriate instruction is executed, the memory control state machine assets the WAIT signal until the next T2 state, and during this T2 state, the MEM RQST ST lines carry the value '4', the code for a STORE instruction. At the end of the T2 state, the LD MIR' signal goes high, and clocks the next microcode instruction into the microcode pipeline latch described in chapter 4. This instruction, which is executed during T3, calculates the value to be written to the memory, and this value is strobed into the memory PCB by the RAS NOW output from the memory control state machine that is also asserted at this time. The following table shows the relative timing of these signals

Time	F	SF	Bookmark	ras	Mem RQST	Wait	MDI Valid
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	15	0	0	0	1	0
2	1	15	7	0	4	0	0
3	1	15	0	1	0	0	0

7.1.2 DMA cycles

The EIO board in a PERQ machine contains a DMA controller that allows several of the I/O subsystems to directly transfer data to the main memory. During such a transfer, the DMA controller provides the memory address and the data words, while the CPU memory control state machine asserts the MEM RQST ST and the RAS NOW signals during the correct T-states. The EIO card is outside the scope of this manual, so only the operation of the memory control state machine will be described here.

There are 2 possible types of DMA cycle - Read and Write. Both types transfer a Quad-word to/from the memory board, although the timing of the 2 cycles is very different, and will therefore be described separately

7.1.2.1 DMA Read

A DMA Read operation occurs when the I/O MEM RQST output from the EIO board goes low and the I/O MEM WR signal is also low. When this occurs, the memory control state machine asserts the GRANT output for one microcycle during the next T2 state, and this signal is used by the EIO card to start the DMA transfer. During the following T3 state, the processor address outputs are disabled, since the PROC RDING MA/MD signal is 0 (This signal is referred to as 'MA' in the following tables), and at this time the DMA controller places the address onto the backplane address lines. The memory control state machine drives '1' onto the MEM RQST ST lines, signifying a FETCH4 operation, and also asserts the RAS NOW signal. The data words are read out of memory in the usual way, and transferred to the peripheral by the DMA controller. This sequence is repeated every 4 microcycles, as indicated by the following table, and thus a 16 bit word is transferred on every cycle.

T	Bookmark	Mem Rqst	ras	IO Req	Grant	MA
1	0	0	0	0	0	1
2	0	0	0	0	0	1
3	0	0	0	0	0	1
0	0	0	0	0	0	1
1	0	0	0	1	0	1
2	1	0	0	1	1	1
3	0	1	1	1	0	0
0	0	0	0	1	0	1
1	0	0	0	1	0	1
2	1	0	0	1	1	1
3	0	1	1	1	0	0
0	0	0	0	1	0	1
1	0	0	0	1	0	1
2	1	0	0	1	1	1
3	0	1	1	1	0	0
0	0	0	0	1	0	1
1	0	0	0	1	0	1
2	1	0	0	1	1	1
3	0	1	1	1	0	0
0	0	0	0	1	0	1

7.1.2.2 DMA Write

This operation is started when the I/O MEM RQST signal goes low and the I/O MEM WR signal is high. The memory control state machine then asserts the GRANT output during the next T2 state, and disables the processor address and data outputs during the subsequent T0-T3 state, during which time the DMA controller transmits the address and data words to the memory card. The memory control state machine places '5' onto the MEM RQST ST lines during the T1 and T2 cycles of this period, thus defining a STORE4 cycle, and then asserts the RAS NOW signal during the T3 period, to start the memory operation. This sequence is repeated by asserting the GRANT output during the next T2 cycle, and thus a 4-word transfer occurs every 8 microcycles. The timing of this operation is shown in the following table.

T	Bookmark	Mem Rqst	ras	IO Req	Grant	MA
1	0	0	0	0	0	1
2	0	0	0	0	0	1
3	0	0	0	0	0	1
0	0	0	0	0	0	1
1	0	0	0	1	0	1
2	2	0	0	1	1	1
3	2	0	0	1	0	1
0	2	0	0	1	0	0
1	2	5	0	1	0	0
2	13	5	0	1	0	0
3	0	0	1	1	0	0
0	0	0	0	1	0	1
1	0	0	0	1	0	1
2	2	0	0	1	1	1
3	2	0	0	1	0	1
0	2	0	0	1	0	0
1	2	5	0	1	0	0
2	13	5	0	1	0	0
3	0	0	1	1	0	0
0	0	0	0	1	0	1

7.2 The Raster Operation State Machine

The data path used to update a graphics bitmap that was described in chapter 6 is controlled by a 4-state finite state machine, the outputs of which are decoded by a set of PROMs to generate the necessary control signals. While this system is controlling the bitmap data path, the main CPU calculates the memory addresses for the bitmap words, and performs the necessary memory cycles to fetch and store them.

The state of this machine is held in 2 sections of the 74S374 D-type flip-flop U170, namely U170d (for the least significant bit, ROP ST<0>) and U170c (for ROP ST<1>, the most significant bit). These flip-flops are clocked by the rising edge of the LD MIR' B signal at the start of each microcycle,and thus at this time, the state of the machine can be changed. The outputs of the flip-flops are permanently enabled since the output enable pin is tied low.

The State machine is controlled by the 512*8bit 27S29 PROM U171 (RTI02), which accepts inputs from the Raster Operation control register, the microcycle T-state lines, and the output of the state latch. This PROM is programmed to load the next state into the state latch, to provide 2 enable signals (P SRC QUAD and P DST QUAD) to the source and destination control systems, and also to provide 2 direct control signals to the data path, namely RO DATA RDY (which enables the raster operation data path outputs to the memory board) and CLK FIFO IN (which loads the memory data word into the source FIFO). The signals handled by this PROM are given in the following table.

Address	Signal	Comments
A0	ROP ST<0>	State Latch Output
A1	ROP ST<1>	
A2	L TIME<1>	Latched Microcode T-state counter
A3	L TIME<0>	
A4	RO/PS	Master Enable signal from control register
A5	Ground	
A6	PHASE<0>	Cycle type select from control register
A7	PHASE<1>	
A8	PHASE<2>	

Data	Signal	Comment
D0	P SRC QUAD	Source Fetch control signal
D1	P DST QUAD	Destination path Enable
D2	P ROP ST<1>	Input to state latch
D3	P ROP ST<0>	
D4	RO DATA RDY	Enable signal to output buffers
D5	CLK FIFO IN	Load memory word into source FIFO
D6	Not Used	
D7	Not Used	

The P ROP ST<n> lines are directly connected to the D inputs on the respective state flip-flops, while the P SRC QUAD and P DST QUAD signals are latched by U170a and U170b (both 74S374 flip-flops) respectively, before being fed to address inputs on the source and destination control PROMs, as described below.

The RO DATA RDY signal is logically NANDed with the CLK-4E CPU clock signal by the 74S00 NAND gate U237b, the output of which drives the output enable pins of the combiner

output latches U243 and U260. Therefore, the outputs of these latches are enabled during the second half of any microcycle when the RO DATA RDY signal is high, thus allowing the output of the combiner to be stored in the main memory

The CLK FIFO IN signal is connected to one of the Input Clock pins (CKA or CKB) of the Shifter source FIFOs U157, U158, U137, U138, while the other clock input is driven by the CLK-4F CPU clock signal. Since CKA and CKB are logically ANDed inside the FIFO chips, the CLK FIFO IN signal enables the FIFO input clock and causes the memory data word (SI<n>) to be loaded into the FIFO half way through the microcycle.

7.2.1 The Destination Control System

The actual updating of a given 16 bit word is controlled by the 512*4bit 7643 RDS00 PROM U169, which provides the control signals for the combiner along with the QUAL FIFO signal that causes a source bitmap word to be clocked out of the Fifo. This PROM is controlled by the DST WRD<n> and WID WRD<n> outputs from the appropriate control registers, which determine where the left and right edges of the updated region occur. The PROM also accepts the SL/SR control signal, which indicates in which direction the bitmap is being scanned. , as this determines whether the start of a line contains a left edge or a right edge. Two of the cycle type signals, namely PHASE<0> and PHASE <1> are also connected to this PROM, along with the T-state counter outputs L TIME <n>. Finally, the whole data path is enabled by the latched DST QUAD signal from the Raster Operation Finite state machine.

Three of the outputs from the RDS00 PROM are the raw signals for the Combiner mask, namely P EL, P ER and P DON'T MASK. These signals are latched by U170e, U170f and U170h respectively to produce the EL, ER and DON'T MASK control signals. The clock input to U170 is driven by the LD MIR'B signal, so that the combiner control signals are updated at the start of each microcycle. The Output enable pin of U170 is, as mentioned above, grounded, so that these outputs are always enabled. The operation of the combiner during graphics operations is described in chapter 6, and the relative timing of these signals is explained below.

The final output of the RDS00 PROM is the QUAL FIFO signal. This signal is used to clock a source bitmap word out of the FIFO, after being processed by the following circuit. The QUAL FIFO signal is not latched by any section of U170, and, owing to the programming of the RDS00 PROM, occurs during the microcycle preceeding the one in which the update takes place.

The QUAL FIFO signal is connected to the D-input of the 74S374 flip-flop U209f. This flip-flop is clocked by the CLK-0R B signal at the start of each microcycle, and the permanently enabled output of U209f, the DLY QUAL FIFO signal, is therefore active on the microcycle when the bitmap updating actually occurs. This output is inverted by the 74S240 NOT gate U206b, to produce the DLY QUAL FIFO' signal.

The QUAL FIFO signal is logically NANDed with the PHASE<4> output from the Raster Operation Control Register by the 74S00 NAND gate U212a. The output of this gate therefore goes low on cycles preceding a bitmap word update, and is therefore used to clock a word from the Source FIFO into the half pipeline latches described in chapter 6.

The LEFTOVER output from the RSC03 source control PROM described below is logically NANDed with the FIFO OR signal (which indicates when there is a valid word in the Source FIFO), by the 74S00 NAND gate U210a. The output goes low when a word has to be removed from the Source FIFO, thus clocking the FIFO to drop the word, but not clocking the Raster Operation memory data registers. This occurs when either fewer than 4 words are to be fetched for the source operand, or to clear the FIFO after a complete line of the graphics bitmap has been updated. The output of U210a is connected to the D input of the 74S74 flip-flop U147b, which is clocked by the rising edge of the CLK-4E signal at the start of the second half of the microcycle. The output of U147b is the CLR FIFO L signal

The 3 signals that cause the FIFO to be clocked, namely DLY QUAL FIFO', CLR FIFO L, and the output of U212a, need to be Logically ORed together. The DLY QUAL FIFO' and CLR FIFO L signals are connected to the inputs of the 74S08 AND gate U73Aa, while the output of this gate is combined with the output of U212a by the 74S00 NAND gate U212b. Since the 3 signals are all active low, the output of U212b, the CLK FIFO OUT signal, goes high whenever at least one of the inputs is active, as required.

As described in chapter 6, the CLK FIFO OUT signal enables the 74S00 NAND gate U203c, and thus allows the UNLD signal to become active. This signal directly drives the output clock pins of the Source FIFO chips. The CLK FIFO OUT also enables the SLIVER CLK via the 74S10 NAND gate U211b, and causes the previous word to be loaded into the appropriate half pipeline latch.

7.2.2 The Source Control System

The purpose of the source control system is to drop unnecessary words from the Source FIFO either at the end of a line of the graphics object, or when fewer than 4 words need to be fetched during a FirstSource operation. The system consists of a small finite state machine controlled by the 74S287 256*4bit RSC03 PROM U172.

The PROM acts as the combinatorial feedback logic for the single-bit state latch U170g, which is clocked by the rising edge of the LD MIR' B signal at the start of each microcycle. The D-input of this 74S374 flip-flop is driven by the P SRC FIFO CLR output of the RSC03 PROM, while the output, the OLD EVEN/ODD signal, is connected to one of the address inputs of the RSC03 PROM, and also to the input of bit 5 of the Source FIFO chip U157. The OLD EVEN/ODD signal is therefore carried through the FIFO along with the source bitmap words, and emerges from U157

on the Q5 pin as the FIFO EVEN/ODD signal, which is fed back to another address input of the RSC03 PROM. By using this signal, the Source Control system can flag the source bitmap words, and change state when the particular word has passed through the FIFO.

The signals connected to the RSC03 PROM are listed in the following table

Address	Signal	Comments
A0	OLD EVEN/ODD	Output of state flip-flop
A1	FIFO EVEN/ODD	Output of FIFO flag bit
A2	SL/SR	Direction control bit from control register
A3	L TIME<0>	Latched T-state counter
A4	L TIME<1>	
A5	SRC WRD<0>	Source Word position from source register
A6	SRC WRD<1>	
A7	SRC QUAD	Source Word Enable from Raster Operation State Machine

Data	Signal	Comments
D1	Not Used	
D2	Not Used	
D3	Leftover	Clear FIFO signal to clock logic
D4	P SRC FIFO CLR	Input to state flip-flop

The only control output of this state machine is the LEFTOVER signal that is used to enable the FIFO output clock to drop a FIFO word, as described above.

When the value of the SRC WRD<n> field indicates that fewer than 4 words are to be loaded into the FIFO on a FirstSource Instruction, the FETCH4 memory cycle is executed as normal, and the 4 words are in fact loaded into the FIFO, but the unnecessary words are dropped from the FIFO by the leftover signal.

7.3 Raster Operation Timing

All the raster operation update instructions have essentially the same timing, and the basic sequence of operations is as follows.

The main CPU starts a FETCH4 cycle to get the next 4 words of the destination bitmap from the main memory. The first word from this fetch appears at the outputs of the combiner input register (U219 and U221) during the T3 microcycle. The raster operation state machine now asserts

the EL, ER and DON'T MASK signals according to the particular part of the graphics object being updated, and the combiner updates the bitmap word in the required way. The resultant word is stored in the combiner output latched (U243 and U260) at the end of the T3 cycle, and is therefore valid throughout the following T0 cycle to be written back to memory. The CPU has started an overlapping STORE4 cycle, and so the result is written back to the memory during the next 4 T-states. The CPU also starts an overlapping FETCH4 cycle to fetch the next 4 words of the source bitmap into the FIFO, controlled by the CLK FIFO output from the Raster Operation State Machine. These memory cycles take a total of 8 microcycles to perform.

When the state of the DON'T MASK signal means that the current output word of the FIFO will be used to update the bitmap, then the Raster Operation state machine asserts the QUAL FIFO output during the preceding microcycle, so that a new word will be presented to the shifter inputs at the end of the cycle.

There then follows a pause of 4 microcycles before the entire sequence is repeated, and thus a 64bit bitmap is updated every 12 microcycles.

The sequence of these memory cycles is given in the following table.

T-state		
2	Destination Fetch	
3	Destination Fetch	
0	Destination Fetch	Destination Store
1	Destination Fetch	Destination Store
2	Source Fetch	Destination Store
3	Source Fetch	Destination Store
0	Source Fetch	
1	Source Fetch	
2	Idle	
3	Idle	
0	Idle	
1	Idle	

And the sequence is now repeated

As was mentioned above, Raster Operation state machine has 4 states, numbered 0-3 and the following table give the approximate correspondence between the state number and the operation being performed.

State	Operation
0	Idle between source and destination fetches
1	Destination Fetch
2	Source Fetch
3	Idle - Raster Operations turned off

It is important to note that in the following tables, the column headed 'state' gives the new state of the state machine at the end of that microcycle, that is the value of the outputs of the ROM that are connected to the D-inputs of the state latch.

There are 8 different Raster Operation instructions which are selected by the state of the PHASE<0>-PHASE<2> signals, while the exact operations performed by each instruction may be modified by the values of the SRC WRD<n>, DST WRD<n> and WID WRD<n> fields together with the state of the SL/SR signal. It is obviously impossible to show all possible instructions in this manual, and therefore a representative sample will be explained.

In the following tables, the state of the LEFTOVER signal should be ignored unless reference is made to it in the text. This is because the state of the source FIFO is unknown at the start of each instruction.

7.3.1 Phase=0 – Begin

This instruction follows the timing cycle just described, and when the SL/SR signal is low, it performs a left-edge update of the destination bitmap. If a particular 16 bit word is entirely outside the area to be updated, then the DON'T MASK signal is asserted, and the QUAL FIFO signal is not, so that the source FIFO is unchanged and the destination word is passed unaltered through the combiner.

If the current 16 bit word contains the left edge of the bitmap, then the DON'T MASK signal is forced to 0, and the QUAL FIFO signal is asserted on the previous cycle, thus causing a normal update cycle to occur. The EL signal is 1, and the ER signal is 0, so that the combiner performs a left-edge masking operation.

Finally, if the current 16 bit word lies entirely inside the updated area, then an update cycle occurs as described in the last paragraph, but both the EL and ER signals are set to 1, so that the combiner performs no mask operation on the destination word.

The particular word that contains the left edge of the region to be updated is selected by the DST WRD field of the Raster Operation Destination Control Register. The following 2 tables show the sequence of operations that occurs for 2 different values of this field.

Time=	0	Sl/Sr=	0	RO/PS=	1	E/O=	0				
Ph0=	0	Ph1=	0	Ph2=	0						
SrcWd=	0	DstWd=	0	WidWd=	0						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	1
2	0	0	0	0	1	0	1	1	0	0	1
3	0	1	1	1	1	0	1	1	0	0	1
0	0	1	1	1	1	0	1	1	0	1	1
1	1	1	1	1	1	0	1	1	0	1	2
2	0	0	1	0	0	0	1	1	1	1	2
3	0	0	0	0	0	0	1	1	1	1	2
0	0	0	0	0	0	0	1	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	1
2	0	0	0	0	1	0	1	1	0	0	1
3	0	1	1	1	1	0	1	1	0	0	1

Time=	1	Sl/Sr=	0	RO/PS=	1	E/O=	0				
Ph0=	0	Ph1=	0	Ph2=	0						
SrcWd=	0	DstWd=	2	WidWd=	0						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
1	0	0	0	0	0	0	1	1	0	0	1
2	0	0	0	0	1	0	1	1	0	0	1
3	1	1	1	1	1	0	1	1	0	0	1
0	1	0	1	0	1	0	1	1	0	1	1
1	1	0	1	1	1	0	1	1	0	1	2
2	0	0	1	1	0	0	1	1	1	1	2
3	0	0	0	0	0	0	1	1	1	1	2
0	0	0	0	0	0	0	1	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	1
2	0	0	0	0	1	0	1	1	0	0	1
3	1	1	1	1	1	0	1	1	0	0	1
0	1	0	1	0	1	0	1	1	0	1	1

Finally, if the SL/SR signal is one, then the system performs a right-edge update (since this is the first edge to be modifies on a right-to-left update), as described for the END operation below. The next table shows how this occurs.

Time=	3	Sl/Sr=	1	RO/PS=	1	E/O=	0				
Ph0=	0	Ph1=	0	Ph2=	0						
SrcWd=	0	DstWd=	0	WidWd=	0						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
2	1	0	0	0	1	0	0	0	0	0	1
3	1	0	0	1	1	0	0	0	0	0	1
0	1	0	1	1	1	0	0	0	0	1	1
1	1	0	1	1	1	0	0	0	0	1	2
2	0	0	1	1	0	0	0	0	1	1	2
3	0	0	0	0	0	0	0	0	1	1	2
0	0	0	0	0	0	0	0	0	1	0	2
1	0	0	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
2	1	0	0	0	1	0	0	0	0	0	1

7.3.2 Phase=1 – Mid

This instruction is used to update graphics words where all 64 bits of the destination quadword lie inside the area to be updated, and therefore no edges are contained within the word. The operation of this instruction is not affected by the contents of the various control registers.

On each update cycle, both the EL and ER inputs to the combiner mask control are set to 1, so that the combiner passes the destination word unchanged. The QUAL FIFO signal goes active during the microcycle before each update cycle, so that a new word is fed into the shifter. These operations are shown in the following table.

Time=	3	Sl/Sr=	0	RO/PS=	1	E/O=	0				
Ph0=	1	Ph1=	0	Ph2=	0						
SrcWd=	0	DstWd=	2	WidWd=	0						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	1
2	1	0	0	0	1	0	1	1	0	0	1
3	1	0	1	1	1	0	1	1	0	0	1
0	1	0	1	1	1	0	1	1	0	1	1
1	1	0	1	1	1	0	1	1	0	1	2
2	0	0	1	1	0	0	1	1	1	1	2
3	0	0	0	0	0	0	1	1	1	1	2
0	0	0	0	0	0	0	1	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	1
2	1	0	0	0	1	0	1	1	0	0	1

7.3.3 Phase=2 – End

When the SL/SR signal is low, this instruction is used to update graphics words which contain the right edge of the are to be altered. The position of this edge is determined by the difference between the DST WRD<n> and WID WRD<n> fields of the raster operation control register. The sequence of operations is similar to that of the previous 2 instructions, and the only new feature is that during the cycle which updates the word containing the edge, the mask control lines are set to the EL=0, ER=1 state, so that the combiner performs a right-edge update. The following 2 tables show how this occurs.

Time=	0	Sl/Sr=	0	RO/PS=	1	E/O=	0				
Ph0=	0	Ph1=	1	Ph2=	0						
SrcWd=	0	DstWd=	2	WidWd=	0						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
0	0	0	0	0	0	0	1	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	1
2	1	0	0	0	1	0	1	1	0	0	1
3	1	0	1	1	1	0	1	1	0	0	1
0	1	0	1	1	1	0	1	1	0	1	1
1	1	0	1	1	1	0	1	1	0	1	2
2	0	0	0	1	0	0	1	1	1	1	2
3	0	0	0	0	0	0	1	1	1	1	2
0	0	0	0	0	0	0	1	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0

Time=	3	Sl/Sr=	0	RO/PS=	1	E/O=	0				
Ph0=	0	Ph1=	1	Ph2=	0						
SrcWd=	0	DstWd=	2	WidWd=	2						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	1
2	1	0	0	0	1	0	1	1	0	0	1
3	1	0	1	1	1	0	1	1	0	0	1
0	0	0	0	1	1	0	1	1	0	1	1
1	0	1	1	1	1	0	1	1	0	1	2
2	0	1	1	1	0	0	1	1	1	1	2
3	0	0	0	0	0	0	1	1	1	1	2
0	0	0	0	0	0	0	1	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	1
2	1	0	0	0	1	0	1	1	0	0	1

If the SL/SR signal is one, the system performs as left edge update as described for the Begin Instruction above, and as shown in the next table

Time=	2	Sl/Sr=	1	RO/PS=	1	E/O=	0				
Ph0=	0	Ph1=	1	Ph2=	0						
SrcWd=	0	DstWd=	0	WidWd=	0						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
2	1	0	0	0	1	0	0	0	0	0	1
3	0	0	1	0	1	0	0	0	0	0	1
0	0	1	1	1	1	0	0	0	0	1	1
1	0	1	1	1	1	0	0	0	0	1	2
2	0	1	1	1	0	0	0	0	1	1	2
3	0	0	0	0	0	0	0	0	1	1	2
0	0	0	0	0	0	0	0	0	1	0	2
1	0	0	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1

7.3.4 Phase=3 – Begin/End

This instruction is really a combination of the Phase=0 (Begin) and Phase=2 (End) instructions described above, and is used when the same 64 bit Quad-word contains both the left and right edges of the area to be updated. When these edges are contained in different 16-bit words, The raster operation machine performs left-edge, middle, and right edge cycles as required. This is demonstrated by the following table

Time=	1	Sl/Sr=	0	RO/PS=	1	E/O=	0				
Ph0=	1	Ph1=	1	Ph2=	0						
SrcWd=	0	DstWd=	2	WidWd=	0						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	1
2	0	0	0	0	1	0	1	1	0	0	1
3	1	1	1	1	1	0	1	1	0	0	1
0	1	0	1	0	1	0	1	1	0	1	1
1	1	0	1	1	1	0	1	1	0	1	2
2	0	0	0	1	0	0	1	1	1	1	2
3	0	0	0	0	0	0	1	1	1	1	2
0	0	0	0	0	0	0	1	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0

However, the 2 edges may occur in the same 16 bit word. When this occurs, the Raster operation state machine sets both the EL and ER signals to 0, so that the combiner performs a both-edge update cycle. The remaining 3 microcycles in the update do not change the destination word as the DON'T MASK signal is set to 1. This sequence of operations is illustrated in this table

Time=	1	Sl/Sr=	0	RO/PS=	1	E/O=	0				
Ph0=	1	Ph1=	1	Ph2=	0						
SrcWd=	0	DstWd=	2	WidWd=	2						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0	1
2	0	0	0	0	1	0	1	1	0	0	1
3	1	1	1	1	1	0	1	1	0	0	1
0	0	0	0	0	1	0	1	1	0	1	1
1	0	1	1	1	1	0	1	1	0	1	2
2	0	1	1	1	0	0	1	1	1	1	2
3	0	0	0	0	0	0	1	1	1	1	2
0	0	0	0	0	0	0	1	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0

7.3.5 Phase=4 – XtraSource

This instruction is used to load quadwords from memory into the source FIFO. The raster operation state machine alternates between states 0 and 2, and while in state 2, the CLK FIFO IN signal is active, so that the memory data is strobed into the FIFO. The main CPU performs FETCH4 cycles, so that the required memory data is presented to the FIFO data inputs.

None of the control fields affect the operation of this instruction. The sequence of operations is given in the next table

Time=	2	Sl/Sr=	0	RO/PS=	1	E/O=	0				
Ph0=	0	Ph1=	0	Ph2=	1						
SrcWd=	0	DstWd=	0	WidWd=	0						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
2	0	0	0	0	0	0	0	0	1	0	2
3	0	0	0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	0	0	0	1	0	2
1	0	0	0	0	0	0	0	0	1	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	2
2	0	0	0	0	0	0	0	0	1	0	2
3	0	0	0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	0	0	0	1	0	2
1	0	0	0	0	0	0	0	0	1	0	0

7.3.6 Phase=5 – FirstSource

This operation makes use of the EVEN/ODD signal, and the flag bit of the source FIFO, to load the appropriate number of 16 bit words into the source FIFO at the start of a graphics operation. The Raster operation state machine toggles between the 0 and 2 states, and loads a Quad-word into the FIFO, after main CPU has executed a FETCH4 memory instruction.

The SRC WRD<n> signals control the relative phase of the OLD EVEN/ODD and the CLK FIFO IN signals, and the result is to cause the LEFTOVER signal to become active during some of the subsequent microcycles, and to drop the appropriate number of words from the FIFO.

The following tables show examples of this operation

Time= 2 Sl/Sr= 0 RO/PS= 1 E/O= 1
Ph0= 1 Ph1= 0 Ph2= 1
SrcWd= 3 DstWd= 0 WidWd= 0

Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
2	0	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	2
2	0	0	0	0	0	1	0	1	1	0	2
3	0	0	0	0	0	1	0	1	1	0	2
0	0	0	0	0	0	1	0	1	1	0	2
1	0	0	0	0	0	0	0	1	1	0	0
2	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	1	0	0	2
2	0	0	0	0	0	1	0	0	1	0	2
3	0	0	0	0	0	1	0	0	1	0	2
0	0	0	0	0	0	1	0	0	1	0	2
1	0	0	0	0	0	0	1	0	1	0	0

Time= 2 Sl/Sr= 0 RO/PS= 1 E/O= 0
Ph0= 1 Ph1= 0 Ph2= 1
SrcWd= 2 DstWd= 0 WidWd= 0

Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	2
2	0	0	0	0	0	1	0	0	1	0	2
3	0	0	0	0	0	1	0	1	1	0	2
0	0	0	0	0	0	1	0	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	1	0	0	0

Time=	2	Sl/Sr=	0	RO/PS=	1	E/O=	1				
Ph0=	1	Ph1=	0	Ph2=	1						
SrcWd=	1	DstWd=	0	WidWd=	0						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
2	0	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	2
2	0	0	0	0	0	1	0	0	1	0	2
3	0	0	0	0	0	1	0	0	1	0	2
0	0	0	0	0	0	1	0	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	0	1	0	0	0

Time=	2	Sl/Sr=	0	RO/PS=	1	E/O=	1				
Ph0=	1	Ph1=	0	Ph2=	1						
SrcWd=	0	DstWd=	0	WidWd=	0						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
2	0	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	2
2	0	0	0	0	0	1	0	0	1	0	2
3	0	0	0	0	0	1	0	0	1	0	2
0	0	0	0	0	0	1	0	0	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	1	0	1	0	0	2
2	0	0	0	0	0	1	0	1	1	0	2
3	0	0	0	0	0	1	0	1	1	0	2
0	0	0	0	0	0	1	0	1	1	0	2
1	0	0	0	0	0	0	1	0	1	0	0

7.3.7 Phase=6 – End,Clear

This instruction is a combination of the End and FirstSource instructions described above, and is used at the end of a graphics bitmap line, to ensure that the FIFO is cleared, ready for the next line of the bitmap.

This operation is identical to the End instruction described earlier (that is, the Phase=2 instruction), except that the LEFTOVER signal goes active during the source fetch phase, and thus the unnecessary source operand words are dropped from the FIFO. The sequence of operations is shown in the next tables

Time=	2	Sl/Sr=	0	RO/PS=	1	E/O=	0				
Ph0=	0	Ph1=	1	Ph2=	1						
SrcWd=	2	DstWd=	1	WidWd=	1						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
2	1	0	0	0	1	0	0	0	0	0	1
3	1	0	1	1	1	0	0	0	0	0	1
0	1	0	1	1	1	0	0	0	0	1	1
1	0	0	0	1	1	1	0	0	0	1	2
2	0	1	1	1	0	1	0	0	1	1	2
3	0	0	0	0	0	1	0	1	1	1	2
0	0	0	0	0	0	1	0	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	0	1	0	0	0
3	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	1	0	0	1
2	1	0	0	0	1	0	0	1	0	0	1
3	1	0	1	1	1	0	0	1	0	0	1
0	1	0	1	1	1	0	0	1	0	1	1
1	0	0	0	1	1	1	0	1	0	1	2

Time=	0	Sl/Sr=	0	RO/PS=	1	E/O=	1				
Ph0=	0	Ph1=	1	Ph2=	1						
SrcWd=	1	DstWd=	1	WidWd=	1						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
0	0	0	0	0	0	1	0	0	1	0	2
1	0	0	0	0	0	0	1	0	1	0	0
2	0	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
2	1	0	0	0	1	0	0	0	0	0	1
3	1	0	1	1	1	0	0	0	0	0	1
0	1	0	1	1	1	0	0	0	0	1	1
1	0	0	0	1	1	1	0	0	0	1	2
2	0	1	1	1	0	1	0	0	1	1	2
3	0	0	0	0	0	1	0	0	1	1	2
0	0	0	0	0	0	1	0	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	0	1	0	0	0

7.3.8 Phase=7 – Begin/End, Clear

This instruction is similar to the End,Clear instruction just described, except for the fact that a Begin/End (Phase=3) instruction replaces the End instruction. It is therefore used when both

edges of the updated region lie in the same 64 bit Quad word, and when the source FIFO needs to be cleared ready for the next line of the bitmap.

The operation of this instruction is shown in the following tables.

Time=	1	Sl/Sr=	0	RO/PS=	1	E/O=	0				
Ph0=	1	Ph1=	1	Ph2=	1						
SrcWd=	1	DstWd=	3	WidWd=	2						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
1	0	1	1	1	1	1	0	0	0	1	2
2	0	1	1	1	0	1	0	0	1	1	2
3	0	0	0	0	0	1	0	0	1	1	2
0	0	0	0	0	0	1	0	1	1	0	2
1	0	0	0	0	0	0	1	1	1	0	0
2	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	1	0	0	1
2	1	0	0	0	1	0	0	1	0	0	1
3	1	0	1	0	1	0	0	1	0	0	1
0	0	0	0	1	1	0	0	1	0	1	1
1	0	1	1	1	1	1	0	1	0	1	2
2	0	1	1	1	0	1	0	1	1	1	2
3	0	0	0	0	0	1	0	1	1	1	2
0	0	0	0	0	0	1	0	0	1	0	2

Time=	3	Sl/Sr=	0	RO/PS=	1	E/O=	0				
Ph0=	1	Ph1=	1	Ph2=	1						
SrcWd=	0	DstWd=	3	WidWd=	1						
Time	Qual Fifo	Dont Mask	El	Er	P Dst Quad	P Src Quad	Left Over	Old E/O	Clk Fifo	Ro Data	State
3	1	0	1	0	1	0	1	1	0	0	1
0	1	0	1	1	1	0	1	1	0	1	1
1	0	0	0	1	1	1	0	1	0	1	2
2	0	1	1	1	0	1	0	1	1	1	2
3	0	0	0	0	0	1	0	1	1	1	2
0	0	0	0	0	0	1	0	1	1	0	2
1	0	0	0	0	0	0	1	0	1	0	0
2	0	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0	1
2	1	0	0	0	1	0	1	0	0	0	1
3	1	0	1	0	1	0	1	0	0	0	1
0	1	0	1	1	1	0	1	0	0	1	1
1	0	0	0	1	1	1	0	0	0	1	2
2	0	1	1	1	0	1	0	0	1	1	2