PS 390 RASTER BACKEND MICROCODE MANUAL
Preliminary Version 1.2

Evans & Sutherland

March 17, 1987

Contents

List of Figures

List of Tables

Part I

Overview

Chapter 1

PS390 Graphics System Overview

The microcode described in this manual controls operations in the Raster Backend section of the PS390 Graphics System. The PS390 Graphics System uses the ACP and PLS from the PS350 along with Shadowfax Technology to produce *anti-aliased* lines on a raster display. The basic components (*cards*) of the PS390 Graphics System include:

- Graphics Control Processor (JCP)
- Mass Memory
- Display Processor, comprising
  - Arithmetic Control Processor (ACP)
  - Pipeline Subsystem (PLS)
  - Raster Backend (RBE)

In general, the purpose of the Raster Backend portion of the PS390 is to:

- Replace the PS350 refresh buffer, and emulate it as much as possible.
- Convert PS300 PLS endpoint data to Shadowfax Technology endpoints.
- Convert PS300 HSI color specification to RGB color.
- Perform LGS functionality of hit detection.
- Provide direct pixel reads and writes.

The microcode that controls Raster Backend (RBE) operations is loaded by the Graphics Control Processor (JCP) into Bitslice writable control store (WCS). WCS is an 80-bit $\times$ 4096 word memory. Each microcode word corresponds to a *state* of the Raster Backend. That is, the execution of a new microcode word changes the state of the Raster Backend. Words are read from the WCS into an execution register from which they are read out to control various Raster Backend components.

The basic components of the Raster Backend which are controlled by the microcode word during each state are:

- Program Sequencer AM2910A
- Branch Condition Select Multiplexer
- Arithmetic & Logic Unit Microprocessor AM29117
- Multiplier AM29517A (or compatible)
- 16-bit D data bus
- 16-bit Y data bus
- Input FIFO Stack Bus Controller to Shadowfax Technology
- Video Control Interface
- Shadowfax Technology Acknowledge (PACK)
- Immediate field

When microcode is written, each state must include control for each one of these components. The assembler has the ability to define default control, hence the programmer does not need to explicitly define the control for each state but should be aware that each component will perform an operation each state.

Other components of the RBE are accessed depending on the control of the D and Y data buses. These components include:

- Scratch Memory
- Function Lookup Table
- Vector RAM
- Common Bus Interface
- FIFO buffer register holding data from PLS

- Pixel processor loaders
- Scanline Buffer Interface

Part II

Microcode Word

Chapter 2

Microcode Word Fields

The microcode word is divided into fields which correspond to the basic components to be controlled. These fields may then be further divided into subfields. The microcode word is organized as follows:

Sequencer			ALU			Multiplier				D-bus		Y-bus		FSBC/	PACK	Immediate	
29110	CondSel	L	D	SR	ALU	Cx	Cy	TYPE	OSel	Src	Dest	Src	Dest	Video		Src	Data

Figure 2.1: Microword Organization

The syntax for one state of microcode can contain specification for each field with a '&' between each field. The order of the fields is not critical but should be in the order of the word for easier maintainance. The order of the subfields within a field are also not critical. Fields which are left blank, will receive the default value; but if any subfield is filled, all subfields for that field must be filled.

2.1 Program Sequencer

2.1.1 29110 Instruction

The program sequencer controls the program flow by selecting the next microword to be executed. The next address can be any of the following:

- microprogram address register
- internal register counter
- Stack
- external Vector RAM
- Immediate Field

This subfield is combined with the branch condition code select MUX subfield to create the single mnemonic which controls the program sequencer. See the specification of the branch condition code select MUX subfield when *.cc* is listed to see the values that it may take.

JZ	Jump to 0 and reset the stack pointer (must be used at power-up to initialize)
CJSI.*cc*	If the condition code passes, PUSH the current microprogram counter onto the stack and jump to the address specified in the immediate field; if the condition code fails, continue with the next microcode word.
CJSV.*cc*	If the condition code passes, PUSH the current microprogram counter onto the stack and jump to the address supplied by the vector RAM; if the condition fails, continue with the next microcode word.
JMPI	Jump to the address in the immediate field.
JMPV	Jump to the address supplied by the vector RAM.
CJI.*cc*	If the condition code passes, jump to the address specified in the immediate field; if the condition code fails, continue with the next microcode word.
PUSH	PUSH the current microprogram counter onto the stack.
PUSHI.*cc*	PUSH the current microprogram counter onto the stack and load the counter with the value specified in the immediate field only if the condition code passes.
JSRI.*cc*	PUSH the current microprogram counter onto the stack and if the condition code passes, jump to the address specified in the immediate field; if the condition code fails, jump to the address specified in the internal register/counter.
JSRV.*cc*	PUSH the current microprogram counter onto the stack and if the condition code passes, jump to the address supplied by the vector RAM; if the condition code fails, jump to the address specified in the internal register/counter.
CJV.*cc*	If the condition code passes, jump to the address supplied by the vector RAM; if the condition code fails, continue with the next microcode word.
JRI.*cc*	If the condition code passes, jump to the address specified in the Immediate field; if the condition code fails, jump to the address specified in the internal register/counter.
JRV.*cc*	If the condition code passes, jump to the address supplied by the vector RAM; if the condition code fails, jump to the address specified in the internal register/counter.

RFCT	If the internal register/counter is not zero, decrement the counter and jump to the address on the top of the stack; if the internal register/counter is zero, POP the stack and continue with the next instruction.
RICT	If the internal register/counter is not zero, decrement the counter and jump to the address specified in the Immediate field; if the internal register/counter is zero, continue with the next instruction.
RVCT	If the internal register/counter is not zero, decrement the counter and jump to the address supplied by the vector RAM; if the internal register/counter is zero, continue with the next instruction.
CRTN.*cc*	If the condition code passes, jump to the address on the top of the stack and POP the stack; if the condition code fails, continue with the next microcode word.
CJIP.*cc*	If the condition code passes, jump to the address specified in the immediate field and POP the stack; if the condition code fails, continue with the next microcode word.
CJVP.*cc*	If the condition code passes, jump to the address supplied by the vector RAM and POP the stack; if the condition code fails, continue with the next microcode word.
LDCTI	Load the internal counter/register with the value specified in the immediate field and continue with the next microcode word.
LDCTV	Load the internal counter/register with the value supplied by vector RAM and continue with the next microcode word.
LOOP.*cc*	If the condition code passes, POP the stack and continue with the next microcode word; if the condition code fails, jump to the address specified on the top of stack.
CONT	Continue with the next microcode word (DEFAULT)
TWBI.*cc*	If the condition code passes, continue with the next microcode word and POP the stack; if the condition code fails and the counter is not 0, jump to the address on the top of the stack and decrement the counter; if the condition code fails and the counter is 0, jump to the address specified in the immediate field and POP the stack
TWBV.*cc*	If the condition code passes, continue with the next microcode word and POP the stack; if the condition code fails and the counter is not 0, jump to the address on the top of the stack and decrement the counter; if the condition code fails and the counter is 0, jump to the address supplied by the vector RAM and POP the stack.

NOTE: The current microprogram counter has a value equal to the address of the *NEXT* microcode word.

2.1.2 Branch Condition Code Select MUX

The condition code select subfield is used to control the MUX which determines which bit in the RBE status register is provided to the sequencer for conditional testing. The condition code selects 1 of 7 possible bits from the status register and supplies that as the condition code to the sequencer. The bits which may be selected from the status register are:

- CT — CT status bit from the 29117
- OVR — overflow bit from the 29117
- N — negative bit from the 29117
- C — carry bit from the 29117
- Z — zero bit from the 29117
- PPL ATTN — pixel processor loader attention request, signifying that there is a command which cannot be given to the pixel processors that must be handled **WHEN THE BIT IS CLEAR**
- FSBC Ready — Input FSBC is ready to accept a command or endpoint **WHEN THE BIT IS CLEAR**

.T	condition code will always pass (DEFAULT)
.F	condition code will always fail
.Z	condition code will pass if ZERO bit is set
.NZ	condition code will pass if ZERO bit is clear
.N	condition code will pass if NEGATIVE bit is set
.NN	condition code will pass if NEGATIVE bit is clear
.C	condition code will pass if CARRY bit is set
.NC	condition code will pass if CARRY bit is clear
.O	condition code will pass if OVERFLOW bit is set
.NO	condition code will pass if OVERFLOW bit is clear
.CT	condition code will pass if CT bit is set
.NCT	condition code will pass if CT bit is clear
.FR	condition code will pass if FSBC READY bit is set
.NFR	condition code will pass if FSBC READY bit is clear
.PPL	condition code will pass if PPL ATTN bit is set
.NPPL	condition code will pass if PPL ATTN bit is clear

2.1.3 Condition Code Latch

Certain bits of the condition code are explicitly latched while others are latched each state. The bits which are explicitly latched are:

- OVR
- N
- C
- Z

The single bit field which controls this is:

No.CC.Latch	do not change the condition code bits (OVR,N,C,Z) (DEFAULT)
CC.Latch	latch the new values of (OVR,N,C,Z)

2.2 Arithmetic & Logic Unit Microprocessor

The arithmetic logic unit of the RBE is a AM29117. This is a 16-bit processor which provides addition, subtraction, boolean operations and barrel shifting along with 32 internal registers.

The microcode word is defined such that dual-address operations are supported, along with control for the D-latch and updating of the status register.

2.2.1 ALU instruction field

The instructions for the ALU are described in the data sheet for an AM29117. The mnemonics described in the data sheet are used as is.

The specification of the register fields is:

RS.xx	source register, where xx is register number [0..31]
RD.xx	destination register, where xx is register number [0..31]

2.2.2 D-latch field

The AM29117 has the ability to read in a 16 bit value from the D-bus and latch it into a register which can then be used in the ALU during the next state.

The single bit field which controls this is:

No.D.Latch	do not change the D-bus (DEFAULT)
D.Latch	latch the D-bus into the D-latch

RESTRICTION: The D-latch cannot be latched and used as a source to an ALU operation in the same state.

2.2.3 Status Register field

The status register in the 29117 is explicitly controlled as to whether it is updated by the ALU operation. This allows for the status to be saved and tested many states later.

The single bit field which controls this is:

Status.No.Update	do not update the status register
Status.Update	update the status register (DEFAULT)

2.3 Multiplier

The multiplier in the RBE is an AM29517A compatible multiplier. The two operands are written into registers using the the D and Y buses. This field controls the actual multiplication which includes:

- unsigned or signed specification for both X and Y
- round the product to the most significant 16 bits
- perform a fractional multiply, this shifts the MSP left one bit, and duplicates the sign bit in the MSB of the LSP
- have either the MSP or LSP available to the D-bus

The first subfield specifies whether X is unsigned or signed:

X.S	X is signed
X.US	X is unsigned (DEFAULT)

The second subfield specifies whether Y is unsigned or signed:

Y.S	Y is signed
Y.US	Y is unsigned (DEFAULT)

The third subfield specifies the type of multiplication:

NO.MPY	do not perform a multiply (DEFAULT)
MPY	perform integer multiply
MPY.R	perform integer multiply rounded
MPY.F	perform fractional multiply
MPY.FR	perform fractional multiply rounded

The fourth subfield specifies which 16 bits are output to Y-bus:

Out.MS	Provide most significant 16 bits as output to Y-bus
Out.LS	Provide least significant 16 bits as output to Y-bus

RESTRICTION: A multiply and reading of the result cannot be performed in the same state.

2.4 16-bit D data bus

The D-bus is one of two data buses in the RBE. There are number of sources to choose data from and different destinations that can receive the data from the D-bus.

2.4.1 Sources for the D-bus

Sources for the D-bus are considered to be *read* onto the bus hence they are all referred to as *read registers.*

S.IMMED	read immediate field of microcode word (DEFAULT)
S.SCRAM	read scratch ram, address supplied by IMMED field of microcode word
S.FIFO	read 16 bit value from the PLS FIFO buffer register
S.STATUS	read 16 bit status register
S.FCNTAB	read the function table, address must be written 2 states prior to reading value
S.YBUS	read Y-bus - transparent mode
S.YTMP	read the temporary register for data from the Y-bus
S.PC	read the program counter (used for diagnostics only, or position independent code)
S.PPL	read pixel processor loader information
S.MPY	read the multiplier output
S.CB.DAT	read common bus data register
S.SLB	read the data from the scanline buffer, first read will return red and green; second will return blue and window

2.4.2 Destinations for the D-bus

Destinations for the D-bus are considered to be *written* from the bus, hence they are all referred to as *write registers*.

D.DNULL	no destination for D-bus (DEFAULT)
D.CLR.CB.ATTN	selecting this as the D-bus destination clears the CB ATTN bit in the maintenance register and the status register
D.CLR.HIT	selecting this as the D-bus destination clears the Hit Detect bit and the LP Hit bit in the status register
D.MPY.Y	write the multiplier Y register
D.SCRAM	write the scratch RAM, address is supplied by the least significant 11 bits of the immediate field of the microcode word (the scratch RAM cannot be written from the 29117 in one state, data must first be written into the YTMP register and then next state from the DTMP register to the scratch RAM)
D.CBADR.MS.R	write common bus most significant address bits and do mass memory read
D.CBADR.MS.W	write common bus most significant address bits and do mass memory write
D.CBADR.LS	write common bus address least significant address bits
D.CB.DAT	write common bus data register
D.IRV	interrupt the JCP with the interrupt vector written
D.VEC.PAGE	write vector RAM page select bits from the three least significant bits on the D-bus
D.DTMP	write the data into the temporary D-bus register which can be source to the Y-bus in a subsequent state
D.PP.ADR	write the pixel processor register to which the data stored in the intermediate register (PP.DATA) is to be loaded and perform the load

2.5 16-bit Y data bus

The Y-bus is one of two data buses in the RBE. There are number of sources to choose data from and different destinations that can receive the data from the Y-bus.

2.5.1 Sources for the Y-bus

Sources for the Y-bus are considered to be *read* onto the bus hence they are all referred to as *read registers.*

S.ALU	read output from the ALU (DEFAULT)
S.DBUS	read data from the D-bus - transparent mode
S.DTMP	read the temporary register for data from the D-bus
S.VDR	read Video Display intermediate register

2.5.2 Destinations for the Y-bus

Destinations for the Y-bus are considered to be *written* from the bus, hence they are all referred to as *write registers*.

D.YNULL	no destination for Y-bus (DEFAULT)
D.MPY.X	write multiplier X register
D.FCNTAB.ADR	write function table address register
D.IMMED	write immediate register, value written can be used in any following state instead of value in the microcode word until is is again written
D.InFSBC.MS	write data to the Input FSBC MSW Intermediate Register; Value is loaded into Input FSBC Register as specified in the FSBC field
D.InFSBC.LS	write data to the Input FSBC LSW Intermediate Register
D.VDR	write the Video Display intermediate register
D.PP.DAT	write the data to loaded into the pixel processors into an intermediate register
D.PP.MASK	write the pixel processor enable mask which determines to which pixel processor data will be written
D.VEC.ADR	write vector ram address 8 bits (taken from most significant byte of the Y-bus)
D.YTMP	write the data into the temporary Y-bus register which can be source to the D-bus in a subsequent state
D.SLB	write the encoded value from the least significant 4 bits of the Y-bus which selects which pixel processor the Scanline Buffer logic will read when the pixel processors request scanline activity
D.HITBOX	writing to this register loads the current value of the hit box

2.6 Input FSBC & Video

Due to timing and bus sizes, writing to either the Input FSBC or the Video card actually goes through intermediate registers. This field then controls data flow between those registers and the Input FSBC or Video Registers.

2.6.1 Input FSBC

The Input FSBC is the FIFO stack bus controller chip which transforms parallel data into bit serial data for the Shadowfax Technology chips. There exists five 32 bit registers:

- Control Word
- X
- Y
- Z
- W

Actual loading of the FSBC chip occurs in the state after the MSW register has been loaded. Hence first the LSW register must be written then the MSW. There is an option to load the MSW and ZERO the LSW for times when only the MSW contains valid data.

> **RESTRICTION:** The two states following the loading of the MSW CANNOT load either the LSW or the MSW of the Input FSBC.

The specification of this field when loading the FSBC is:

NOP	nothing is to occur with respect to FSBC and Video (DEFAULT)
Ld.InFSBC.CNTL	Load the MSW with data and then load the control word of the Input FSBC
Ld.InFSBC.CNTL.ZL	Load the MSW with data, zero the LSW and then load the control word of the Input FSBC
Ld.InFSBC.X	Load the MSW with data and then load the X register of the Input FSBC
Ld.InFSBC.X.ZL	Load the MSW with data, zero the LSW and then load the X register of the Input FSBC
Ld.InFSBC.Y	Load the MSW with data and then load the Y register of the Input FSBC
Ld.InFSBC.Y.ZL	Load the MSW with data, zero the LSW and then load the Y register of the Input FSBC
Ld.InFSBC.Z	Load the MSW with data and then load the Z register of the Input FSBC
Ld.InFSBC.Z.ZL	Load the MSW with data, zero the LSW and then load the Z register of the Input FSBC
Ld.InFSBC.W	Load the MSW with data and then load the W register of the Input FSBC
Ld.InFSBC.W.ZL	Load the MSW with data, zero the LSW and then load the W register of the Input FSBC

2.6.2 Video

The video card has a number of read and write registers to control various things such as:

- cursor definition
- cursor location

- LUT
- *etc.*

This field controls data flow between the intermediate register and specific registers on the video card.

R.VC	read video control register
W.VC	write video control register
W.MC.LA	write video logic array/memory control register
W.CLR.VBLANK	clear vertical blank in status register
R.LP.X	read light pen X register
R.LP.Y	read light pen Y register
W.CRS.RAM.ADR	write cursor RAM address register
R.SIGNATURE	read signature register
W.BKG.R	write RED background color
W.BKG.GB	write GREEN & BLUE background color
W.CLUTAB.ADR	write video color lookup table address register
R.CLUTAB.R	read video color lookup table RED
W.CLUTAB.R	write video color lookup table RED
R.CLUTAB.G	read video color lookup table GREEN
W.CLUTAB.G	write video color lookup table GREEN
R.CLUTAB.B	read video color lookup table BLUE
W.CLUTAB.B	write video color lookup table BLUE
R.CRS.RAMOW	read cursor RAM Overlay/White data
W.CRS.RAMOW	write cursor RAM Overlay/White data
W.BLINK.RATE	write blink rate with values in bits 8-15
R.BLINK.COUNT	read value of blink counter
R.CRS.X	read cursor X register
W.CRS.X	write cursor X register
R.CRS.Y	read cursor Y register
W.CRS.Y	write cursor Y register

All *writes* take the data in the intermediate register present at the beginning of the state (*i.e.* data that has been written in a previous state) and stores it in the corresponding register. Reads take the data from the specified register and stores it in the intermediate register which then can be a source to the Y-bus in a subsequent state.

NOTE: Both reads and writes can be pipelined if the corresponding video registers are readable or writable in a single state.

RESTRICTION: The LUT address must be written at least two states prior to reading or writing the location. To read a LUT location, the same read must occur in two consecutive states with the data valid in the Video Intermediate Register after the second read.

2.7 Shadowfax Handshake Interface Technique

The Shadowfax Technology chips are designed such that each chip handles multiple registers. When all registers are either loaded or read, an acknowledge pin *PACK* to the chip must be asserted which then enables the chip to continue.

NO.PACK	no PACK to be asserted (DEFAULT)
IN.PACK	give *PACK* to the Input-FSBC chip.
PPL.PACK	give *PACK* to the pixel processor loaders to signal that the command has been read from the FSBC at the bottom of the DDCC.
PP.PACK	give *PACK* to the pixel processors.

RESTRICTION: PPL.PACK cannot be given in the state following a read from the PPL.

2.8 Immediate field

The Immediate data field is a 16 bit data field which can supply data to:

- sequencer for an address
- scratch RAM for an address
- D-bus for 16 bit constant

In addition to the the data field there is a single bit field which determines for the next state the source of the immediate data:

IMMED.MW	immediate data to be from the microcode word (DEFAULT)
IMMED.REG	immediate data to be from the immediate register

Part III

Register Definitions

Chapter 3

Register Definitions

3.1 Maintenance Register

HMS	clock control				shift	RESERVED			wrt	*busyd	WCS				vec RAM
IREQ	*reset	*halt	step	shift clk	bit				decod en		mode	dclk	*oe	r/*w	r/*w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 3.1: Maintenance Register

- HMS IREQ (r/w) — interrupt request from JCP to bitslice, setting bit will set CB ATTN bit in status register (bit is cleared by bitslice specifying D.CB.ATTN as destination of D-bus)
- clock control
- *reset (r/w) — clearing bit resets the Raster Backend
- *halt (r/w) — clearing bit causes the HMS processor to stop
- step (w) — setting bit causes one state of microcode to be executed always read as 0
- shiftclk (r/w) — setting and then clearing bit causes shiftbit to be shifted into the shift loop
- shiftbit (r/w) — bit value to be shifted into the shift loop when written, when read returns bit shifted out of the shift loop

- wrt decod en (r/w) — bit when clear inhibits the storing of data in the Y & D buses.
- *busyd (r/w) — bit connected to *BID of Shadowfax Technology chips
- WCS mode (r/w) — 0 = Run, 1 = Load
- WCS dclk (r/w) — clocks the shift loop
- WCS *oe (r/w) — 0 = enable, 1 = disable
- WCS r/*w (r/w) — bit when clear causes value in execution register to be written into writable control store
- vecRAM r/*w (r/w) — bit when clear causes the value in the immediate field of the execution register to be written to the vector RAM

NOTE: All bits go to zero on hard reset (power-on).

3.2 RBE Registers

3.2.1 IMMED

16 bit register which can source the immediate bus instead of the immediate field of the microcode word if IMMED.REG is selected in the previous stat.

3.2.2 FIFO buffer register

16 bit read-only register returns the next PLS value when read if the FIFO Ready bit is set in the STATUS register.

RESTRICTION: FIFO can not be read in two consecutive states.

3.2.3 STATUS

The Status register is a 16 bit read-only register which contains bits corresponding to the status of various parts of the raster backend as shown in Figure 3.2.3.

CB ATTN	FIFO Buffer Ready	CB Busy	PP Ready	PP Active	Vert Blank	Even Field	Hit Detect	SLB Busy	*RC Exist	RESERVED					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 3.2: Status Register Definition

- CB ATTN — common bus attention request (cleared by bitslice specifying D.CLR.CB.ATTN as destination of D-bus)
- FIFO Buffer Ready — FIFO buffer register has data to be processed (invalid for state following third data read)
- CB Busy — common bus state machine indication when clear means data ready to be read or write has completed (invalid for second state following loading of CB.ADR.MS)
- PP Ready — pixel processors are able to accept a new command when bit is clear (invalid for two states after PP.PACK)
- PP Active — pixel processors are active when high, clear means Idle (invalid for two states after PP.PACK)
- Vert Blank — video card is doing vertical retrace (cleared by selecting W.CLR.VBLANK in the FSBC/VIDEO Field)
- Even Field — video card is displaying an even field
- Hit Detect — hit detect logic saw pixel written within pick box (cleared by selecting D.CLR.HIT as destination of D-bus)
- SLB Busy — when clear SLB has 32 bits (28 read data) ready from read of pixel data
- *RC Exist — bit is clear if Raster (video) Card exists

3.2.4 PPL

16 bit read-only register which returns values from the endpoint FSBC and color FSBC. The PPL ATTN bit in the STATUS register is set when there is a command which requires the RBE bitslice to process before more endpoint processing can happen. This method is used to

determine when the DDCC have flushed. (MSW and LSW pairs must be read in the following manner: READ MSW, NO OP STATE, READ LSW) Reads from the register will return the following data in order:

- endpoint-FSBC Command MSW
- endpoint-FSBC Command LSW
- endpoint-FSBC X MSW
- endpoint-FSBC X LSW
- endpoint-FSBC Y MSW
- endpoint-FSBC Y LSW
- endpoint-FSBC Z MSW
- endpoint-FSBC Z LSW
- color-FSBC W MSW
- color-FSBC W LSW
- color-FSBC Y MSW
- color-FSBC Y LSW
- color-FSBC Z MSW
- color-FSBC Z LSW

RESTRICTION: A PPL.PACK cannot be asserted in the state following an LSW READ. Data must be read in pairs.

3.2.5 MPY.X

16 bit write-only register supplying one input to the multiplier.

3.2.6 MPY.Y

16 bit write-only register supplying one input to the multiplier.

3.2.7 MPY

16 bit read-only register returning the output of the last multiplication performed selected by the multiplier OSel subfield.

3.2.8 CBADR.MS.R

16 bit register specifying the most significant bits of the mass memory address and initiate a read from mass memory. Data is in CB.DAT when the CB Ready bit in the STATUS register is set. It is only valid to write this register when the CB Busy bit in the STATUS register is clear.

3.2.9 CBADR.MS.W

16 bit register specifying the most significant bits of the mass memory address and initiate a write to mass memory. Data in CB.DAT is written to mass memory. Another mass memory operation can occur when the CB Ready bit in the STATUS register is set. It is only valid to write this register when the CB Busy bit in the STATUS register is clear.

3.2.10 CBADR.LS

16 bit register specifying the least significant bits of the mass memory address to which a read or write will happen. It is only valid to write this register when the CB Busy bit in the STATUS register is clear.

3.2.11 CB.DAT

16 bit register which supplies the data to be written to mass memory, or if a read will return the data in mass memory specified by the mass memory address registers. It is only valid to read or write this register when the CB Busy bit in the STATUS register is clear.

3.2.12 VEC.ADR

8-bit write-only register used to index into vector RAM which returns the address of a microcode module. The vector RAM is to be loaded such that it can be used to decode command received either from the FIFO or from the common bus interface. The 8 bits written are from the MSB of the bus.

> **RESTRICTION:** This register CANNOT be used (loaded) in two consecutive states.

3.2.13 VEC.PAGE

3-bit write-only register used to determine the *page* of vector RAM used to decode the command. This mode is envisioned to be set such that endpoints will jump to the appropriate code to handle either:

- 000 — normal endpoint
- 001 — textured endpoint
- 010 — viewport center specification
- 011 — raster mode

3.2.14 PP.ADR

16-bit write-only register which specifies to which pixel processor register the data contained in the PP.DATA register is to be written and triggers the write.

> **NOTE:** the pixel processors can only be written to when *BOTH* the PPL ATTN is clear and PP Ready bit is set in the STATUS register.

3.2.15 PP.DATA

16-bit write-only register to hold the data to be written to a pixel processor register until the address is written.

NOTE: the pixel processors can only be written to when *BOTH* the PPL ATTN is clear and PP Ready bit is set in the STATUS register.

3.2.16 PP.MASK

16-bit write-only register which contains the bit mask used to determine which pixel processors are written to (0-enabled, 1-disabled).

NOTE: this register should only be non-zero during the power-on sequence during which the pixel processors are given their unique number and the GEN pin is being defined.

Pixel Processor Number	GEN Pin Definition
0	*SLR
1	NSCN
4	BUSY
5	ST1
8	*SLW
9	*RBA
12	TI
13	READ

Figure 3.3: GEN Pin Definitions

3.2.17 IRV

8-bit write-only register which when written generates a common bus interrupt and supplies the value *in the most significant 8 bits* as the interrupt vector.

3.2.18 VDR

16-bit register which is an intermediate register between the video registers and the Bitslice. The actual video register read or written is specified in the FSBC/Video field of the microcode word. This register is written the state *before* the FSBC/Video field specifies which register on the Video card the data is written to. When reading a video register, this register can be read the state after the Video field specifies the video register to be read.

3.2.19 InFSBC.LS

16-bit write-only register which is an intermediate register to hold the LSW portion of the data so that the 32 bit value can can be assembled and then written into the input FSBC.

NOTE: the FSBC can only be written when the FSBC Ready bit is clear.

3.2.20 InFSBC.MS

16-bit write-only register which is an intermediate register to hold the MSW portion of the data so that the 32 bit value can can be assembled and then written into the input FSBC. The FSBC field of the microcode word specifies where this particular data is to be loaded in the Input FSBC chip.

NOTE: the FSBC can only be written when the FSBC Ready bit is clear.

RESTRICTION: after writing the MSW, two states must be executed before either the LSW or MSW can again be written.

3.2.21 DTMP

16-bit register which can be destination of the D-bus in one state and then source for the Y-bus in a subsequent state.

RESTRICTION: neither DBUS or DTMP can be selected as the source for the Y-bus when DTMP is selected as as the destination of the D-bus.

3.2.22 YTMP

16-bit register which can be destination of the Y-bus in one state and then source for the D-bus in a subsequent state.

> **RESTRICTION:** neither YBUS or YTMP can be selected as the source for the D-bus when YTMP is selected as as the destination of the Y-bus.

3.2.23 SLB

16-bit read-only register which during the first read returns the red value in the most significant byte and green in the least significant byte. During the second read the blue value is returned in the most significant byte and window is returned in the least significant byte.

> **RESTRICTION:** Once the PP.PACK has be asserted to request the pixel processors to perform a read, *NO REGISTER* in the pixel processors may be written until the SLB Busy bit is clear.

> **NOTE:** Data is valid in this register only when the the SLB Busy bit in the status register is clear.
> **RESTRICTION:** One state must be executed between the two SLB reads.

3.2.24 HITBOX

16-bit write-only register which loads the values of the corners of the hit box to be used for hit testing. The values must be written in the following order:

- X-min in MSB, Y-max in LSB
- X-max in MSB, Y-min in LSB

The values should be the most significant 8 bits of the 10 bit pixel address. Comparison for hit happens on the most significant address bits of one of the pixel processors.

3.2.25 FCNTAB.ADR

16-bit write-only register which is written with the address of the location of entry in the function table to be returned. The data will be available to be read two states after writing this register.

3.3 InFSBC Interface

FSBC internal chip registers are 32 bit registers. There are two intermediate registers a LSW and MSW register. The FSBC field of the microcode word specifies which of these registers the data is to be written. Writing the MSW causes the appropriate register in the input FSBC to be loaded in the next state. For specific details of the FSBC registers see the Shadowfax VLSI Manual.

> **RESTRICTION:** after writing the MSW, two states must be executed before either the LSW or MSW can again be written.

3.4 Video Control Interface

3.4.1 VC

Buffer Select	Video Enable	Diag Clk	Blink	*Tip Switch	VLA Mode	CT Mode	Col Sig	Cursor Enable	Cursor Select	Video Format		Screen Blast Enable	Cursor Color Red	Cursor Color Green	Cursor Color Blue
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 3.4: Video Control Register (Read/Write)

- Buffer Select — selects buffer displayed (A=0,B=1) Read/Write
- Video Enable — enables picture for display (Enabled = 1) Read/Write
- Diag Clk — bit which when video format is in diagnostic mode is the clock. Read/Write
- Blink — bit when clear disables blinking and when set enables blinking.

- *Tip Switch — Depressed = 0 Read/(Write); Note that writing 0 to this bit clears the Light Pen Hit Y register.
- VLA Mode — video logic array Display/*Maintenance bit. Read/Write
- CT Mode — color table write/*read mode. Read/Write
- Col Sig — color signature Red/*Green-Blue. Read/Write
- Cursor Enable — enable display of cursor (Enabled = 1) Read/Write
- Cursor Select — selects one of two definable cursors Read/Write
- Video Format — selects video format to be displayed Read/Write
  - 0 = High Resolution
  - 1 = Diagnostic
  - 2 = Camera
  - 3 = Television
- Screen Blast Enabled — enables screen blast (Enabled = 1) Read/Write
- Cursor Color Red — enables red gun for cursor Read/Write
- Cursor Color Green — enables green gun for cursor Read/Write
- Cursor Color Blue — enables blue gun for cursor Read/Write

3.4.2 MC.LA

Memory Controller Mode								VLA Control							
RESERVED		*PP Reset	RAS BankW	RAS BankB	RAS BankA	*Long Cycle	Xfer Cycle	RESERVED							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 3.5: Video Logic Array/Memory Control Register (Write-Only)

- *PP Reset — bit when clear resets the pixel processors
- RAS BankW — enable write to window planes
- RAS BankB — enable write to buffer B
- RAS BankA — enable write to buffer A
- *Long Cycle — when clear enables long memory cycle
- Xfer Cycle — when set enables transfer cycle, used to clear valid planes
- VLA Control — value should be b'11100000' (or h'E0')

S/D	Blink	Reserved	Reserved
3	2	1	0

Figure 3.6: RBE Window Bit Definitions

NOTE: The RBE chooses the following definition for the window bits:

Bit #3 when clear specifies that the window is double buffered and when set is single buffered.

Bit #2 when set specifies that the data is blinkable in double buffered windows.

3.4.3 LP.X

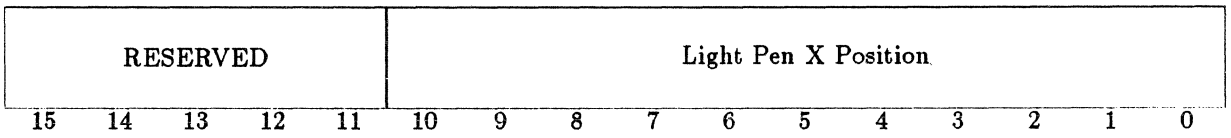

Figure 3.7: Light Pen X Register (Read-Only)

- Light Pen X Position — Horizontal address at time of last light pen hit is contained in bits 0-10, with bit 10 the MSB. Bits 11-15 are undefined. The address is related to the screen position by some displacement that depends on the screen and light pen characteristics, as well as the video format.

NOTE: formula for translating this value to actual screen coordinate
YET TO BE DETERMINED.

3.4.4 LP.Y

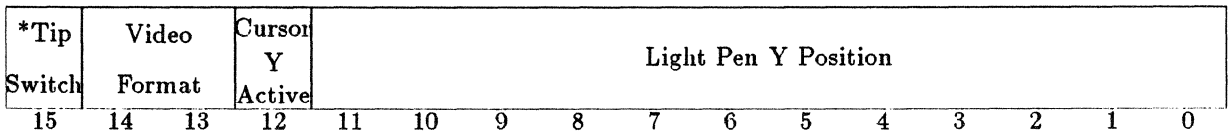

Figure 3.8: Light Pen Y Register (Read-Only)

- *Tip Switch — bit clear if tip switch was depressed at the time of the last light pen hit
- Video format — returns the video format active when the hit occurred
- Cursor Y active — bit is set if cursor is displayed in row in which hit was detected
- Light Pen Y Position — vertical address at time of the last light pen hit is contained in bits 0-11, with bit 11 the MSB. This register is cleared by writing a zero to bit 11 of the Video Control Register.

NOTE: formula for translating this value to actual screen coordinate
YET TO BE DETERMINED.

3.4.5 CRS.RAM.ADR

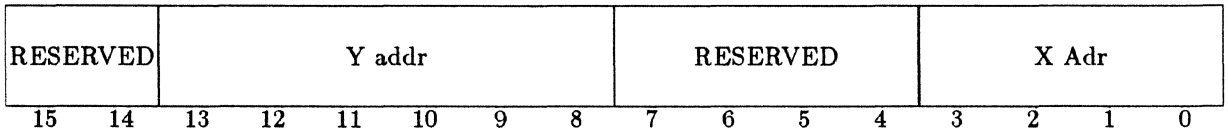

Figure 3.9: Cursor RAM Address (Write-Only)

- X addr — The 8 pixel X address into the Cursor Definition RAM is contained in bits 0-3. Address 0 is at the left-hand edge of the cursor.
- Y addr — The Y address into the Cursor Definition RAM is contained in bits 8-13. Address 0 is at the top of the cursor. If the Cursor Enable bit is set, this specifies the Cursor Row Start Address, which is used in with interlaced displays and also to make the cursor go off the top of the screen.

If the Even Field bit of the Video Control Register is set, the address must be written twice into this register. This register is used for Cursor RAM addressing only if the Cursor Enable bit in the Video Control Register is not set. If the Cursor Enable bit is set, bits 8-13 specify the Cursor Row Start Address, which is used in with interlaced displays and also to make the cursor go off the top of the screen. In interlaced mode, the Cursor Row Start address for the Even Field must be written first, followed by the Cursor Row Start address for the Odd Field. Bits 14-15 must always be written to 0.

NOTE: If interlaced format is being used, the top of the cursor will be at address 0 and address 32.

3.4.6 BKG.R

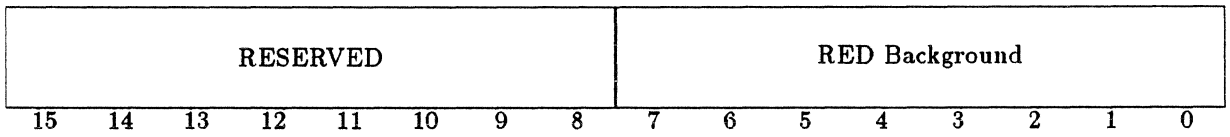

Figure 3.10: Background Color Red Register (Write-Only)

- RED Background — red background color value to be used for pixels in double-buffered windows which have validity bit clear.

3.4.7 BKG.GB

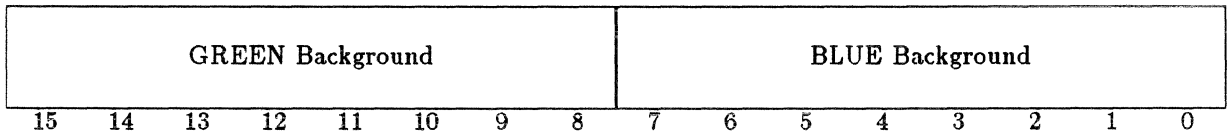

Figure 3.11: Background Color Green/Blue Register (Write-Only)

- GREEN Background — green background color value to be used for pixels in double-buffered windows which have validity bit clear.
- BLUE Background — blue background color value to be used for pixels in double-buffered windows which have validity bit clear.

3.4.8 CLUTAB.ADR

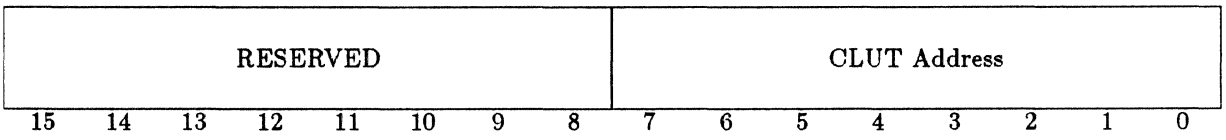

Figure 3.12: Color Lookup Table Address (Read/Write)

- CLUT Address — contains the color lookup table address used for reading and writing the color lookup tables.

  RESTRICTION: This address is not immediately available to the CLUTs until two states after it is written.

3.4.9 CLUTAB.R

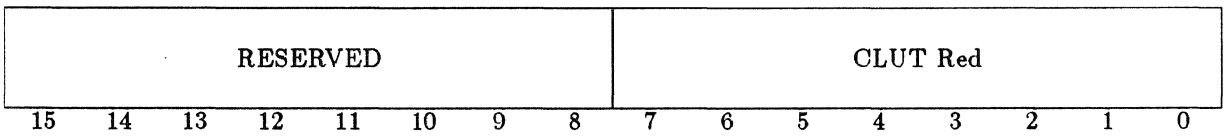

Figure 3.13: Color Lookup Table Red (Read/Write)

- CLUT Red — contains the data from/for the Red color lookup table at the addressed location.

  RESTRICTION: The VLA must be in the Maintenance Mode to read and write the CLUT. The LUT address must be written at least two states prior to reading or writing the location. To read a LUT location, the same read must occur in two consecutive states with the data valid in the video intermediate register after the second read.

3.4.10 CLUTAB.G

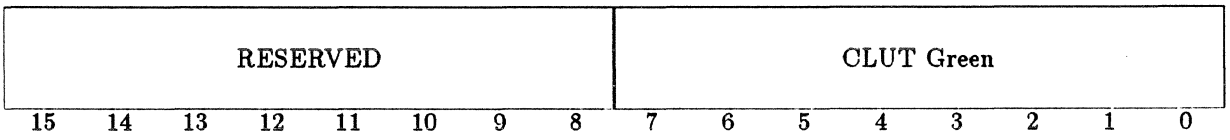

Figure 3.14: Color Lookup Table Green (Read/Write)

- CLUT Green — contains the data from/for the Green color lookup table at the addressed location.

RESTRICTION: The VLA must be in the Maintenance Mode to read and write the CLUT. The LUT address must be written at least two states prior to reading or writing the location. To read a LUT location, the same read must occur in two consecutive states with the data valid in the video intermediate register after the second read.

3.4.11 CLUTAB.B

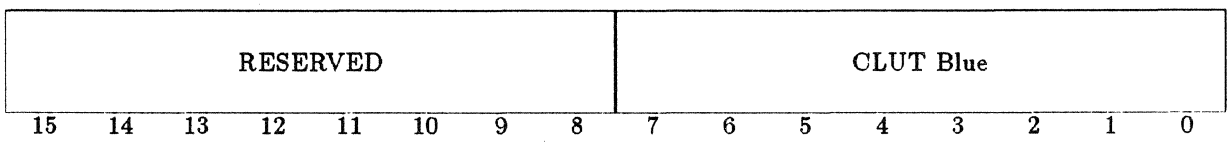

Figure 3.15: Color Lookup Table Blue (Read/Write)

- CLUT Blue — contains the data from/for the Blue color lookup table at the addressed location.

RESTRICTION: The VLA must be in the Maintenance Mode to read and write the CLUT. The LUT address must be written at least two states prior to reading or writing the location. To read a LUT location, the same read must occur in two consecutive states with the data valid in the video intermediate register after the second read.

3.4.12 CRS.RAMOW

Overlay Plane								White Plane							
Pxl 7	Pxl 6	Pxl 5	Pxl 4	Pxl 3	Pxl 2	Pxl 1	Pxl 0	Pxl 7	Pxl 6	Pxl 5	Pxl 4	Pxl 3	Pxl 2	Pxl 1	Pxl 0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 3.16: Cursor Overlay/White (Read/Write)

- Overlay Plane — Bits 8-15 are the data from/for the Cursor Definition RAM, Overlay Plane, at the addressed location. Bit 8 is the left-most bit, bit 15 is the right-most bit.
- White Plane — Bits 0-7 are the data from/for the Cursor Definition RAM, White Plane, at the addressed location. Bit 0 is the left-most bit, bit 7 is the right-most bit. If the corresponding overlay bit is set, this bit, if set, will select the color specified in the video control register for the cursor color; while if clear black will be used.

NOTE: the cursor must be disabled before reading and writing these registers.

3.4.13 BLINK.RATE

RATE								RESERVED							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 3.17: Blink Rate

- Load Up Counter with value (FF - RATE) in the upper 8 bits to set the ON/OFF cycle time for blinking pixels with window bit # 2 set.

NOTE: Writing this register will not change the blink rate until the blinking cycles from the previous count.

3.4.14 CRS.X

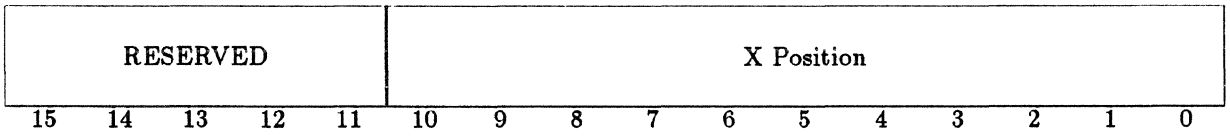

Figure 3.18: Cursor X Start (Read/Write)

- X Position — Bits 0-10 contain the X address at which the cursor is to start on the screen. This is not the actual pixel address, but is related to the pixel address by a displacement that is constant for a given format.

  NOTE: The formulas which convert from requested position to value to be put in this register depending on current video display format. For high resolution video format, the leftmost pixel is x'EF'.

3.4.15 CRS.Y

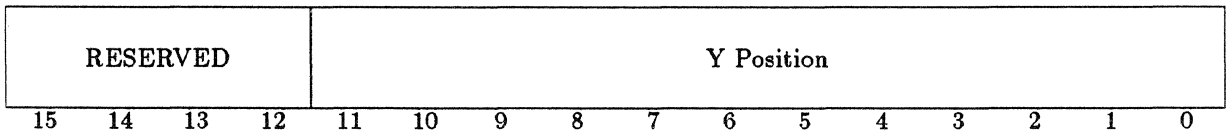

Figure 3.19: Cursor Y Start (Read/Write)

- Y Position — Bits 0-11 contain the Y address at which the cursor is to start on the screen. This is not the actual row address, but is related to the row address by a displacement that is constant for a given format. If interlaced mode is being used, the following applies: The cursor Y start for the Even field must be written first, followed by the cursor Y start for the Odd field. In addition, if the Even Field bit in the Video Control register is set, the value that is read back will be the second to last value written. If the Even Field bit in the Video Control register is not set, the value that is read back will be the last value written.

  NOTE: The formulas which convert from requested position to value to be put in this register depending on current video display format. For high resolution video format, the top row of pixels is at X'24'.

Part IV

Writing Microcode

Chapter 4

Writing Microcode

4.1 Overview

The microcode assembler used will be the preprocessor being developed by Bob Pendelton. This removes the strict ordering of fields. Macros can also be defined for ease of programming and portability. The general order of a microcode word should look like:

```
SEQ( 2910_inst, cc_latch ) &
ALU( d_latch, status_update, alu instruction fields ) &
MULT( x_type, y_type, mult_type, out_select ) &
D_BUS( source, destination ) &
Y_BUS( source, destination ) &
F_V( fsbc/video specification ) &
PACK( pack specification ) &
IMM( imm_src, value ) ;
```

The format that the microcode assembler requires is not context-sensitive. This means that any of the above fields or subfields may be missing, along with the order not being fixed. Programming practice may suggest to use the above format, making sure to use the field identifier which is capitalized and then filling in the appropriate mnemonic for the subfield. The ALU instructions are the exception. Macros have been defined to handle the basic operations internal to the 29117 and these should be used for easier reading of the code.
The mnemonics for all fields except the ALU have already been defined and are summarized with their specific values in Appendix B.

4.2 ALU Instruction Macros

The ALU macros were designed to be easily readable and patterned after existing programming languages. The basic format of a macro is:

[*result*] = [*operand*] **operation** [*operand*]

The result and operands are one of :

Y	y-bus output (result only)
ACC	accumulator
D	d-latch (operand only)
Rn	RAM address where n is [0...31]

The basic macros provided are:

$result = operand$	move instruction
$result = \sim operand$	complement instruction
$result = operand + 1$	increment instruction
$result = -operand$	negate instruction
$result = operand - operand$	subtract instruction
$result = operand + operand$	addition instruction
$result = operand \;\&\&\; operand$	logical AND instruction
$result = operand \sim \& \; operand$	logical NAND instruction
$result = operand \wedge operand$	logical EXOR instruction
$result = operand \sim \mid operand$	logical NOR instruction
$result = operand \mid\mid operand$	logical OR instruction
$result = operand \sim \wedge operand$	logical EXNOR instruction
$result = operand < 0$	shift up 1 bit zero fill
$result = operand < 1$	shift up 1 bit one fill
$result = operand < L$	shift up 1 bit link fill
$result = operand > 0$	shift down 1 bit zero fill
$result = operand > 1$	shift down 1 bit one fill
$result = operand > L$	shift down 1 bit link fill
$result = operand > C$	shift down 1 bit carry fill
$result = operand << n$	barrell shift left n bits
$result = SETBIT(operand, n)$	set bit n
$result = CLRBIT(operand, n)$	clear bit n
$TSTBIT(operand, n)$	test bit n
$result = operand + 2 \wedge n$	add 2^n
$result = operand - 2 \wedge n$	subtract 2^n
$result = 2 \wedge n$	load with 2^n
$result = \sim 2 \wedge n$	load with complement of 2^n
ROTM(U, n, mask, R)	$((U << n)\&mask)\mid(R\& \sim mask)$
ROTC(U, n, mask, R)	$((U << n)\& \sim mask) \wedge (R\& \sim mask)$
PRIOR(R, mask)	prioritize R using mask

Part V

Appendices

Appendix A

Microword Bit Definitions

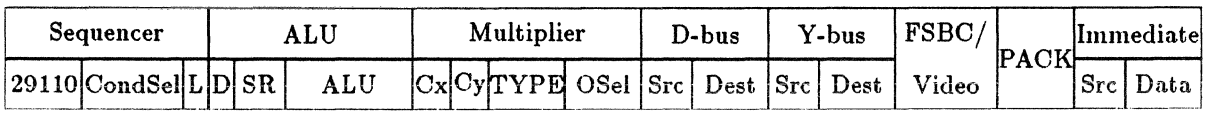

Figure A.1: Microword Organization

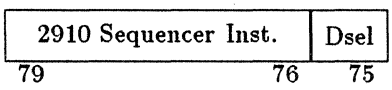

Figure A.2: Sequencer Field Definition

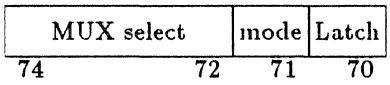

Figure A.3: Condition Field Definition

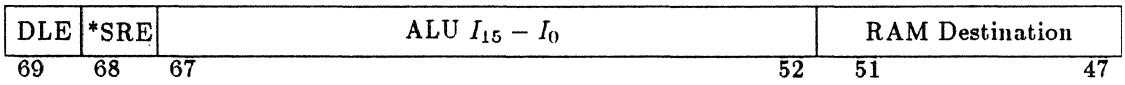

Figure A.4: ALU Field Definition

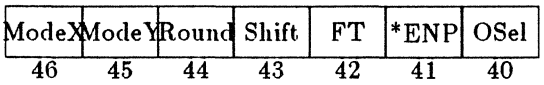

Figure A.5: Multiplier Field Definition

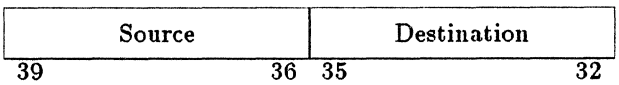

Figure A.6: D-bus Field Definition

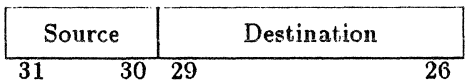

Figure A.7: Y-bus Field Definition

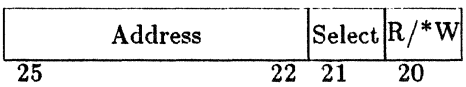

Figure A.8: FSBC/Video Field Definition

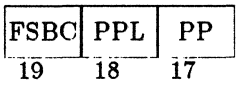

Figure A.9: PACK Field Definition

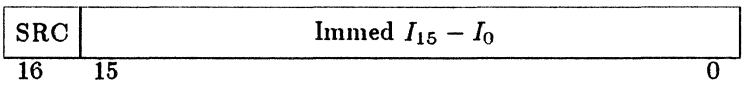

Figure A.10: Immediate Field Definition

Appendix B

Microword Mnemonic Definitions

- BRSEL = 0 — Immed
- BRSEL = 1 — Vector

Inst.BRSEL	Mnemonic	Description
0000.0	**JZ**	Jump to 0 and reset the stack pointer (must be used at power-up to initialize)
0001.0	**CJSI**.*cc*	If the condition code passes, PUSH the current microprogram counter onto the stack and jump to the address specified in the immediate field; if the condition code fails, continue with the next microcode word.
0001.1	**CJSV**.*cc*	If the condition code passes, PUSH the current microprogram counter onto the stack and jump to the address supplied by the vector RAM; if the condition fails, continue with the next microcode word.
0010.0	**JMPI**	Jump to the address in the immediate field.
0010.1	**JMPV**	Jump to the address supplied by the vector RAM.
0011.0	**CJI**.*cc*	If the condition code passes, jump to the address specified in the immediate field; if the condition code fails, continue with the next microcode word.
0100.0	**PUSH**	PUSH the current microprogram counter onto the stack.
0100.0	**PUSHI**.*cc*	PUSH the current microprogram counter onto the stack and load the counter with the value specified in the immediate field only if the condion code passes.
0101.0	**JSRI**.*cc*	PUSH the current microprogram counter onto the stack and if the condition code passes, jump to the address specified in the immediate field; if the condition code fails, jump to the address specified in the internal register/counter.

Table B.1: Sequencer 1 of 3

Inst.BRSEL	Mnemonic	Description
0101.1	**JSRV**.*cc*	PUSH the current microprogram counter onto the stack and if the condition code passes, jump to the address supplied by the vector RAM; if the condition code fails, jump to the address specified in the internal register/counter.
0110.1	**CJV**.*cc*	If the condition code passes, jump to the address supplied by the vector RAM; if the condition code fails, continue with the next microcode word.
0111.0	**JRI**.*cc*	If the condition code passes, jump to the address specified in the Immediate field; if the condition code fails, jump to the address specified in the internal register/counter.
0111.1	**JRV**.*cc*	If the condition code passes, jump to the address suplied by the vector RAM; if the condition code fails, jump to the address specified in the internal register/counter.
1000.0	**RFCT**	If the internal register/counter is not zero, decrement the counter and jump to the address on the top of the stack; if the internal register/counter is zero, POP the stack and continue with the next instruction.
1001.0	**RICT**	If the internal register/counter is not zero, decrement the counter and jump to the address specified in the Immediate field; if the internal register/counter is zero, continue with the next instruction.
1001.1	**RVCT**	If the internal register/counter is not zero, decrement the counter and jump to the address supplied by the vector RAM; if the internal register/counter is zero, continue with the next instruction.
1010.0	**CRTN**.*cc*	If the condition code passes, jump to the address on the top of the stack and POP the stack; if the condtion code fails, continue with the next microcode word.
1011.0	**CJIP**.*cc*	If the condition code passes, jump to the address specified in the immediate field and POP the stack; if the condition code fails, continue with the next microcode word.

Table B.2: Sequencer 2 of 3

Inst.BRSEL	Mnemonic	Description
1011.1	**CJVP**.*cc*	If the condition code passes, jump to the address supplied by the vector RAM and POP the stack; if the condition code fails, continue with the next microcode word.
1100.0	**LDCTI**	Load the internal counter/register with the value specified in the immediate field and continue with the next microcode word.
1100.1	**LDCTV**	Load the internal counter/register with the value supplied by vector RAM and continue with the next microcode word.
1101.0	**LOOP**.*cc*	If the condition code passes, POP the stack and continue with the next microcode word; if the condition code fails, jump to the address specified on the top of stack.
1110.0	**CONT**	Continue with the next microcode word (DEFAULT)
1111.0	**TWBI**.*cc*	If the condition code passes, continue with the next microcode word and POP the stack; if the condtion code fails and the counter is not 0, jump to the address on the top of the stack and decrement the counter; if the condition code fails and the counter is 0, jump to the address specified in the immediate field and POP the stack
1111.1	**TWBV**.*cc*	If the condition code passes, continue with the next microcode word and POP the stack; if the condtion code fails and the counter is not 0, jump to the address on the top of the stack and decrement the counter; if the condition code fails and the counter is 0, jump to the address supplied by the vector RAM and POP the stack.

Table B.3: Sequencer 3 of 3

- I = 0 — invert
- I = 1 — don't invert

$S_3S_2S_1.I$	Mnemonic	Description
011.1	**.T**	condition code will always pass (DEFAULT)
011.0	**.F**	condition code will always fail
100.0	**.Z**	condition code will pass if ZERO bit is set
100.1	**.NZ**	condition code will pass if ZERO bit is clear
110.0	**.N**	condition code will pass if NEGATIVE bit is set
110.1	**.NN**	condition code will pass if NEGATIVE bit is clear
101.0	**.C**	condition code will pass if CARRY bit is set
101.1	**.NC**	condition code will pass if CARRY bit is clear
111.0	**.O**	condition code will pass if OVERFLOW bit is set
111.1	**.NO**	condition code will pass if OVERFLOW bit is clear
000.0	**.CT**	condition code will pass if CT bit is set
000.1	**.NCT**	condition code will pass if CT bit is clear
001.0	**.FR**	condition code will pass if FSBC READY bit is set
001.1	**.NFR**	condition code will pass if FSBC READY bit is clear
010.0	**.PPL**	condition code will pass if PPL ATTN bit is set
010.1	**.NPPL**	condition code will pass if PPL ATTN bit is clear

Table B.4: Condition Code — Branch MUX Select

Value	Mnemonic	Description
1	**No.CC.Latch**	do not change the condition code bits (OVR,N,C,Z) (DEFAULT)
0	**CC.Latch**	latch the new values of (OVR,N,C,Z)

Table B.5: Condition Code — Latch

Value	Mnemonic	Description
0	**No.D.Latch**	do not change the D-bus (DEFAULT)
1	**D.Latch**	latch the D-bus into the D-latch

Table B.6: ALU — D-latch

Value	Mnemonic	Description
1	**Status.No.Update**	do not update the status register
0	**Status.Update**	update the status register (DEFAULT)

Table B.7: ALU — Status Register Update

Value	Mnemonic	Description
1	**X.S**	X is signed
0	**X.US**	X is unsigned (DEFAULT)

Table B.8: Multiplier — Cx

Value	Mnemonic	Description
1	**Y.S**	Y is signed
0	**Y.US**	Y is unsigned (DEFAULT)

Table B.9: Multiplier — Cy

Value	Mnemonic	Description
0001	**NO.MPY**	do not perform a multiply (DEFAULT)
0110	**MPY**	perform integer multiply
1110	**MPY.R**	perform integer multiply rounded
0010	**MPY.F**	perform fractional multiply
1010	**MPY.FR**	perform fractional multiply rounded

Table B.10: Multiplier — TYPE

Value	Mnemonic	Description
1	**Out.MS**	Provide most significant 16 bits as output to Y-bus
0	**Out.LS**	Provide least significant 16 bits as output to Y-bus

Table B.11: Multiplier — Output Select

Value	Mnemonic	Description
0000	**S.IMMED**	read immediate field of microcode word (DEFAULT)
0001	**S.SCRAM**	read scratch ram, address supplied by IMMED field of microcode word
0010	**S.FIFO**	read 16 bit value from the PLS FIFO buffer register. FIFO cannot be read in two consecutive states
0011	**S.STATUS**	read 16 bit status register
0100	**S.FCNTAB**	read the function table, address must be written 2 states prior to reading value
0101	**S.YBUS**	read Y-bus - transparent mode
0110	**S.YTMP**	read the temporary register for data from the Y-bus
0111	**S.PC**	read the program counter (used for diagnostics only, or position independent code)
1000	**S.PPL**	read pixel processor loader information
1001	**S.MPY**	read the multiplier output
1010	**S.CB.DAT**	read common bus data register
1011	**S.SLB**	read the data from the scanline buffer, first read will return red and green; second will return blue and window

Table B.12: D-bus — Source

Value	Mnemonic	Description
0000	**D.DNULL**	no destination for D-bus (DEFAULT)
0001	**D.CLR.CB.ATTN**	selecting this as the D-bus destination clears the CB ATTN bit in the maintenance register and the status register
0010	**D.CLR.HIT**	selecting this as the D-bus destination clears the Hit Detect bit and the LP Hit bit in the status register
0011	**D.MPY.Y**	write the multiplier Y register
0100	**D.SCRAM**	write the scratch RAM, address is supplied by the least significant 11 bits of the immediate field of the microcode word (the scratch RAM cannot be written from the 29117 in one state, data must first be written into the YTMP register and then next state from the DTMP register to the scratch RAM)
0101	**D.CBADR.MS.R**	write common bus most significant address bits and do mass memory read
0110	**D.CBADR.MS.W**	write common bus most significant address bits and do mass memory write
0111	**D.CBADR.LS**	write common bus address least significant address bits
1000	**D.CB.DAT**	write common bus data register
1001	**D.IRV**	interrupt the JCP with the interrupt vector written
1010	**D.VEC.PAGE**	write vector RAM page select bits from the three least significant bits on the D-bus
1011	**D.DTMP**	write the data into the temporary D-bus register which can be source to the Y-bus in a subsequent state
1100	**D.PP.ADR**	write the pixel processor register to which the data stored in the intermediate register (PP.DATA) is to be loaded and perform the load

Table B.13: D-bus — Destination

Value	Mnemonic	Description
00	**S.ALU**	read output from the ALU (DEFAULT)
01	**S.DBUS**	read data from the D-bus
10	**S.DTMP**	read the temporary register for data from the D-bus
11	**S.VDR**	read Video Display intermediate register

Table B.14: Y-bus — Source

Value	Mnemonic	Description
0000	**D.YNULL**	no destination for Y-bus (DEFAULT)
0001	**D.MPY.X**	write multiplier X register
0010	**D.FCNTAB.ADR**	write function table address register
0011	**D.IMMED**	write immediate register, value written can be used in any following state instead of value in the microcode word until is is again written
0100	**D.InFSBC.MS**	write data to the Input FSBC MSW Intermediate Register; Value is loaded into Input FSBC Register as specified in the FSBC field
0101	**D.InFSBC.LS**	write data to the Input FSBC LSW Intermediate Register
0110	**D.VDR**	write the Video Display intermediate register
0111	**D.PP.DAT**	write the data to loaded into the pixel processors into an intermediate register
1001	**D.PP.MASK**	write the pixel processor enable mask which determines to which pixel processor data will be written
1010	**D.VEC.ADR**	write vector ram address 8 bits (taken from most significant byte of the Y-bus)
1011	**D.YTMP**	write the data into the temporary Y-bus register which can be source to the D-bus in a subsequent state
1100	**D.SLB**	write the encoded value from the least significant 4 bits of the Y-bus which selects which pixel processor the Scanline Buffer logic will read when the pixel processors request scanline activity
1101	**D.HITBOX**	writing to this register loads the current value of the hit box

Table B.15: Y-bus — Destination

adr.0.*zerolsw	Mnemonic	Description
1000.0.1	**NOP**	nothing is to occur with respect to FSBC and Video (DEFAULT)
0000.0.1	**Ld.InFSBC.CNTL**	Load the MSW with data and then load the control word of the Input FSBC
0000.0.0	**Ld.InFSBC.CNTL.ZL**	Load the MSW with data, zero the LSW and then load the control word of the Input FSBC
0001.0.1	**Ld.InFSBC.X**	Load the MSW with data and then load the X register of the Input FSBC
0001.0.0	**Ld.InFSBC.X.ZL**	Load the MSW with data, zero the LSW and then load the X register of the Input FSBC
0010.0.1	**Ld.InFSBC.Y**	Load the MSW with data and then load the Y register of the Input FSBC
0010.0.0	**Ld.InFSBC.Y.ZL**	Load the MSW with data, zero the LSW and then load the Y register of the Input FSBC
0011.0.1	**Ld.InFSBC.Z**	Load the MSW with data and then load the Z register of the Input FSBC
0011.0.0	**Ld.InFSBC.Z.ZL**	Load the MSW with data, zero the LSW and then load the Z register of the Input FSBC
0100.0.1	**Ld.InFSBC.W**	Load the MSW with data and then load the W register of the Input FSBC
0100.0.0	**Ld.InFSBC.W.ZL**	Load the MSW with data, zero the LSW and then load the W register of the Input FSBC

Table B.16: FSBC/Video — Input FSBC

adr.1.r/*w	Mnemonic	Description
0000.1.1	**R.VC**	read video control register
0000.1.0	**W.VC**	write video control register
0001.1.0	**W.MC.LA**	write video logic array/memory control register
0010.1.0	**W.CLR.VBLANK**	clear vertical blank in status register
0010.1.1	**R.LP.X**	read lightpen X register
0011.1.1	**R.LP.Y**	read lightpen Y register
0100.1.0	**W.CRS.RAM.ADR**	write cursor RAM address register
0101.1.1	**R.SIGNATURE**	read signature register
0110.1.0	**W.BKG.R**	write RED background color
0111.1.0	**W.BKG.GB**	write GREEN & BLUE background color
1000.1.0	**W.CLUTAB.ADR**	write video color lookup table address register
1001.1.1	**R.CLUTAB.R**	read video color lookup table RED
1001.1.0	**W.CLUTAB.R**	write video color lookup table RED
1010.1.1	**R.CLUTAB.G**	read video color lookup table GREEN
1010.1.0	**W.CLUTAB.G**	write video color lookup table GREEN
1011.1.1	**R.CLUTAB.B**	read video color lookup table BLUE
1011.1.0	**W.CLUTAB.B**	write video color lookup table BLUE
1100.1.1	**R.CRS.RAMOW**	read cursor RAM Overlay/White data
1100.1.0	**W.CRS.RAMOW**	write cursor RAM Overlay/White data
1101.1.0	**W.BLINK.RATE**	write blink rate with values in bits 8-15
1101.1.1	**R.BLINK.RATE**	read value of blink counter
1110.1.1	**R.CRS.X**	read cursor X register
1110.1.0	**W.CRS.X**	write cursor X register
1111.1.1	**R.CRS.Y**	read cursor Y register
1111.1.0	**W.CRS.Y**	write cursor Y register

Table B.17: FSBC/Video — Video

Value	Mnemonic	Description
001	**NO.PACK**	no PACK to be asserted (DEFAULT)
101	**IN.PACK**	give *PACK* to the Input-FSBC chip.
011	**PPL.PACK**	give *PACK* to the pixel processor loaders to signal that the command has been read from the FSBC at the bottom of the DDCC.
000	**PP.PACK**	give *PACK* to the pixel processors.

Table B.18: PACK

Value	Mnemonic	Description
0	**IMMED.REG**	immediate data to be from the immediate register
1	**IMMED.MW**	immediate data to be from the micrcode word (DEFAULT)

Table B.19: Immediate — Src

Appendix C

ALU Microcode Reference

Inst	Opcode	SRC-Dest	RAM Src	RAM Dest
WSOR	MOVE	SORA	RS.xx	RD.xx
BSOR	COMP	SORY		
	INC	SORS		
	NEG	SOAR		
		SODR		
		SOIR		
		SOZR		
		SOZER		
		SOSER		
		SORR		

Inst	Opcode	SRC	Dest
WSONR	MOVE	SOA	NRY
BSONR	COMP	SOD	NRA
	INC	SOI	NRS
	NEG	SOZ	NRAS
		SOZE	
		SOSE	

Table C.1: ALU — SINGLE OPERAND INSTRUCTION

Inst	SRC-SRC-Dest	Opcode	RAM Src	RAM Dest
BTOR1	TORAA	SUBR	RS.xx	RD.xx
WTOR1	TORIA	SUBRC		
	TODRA	SUBS		
	TORAY	SUBSC		
	TORIY	ADD		
	TODRY	ADDC		
	TORAR	AND		
	TORIR	NAND		
	TODRR	EXOR		

Inst	SRC-SRC-Dest	Opcode	RAM Src	RAM Dest
BTOR2	TODAR	NOR	RS.xx	RD.xx
WTOR2	TOAIR	OR		
	TODIR	EXNOR		

Inst	SRC-SRC	Opcode	Dest
BTONR	TODA	SUBR	NRY
WTONR	TOAI	SUBRC	NRA
	TODI	SUBS	NRS
		SUBSC	NRAS
		ADD	
		ADDC	
		AND	
		NAND	
		EXOR	
		NOR	
		OR	
		EXNOR	

Table C.2: ALU — TWO OPERAND INSTRUCTIONS

Inst	SRC-Dest	Opcode	RAM Src	RAM Dest
BSHFTR	SHRR	SHUPZ	RS.xx	RD.xx
WSHFTR	SHDR	SHUP1		
		SHUPL		
		SHDNZ		
		SHDN1		
		SHDNL		
		SHDNC		
		SHDNOV		

Inst	SRC	Opcode	Dest
BSHFTNR	SHRR	SHUPZ	NRY
WSHFTNR	SHDR	SHUP1	NRA
		SHUPL	
		SHDNZ	
		SHDN1	
		SHDNL	
		SHDNC	
		SHDNOV	

Table C.3: ALU — SINGLE BIT SHIFT

Inst	n	Opcode	RAM Src	RAM Dest
BBOR1 WBOR1	n	SETNR RSTNR TSTNR	RS.xx	RD.xx
BBOR2 WBOR2	n	LD2NR LDC2NR A2NR S2NR		

Inst	n	Opcode
BBONR WBONR	n	TSTNA RSTNA SETNA A2NA S2NA LD2NA LDC2NA TSTND RSTND SETND A2NDY S2NDY LS2NY LDC2NY

Table C.4: ALU — BIT ORIENTED INSTRUCTIONS

Inst	n	SRC-Dest	RAM Src	RAM Dest
BROTR1 WROTR1	n	RTRA RTRY RTRR	RS.xx	RD.xx

Inst	n	SRC-Dest	RAM Src	RAM Dest
BROTR2 WROTR2	n	RTAR RTDR	RS.xx	RD.xx

Inst	n	Src-Dest
BROTNR WROTNR	n	RTDY RTDA RTAY RTAA

Table C.5: ALU — ROTATE BY n BIT INSTRUCTIONS

Inst	n	U - R/Dest - S	RAM Src	RAM Dest
BROTM	n	MDAI	RS.xx	RD.xx
WROTM		MDAR		
		MDRI		
		MDRA		
		MARI		
		MRAI		

Table C.6: ALU — ROTATE AND MERGE INSTRUCTIONS

Inst	n	U - R - S	RAM Src	RAM Dest
BROTC	n	CDAI	RS.xx	RD.xx
WROTC		CDRI		
		CDRA		
		CRAI		

Table C.7: ALU — ROTATE AND COMPARE INSTRUCTIONS

Inst	Dest	Src	RAM MASK	RAM Dest
BPRT1	PR1A	PRT1A	RS.xx	RD.xx
WPRT1	PR1Y	PR1D		
	PR1R			

Inst	Mask	Dest	RAM Src	RAM Dest
BPRT2	PRA	RP2A	RS.xx	RD.xx
WPRT2	PRZ	PR2Y		
	PRI			

Inst	Mask	Src	RAM Src	RAM Dest
BPRT3	PRA	PR3R	RS.xx	RD.xx
WPRT3	PRZ	PR3A		
	PRI	PR3D		

Inst	Mask	Src	Dest
BPRTNR	PRA	PRTA	NRY
WPRTNR	PRZ	PRTD	NRA
	PRI		

Table C.8: ALU — PRIORITIZE INSTRUCTION

Inst	Opcode
SETST	SONCZ
	SL
	SF1
	SF2
	SF3

Inst	Opcode
RSTST	RONCZ
	RL
	RF1
	RF2
	RF3

Inst	RAM Src	RAM Dest
BSVSTR	RS.xx	RD.xx
WSVSTR		

Inst	Dest
BSVSTNR	NRY
WSBSTNR	NRA

Inst	Opcode
TEST	TNOZ
	TNO
	TZ
	TOVR
	TLOW
	TC
	TZC
	TN
	TL
	TF1
	TF2
	TF3

Table C.9: ALU — STATUS INSTRUCTIONS

Appendix D

Summary of Mnenomics

Instruction Type	
SOR	Single Operand RAM
SONR	Single Operand Non-RAM
TOR1	Two Operand RAM (Quad 0)
TOR2	Two Operand RAM (Quad 2)
TONR	Two Operand Non-RAM
SHFTR	Single Bit Shift RAM
SHFTNR	Single Bit Shift Non-RAM
ROTR1	Rotate n Bits RAM (Quad 0)
ROTR2	Rotate n Bits RAM (Quad 1)
ROTNR	Rotate n Bits Non-RAM
BOR1	Bit Oriented RAM (Quad 3)
BOR2	Bit Oriented RAM (Quad 2)
BONR	Bit Oriented Non-RAM
ROTM	Rotate and Merge
ROTC	Rotate and Compare
PRT1	Prioritize RAM; Type 1
PRT2	Prioritize RAM; Type 2
PRT3	Prioritize RAM; Type 3
PRTNR	Prioritize Non-RAM
CRCF	Cyclic Redundancy Check Forward
CRCR	Cyclic Redundancy Check Forward
NOOP	No Operation
SETST	Set Status
RSTST	Reset Status
SVSTR	Save Status RAM
SVSTNR	Save Status Non-RAM
TEST	Test Status

Table D.1: Summary of Mnemonics (1 of 16)

Source and Destination Single Operand	
SORA	Single Operand RAM to ACC
SORY	Single Operand RAM to Y Bus
SORS	Single Operand RAM to Status
SOAR	Single Operand ACC to RAM
SODR	Single Operand D to RAM
SOIR	Single Operand I to RAM
SOZR	Single Operand O to RAM
SOZER	Single Operand D(OE) to RAM
SOSER	Single Operand D(SE) to RAM
SORR	Single Operand RAM to RAM
SOA	Single Operand ACC
SOD	Single Operand D
SOI	Single Operand I
SOZ	Single Operand O
SOZE	Single Operand D(OE)
SOSE	Single Operand D(SE)
NRY	Non-RAM Y Bus
NRA	Non-RAM ACC
NRS	Non-RAM Status
NRAS	Non-RAM ACC, Status

Table D.2: Summary of Mnemonics (2 of 16)

Source and Destination Two Operand	
TORAA	Two Operand RAM, ACC to ACC
TORIA	Two Operand RAM, I to ACC
TODRA	Two Operand D, RAM to ACC
TORAY	Two Operand RAM, ACC to Y Bus
TORIY	Two Operand RAM, I to Y Bus
TODRY	Two Operand D, RAM to Y Bus
TORAR	Two Operand RAM, ACC to RAM
TORIR	Two Operand RAM, I to RAM
TODRR	Two Operand D, RAM to RAM
TODAR	Two Operand D, ACC to RAM
TOAIR	Two Operand ACC, I to RAM
TODIR	Two Operand D, I to RAM
TODA	Two Operand D, ACC
TOAI	Two Operand ACC, I
TODI	Two Operand D, I

Table D.3: Summary of Mnemonics (3 of 16)

Source and Destination Single Bit Shift	
SHRR	Shift RAM, Store in RAM
SHDR	Shift D, Stor in RAM
SHA	Shift ACC
SHD	Shift D

Table D.4: Summary of Mnemonics (4 of 16)

Source and Destination Rotate n Bits	
RTRA	Rotate RAM, Store in ACC
RTRY	Rotate RAM, Place on Y Bus
RTRR	Rotate RAM, Store in RAM
RTAR	Rotate ACC, Store in RAM
RTDR	Rotate D, Store in RAM
RTDY	Rotate D, Place on Y Bus
RTDA	Rotate D, Store in ACC
RTAY	Rotate AcC, Place on Y Bus
RTAA	Rotate ACC, Store in ACC

Table D.5: Summary of Mnemonics (5 of 16)

Source and Destination Rotate and Merge	
MDAI	Merge Disjoint Bits of D and ACC Using I as Mask and Store in ACC
MDAR	Merge Disjoint Bits of D and ACC Using RAM as Mask and Store in ACC
MDRI	Merge Disjoint Bits of D and RAM Using I as Mask and Store in RAM
MDRA	Merge Disjoint Bits of D and RAM Using ACC as Mask and Store in RAM
MARI	Merge Disjoint Bits of ACC and RAM Using I as mask and Store in RAM
MRAI	Merge Disjoint Bits of RAM and ACC Using I as Mask and store in ACC

Table D.6: Summary of Mnemonics (6 of 16)

Source and Destination Rotate and Compare	
CDAI	Compare Unmasked Bits of D and ACC Using I as Mask
CDRI	Compare Unmasked Bits of D and RAM Using I as Mask
CDRA	Compare Unmasked Bits of D and RAM Using ACC as Mask
CRAI	Compare Unmasked Bits of RAM and ACC Using I as Mask

Table D.7: Summary of Mnemonics (7 of 16)

Source and Destination Prioritize	
PR1A	ACC as Destination for Prioritize Type 1
PR1Y	Y Bus as Destination for Prioritize Type 1
PR1R	RAM as Destination for Prioritize Type 1
PRT1A	ACC as Source for Prioritize Type 1
PR1D	D as Source for Prioritize Type 1
PR2A	ACC as Destination for Prioritize Type 2
PR2Y	Y Bus as Destination for Prioritize Type 2
PR3R	RAM as Source for Prioritize Type 3
PR3A	ACC as Source for Prioritize Type 3
PR3D	D as Source for Prioritize Type 3
PRTA	ACC as Source for Prioritize Type Non-RAM
PRTD	D as Source for Prioritize Type Non-RAM
PRA	ACC as Mask for Prioritize Type 2, 3 and Non-RAM
PRZ	Mask Equal to Zero for Prioritize Type 2, 3 and Non-RAM
PRI	I as Mask for Prioritize Type 2, 3 and Non-RAM

Table D.8: Summary of Mnemonics (8 of 16)

Opcode Addition	
ADD	Add without Carry
ADDC	Add with Carry
A2NA	Add 2^n to ACC
A2NR	Add 2^n to RAM
A2NDY	Add 2^n to D, Place on Y Bus

Table D.9: Summary of Mnemonics (9 of 16)

Opcode Subtraction	
SUBR	Subtract R from S without Carry
SUBRC	Subtract R from S with Carry
SUBS	Subtract S from R without Carry
SUBSC	Subtract S from R with Carry
S2NR	Subtract 2^n from RAM
S2NA	Subtract 2^n from ACC
S2NDY	Subtract 2^n from D, Place on Y Bus

Table D.10: Summary of Mnemonics (10 of 16)

Opcode Logical Operations	
AND	Boolean AND
NAND	Boolean NAND
EXOR	Boolean EXOR
NOR	Boolean NOR
OR	Boolean OR
EXNOR	Boolean EXNOR

Table D.11: Summary of Mnemonics (11 of 16)

Opcode Shifts	
SHUPZ	Shift Up Towards MSB with 0 Insert
SHUP1	Shift Up Towards MSB with 1 Insert
SHUPL	Shift Up Towards MSB with LINK Insert
SHDNZ	Shift Down Towards LSB with 0 Insert
SHDN1	Shift Down Towards LSB with 1 Insert
SHDNL	Shift Down Towards LSB with LINK Insert
SHDNC	Shift Down Towards LSB with Carry Insert
SHDNOV	Shift Down Towards LSB with Sign EXOR Overflow Insert

Table D.12: Summary of Mnemonics (12 of 16)

Opcode Loads	
LD2NR	Load 2^n into RAM
LDC2NR	Load $\overline{2^n}$ into RAM
LD2NA	Load 2^n into ACC
LDC2NA	Load $\overline{2^n}$ into ACC
LD2NY	Place 2^n on Y Bus
LDC2NY	Place $\overline{2^n}$ on Y Bus

Table D.13: Summary of Mnemonics (13 of 16)

Opcode Bit Oriented	
SETNR	Set RAM, Bit n
SETNA	Set ACC, Bit n
SETND	Set D, Bit n
SONCZ	Set OVR, N, C, Z, in Status Register
SL	Set LINK Bit in Stutus Register
SF1	Set Flag1 Bit in Status Register
SF2	Set Flag2 Bit in Status Register
SF3	Set Flag3 Bit in Status Register
RSTNR	Reset RAM, Bit n
RSTNA	Reset ACC, Bit n
RSTND	Reset D, Bit n
RONCZ	Reset OVR, N, C, Z, in Status Register
RL	Reset LINK Bit in Status Register
RF1	Reset Flag1 Bit in Status Register
RF2	Reset Flag2 Bit in Status Register
RF3	Reset Flag3 Bit in Status Register
TSTNR	Test RAM, Bit n
TSTNA	Test ACC, Bit n
TSTND	Test D, Bit n

Table D.14: Summary of Mnemonics (14 of 16)

Opcode Arithmetic Operations	
MOVE	Move and Update Status
COMP	Complement (1's Complement)
INC	Increment
NEG	Two's Complement

Table D.15: Summary of Mnemonics (15 of 16)

Opcode Conditional Test	
TNOZ	Test (N $\oplus$ OVR) + Z
TNO	Test N $\oplus$ OVR
TZ	Test Zero Bit
TOVR	Test Overflow Bit
TLOW	Test for Low
TC	Test Carry Bit
TZC	Test Z = $\bar{C}$
TN	Test Negative Bit
TL	Test LINK Bit
TF1	Test Flag1 Bit
TF2	Test Flag2 Bit
TF3	Test Flag3 Bit

Table D.16: Summary of Mnemonics (16 of 16)