# SHARED MODULE

This info is not so accurate

Triton engineering note section 5

## Figures

## Tables

## Shared Module

There is one shared module in a CRAY T916 or smaller mainframe and two shared modules in a CRAY T932 mainframe. Figure 1 shows the location of these modules in a CRAY T932 mainframe. One shared module is located in each half of the mainframe, and they are referred to as the local and remote shared modules.

All logic for the shared registers is contained on the shared module. The shared modules also include much of the logic for I/O, including the data paths for channel addresses and error information. These modules also synchronize the CPUs when setting the real-time clock (RTC). In addition, they hold the mainframe configuration data specifying which channels and CPUs are assigned to each cluster group.

## Shared Registers

The shared registers provide intermediate data storage and control between CPUs. The shared registers are divided into groups called clusters. Each cluster contains 16 shared B (SB) registers, 16 shared T (ST) registers, and 64 ($100_8$) semaphore (SM) registers. Figure 2 shows the clusters and the registers within each cluster. Each SM register is 1 bit long, and each SB and ST register is 64 bits long. The SB and ST registers pass address (A) and scalar (S) register values between CPUs attached to the same cluster.

CPUs are attached to a cluster by specifying the cluster number (CLN) in the exchange package; or when a CPU is in monitor mode, by issuing a 0014*j*3 instruction to change the CLN. CPUs that do not use the shared registers are attached to cluster 0. If a CPU is attached to cluster 0, and a shared register instruction is issued, all hold issue conditions that would occur for a valid shared register instruction still apply; and any shared register read operations result in the destination register being zeroed out (filled with zeroes).

This section includes information on the operation of the shared registers and the RTC. In some of the diagrams, I/O data paths are also shown, especially when they are shared with the shared registers.

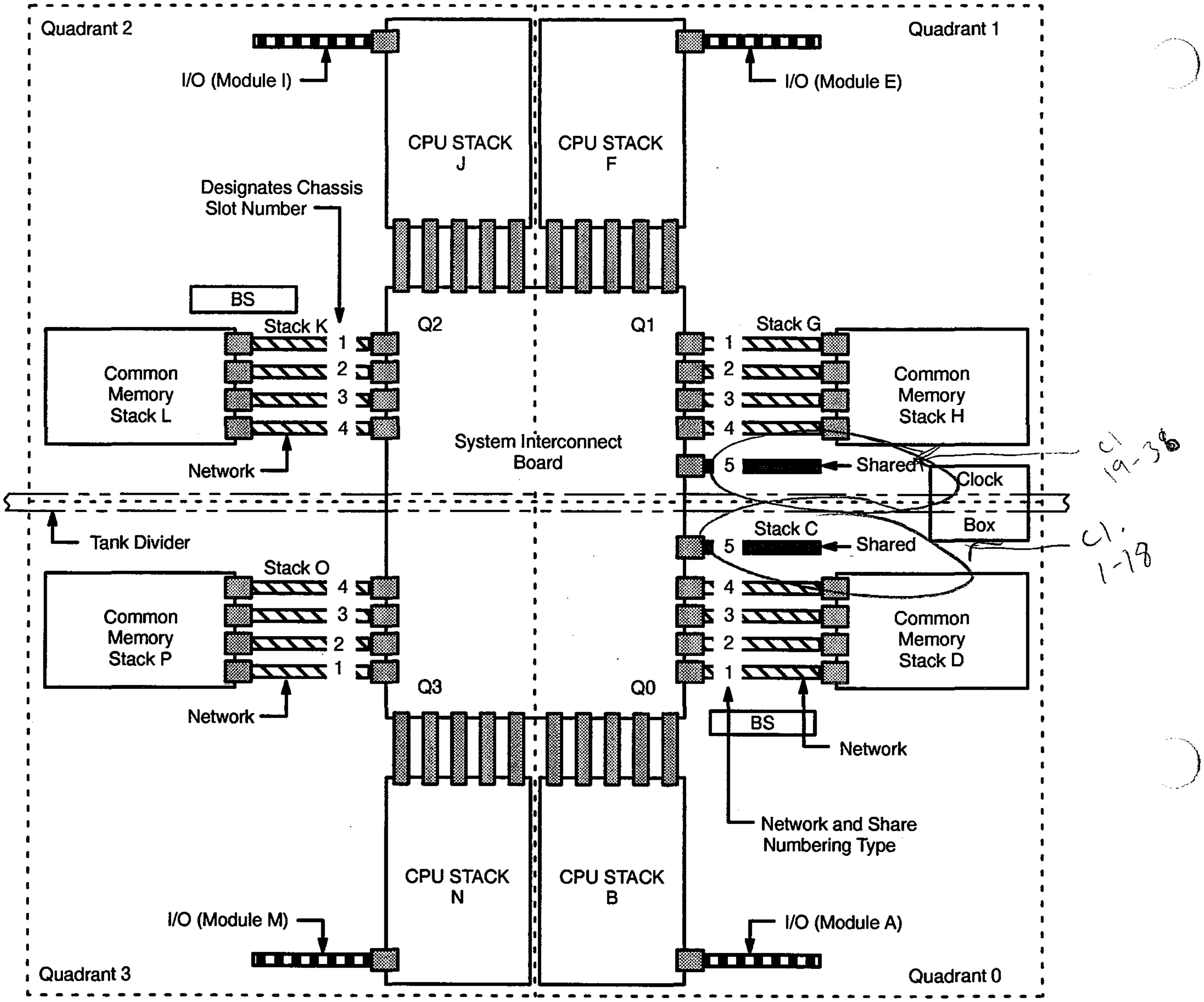
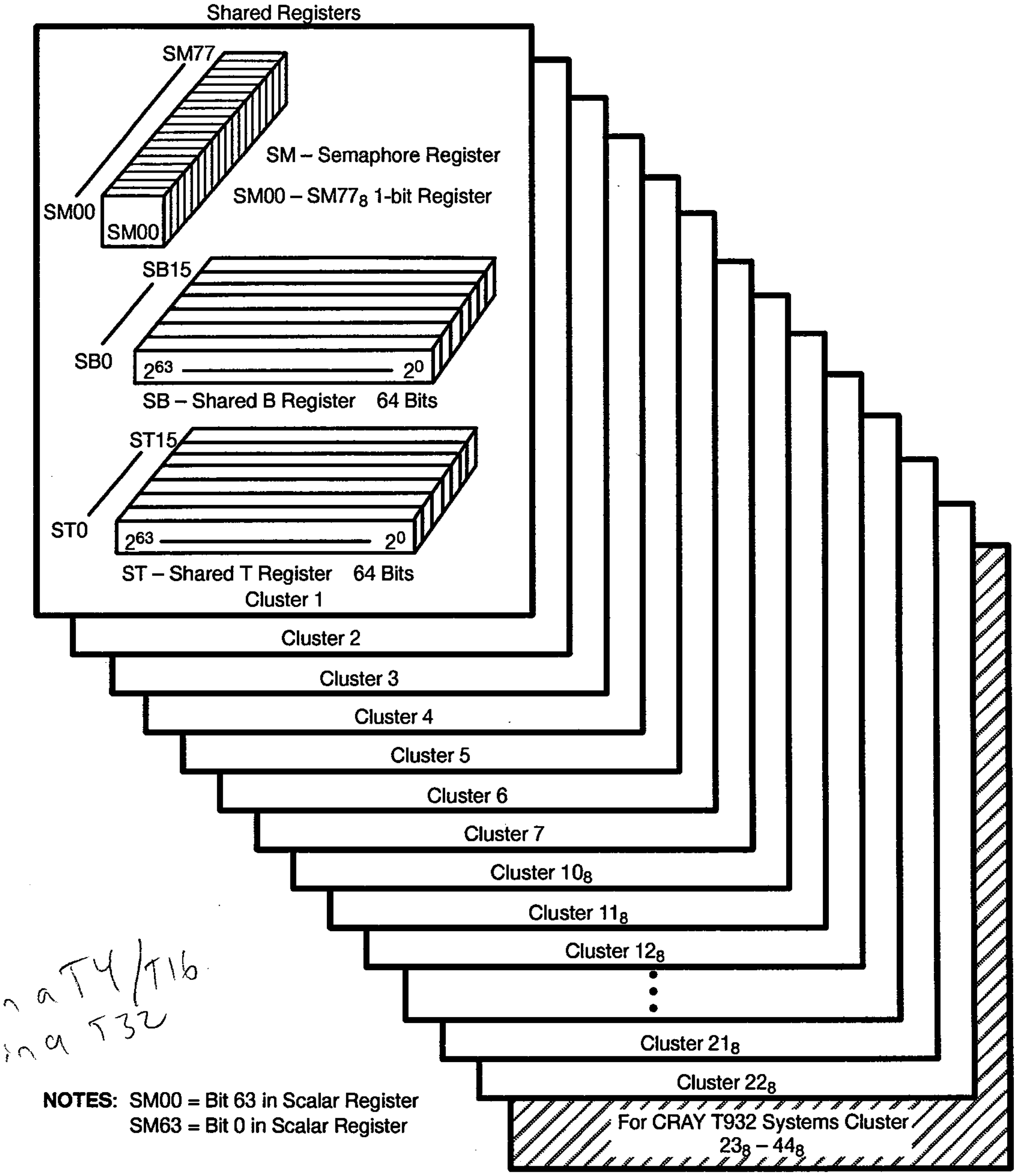


Figure 1. CRAY T932 System Module Locations



18 clusters in a T4/T16
36 clusters in a T32

**NOTES:** SM00 = Bit 63 in Scalar Register
SM63 = Bit 0 in Scalar Register

Figure 2. Shared Registers

## Shared Register Instructions

The shared register instructions are listed in three groups: instructions that move data between the CPU and shared registers, semaphore register instructions, and cluster number (CLN) instructions.

in C90 mode:
The upper 32 bits of SB's are Ø

## SB and ST Register Instructions

| Instruction | CAL | Description |
|---|---|---|
| 026*ij*4 | A*i* SB,A*j*,+1 | Transmit (SB(A*j*)) to A*i*; increment SB(A*j*) by 1 |
| 026*ij*5 | A*i* SB*j*,+1 | Transmit (SB*j*) to A*i*; increment (SB*j*) by 1 |
| 026*ij*6 | A*i* SB,A*j* | Transmit (SB(A*j*)) to A*i* |
| 026*ij*7 | A*i* SB*j* | Transmit (SB*j*) to A*i* |
| 027*ij*6 | SB,A*j* A*i* | Transmit (A*i*) to SB(A*j*) |
| 027*ij*7 | SB*j* A*i* | Transmit (A*i*) to SB*j* |
| 072*ij*3 | S*i* ST*j* | Transmit (ST*j*) to S*i* |
| 072*ij*6 | S*i* ST,A*j* | Transmit (ST(A*j*)) to S*i* |
| 073*ij*3 | ST*j* S*i* | Transmit (S*i*) to ST*j* |
| 073*ij*6 | ST,A*j* S*i* | Transmit (S*i*) to ST(A*j*) |

## Semaphore Instructions

| Instruction | CAL | Description |
|---|---|---|
| 0034*jk* (*j*2=0) | SM*jk* 1, TS | Test and set semaphore *jk* (*jk* = 0 –$37_8$) |
| 0034*jk* (*j*2=1) | SM,A*k* 1, TS | Test and set semaphore (A*k*) |
| 0036*jk* (*j*2=0) | SM*jk* 0 | Clear semaphore *jk* (*jk* = 0 –$37_8$) |
| 0036*jk* (*j*2=1) | SM,A*k* 0 | Clear semaphore (A*k*) |
| 0037*jk* (*j*2=0) | SM*jk* 1 | Set semaphore *jk* (*jk* = 0 –$37_8$) |
| 0037*jk* (*j*2=1) | SM,A*k* 1 | Set semaphore (A*k*) |
| 0064*jknm* (*j*2=0) | JTS*jk* *exp* | Jump to exp if SM*jk* = 1; else set SM*jk* |
| 0064*jknm* (*j*2=1) | JTS,A*k* *exp* | Jump to exp if SM(A*k*) = 1; else set SM(A*k*) |
| 072*i*02 | S*i* SM | Transmit semaphores to S*i* |
| 073*i*02 | SM S*i* | Transmit (S*i*) to semaphores |

### CLN Instructions

| Instruction | CAL | Description |
|---|---|---|
| 0014*j*3 | CLN A*j* | Transmit (A*j*) to cluster number register |
| 001640 | BCD | Broadcast cluster detach Triton mode only |

when you execute this, the syst runs like a C90

## SB and ST Register Operations

The different types of shared register instructions allow data to be transferred between the A registers and the SB registers and between the S registers and the ST registers. Any of the eight A registers can read from or write to any of the 16 SB registers; this is also true for the S registers and ST registers. The shared register instructions differ from other instructions in that the *i* field always designates the A or S register for both read and write operations involving the shared registers. The *j* field is used to specify the SB or ST register. The CRAY T90 series instruction set includes a new instruction, 001640, that releases all cluster assignments previously established by user-mode programs running in the particular CPU.

Indexing is accomplished by using the *j* field of the instruction to specify an A register. The contents of the A register specify the SB or ST register to be used. By incrementing the contents of the A register in a program loop, a series of shared registers can be read from or written to with only a few lines of code. This feature thus enables some programs to be simplified.

The fetch and increment instructions 026*ij*4 and 026*ij*5 work only with the SB registers. When these instructions are issued, the contents of the SB register are read and sent to the A register and the contents of the SB register are incremented by 1.

## Semaphore Operations

The semaphore (SM) registers are a set of sixty-four 1-bit registers used to control program flow among two or more processors (processes) in a multitasking program.

## Reading from and Writing to SM Registers

Each semaphore register may be set or cleared by using instruction 0036*jk* or 0037*jk*, respectively. The SM registers can also be read from or written to as an entire set. Instruction 072*i*02 reads all 64 SM registers to S*i*. SM0 is written to bit 63 of S*i*, and SM63 is written to bit 0. Instruction 073*i*02 writes S*i* to the SM registers.

## Test and Set Instruction

The test and set instruction, 0034*jk*, tests the content of SM*jk* or SM(A*k*), depending on the value of bit 2 of the *j* field. If bit 2 of the *j* field is a 0, the *jk* field of the instruction specifies the SM register; if bit 2 of the *j* field is a 1, the content of register A*k* specifies the SM register to test.

The 0034*jk* instruction tests the content of the SM register to determine whether it is a 1 or a 0. The processor sets the waiting on semaphore flag and holds issue until the SM register is checked. If the SM register is cleared, this instruction sets the SM register, and the processor continues issuing instructions. If the SM register is set, the processor continues to hold issue until the SM register is cleared. The SM register is cleared by some other CPU or process attached to the same cluster in a multitasking program. As soon as the SM register is cleared, it is immediately reset, and the processor continues issuing instructions.

## Test and Set Branch Instruction

The test and set branch instruction, 0064*jkmn*, examines the content of SM*jk* or SM(A*k*), depending on the value of bit 2 of the *j* field. If bit 2 of the *j* field is a 0, the *jk* field of the instruction specifies the SM register; if bit 2 of the *j* field is a 1, the content of register A*k* specifies the SM register to test.

The 0064*jkmn* instruction operates in a similar manner to the 0034*jk* instruction, except that if the SM register is set, the CPU branches to location *mn*.

## Deadlock

A deadlock condition occurs when all CPUs attached to the same cluster are in a hold issue condition on a test and set instruction (0034*jk*) directed to the same SM register. A deadlock condition sets the deadlock interrupt flag if the CPU's IDL (Interrupt on Deadlock) mode bit is set and if the EIM (Enable Interrupt Modes) bit is set.

## CLN Operations

Instruction 0014*j*3 is a monitor-mode instruction that sets the cluster number register on the shared module. Valid cluster numbers are 0 through *n*, where *n* is the number of clusters in the system. Cluster 0 is a *dummy* cluster without registers. An instruction that attempts to read data from a shared register in cluster 0 returns all 0's. An instruction that attempts to write to a shared register in cluster 0 executes as a no-op.

The broadcast cluster detach instruction (001640) releases all cluster attachments previously established by user-mode programs running in the particular CPU. This instruction is normally used only in Triton mode; in C90 mode, cluster attachments are automatically released on every exchange.

When a user-mode program exchanges in, the CLN field in its exchange package specifies the shared register cluster to which it is attached. If the user-mode program exchanges out to a monitor-mode program and a new user is to be brought into the CPU, the monitor program can execute instruction 001640 to release the old cluster attachment.

If a user-mode program exchanges out to another user-mode program, the cluster attachment is not released. The new cluster attachment takes precedence, but the old attachment is still active. Thus, when a monitor-mode program starts, the CPU can be attached to several clusters. Instruction 001640 releases all these attachments.

Shared module configuration functions can enable and disable the capability of having multiple cluster attachments active simultaneously.

## Shared Register Operation

The following subsections describe the flow of data and control signals moving from the CPU to the shared register logic and back. No specific references to clock periods are made and each event is tracked through to its conclusion. You should refer to the numerous figures and tables as you read this subsection.

### Control and Data Flow

Before any instructions pertaining to the operation of the shared module are passed to that module by the CPU, they are first converted on the HF option to either a 6-bit CPU shared command code (ccmd) or a 6-bit shared command code (cmd). The ccmd codes are used for I/O and CPU instructions, and the cmd codes are used for shared register instructions. Table 1 lists the shared command codes for the shared register instructions.

Table 1. Shared Command Codes

| Command Code | Instruction | Description |
|---|---|---|
| c1 | Exchange sequence | Flush |
| c5 | Exchange sequence | Flush |
| c7 | Exchange sequence | Set CLN and MM from exchange |
| c27 | 0014*j*3 | Transmit (A*j*) to cluster number register |
| c30 | 0064*jknm* | Jump to exp if SM*jk* or SM(A*k*)= 1; else set SM |
| c31 | 072*i*02 | Transmit semaphores to S*i* |
| c32 | 001640 | Broadcast cluster detach |
| c33 | 073*i*02 | Transmit (S*i*) to semaphores |
| c34 | 0034*jk* | Test and set SM*jk* or SM(A*k*) |
| c36 | 0036*jk* | Clear SM*jk* or SM(A*k*) |
| c37 | 0037*jk* | Set SM*jk* or SM(A*k*) |
| c41 | 026*ij*4, 026*ij*5 | Transmit (SB*j*) or (SB(A*j*)) to A*i* and increment (SB) by 1 |
| c42 | 026*ij*6, 026*ij*7 | Transmit (SB*j*) or SB(A*j*) to A*i* |
| c43 | 027*ij*6, 027*ij*7 | Transmit (A*i*) to SB*j* or SB(A*j*) |
| c44 | 072*ij*3, 072*ij*6 | Transmit (ST*j*) or (ST(A*j*)) to S*i* |
| c45 | 073*ij*3, 073*ij*6 | Transmit (S*i*) to ST*j* or ST(A*j*) |

Figure 3 shows the data and control paths on the shared module. There are 16 CPUs associated with each shared module, and 18 clusters of shared registers are located on each shared module. The CPUs are grouped on the shared module into four groups, as shown in Table 2. In a CRAY T932 system, CPUs 0 through 15 access a shared module through ports 0 through 15, and CPUs 16 through 31 access the other shared module through ports 0 through 15. In a CRAY T916 system, CPU *x* accesses the shared module through port *x*. In a CRAY T94 system, CPUs 0 through 3 access the shared module through ports 2, 3, 4, and 5.

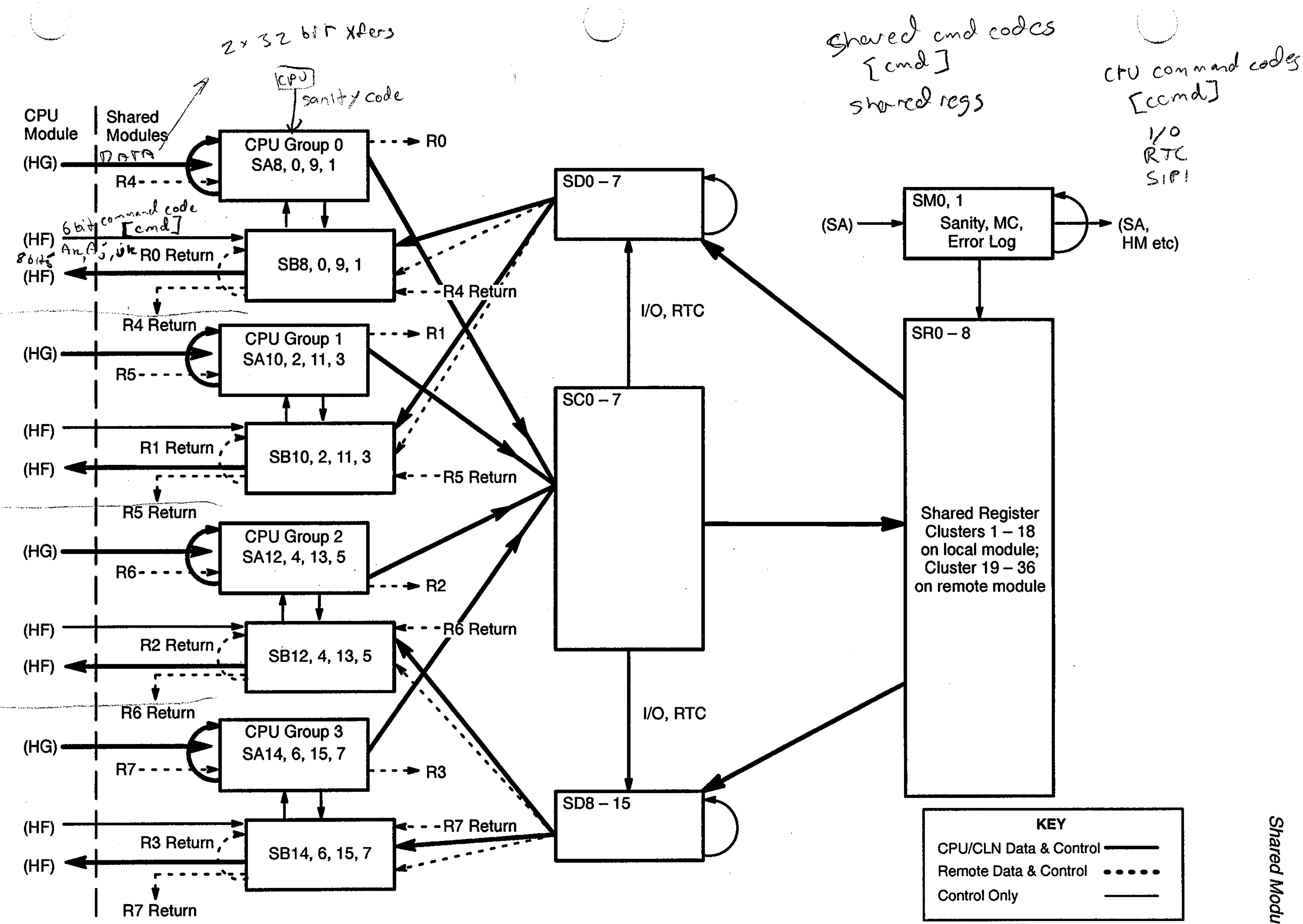


Figure 3. Shared Module Data and Control Paths

Table 2. Shared Module CPU Groups

| CPU Group | Decimal Port No. | Octal Port No. | Binary Port No. |
|---|---|---|---|
| 0 | 8<br>0<br>9<br>1 | 10<br>0<br>11<br>1 | 1 0 0 0<br>0 0 0 0<br>1 0 0 1<br>0 0 0 1 |
| 1 | 10<br>2<br>11<br>3 | 12<br>2<br>13<br>3 | 1 0 1 0<br>0 0 1 0<br>1 0 1 1<br>0 0 1 1 |
| 2 | 12<br>4<br>13<br>5 | 14<br>4<br>15<br>5 | 1 1 0 0<br>0 1 0 0<br>1 1 0 1<br>0 1 0 1 |
| 3 | 14<br>6<br>15<br>7 | 16<br>6<br>17<br>7 | 1 1 1 0<br>0 1 1 0<br>1 1 1 1<br>0 1 1 1 |

The binary representation of the port number shows that bits 1 and 2 determine the group to which the CPU belongs, and that bits 0 and 3 select the particular CPU within that group. CPUs 16 – 31 and clusters 19 – 36 are located on the other shared module, referred to as the remote shared module. When data and controls must be returned to the local shared module from a cluster on the remote shared module, a fifth bit is used in the CPU designator to steer the data and controls to the proper shared module.

There are 16 SA options and 16 SB options on a shared module as shown in Figure 3. For local references, each SA option receives data from the HG option on the CPU with the same number, and each SB option receives controls from the HF option on the CPU with the same number. Thus, SA8 receives data from the HG option on CPU 8, and SB14 receives controls from the HF option on CPU 14. Each SB option also returns both data and controls to the HF option on the CPU with the same number.

For remote references, the data and control paths for CPU group 0 are shown in Figure 4. For other CPU groups, the terminology is similar. The letter R is used to indicate remote data and controls, followed by a number identifying the CPU group of the issuing CPU. Thus, the data and controls leaving CPU group 0 are referred to as R0 data and controls, the data and controls leaving CPU group 1 are referred to as R1 data and controls, and so on. Data and controls entering the corresponding CPU group on the remote shared module are referred to as R4, R5, R6, or R7 data and controls, respectively. When the data and controls leave the remote shared module, they are referred to as R4 Return, R5 Return, R6 Return, or R7 Return data and controls, depending on the CPU group from which they exit the module. The data and controls re-enter the local shared module through the SB options in the same CPU group that initiated the reference; on re-entry, they are referred to as R4 Return, R5 Return, etc. The steering controls are then used to fan the data and other controls into the single SB option that will forward them to its associated CPU. The data and controls enter this single SB option as R0 Return, R1 Return, etc.

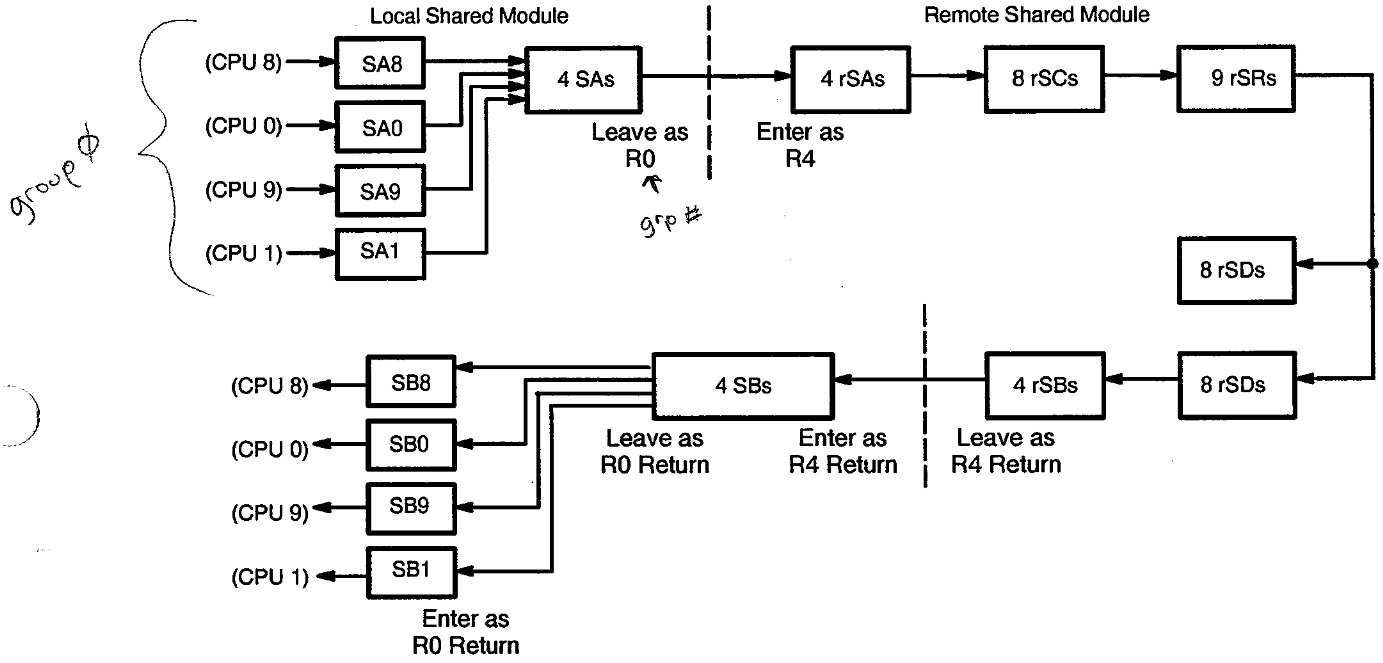


Figure 4. Remote Reference Data and Control Paths for CPU Group 0

In order to describe more closely the operation of the shared registers, the shared register block diagram shown in Figure 5 shows the detail for only the data and control paths for CPU 8 references. References from other CPUs follow the same operational paths and differ only in the numbering of the SA and SB options and the remote data references. A detailed description of the control signal paths for the CPU 8 references is shown in Table 3 through Table 6 on pages 17 through 22. Conflicts may occur on the SA, SC, SD, and SB options in that order as data moves through the module. The conflict resolution logic for each of these options is shown in Figure 6 through Figure 10. A generalized overview of the data and control paths used for remote references is shown in Figure 4.

On shr. module
a group is a group of 4 ports
a port is an SA and SB option
ex. cpu 11 = port 11 = SA11 = S

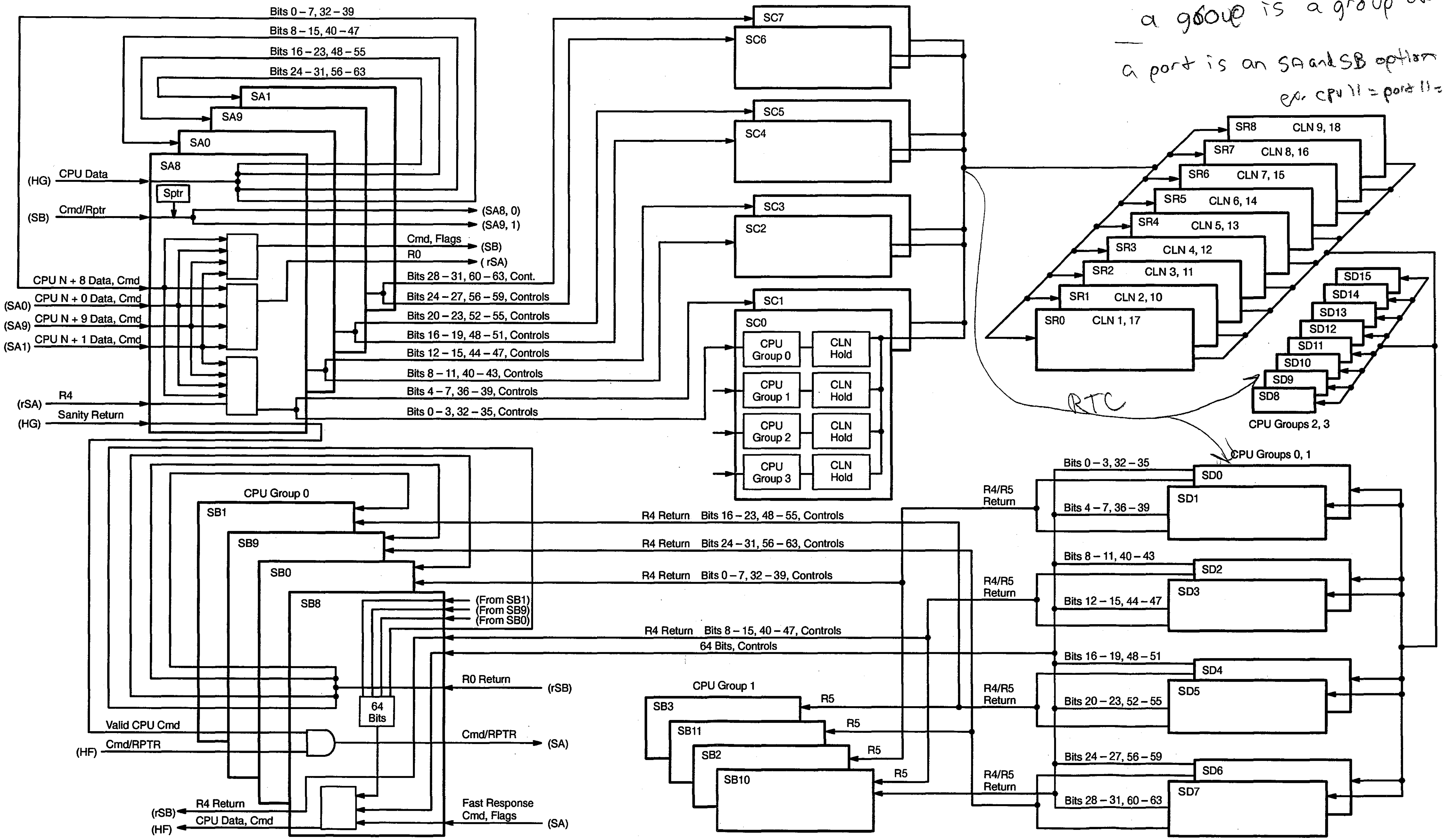


Figure 5. Shared Register Block Diagram

16 SD options SD0-7 = grps 0&1
SD8-15 = grps 2&3

Table 3. Cmd Control Signal Path for CPU 8

| Source Option | Output Terms | Destination Option(s) | Input Terms | Comments |
|---|---|---|---|---|
| HF | ODA – ODF | SB8 | ISA – ISF | 6-bit command code |
| SB8 | ONA – ONH | SA8 | ICA – ICH | Cmd signal uses same 8-bit path as Rptr signal, but one even phase earlier |
| SA8 | OIA – OIF<br>OJA – OJF | SA8, 0<br>SA9, 1 | IIA – IIF | Cmd signal uses same 8-bit path as Sptr/Rptr signal, but one even phase earlier |
| SA8 | OQA – OQF | SC0, 1 | IAA – IAF | 6-bit command code |
| SA0 | OQA – OQF | SC2,3 | IAA – IAF | " |
| SA9 | OQA – OQF | SC4,5 | IAA – IAF | " |
| SA1 | OQA – OQF | SC6,7 | IAA – IAF | " |
| SC0 | OHJ, OIJ,<br>OJJ, OKJ,<br>OLJ, OMJ,<br>ONJ, OOJ,<br>OPJ | SR0, 1,<br>SR2, 3<br>SR4, 5<br>SR6, 7<br>SR8 | IAA | Bit 0 of cmd signal (Each output term is sent to one SR option.) |
| SC1 | OHJ, OIJ,<br>OJJ, OKJ,<br>OLJ, OMJ,<br>ONJ, OOJ,<br>OPJ | SR0, 1,<br>SR2, 3<br>SR4, 5<br>SR6, 7<br>SR8 | IAB | Bit 1 of cmd signal (Each output term is sent to one SR option.) |
| SC2 | OHJ, OIJ,<br>OJJ, OKJ,<br>OLJ, OMJ,<br>ONJ, OOJ,<br>OPJ | SR0, 1,<br>SR2, 3<br>SR4, 5<br>SR6, 7<br>SR8 | IAC | Bit 2 of cmd signal (Each output term is sent to one SR option.) |
| SC3 | OHJ, OIJ,<br>OJJ, OKJ,<br>OLJ, OMJ,<br>ONJ, OOJ,<br>OPJ | SR0, 1,<br>SR2, 3<br>SR4, 5<br>SR6, 7<br>SR8 | IAD | Bit 3 of cmd signal (Each output term is sent to one SR option.) |
| SR0 – 8 | | | | The SR options convert the command code into a special 4-bit code used internally. The original code is passed on to the SD options. |
| SR0 | ODA – ODD<br>OEA – OED | SD0, 1<br>SD2, 3 | IEA – IED | 4-bit code from CLN M+0, N+0 |
| SR1 | ODA – ODD<br>OEA – OED | SD0, 1<br>SD2, 3 | IGA – IGD | 4-bit code from CLN M+1, N+1 |
| SR2 | ODA – ODD<br>OEA – OED | SD0, 1<br>SD2, 3 | IIA – IAD | 4-bit code from CLN M+2, N+2 |

Table 3. Cmd Control Signal Path for CPU 8 (continued)

| Source Option | Output Terms | Destination Option(s) | Input Terms | Comments |
|---|---|---|---|---|
| SR3 | ODA – ODD<br>OEA – OED | SD4, 5<br>SD6, 7 | IKA – IKD | 4-bit code from CLN M+3, N+3 |
| SR4 | ODA – ODD<br>OEA – OED | SD4, 5<br>SD6, 7 | IMA – IMD | 4-bit code from CLN M+4, N+4 |
| SR5 | ODA – ODD<br>OEA – OED | SD4, 5<br>SD6, 7 | IOA – IOD | 4-bit code from CLN M+5, N+5 |
| SR6 | ODA – ODD<br>OEA – OED | SD4, 5<br>SD6, 7 | IQA – IQD | 4-bit code from CLN M+6, N+6 |
| SR7 | ODA – ODD<br>OEA – OED | SD4, 5<br>SD6, 7 | ISA – ISD | 4-bit code from CLN M+7, N+7 |
| SR8 | ODA – ODD<br>OEA – OED | SD4, 5<br>SD6, 7 | IUA – IUD | 4-bit code from CLN M+8, N+8 |
| SD1 | OQA – OQD<br>OSA – OSD<br>OUA – OUD | SD4, 5 | IEA – IED<br>IGA – IGD<br>IIA – IID | 4-bit code from CLN M+0, N+0<br>M+1, N+1<br>M+2, N+2 |
| SD3 | OQA – OQD<br>OSA – OSD<br>OUA – OUD | SD6, 7 | IEA – IED<br>IGA – IGD<br>IIA – IID | 4-bit code from CLN M+0, N+0<br>M+1, N+1<br>M+2, N+2 |
| SD4 | OQA – OQD<br>OSA – OSD<br>OUA – OUD | SD0, 1 | IQA – IQD<br>ISA – ISD<br>IUA – IUD | 4-bit code from CLN M+6, N+6<br>M+7, N+7<br>M+8, N+8 |
| SD5 | OQA – OQD<br>OSA – OSD<br>OUA – OUD | SD0, 1 | IKA – IKD<br>IMA – IMD<br>IOA – IOD | 4-bit code from CLN M+3, N+3<br>M+4, N+4<br>M+5, N+5 |
| SD6 | OQA – OQD<br>OSA – OSD<br>OUA – OUD | SD2, 3 | IQA – IQD<br>ISA – ISD<br>IUA – IUD | 4-bit code from CLN M+6, N+6<br>M+7, N+7<br>M+8, N+8 |
| SD7 | OQA – OQD<br>OSA – OSD<br>OUA – OUD | SD2, 3 | IKA – IKD<br>IMA – IMD<br>IOA – IOD | 4-bit code from CLN M+3, N+3<br>M+4, N+4<br>M+5, N+5 |
| SD0 | OAA – OAF | SB1 | IMA – IMF | 6-bit command code |
| SD1 | OAA – OAF | SB0 | IMA – IMF | " |
| SD2 | OAA – OAF | SB9 | IMA – IMF | " |
| SD3 | OAA – OAF | SB8 | IMA – IMF | " |
| SD4 | OAA – OAF | SB2 | IMA – IMF | " |
| SD5 | OAA – OAF | SB3 | IMA – IMF | " |
| SD6 | OAA – OAF | SB10 | IMA – IMF | " |
| SD7 | OAA – OAF | SB11 | IMA – IMF | " |
| SB8 | OAA – OAF | HF | ICQ – ICV | 6-bit response code |

Table 4. Rptr Control Signal Path for CPU 8

| Source Option | Output Terms | Destination Option | Input Terms | Comments |
|---|---|---|---|---|
| HF | ODM – ODT | SB8 | ITA – ITH | 6-Bit shared register designator from A*j*, A*k*, or *jk* field (Of 8 bits sent, only 6 are used for shared operations.) |
| SB8 | ONA – ONH | SA8 | ICA – ICH | Rptr signal uses same 8-bit path as Cmd signal, but 1 even phase later |
| SA8 | OIG – OIH | SA8, 0 | IIG – IIH | Rptr signal is sent in separate bits to other SA options in 2 even clock phases using the same path. In the first phase, SA8 and SA0 receive bits 0 and 1; in the second phase, they receive bits 4 and 5. |
| SA8 | OJG – OJH | SA9, 1 | IIG – IIH | In the first phase, SA9 and SA1 receive bits 2 and 3; in the second phase, they receive bits 4 and 5. |
| SA0 | OSI – OSJ | SC5 | ICE – ICF | Bits 0 and 1 of Rptr signal |
| SA9 | OSI – OSJ | SC6 | ICE – ICF | Bits 2 and 3 of Rptr signal |
| SA1 | OSI – OSJ | SC7 | ICE – ICF | Bits 4 and 5 of Rptr signal |
| SC5 | OHI – OHJ<br>OII – OIJ<br>OJI – OJJ<br>OKI – OKJ<br>OLI – OLJ<br>OMI – OMJ<br>ONI – ONJ<br>OOI – OOJ<br>OPI – OPJ | SR0<br>SR1<br>SR2<br>SR3<br>SR4<br>SR5<br>SR6<br>SR7<br>SR8 | ICA – ICB | Bits 0 and 1 of Rptr signal |
| SC6 | OHI – OHJ<br>OII – OIJ<br>OJI – OJJ<br>OKI – OKJ<br>OLI – OLJ<br>OMI – OMJ<br>ONI – ONJ<br>OOI – OOJ<br>OPI – OPJ | SR0<br>SR1<br>SR2<br>SR3<br>SR4<br>SR5<br>SR6<br>SR7<br>SR8 | ICC – ICD | Bits 2 and 3 of Rptr signal |
| SC7 | OHI – OHJ<br>OII – OIJ<br>OJI – OJJ<br>OKI – OKJ<br>OLI – OLJ<br>OMI – OMJ<br>ONI – ONJ<br>OOI – OOJ<br>OPI – OPJ | SR0<br>SR1<br>SR2<br>SR3<br>SR4<br>SR5<br>SR6<br>SR7<br>SR8 | ICE – ICF | Bits 4 and 5 of Rptr signal |

Table 5. Sptr(eCLN) Control Signal Path for CPU 8

| Source Option | Output Terms | Destination Option | Input Terms | Comments |
|---|---|---|---|---|
| SA8 | OIA – OIF<br>OJA – OJF | SA8, 0<br>SA9, 1 | IIA – IIF | Sptr = eCLN from SA to SR options; CLN is stored on SA options<br>Sptr signal uses same 6-bit path as Cmd signal, but one even clock phase later |
| SA8 | ORA – ORE | SC0, 1 | IBA – IBE | 5-bit CLN |
| SA0 | ORA – ORE | SC2, 3 | IBA – IBE | " |
| SA9 | ORA – ORE | SC4, 5 | IBA – IBE | " |
| SA1 | ORA – ORE | SC6, 7 | IBA – IBE | " |
| SC4 | OHJ<br>OIJ<br>OJJ<br>OKJ<br>OLJ<br>OMJ<br>ONJ<br>OOJ<br>OPJ | SR0<br>SR1<br>SR2<br>SR3<br>SR4<br>SR5<br>SR6<br>SR7<br>SR8 | IBA | 1-bit cluster pointer; only 1 bit is needed because there are only two clusters on each SR option. |

Table 6. Sptr(Scpu) Control Signal Path for CPU 8

| Source Option | Output Terms | Destination Option | Input Terms | Comments |
|---|---|---|---|---|
| SA8 | OSI | SC0 | ICE | Bit 0 of source cpu (Scpu) signal |
| SA8 | OSJ | SC1 | ICE | Bit 3 of Scpu signal |
| SA8 | OSK | SC2 | ICE | Bit 4 of Scpu signal |
| SC0 | OHI<br>OII<br>OJI<br>OKI<br>OLI<br>OMI<br>ONI<br>OOI<br>OPI | SR0<br>SR1<br>SR2<br>SR3<br>SR4<br>SR5<br>SR6<br>SR7<br>SR8 | IDA | Bit 0 of Scpu signal |

Table 6. Sptr(Scpu) Control Signal Path for CPU 8 (continued)

| Source Option | Output Terms | Destination Option | Input Terms | Comments |
|---|---|---|---|---|
| SC1 | OHI<br>OII<br>OJI<br>OKI<br>OLI<br>OMI<br>ONI<br>OOI<br>OPI | SR0<br>SR1<br>SR2<br>SR3<br>SR4<br>SR5<br>SR6<br>SR7<br>SR8 | IDB | Bit 3 of Scpu signal |
| SC2 | OHI<br>OII<br>OJI<br>OKI<br>OLI<br>OMI<br>ONI<br>OOI<br>OPI | SR0<br>SR1<br>SR2<br>SR3<br>SR4<br>SR5<br>SR6<br>SR7<br>SR8 | IDC | Bit 4 of Scpu signal |
| SC3 | OHI<br>OII<br>OJI<br>OKI<br>OLI<br>OMI<br>ONI<br>OOI<br>OPI | SR0<br>SR1<br>SR2<br>SR3<br>SR4<br>SR5<br>SR6<br>SR7<br>SR8 | IDD | Bit 1 of Scpu signal; SC3 and SC4 generate bits 1 and 2 of the Scpu signal depending on the CPU group sending data to the SC options. |
| SC4 | OHI<br>OII<br>OJI<br>OKI<br>OLI<br>OMI<br>ONI<br>OOI<br>OPI | SR0<br>SR1<br>SR2<br>SR3<br>SR4<br>SR5<br>SR6<br>SR7<br>SR8 | IDE | Bit 2 of Scpu signal |
| SR0 | OFA – OFE<br>OGA – OGE | SD0, 1<br>SD2, 3 | IEQ – IEU | Bits 0, 1, 3, and 4 of Scpu signal from CLN M+0, N+0; bit 2 is not used |
| SR1 | OFA – OFE<br>OGA – OGE | SD0, 1<br>SD2, 3 | IGQ – IGU | Bits 0, 1, 3, and 4 of Scpu signal from CLN M+1, N+1 |
| SR2 | OFA – OFE<br>OGA – OGE | SD0, 1<br>SD2, 3 | IIQ – IAU | Bits 0, 1, 3, and 4 of Scpu signal from CLN M+2, N+2 |
| SR3 | OFA – OFE<br>OGA – OGE | SD4, 5<br>SD6, 7 | IKQ – IKU | Bits 0, 1, 3, and 4 of Scpu signal from CLN M+3, N+3 |

Table 6. Sptr(Scpu) Control Signal Path for CPU 8 (continued)

| Source Option | Output Terms | Destination Option | Input Terms | Comments |
|---|---|---|---|---|
| SR4 | OFA – OFE<br>OGA – OGE | SD4, 5<br>SD6, 7 | IMQ – IMU | Bits 0, 1, 3, and 4 of Scpu signal from CLN M+4, N+4 |
| SR5 | OFA – OFE<br>OGA – OGE | SD4, 5<br>SD6, 7 | IOQ – IOU | Bits 0, 1, 3, and 4 of Scpu signal from CLN M+5, N+5 |
| SR6 | OFA – OFE<br>OGA – OGE | SD4, 5<br>SD6, 7 | IQQ – IQU | Bits 0, 1, 3, and 4 of Scpu signal from CLN M+6, N+6 |
| SR7 | OFA – OFE<br>OGA – OGE | SD4, 5<br>SD6, 7 | ISQ – ISU | Bits 0, 1, 3, and 4 of Scpu signal from CLN M+7, N+7 |
| SR8 | OFA – OFE<br>OGA – OGE | SD4, 5<br>SD6, 7 | IUQ – IUU | Bits 0, 1, 3, and 4 of Scpu signal from CLN M+8, N+8 |
| SD1 | ORA – ORE<br>OTA – OTE<br>OVA – OVE | SD4, 5 | IEQ – IEU<br>IGQ – IGU<br>IIQ – IIU | 4 bits of Scpu from CLN M+0, N+0<br>M+1, N+1<br>M+2, N+2 |
| SD3 | ORA – ORE<br>OTA – OTE<br>OVA – OVE | SD6, 7 | IEQ – IEU<br>IGQ – IGU<br>IIQ – IIU | 4 bits of Scpu from CLN M+0, N+0<br>M+1, N+1<br>M+2, N+2 |
| SD4 | ORA – ORE<br>OTA – OTE<br>OVA – OVE | SD0, 1 | IQQ – IQU<br>ISQ – ISU<br>IUQ – IUU | 4 bits of Scpu from CLN M+6, N+6<br>M+7, N+7<br>M+8, N+8 |
| SD5 | ORA – ORE<br>OTA – OTE<br>OVA – OVE | SD0, 1 | IKQ – IKU<br>IMQ – IMU<br>IOQ – IOU | 4 bits of Scpu from CLN M+3, N+3<br>M+4, N+4<br>M+5, N+5 |
| SD6 | ORA – ORE<br>OTA – OTE<br>OVA – OVE | SD2, 3 | IQQ – IQU<br>ISQ – ISU<br>IUQ – IUU | 4 bits of Scpu from CLN M+6, N+6<br>M+7, N+7<br>M+8, N+8 |
| SD7 | ORA – ORE<br>OTA – OTE<br>OVA – OVE | SD2, 3 | IKQ – IKU<br>IMQ – IMU<br>IOQ – IOU | 4 bits of Scpu from CLN M+3, N+3<br>M+4, N+4<br>M+5, N+5 |
| SD0 | OBA – OBB | SB1 | INA – INB | Bits 0 and 3 of Scpu signal |
| SD1 | OBA – OBB | SB0 | INA – INB | " |
| SD2 | OBA – OBB | SB9 | INA – INB | " |
| SD3 | OBA – OBB | SB8 | INA – INB | " |
| SD4 | OBA – OBB | SB2 | INA – INB | " |
| SD5 | OBA – OBB | SB3 | INA – INB | " |
| SD6 | OBA – OBB | SB10 | INA – INB | " |
| SD7 | OBA – OBB | SB11 | INA – INB | " |

## Shared Register Write Operations

A shared register write operation starts when the shared command code (Cmd) and the destination register pointer (Rptr) are sent from the HF option on CPU 8 to the SB8 option. The HF option constructs the Rptr signal from data contained in the A*j* or A*k* register or the *jk* field of the instruction. If SB8 receives a valid command code from the SA8 option indicating that CPU 8 has returned a valid sanity code to the SA option, then the command and Rptr are forwarded to the SA8 option. These two control signals are sent over the same path to the SA option in two successive even clock phases, with the Cmd signal followed by the Rptr signal.

Some of the shared command codes are changed on the SA8 option. These are mostly codes for I/O instructions and the real-time clock. For example, 024, which is the code for a 0014*j*0 RTC instruction, is changed to [illegible]. None of the codes for the shared register operations are changed. The final command code to be used is then checked to verify that it is a valid code for the shared module.

At the same time the Cmd and Rptr signals enter the SA8 option, the data from the HG option on CPU 8 also enters the SA8 option. The data arrives 32 bits at a time, in two successive even clock phases over the same data paths. If the command code is valid, this data is then split into 16-bit slices, which are sent 8 bits at a time in successive even clock phases to each of the four SA options in CPU group 0.

Before the Cmd and Rptr signals are sent to the other SA options, the Sptr (steering pointer) control signal is generated on the SA8 option. The Sptr signal used between the SA and SR options is the effective cluster number (eCLN) of the destination shared register.

The eCLN is generated on the SA8 option in a sequence of steps. The assigned cluster number for CPU 8 is first read from a holding register on the option and added to the cluster offset that was set during system startup. The resulting CLN is then checked to be sure it is between 1 and 36 inclusive for a system with two shared modules or between 1 and 18 inclusive for a system with one shared module. The logic also checks that the assigned CLN for CPU 8 is less than the cluster range set during system startup. If either of these tests fails, the eCLN is set to 77 to denote an illegal Sptr signal. If the resulting CLN is not illegal, then a 6-bit eCLN is generated from the CLN as shown in Table 7.

The three control signals (Cmd, Rptr, and Sptr) are sent to the other SA options over two paths. One path sends the controls to the SA8 and SA0 options, and the other path sends the controls to the SA9 and SA1 options. Both paths send the controls during two successive even clock phases.

The Cmd signal and 4 bits of the Rptr signal are sent in the first clock phase, followed by the Sptr signal and the remaining 2 bits of the Rptr signal. Refer to Table 4 for the distribution of the Rptr bits to each option.

Table 7. eCLN Generation from CLN

| eCLN Bit | CLN (Decimal) | CLN (Octal) |
|---|---|---|
| 5 | 19 – 36 | 23 – 44 |
| 4 | 9 or 18<br>27 or 36 | 11 or 22<br>33 or 44 |
| 3 | 5, 6, 7, 8, 13, 14, 15, or 16<br>21, 22, 23, 24, 29, 30, 31, or 32 | 5, 6, 7, 10, 15, 16, 17, or 20<br>25, 26, 27, 30, 35, 36, 37, or 40 |
| 2 | 3, 4, 7, 8, 11, 12, 15, or 16<br>19, 20, 23, 24, 28, 31, 32, or 35 | 3, 4, 7, 10, 13, 14, 17, or 20<br>23, 24, 27, 30, 34, 37, 40, or 43 |
| 1 | 2, 4, 6, 8, 10, 12, 14, or 16<br>20, 22, 24, 26, 28, 30, 32, or 34 | 2, 4, 6, 10, 12, 14, 16, or 20<br>24, 26, 30, 32, 34, 36, 40, or 42 |
| 0 | 1, 3, 5, 7, 9, 10, 12, 14, or 16<br>19, 21, 23, 25, 27, 28, 30, 32, or 34 | 1, 3, 5, 7, 11, 12, 14, 16, or 20<br>23, 25, 27, 31, 33, 34, 36, 40, or 42 |

When the data and controls enter all the SA options, they are placed in separate FIFO (first in, first out) queues, as shown in Figure 6. As the data and controls move through the FIFO ranks A, B and C, they can be held at any time because of a conflict at ranks M and N. Note that I/O interrupts move immediately to rank C in a separate FIFO queue.

As the data and controls leave rank C, bit 5 of the Sptr(eCLN) signal is checked to determine whether the data is intended for a cluster in the remote shared module. If so, it is steered to an output buffer where it must compete with the other CPUs for the output path. Initially, the priority is CPU 8, CPU 0, CPU 9, and CPU 1. Whenever a path is selected to proceed, its priority is moved to the lowest level, while still maintaining the cyclic order. Thus, if CPU 0 makes a reference, the new order of priority would be CPU 9, CPU 1, CPU 8, and CPU 0.

When the data and controls are ready to enter rank M, they are joined in vying for a slot in rank M by controls and data from the remote shared module. The potential conflict to enter rank M is resolved by assigning a priority to each of the five input paths. Initially, the priority is CPU 8, CPU 0, CPU 9, CPU 1, and R4 (remote reference). Whenever a path is selected to proceed, its priority is moved to the lowest level, while still maintaining the cyclic order. Thus, if CPU 0 makes a reference, the new order of priority would be CPU 9, CPU 1, R4, CPU 8, and CPU 0.

Before the data and controls move from rank C to rank M, a check is first made to be sure the fast response output buffer for CPU 8 is not busy. This buffer is used to send a copy of the command code together with five flags to the SB8 option. The functions of the flags are shown in Table 8. This information is collectively referred to as the *fast response data.* If the fast response output buffer for CPU 8 is busy, it will prevent the data and controls from advancing from rank C to rank M.

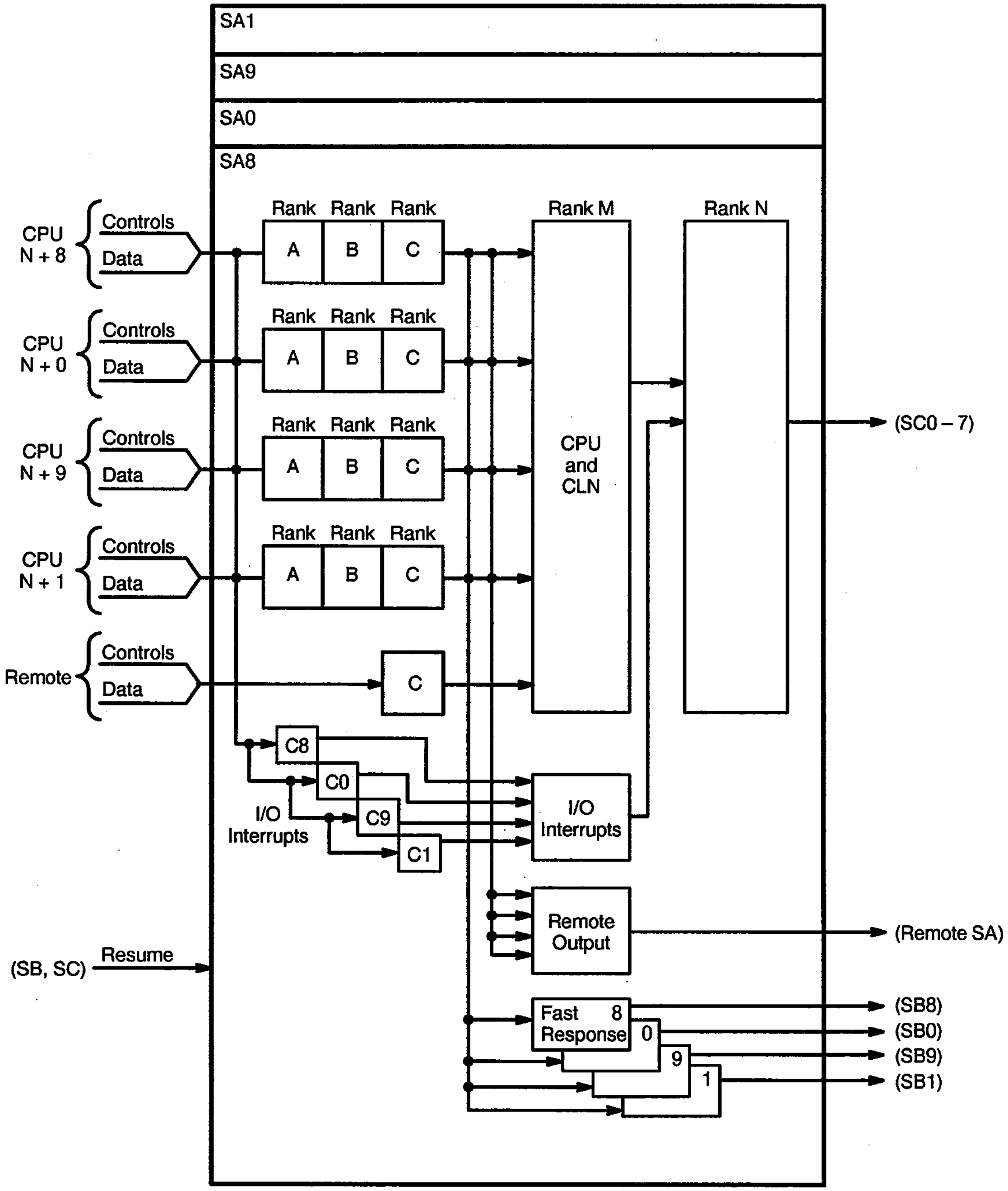


Figure 6. SA Option Operations

When the fast response data enters the SB8 option, the Cmd signal is decoded to determine the type of shared register reference. For all valid write references and for valid read references with an illegal Sptr signal, the command code is forwarded to the HF option on CPU 8. There, it is used to release the reservation on the shared register path allowing additional references from CPU 8.

Table 8. Fast Response Flags

| Bit Position | Function |
|---|---|
| 0 | Valid reference |
| 1 | Not used |
| 2 | Illegal Sptr |
| 3 | Command code c1 |
| 4 | Bit 0 of Sptr |

When the data and controls are ready to enter rank N, they must compete with I/O interrupts. Priority is given to the I/O interrupts.

Before the data and controls leave the SA options for the SC options, the Scpu (source CPU) control signal must be generated. The Scpu signal is a 5-bit signal identifying the CPU that issued the instruction and the CPU to which any read data must be returned. Three bits of this signal are read from hard-wired position constants on the SA8 option and are output as shown in Table 6 on page 20. The SC3 and SC4 options each generate one of the remaining 2 bits.

Each SA option sends 8 bits of data to each of two SC options. The data is sent in two successive clock phases over different paths, with 4 bits sent during each phase. The block diagram shows the distribution of data bits to the SC options. Of the four control signals (Cmd, Rptr, Sptr, and Scpu), the Cmd and Sptr signals are sent from SA8 to SC0 and SC1, from SA0 to SC2 and SC3, etc. The Rptr and Scpu signals are sent in separate bits from the SA options to specific SC options. Table 4 on page 19 and Table 6 on page 20 show the distribution of these bits.

The SC options receive data and controls from each of the four CPU groups and route it to one of the nine SR options. The internal operations of the SC options are shown in Figure 7. When the data and controls for a reference to a particular cluster enter the SC options, they are gated into one of three input buffers for the appropriate CPU group. The buffers are labeled A, B, and C. Buffer A has the highest priority. If it is full, the data and controls are routed to buffer B, and so on. As buffer A empties, data and controls in buffers B and C move up to buffers A and B. I/O and RTC references are handled in the same manner, but use only two buffers.

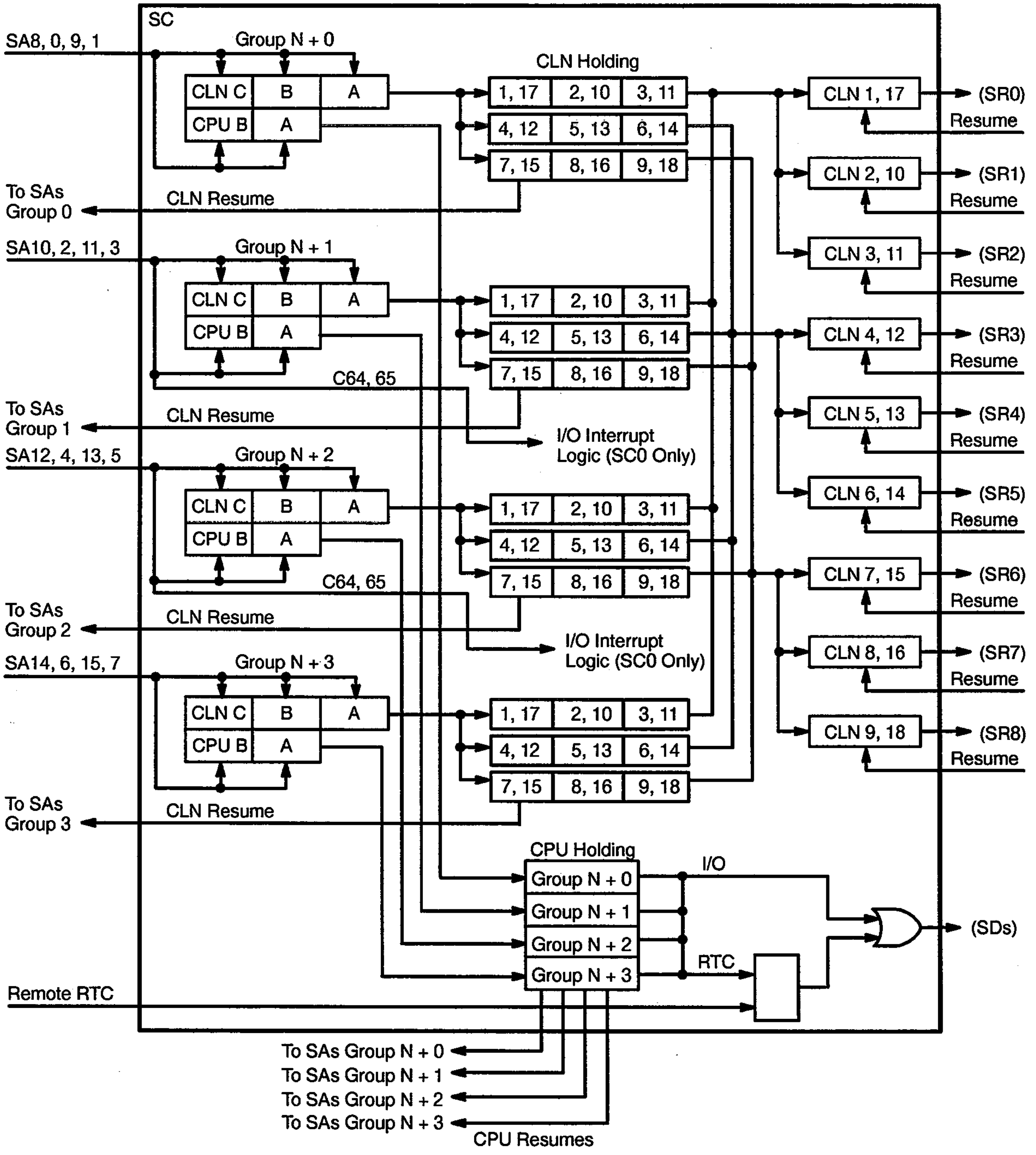


Figure 7. SC Option Operations

Before the control signals leave buffer A, the SC3 and SC4 options generate bits 1 and 2 of the Scpu signal. These bits identify the CPU group from which the data is arriving and are determined from the input path used.

As the data and control signals leave buffer A, bits 1 through 4 of the Sptr signal are used to steer them into one of three CLN holding buffers. Anytime data is latched into a holding buffer on an option, a Resume signal is sent to the option from which the data came. This allows other references to move through the sending option.

Each of the CLN holding buffers holds data and controls intended for one of six clusters. Each buffer is subdivided into three output buffers, with each output buffer holding data and controls for the two clusters contained on one of the SR options. Table 9 shows the clusters contained on each of the SR options. Each CLN holding buffer can hold the data and controls for only a single reference to one of its six clusters. Thus, only one of the three output buffers in each holding buffer can be full at any given time.

Table 9. SR Option Cluster Assignment

| Option | | Clusters (Decimal) | Clusters (Octal) |
|---|---|---|---|
| SR0 | Local | 1,17 | 1, 21 |
| | Remote | 25, 33 | 31, 41 |
| SR1 | Local | 2, 10 | 2, 12 |
| | Remote | 26, 34 | 32, 42 |
| SR2 | Local | 3, 11 | 3, 13 |
| | Remote | 19, 35 | 23, 43 |
| SR3 | Local | 4, 12 | 4, 14 |
| | Remote | 20, 28 | 24, 34 |
| SR4 | Local | 5, 13 | 5, 15 |
| | Remote | 21, 29 | 25, 35 |
| SR5 | Local | 6, 14 | 6, 16 |
| | Remote | 22, 30 | 26, 36 |
| SR6 | Local | 7, 15 | 7, 17 |
| | Remote | 23, 31 | 27, 37 |
| SR7 | Local | 8, 16 | 10, 20 |
| | Remote | 24, 32 | 30, 40 |
| SR8 | Local | 9, 18 | 11, 22 |
| | Remote | 27, 36 | 33, 44 |

Because up to four CPU groups can vie for a single output path to a particular SR option, conflicts can occur. These are resolved by assigning a priority to each of the four CPU groups. Initially, the priority is CPU group 0, CPU group 1, CPU group 2, and CPU group 3. Whenever a CPU group is selected to proceed, its priority is moved to the lowest level, while still maintaining the cyclic order. Thus, if CPU group 1 makes a reference to a particular SR option, the new order of priority would be CPU group 2, CPU group 3, CPU group 0, and CPU group 1. Each output path to an SR option has its own priority circuit which operates independently of the other paths.

Each SC option sends its 8 bits of data to the selected SR option in 2 even clock phases over two separate paths. The control signals are sent 1 even clock phase earlier, with each SC option sending 2 bits of the 16 total control bits needed. All 6 bits of the Rptr signal and all 5 bits of the Scpu signal are sent to the SR option. Only the lower 4 bits of the Cmd signal are sent, because 4 bits are sufficient to determine one of the Cmd codes exactly. Only bit 0 of the Sptr signal is sent to the SR option, because each SR option contains only two clusters, and a single bit can select the correct one.

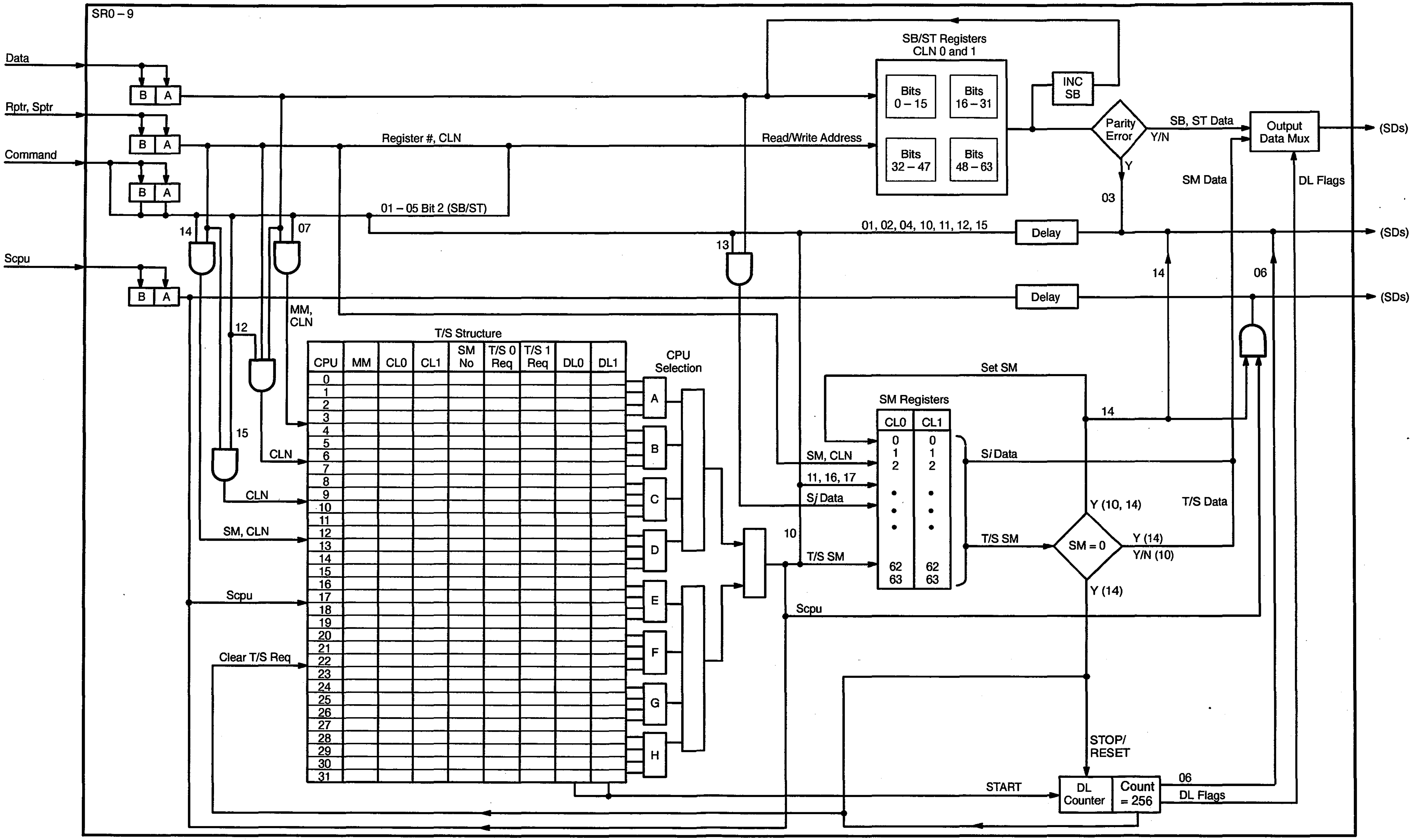
When the data and the Rptr, Sptr, and Scpu control signals enter the SR option, they are gated into one of two input buffers labeled A and B, as shown in Figure 8. Buffer A has the highest priority. If it is full, the data and controls are routed to buffer B. As buffer A empties, the data and controls in buffer B move up to buffer A. The Cmd control signal also uses two buffers, but the Cmd signal can be read out of either buffer or directly from the input signal. Before the Cmd signal is used on the SR option, it is converted into a special 4-bit code that is used for internal control on the option. Table 10 lists these special SR option codes.

The 6-bit read/write address for SB and ST register references is generated from the Rptr and Sptr signals and the SR option code. The lower 4 bits of the Rptr signal contain the register number, the 1-bit Sptr signal determines the cluster, and bit 2 of the SR option code selects between an SB or an ST reference. The 2 upper bits of the Rptr signal steer the read/write address to one of the four high-speed register storage arrays containing the SB and ST registers. These storage arrays are 18 bits wide by 64 bits deep. They store 16 data bits and 1 parity bit for each of the 16 SB and 16 ST registers in each of the two clusters contained on the particular SR option.

For SM register read/write operations to a single register, the 6 bits of the Rptr signal select one of the 64 SM registers, and the 1-bit Sptr signal selects the proper cluster. For the SM register broadside write operation, the SR option code selects all 64 SM registers simultaneously and gates the S*j* data to them, and the 1-bit Sptr signal selects the proper cluster.

Table 10. SR Option Codes

| SR Option Code | Command Code | Description |
|---|---|---|
| 01 | c41 | Transmit (SB*j*) or (SB(A*j*)) to A*i* and increment (SB) by 1 |
| 02 | c42 | Transmit (SB*j*) or SB(A*j*) to A*i* |
| 03 | c43 | Transmit (A*i*) to SB*j* or SB(A*j*) |
| 03 | | RAM parity error (generated on SR option) |
| 04 | c44 | Transmit (ST*j*) or (ST(A*j*)) to S*i* |
| 05 | c45 | Transmit (S*i*) to ST*j* or ST(A*j*) |
| 06 | | Deadlock interrupt (generated on SR option) |
| 07 | c7 | Attach CPU to CLN (from exchange) |
| 10 | c30 | Jump to exp if SM*jk* or SM(A*k*)= 1; else set SM |
| 11 | c31 | Transmit semaphores to S*i* |
| 12 | c32 | Broadcast cluster detach |
| 13 | c33 | Transmit (S*i*) to semaphores |
| 14 | c34 | Test and set SM*jk* or SM(A*k*) |
| 15 | c5 | Flush |
| 16 | c36 | Clear SM*jk* or SM(A*k*) |
| 17 | c37 | Set SM*jk* or SM(A*k*) |



| CPU | MM | CL0 | CL1 | SM No | T/S 0 Req | T/S 1 Req | DL0 | DL1 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | |
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | | | | | | | |
| 9 | | | | | | | | |
| 10 | | | | | | | | |
| 11 | | | | | | | | |
| 12 | | | | | | | | |
| 13 | | | | | | | | |
| 14 | | | | | | | | |
| 15 | | | | | | | | |
| 16 | | | | | | | | |
| 17 | | | | | | | | |
| 18 | | | | | | | | |
| 19 | | | | | | | | |
| 20 | | | | | | | | |
| 21 | | | | | | | | |
| 22 | | | | | | | | |
| 23 | | | | | | | | |
| 24 | | | | | | | | |
| 25 | | | | | | | | |
| 26 | | | | | | | | |
| 27 | | | | | | | | |
| 28 | | | | | | | | |
| 29 | | | | | | | | |
| 30 | | | | | | | | |
| 31 | | | | | | | | |

Figure 8. SR Option Operations

## Shared Register Read Operations

A shared register read operation begins the same way as a write operation and goes through most of the same steps as it moves from the CPU through the SA, SC and SR options. The only difference in this first part of the read operation is that the fast response command code is not sent to the HF option unless an illegal Sptr signal is detected. In a read operation, the command code is usually returned to the CPU along with the requested data, and only then is the CPU shared path reservation released. The major differences in the read operation begin exactly where the write operation stopped – in the shared registers on the SR options.

For every SB/ST register read operation, 4 parity bits are generated, one for each 16-bit block of data read from each of the four storage arrays. Each parity bit is compared to the one generated when the data was written into the array. If all 4 parity bits are identical to those stored in the arrays, the data is valid. If any of the 4 parity bits differs from the parity bit stored in its array, a parity error code of 03 is generated. This code is sent to the SD options, where it is converted to a Cmd code of c43. In either case, the readout data is sent to the SD options. Note that the Cmd code of c43 is the same as that used for a write operation to the SB registers. However, for the write operation, no Cmd code is sent to the SD options.

For a broadside SM register read operation, all 64 SM registers are read simultaneously, and the readout data is treated as a single word. Test and set operations are discussed in the next section.

Two control signals accompany the data as it makes its way through the SD and SB options en route to CPU 8. One is the Cmd code, and the other is the Scpu signal. Because the Scpu signal is used to steer the data and Cmd signal to CPU 8, it is renamed the Sptr(scpu) signal, referred to simply as the Sptr signal. This is not to be confused with the Sptr(eCLN) signal used to steer write data to the SR option.

The SR option sends 8 bits of data to each of the 16 SD options. The data is sent 4 bits at a time in 2 successive even clock phases. The distribution of the data bits is shown in Figure 5 on page 15. When the data enters the SD options, it is placed in input buffer B, from where it must first move to input buffer A before going further, as shown in Figure 9.

The SR option also sends the SR option code and Sptr signals to four SD options in CPU groups 0, 1 and four SD options in CPU groups 2, 3. The distribution of the SR option code and Sptr signals to the SD options in CPU groups 0, 1 is shown in Table 3 on page 18 and Table 6 on pages 21 and 22. Only bits 0, 1, 3, and 4 of the Sptr signal are used, because bit 2 was already used to distinguish between CPU groups 0, 1 and CPU groups 2,3, as shown in Table 2.

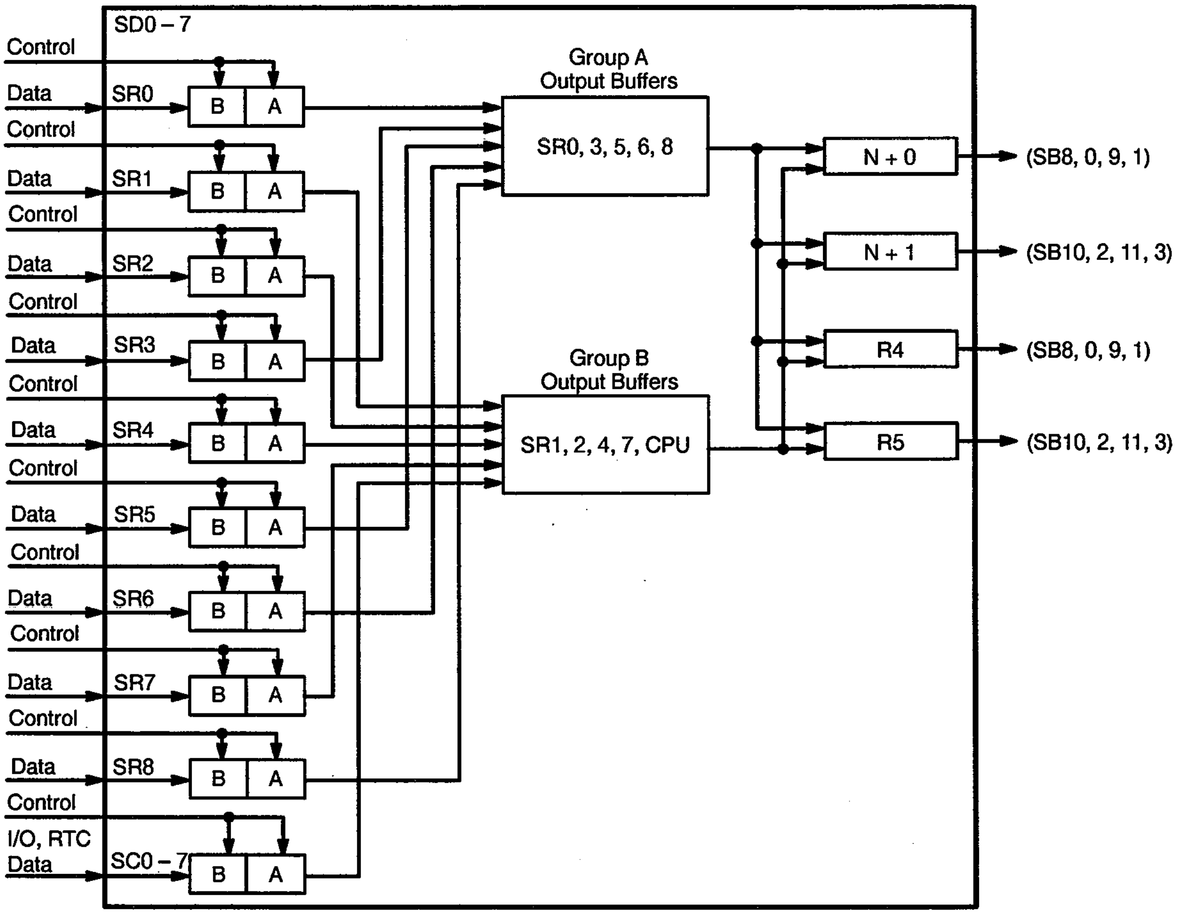


Figure 9. SD Option Operations

The two control signals are immediately fanned out from the four SD options to the remaining SD options, as shown in Table 3 on page 18 and Table 6 on page 22. The control signals, however, can be gated into either input buffer A or input buffer B. In either case, just before the 4-bit SR option code is latched in, it is converted back to the original 6-bit Cmd signal that left CPU 8. This is done by using bit 3 of the signal to set bits 4 and 5 to 1 0 or 0 1, depending on whether the code should be 40 – 47 or 30 – 37, respectively.

As the data and controls move out of input buffer A, they are gated into one of 10 output holding buffers corresponding to each of the input buffers. Because there is only one output path to CPU group 0 for a local reference and ten output buffers, the output buffers are divided into two groups, as shown in Figure 9. A priority is then set for each of the buffers in group A and each of the buffers in group B. Initially, the priority for each of the groups is as shown in Figure 9. Whenever a buffer is selected to send its data and controls, its priority is moved to the lowest level, while still maintaining the cyclic order. Thus, if SR5 makes a reference, the new order of priority would be SR6, SR8, SR0, SR3, and SR5. Priority also alternates between each of the two groups.

Each output path competes separately for information from one of the output buffers. Bits 1 and 4 of the Sptr signal select one of the four output paths. Bit 1 selects CPU group 0 or 1, and bit 4 selects the local or remote return path. R4, for example, denotes the read reference was made from a CPU in group 0 on the other shared module.

For a local read reference using the N + 0 output path, each of the SD options sends 8 bits of data to each of the SB options. The data is sent 4 bits at a time in 2 successive even clock phases over different output paths. Thus, each of the SB options receives all 64 bits of the data. The 6-bit Cmd signal and the 2-bit Sptr signal are sent from each of the SD options to one of the SB options. The distribution of these bits is shown in Table 3 and Table 6.

For a remote read reference using the R4 output path, each of two SD options sends 8 bits of data to a single SB option. The data is sent 4 bits at a time in 2 successive even clock phases over different output paths. Thus, each SB option receives 16 bits of data. The 6-bit R4 Cmd signal and the 2-bit R4 Sptr signal are sent from each of the SD options to one of the SB options. The data and controls are immediately sent to the corresponding remote SB options in CPU group 0 on the other shared module. The R4 data is sent 8 bits at a time in 2 successive even clock phases. The incoming data and control signals are referred to as R4 Return data and controls. This information is immediately fanned out to the other SB options in the group as R0 data and controls, so that each SB option can reconstruct the 64-bit data word.

The 2 remaining bits of the Sptr signal select one of the four CPUs in the group to receive the data and Cmd signal. Because there is only one output path to the CPU from the SB option and three sources of CPU data, a conflict may occur, as shown in Figure 10. Initially, the priority for each of the sources is Remote Response, Fast Response, and Local CLN Response. Whenever data from one of these sources is selected, its priority is moved to the lowest level, while still maintaining the cyclic order.

The data is sent to the HF option 32 bits at a time in 2 successive even clock phases over the same output path. The Cmd signal is also sent to the HF option to release the reservation on the CPU shared path.

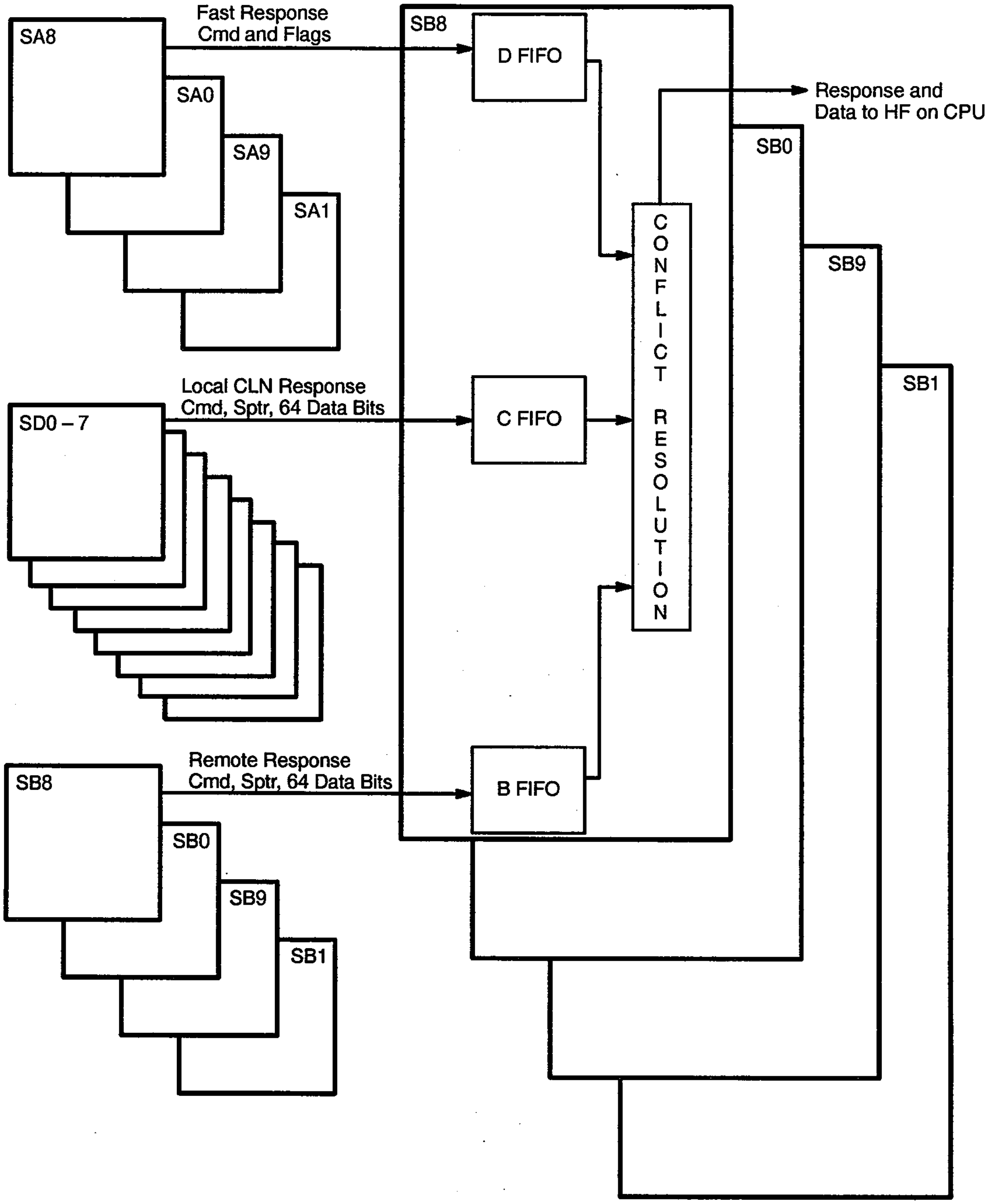

SA8
SA0
SA9
SA1
Fast Response Cmd and Flags
SB8
D FIFO
Response and Data to HF on CPU
SB0
SB9
SB1
CONFLICT RESOLUTION
SD0 – 7
Local CLN Response Cmd, Sptr, 64 Data Bits
C FIFO
SB8
Remote Response Cmd, Sptr, 64 Data Bits
B FIFO
SB0
SB9
SB1

Figure 10. SB Option Operations

## SM Register Test and Set Operations

Instructions 0034*jk* and 0064*jknm* perform a test and set operation on a selected SM register, but use the results of the test differently. The general flow of these instructions was described previously in the "Shared

Register Instructions" section. In the following paragraphs, the execution of these instructions in the SR option hardware is explained more fully. Refer to Figure 8 while reading these paragraphs.

### Instruction 0034*jk*

Instruction 0034*jk* (SR option code 14) must first pass through the Test and Set (T/S) structure before the semaphore is checked. The T/S structure on each SR option stores information about each of the 32 CPUs, including whether or not the CPU is in monitor mode (MM), if it is attached to either or both of the clusters on the SR option (referred to as CL0 and CL1), if there is a T/S request currently being processed in the CPU, and the number of the SM register to be checked.

SR option code 14 loads the SM register number and CLN into the T/S structure for the CPU specified in the Scpu signal and sets the T/S request for the specified CPU and cluster. The request must then compete with requests from other CPUs for the single output path to the SM registers. To resolve any conflicts, CPUs 0 through 3 are assigned to group A, CPUs 4 through 7 are assigned to group B, and so on, as shown in Figure 8. For second level conflict resolution, groups A through D are assigned to the lower group, and groups E through H are assigned to the upper group.

Initially, the first-level priority for group A is CPU 0, CPU 1, CPU 2, and CPU 3. The initial first-level priority for each of the other groups is also in numerical order. Whenever a CPU is selected to proceed, its priority is moved to the lowest level, while still maintaining the cyclic order. Thus, if CPU 1 makes a reference, the new order of priority would be CPU 2, CPU 3, CPU 0, and CPU 1.

The second-level priority for the lower group is initially group A, group B, group C, and group D. Likewise, the initial priority for the upper group is in alphabetical order. The priorites are maintained in a cyclic order identical to that used at the first level. The third-level priority alternates between the lower group and the upper group, with the lower group having initial priority.

Since there is no way to know beforehand which CPU's request will make it successfully through the CPU selection process, the Scpu number is reconstructed from bits used to select the priority at each level of the selection process. This Scpu signal can then be used to steer a response back to the proper CPU if the semaphore is clear or to retest the same semaphore if it is set.

When a CPU has been selected to forward its T/S request to the SM registers, the T/S Sequence Busy flag is set. The test is then performed. If the semaphore is set, no response is sent to the CPU, and the T/S request is retried in the T/S structure. If the semaphore is clear, a data word containing a value of 1, an SR option code of 14, and the Scpu signal are sent to the SD options. The selected semaphore is also set, and the T/S request is cleared in the T/S structure. The T/S Sequence Busy flag is then cleared to allow other SM register references.

The SR option code is converted to a Cmd code of c34 on the SD options, and the data and controls are passed on to the SB options. The Scpu signal becomes the Sptr signal and is used to steer the Cmd code and the data word to the CPU that issued the test and set instruction. Only after the CPU receives this response can it issue the next instruction.

### Deadlock

If all CPUs attached to the same cluster are in non-monitor mode and have a T/S request pending in the T/S structure, a deadlock counter is started for that cluster. Note that each CPU's pending T/S request could be for a different SM register. Because it can take some time for each CPU's request to get through the CPU selection process and get its selected semaphore checked, the deadlock counter must reach a count of 256 before a deadlock occurs. If at any time before the deadlock counter reaches a count of 256, one of the selected semaphores is checked and found to be clear, or if any changes are made to the T/S structure, the counter will stop and be reset to 0.

If a deadlock occurs, all pending T/S requests for CPUs in the cluster are cleared to prevent multiple deadlock interrupts. Also, an SR option code of 06 and deadlock flags for the cluster are sent to the SD options. The deadlock flags are contained in a data word that is zero-filled except for bit positions corresponding to CPUs in the cluster. The SR option code is converted to a Cmd code of c46 on the SD options. The Cmd code and data word are then passed on to the SB options, where the deadlock flags are used to steer a deadlock interrupt to each of the CPUs in the cluster.

### Instruction 0064*jknm*

For instruction 0064*jknm*, the Rptr and Sptr signals, containing the SM number and CLN, are sent directly to the SM registers if the T/S Sequence Busy flag is clear. If the selected semaphore is clear, then the semaphore will be set, and a data word with a value of 0 along with the SR option code of 10 will be sent to the SD options. If the selected semaphore is set, then a data word with a value of 1 along with the SR option code of 10 will be sent to the SD options. The SD options convert the SR option code to a Cmd code of c30 and send the Cmd code and the data word to the SB options. The SB options send this information to the CPU that issued the instruction. If the data word is 0, the CPU will continue processing instructions. If the data word is 1, the CPU sets the P register to the 32-bit parcel address specified by *nm*, and program execution continues at that point.

## Shared Register CLN Operations

There are five Cmd codes which set or clear the CLN attachments for each CPU. Two of these codes are generated by explicit instructions, and three of them are generated by exchange sequences.

### Instruction 0014*j*3

Instruction 0014*j*3 (Cmd code c27) sets the CLN register and the monitor mode (MM) bit on the SA option. The CLN is sent from the HF option on the CPU module to the SB option on the shared module as the Rptr signal. The Cmd code is also sent from the HF option to the SB option. Both signals are then sent to the SA option, where the Cmd code sets the MM bit, and the CLN updates the CLN register. No response is sent to the CPU.

### Exchange Sequence

An exchange sequence that does not occur while a CPU is holding issue on a test and set instruction generates a Cmd code of c7. The Cmd code is sent from the HF option via the SB option to the SA option. The data sent from the HG option on the CPU module to the SA option on the shared module contains the incoming CLN in bits 0 – 7, the MM bit in bit 23, and the Triton mode bit in bit 21. The SA option uses the CLN and MM bits to update its CLN and MM registers. If the MM bit is not set, then the Cmd, Scpu, and eCLN (derived from the incoming CLN) signals are sent to the appropriate SR option via the SC options. On the SR option, the Cmd code is changed to an SR option code of 07 and is used to gate the MM and CLN into the T/S structure. There the Scpu signal is used to attach the source CPU to the specified cluster.

### Broadcast Cluster Detach and Flush

The broadcast cluster detach instruction (001640, Cmd code c32) and the two flush Cmd codes (c1 and c5) perform similar operations in the T/S structure on the SR options. The flush Cmd codes are generated by an exchange sequence that occurs while a CPU is holding issue on a test and set instruction. If a master clear sequence is in progress at the same time, a Cmd code of c1, indicating a multiple CLN flush, is generated. If there is no master clear sequence in progress, a Cmd code of c5, indicating a single CLN flush, is generated.

Because the c1 flush performs almost the same functions as Cmd code c32, they are grouped together in the following discussion. The Cmd code for either of these operations is initially sent from the HF option via the SB option to the SA option. Cmd code c1 is changed on the SA option to Cmd code Tc32. The data and control signals used for both operations follow exactly the same broadcast paths to the T/S structure on the SR options. The SA option, however, sets the contents of the data and control signals to different values before passing them on. For Cmd code c32, the Rptr signal is set to –1, and the Sptr signal and the data are left unchanged. For Cmd code Tc32, the Rptr signal is set to the eCLN derived from the incoming CLN, bit 0 of the Sptr signal is set to 1, and the data field is set to the present eCLN. For both operations, the Scpu signal indicates the source CPU, and the Cmd code forwarded is c32.

For the original Cmd code 32, fast response data is sent to the SB option to be forwarded to the CPU to release its reservation on the shared path. Since both operations now have the same Cmd code, the SB option uses the fast response flags shown in Table 8 to determine if Cmd code c32 should be sent to the CPU.

The data and controls are broadcast from the SA options to the SC options on both the local and remote shared modules. This is done by sending the same information out over both the local and remote paths. As the information passes through the SC options, it is not steered to a particular SR option, but rather is sent to all SR options.

On the SR options, the Cmd code c32 is converted to the SR option code of 12. This code gates the Rptr signal and the lower 6 bits of the data field into the T/S structure. There the T/S requests and deadlock flags for all clusters attached to the source CPU are cleared. The Rptr and CLNs attached to the source CPU are compared. If they do not match, the clusters are detached from the CPU; if they do match, that cluster is attached to the source CPU. Since the original Cmd code c32 will not have any CLN matches, this test results in the detachment of all clusters attached to the source CPU. The original Cmd code c1, however, will have a match for one cluster, and thus this code results in the detachment of all clusters from the source CPU except the cluster specified in the exchange package.

The CLNs attached to the source CPU are also compared with the lower 6 bits of the data field. If there is a match, and there will only be a match for the original Cmd code c1, the SR option code of 12 is sent to the SD options. There it is converted to the Cmd code c32 and sent to the SB options. On the SB options, this code is changed to Cmd code c5 and sent to the HF option on the CPU that originated the Cmd code of c1.

The c5 flush operates similarly to the c1 flush, except that its action is limited to the cluster specified in the CLN register on the SA option. When the Cmd code of c5 reaches the SR option, it is converted to an SR option code of 15. This code gates the CLN into the T/S structure, where the T/S request and deadlock flags for the specified cluster attached to the source CPU are cleared. The SR option code of 15 is then sent to the SD options, where it is changed to its original Cmd code of c5 before being sent to the CPU via the SB options.

## Real-Time Clock Operation

Instruction 0014*j*0 is used to set the real-time clock (RTC) by transmitting the value stored in the S*j* register to the RTC. This instruction uses the logic on the shared module to synchronize the CPUs so the new RTC value can be loaded into all CPUs in the same cluster group simultaneously. This process of synchronizing the relevant CPUs is explained in the following paragraphs. *x* refers to the number of the CPU that originated instruction 0014*j*0.

Instruction 0014*j*0 is converted on the HF option to a Cmd code of c24. The Cmd signal is then sent to the SB*x* option on the shared module. If the SB*x* option receives a valid command code from the SA*x* option, the Cmd signal is then passed to the SA*x* option. There it is converted to a c6 code and checked to verify that it is a valid code for the shared module. The Sptr signal, consisting of the two bits designating the cluster group (CLN partition), is also generated on the SA*x* option. If the command code is valid, the Cmd and Sptr signals are sent to all four SA options in the CPU group. The paths from CPU*x* through the SB*x* and SA*x* options to all four SA options are the same as those used for all shared instructions.

At the same time the Cmd signal enters the SA*x* option, the S*j* data from the HG option on CPU*x* enters the SA*x* option. If the command code is valid, the data is split into four 16-bit slices, with one slice sent to each of the four SA options in the CPU group. The data paths used from CPU*x* through the SA*x* option to all four SA options are the same as those used for all shared instructions.

Before the data and controls leave the SA options for the SC options, the Scpu (source CPU) control signal must be generated. The Scpu signal is a 5-bit signal identifying the CPU that issued the instruction. This signal is generated in the same manner as it is for all shared instructions, with 3 bits generated on the SA8 option (for CPU group 0) and the remaining 2 bits generated on the SC3 and SC4 options.

The SC options receive data and controls from each of the four CPU groups and route them to the 16 SD options. The internal operations of the SC options are shown in Figure 7. When the data and controls for a CPU*x* reference enter the SC options, they are gated into one of two input buffers for the relevant CPU group. The buffers handle I/O and RTC references and are labeled A and B. Buffer A has the highest priority. If it is full, the data and controls are routed to buffer B. As buffer A empties, the data and controls in buffer B move up to buffer A.

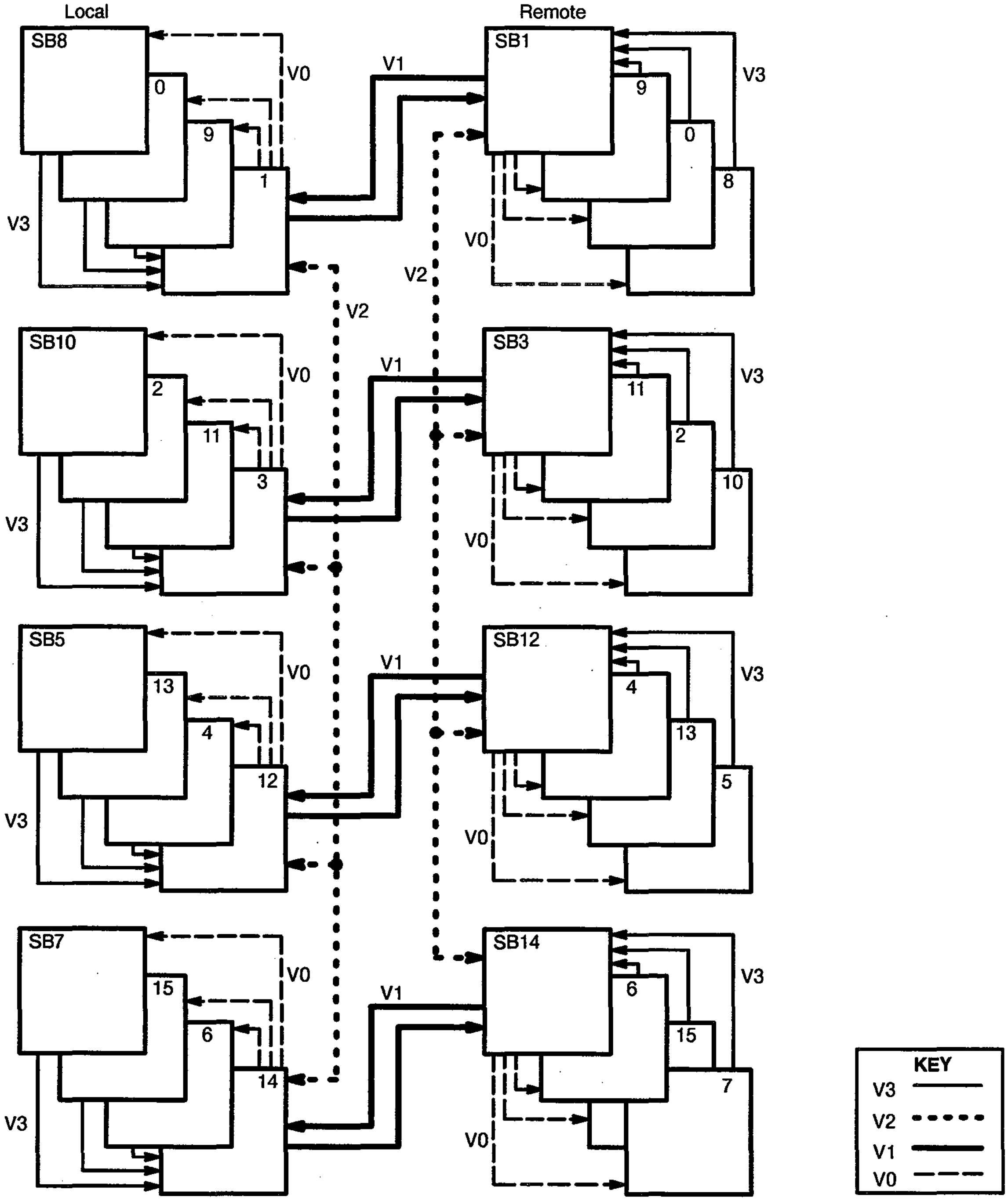
Because up to four CPU groups compete for a single output path to the SD options, conflicts can occur. These are resolved by assigning a priority to each of the four CPU groups. Initially, the priority is CPU group 0, CPU group 1, CPU group 2, and CPU group 3. Whenever a CPU group is selected to proceed, its priority is moved to the lowest level, while still maintaining the cyclic order. Thus, if CPU group 1 makes a reference, the new order of priority would be CPU group 2, CPU group 3, CPU group 0, and CPU group 1.

After a CPU group has been selected, a signal is sent to both the local and remote SC options to stop any additional RTC references. The selected group must then vie for the output path with an RTC reference from the remote shared module. The priority in this case toggles between the local and remote shared modules, with the local shared module having initial priority. I/O references do not have this additional level of conflict resolution.

The three control signals (Cmd, Sptr, and Scpu) and the S*j* data are sent to all 16 SD options. Figure 9 on page 33 shows the SD option operations. The data and controls follow the same general paths through the options as the data and controls for all shared instructions with one exception. The c6 Cmd selects all four output paths and broadcasts the data and controls to all the local SB options as both local and remote information.

The data and controls enter the 16 local SB options on both the local and remote input paths. The remote information is immediately passed on to the 16 remote SB options, so that all 32 SB options receive the information. As each SB option detects the c6 Cmd, it holds the command and data and shuts down all operations on the option. This shutdown prevents all 32 CPUs from receiving any additional commands or data from a shared module until the RTC has been set. The SB options send signals between themselves to determine when all 32 SB options have shut down and the RTC can be set. Figure 11 shows the voting logic used between the SB options.

One SB option in each CPU group is the designated controller to monitor SB option shutdown. The controlling options on the local shared module are SB1, SB3, SB12, and SB14. As each SB option detects the c6 Cmd, it shuts down and sends a V3 signal to its controller. When the controller receives the V3 signals from all the SB options in its group including itself, it sends a V2 signal to the other local SB controllers. When a controller receives all four V2 signals, it sends a V1 signal to its corresponding controller on the remote shared module. When a controller receives a V1 signal in return, it simultaneously sends V0 signals to each of the SB options in its group.



V3 – Sent when the c6 arrives; SB is shut down until the V0 arrives.

V2 – Sent between controlling SBs, indicating they have received all the V3s.

V1 – Sent between controlling SBs on opposite shared modules as they receive their V2s.

V0 – Simultaneously sent to all SBs from controlling SBs. This causes the SBs in the same partition as the Scpu to send their CPU a c6 command and causes 64 bits of data to be entered into the RTC register.

Figure 11. RTC Voting Logic

When an SB option receives the V0 signal, it does several things. If the SB option is in the same cluster group as CPU*x*, it forwards the c6 Cmd and the S*j* data to its CPU's HF option. From there, it is passed on to the HD option where the RTC is set. After receiving the V0 signal, the SB options also begin a 10 CP countdown before ending the shutdown condition and resuming normal option operations. The SB options listed in Table 11 also send the Send Next RTC signal to the SC options, enabling more RTC references. The SB*x* option also converts the c6 Cmd to the original c24 Cmd and sends it to CPU*x* to complete the instruction.

Table 11. Send Next RTC Signal Path

| Source Option | Output Term | Destination Option | Input Term |
|---|---|---|---|
| SB2 | OQQ | SC0, 1 | IVX |
| SB4 | OQQ | SC4, 5 | IVX |
| SB11 | OQQ | SC2, 3 | IVX |
| SB13 | OQQ | SC6, 7 | IVX |

# Reader Comment Form

**Title: Shared Module**
***Preliminary Information***

**Number: HTM-xxx-0**
**December 1994**

Your feedback on this publication will help us provide better documentation in the future. Please take a moment to answer the few questions below.

For what purpose did you primarily use this document?

______Troubleshooting
______Tutorial or introduction
______Reference information
______Classroom use
______Other - please explain ______________________________

Using a scale from 1 (poor) to 10 (excellent), please rate this document on the following criteria and explain your ratings:

______Accuracy ______________________________
______Organization ______________________________
______Readability ______________________________
______Physical qualities (binding, printing, page layout) ______________
______Amount of diagrams and photos ______________________
______Quality of diagrams and photos ______________________

Completeness (Check one)

______Too much information ______________________________

______Too little information ______________________________

______Just the right amount of information

Your comments help Hardware Publications and Training improve the quality and usefulness of your publications. Please use the space provided below to share your comments with us. When possible, please give specific page and paragraph references. We will respond to your comments in writing within 48 hours.

NAME ______________________________
JOB TITLE ______________________________
FIRM ______________________________
ADDRESS ______________________________
CITY ______________ STATE ________ ZIP ________
DATE ______________________________
[or attach your business card]

CRAY
RESEARCH, INC.

CUT ALONG THIS LINE

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

**BUSINESS REPLY CARD**

FIRST CLASS PERMIT NO 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

**Attn: Hardware Publications and Training**
**890 Industrial Boulevard**
**Chippewa Falls, WI 54729**

Fold

STAPLE