SHARED MODULE

Figures

Tables

Tables (continued)

Shared Module Overview

The CRAY T932™ systems each contain two shared modules; the CRAY T916™ and CRAY T94™ systems each contain one shared module. Figure 1 shows the chassis location of the shared module in a CRAY T94 system. Figure 2 shows the location of the shared module(s) in CRAY T916 and CRAY T932 systems. Only quadrants 0 and 1 are populated in CRAY T916 systems. One shared module is located in each physical half of the CRAY T932 system.

Each of two shared modules in the CRAY T932 system can be referred to as either the *local* or *remote* shared module. The reference depends on the orientation of the CPU issuing a shared module instruction. The shared module on the same side of the chassis as the CPU is referred to as the local shared; the shared module on the other side of the chassis is referred to as the remote shared. Therefore, either module can be referred to as local or remote by different CPUs.

All logic for the shared registers is contained on the shared module. The shared module also contains logic for I/O functions and synchronizes the CPUs when it sets the real-time clock (RTC). In addition, the shared module holds much of the configuration information.

Figure 1. CRAY T94 Module Locations

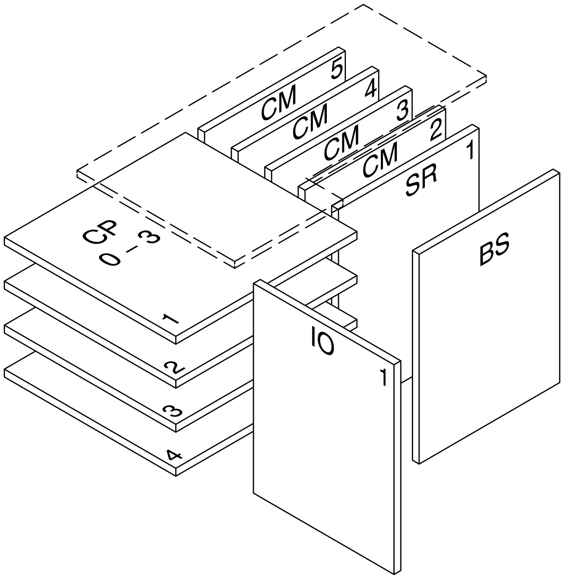
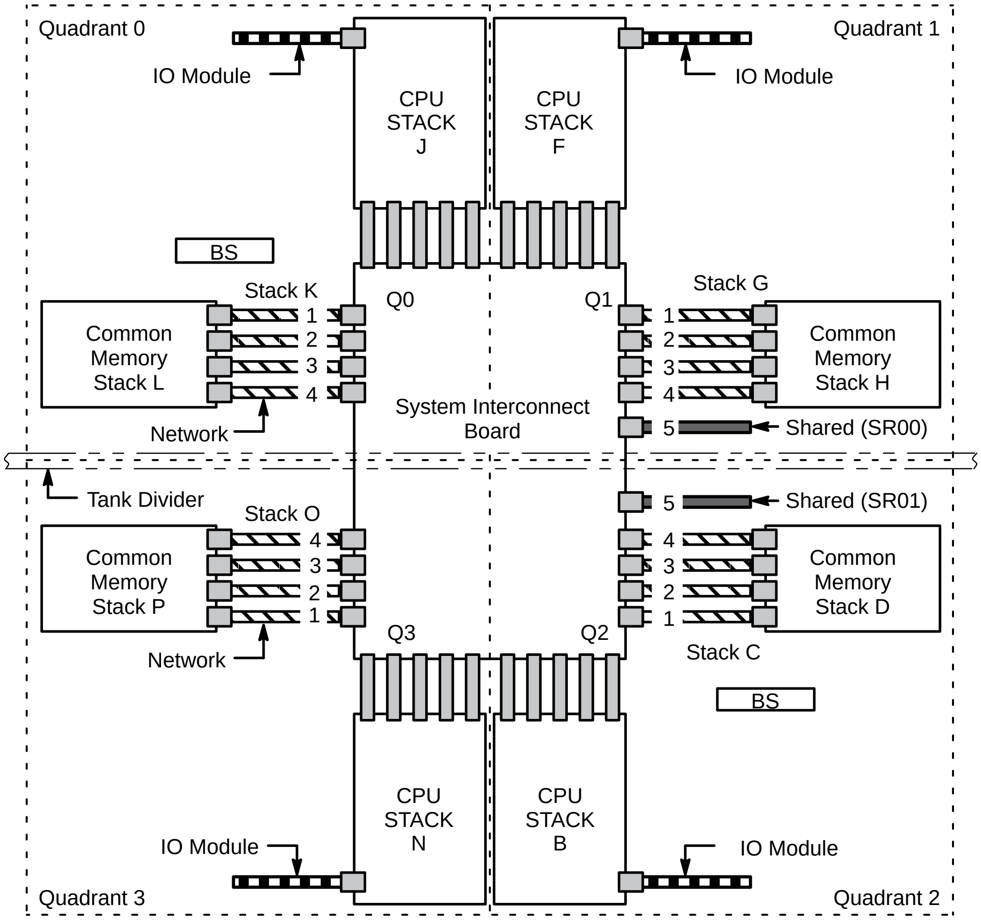


Figure 2. CRAY T932 and CRAY T916 Chassis



Physical Description

The shared module is a single 52-layer printed circuit board module that contains logic on both sides. The sides of the module are referred to as board 1 and board 2. It also contains eight orthogonal interconnect module (OIM) connector pads for physical connection to the eight system interconnect boards (SIBs). Because the CRAY T94 systems do not contain SIBs, the shared module is connected directly to the CP modules in these systems.

The shared module is positioned vertically in the CRAY T90™ series systems. Figure 3 and Figure 4 show module maps for the two boards of the shared module.

Figure 3. Shared Module Board 1

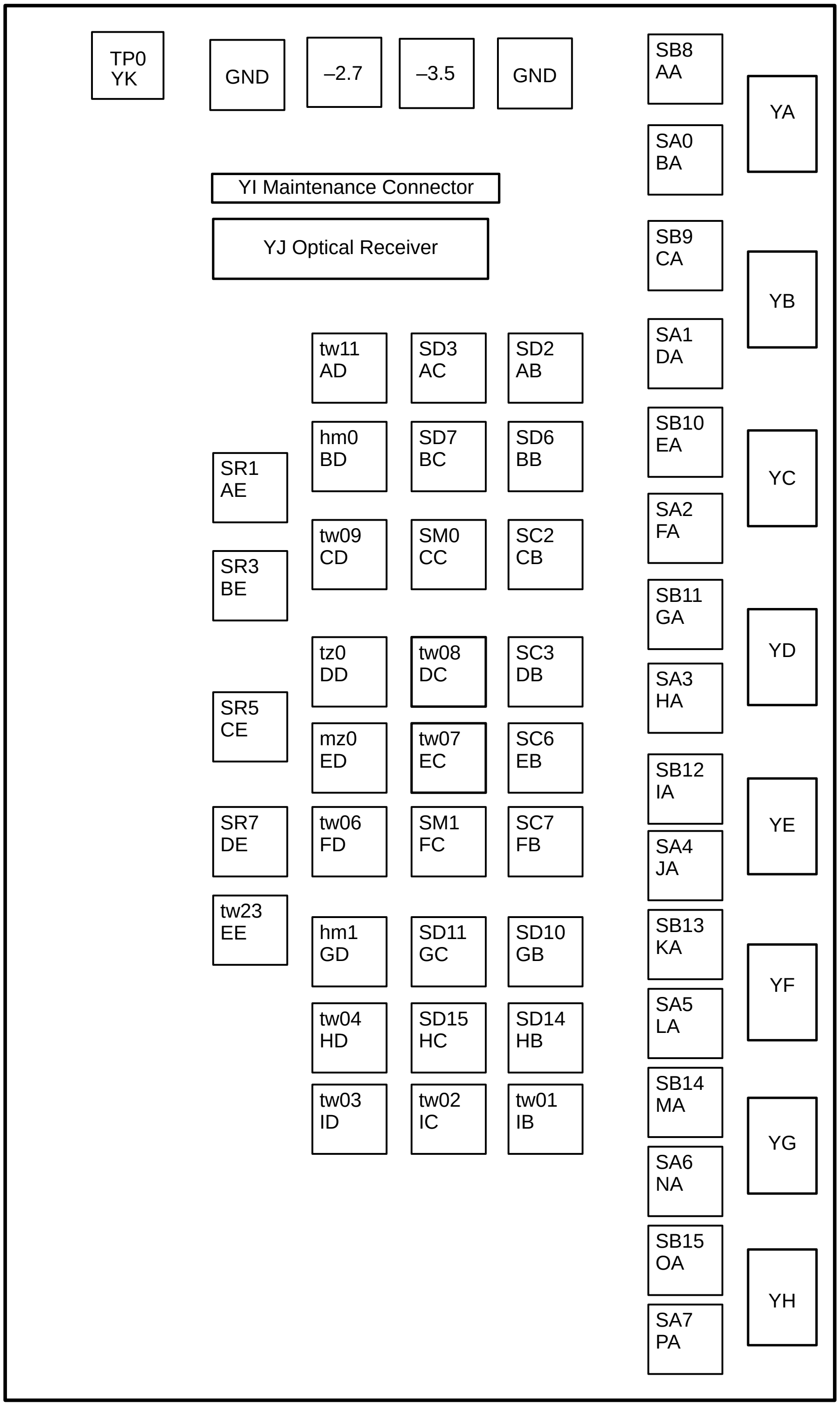
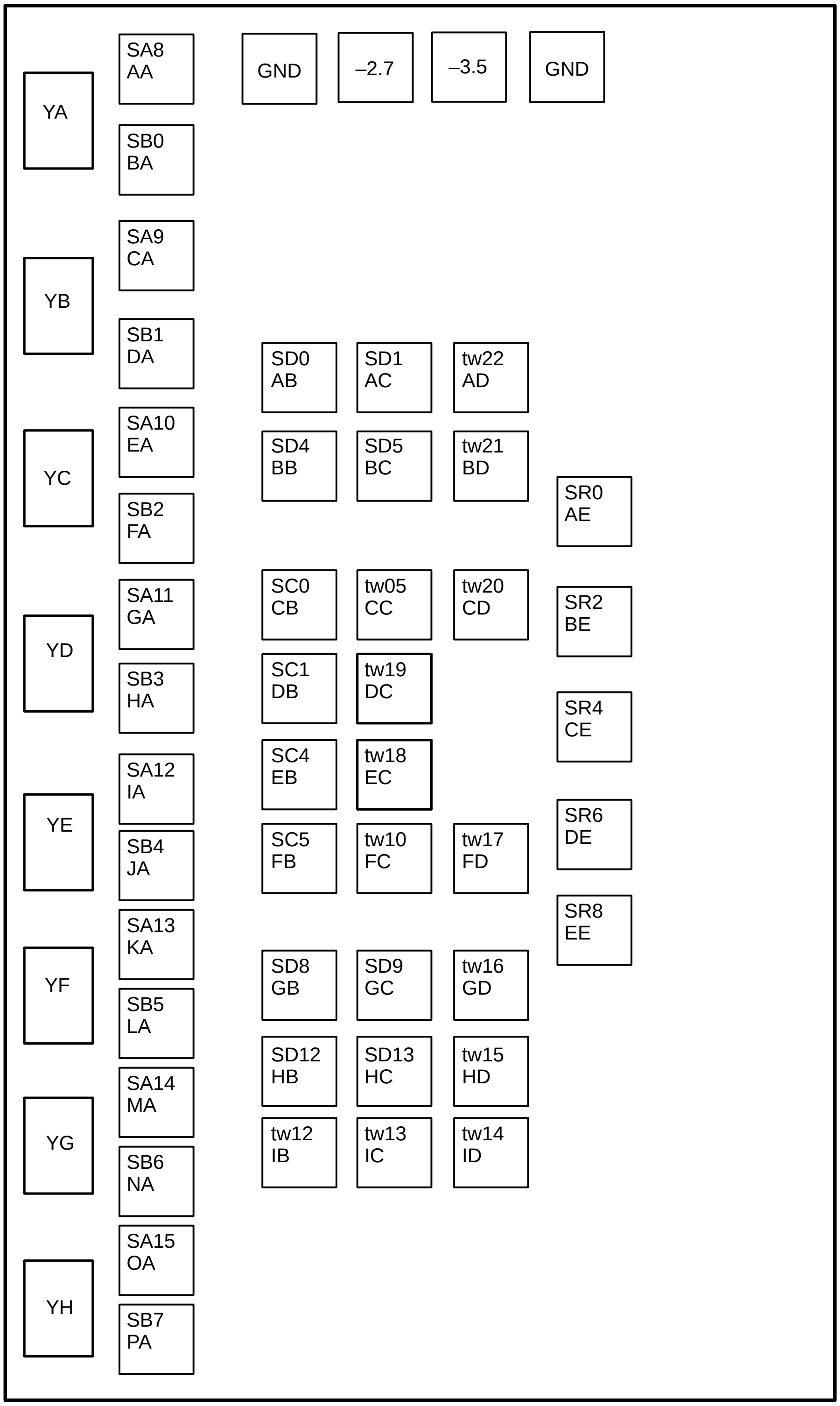


Figure 4. Shared Module Board 2



Shared Module Options

There are seven option types on the shared module: the SA, SB, SC, SD, SM, SR, and HM. The following paragraphs describe each option type.

SA Options

The SA options are one of two option types that make up the 16 ports located on the shared module. In addition to receiving data and command codes, the SA options handle conflict resolution between CPU requests for cluster activity, I/O interrupts, remote references, and fast response data.

These options hold the cluster number (CLN) register, the monitor mode bit and the Triton mode bit. They use the CLN to generate the effective CLN, which is used as the Steering Pointer (Sptr) signal. They also generate 3 bits of the Source CPU (scpu) signal.

There are 16 SA options, numbered SA00 through SA15, on the shared module.

SB Options

The SB options are the second of two option types that make up the 16 ports located on the module. The SB options receive A*j*, A*k*, or *jk* data and the initial command code from the CPU module. They convert the register data to the Register Pointer (Rptr) signal and validate the command code. They then send both signals to the SA options.

These options resolve conflicts between the read data they receive from local and remote clusters and from the SA options. They then forward the data to the CPU module. The SB options also synchronize the CPUs for RTC instructions.

There are 16 SB options, numbered SB00 through SB15, on the shared module.

SC Options

The SC options receive data and control information from each of the four CPU groups and route it to one of the nine SR options. They also receive data and control information for I/O and RTC references and route it to the SD options. Channel interrupts and the System Interrupt Enable (SIE) scoreboard are held on SC0.

There are eight SC options, numbered SC0 through SC7, on the shared module.

SD Options

The SD options receive read data and control information from either the SR options or SC options and route it to the SB options. They also receive write data for the RTC instruction and route it to the SB options.

There are 16 SD options, numbered SD00 through SD15, on the shared module.

SM Options

The SM options handle sanity codes, maintenance channels, and error logger and master clear functions for the shared module. There are two SM options, numbered SM0 and SM1, on board 1 of the shared module.

SR Options

The SR options receive data and control information from the SC options and route it to the shared registers. Each option holds two clusters of shared registers. The SR options detect deadlock conditions and read parity errors. They also forward read data and controls to the SD options.

There are nine SR options, numbered SR0 through SR8, on the shared module.

HM Options

The HM options control the logic monitor functions for the shared module. Each HM option has 64 data inputs for receiving test point data from the other options on the shared module. There are two HM options, numbered HM0 and HM1, on the shared module.

Shared Module Ports

The shared module contains 16 logical ports for communication with the CPUs. The ports are numbered 0 through 17 octal. Each port communicates with one CPU. Any data passed between the shared module and a CPU passes through a specific port.

Each port is made up of an SA/SB option combination. The SA option receives data from the CPU, and the SB option receives control information from the CPU. The SB option also returns data and control information to the CPU. Table 1 identifies the port assignments for a CRAY T94 system.

Shared Module Groups

The shared module ports are arranged into four logical groups. A group is made up of four ports. The following tables identify the logical grouping of the ports and the CPUs that are associated with the ports.

Figure 5 illustrates the options on the shared module that make up the ports. The diagram expands group 0 to show data and command flow from each individual port within the group to the other ports within that group.

Table 1. CRAY T94 Port Assignments

Port	Options	Physical CPU	Group
2	SA2, SB2	0	1
3	SA3, SB3	1	1
4	SA4, SB4	2	2
5	SA5, SB5	3	2

The shared module(s) in CRAY T916 and CRAY T932 systems utilize all 16 ports. Table 2 lists the port assignments for the CRAY T916 system, and Table 3 lists the port assignments for both shared modules in the CRAY T932 system.

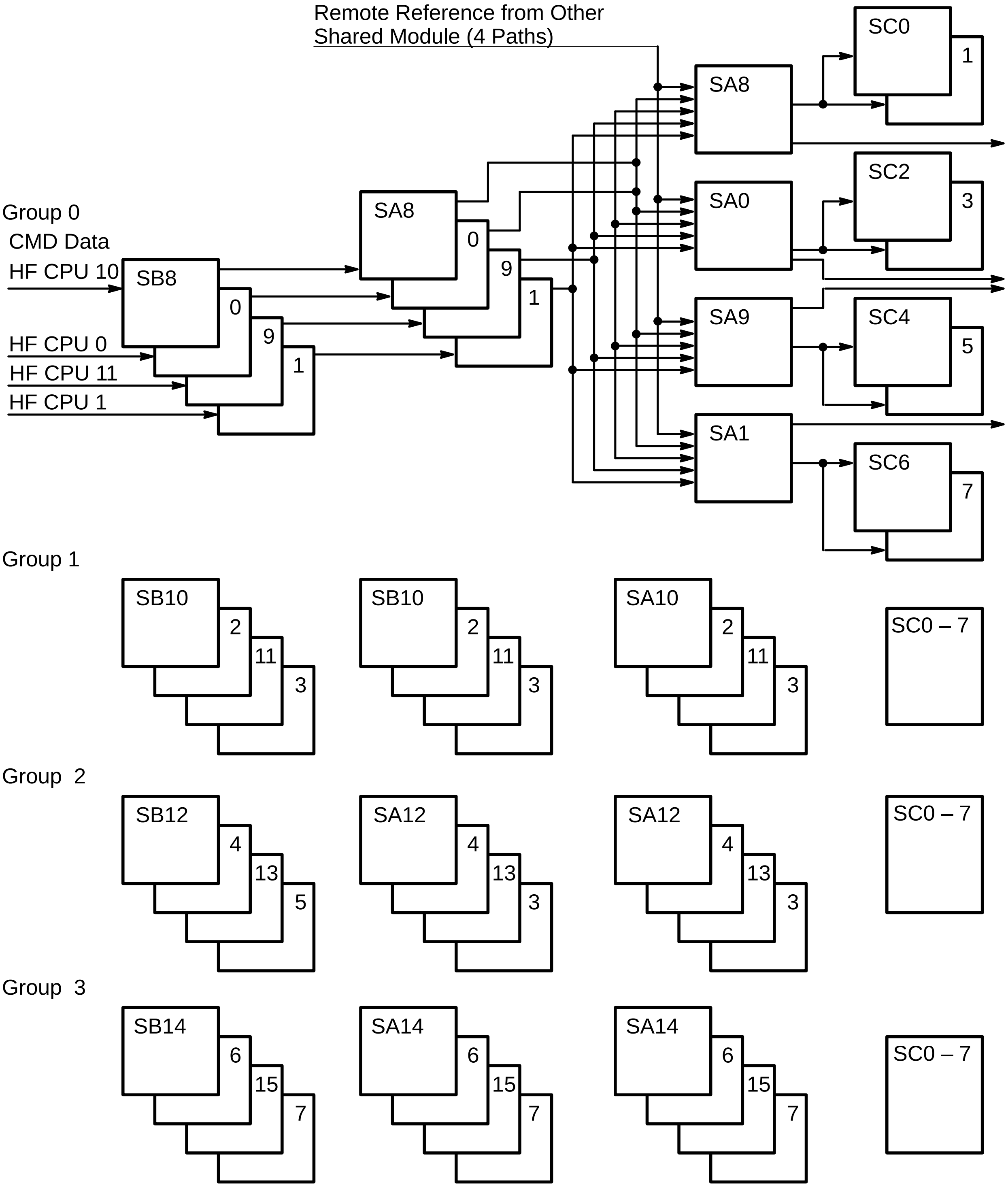
Table 2. CRAY T916 Port Assignments

Port	Options	Physical CPU	Group
0	SA0, SB0	0	0
1	SA1, SB1	1	0
2	SA2, SB2	2	1
3	SA3, SB3	3	1
4	SA4, SB4	4	2
5	SA5, SB5	5	2
6	SA6, SB6	6	3
7	SA7, SB7	7	3
10	SA8, SB8	10	0
11	SA9, SB9	11	0
12	SA10, SB10	12	1
13	SA11, SB11	13	1
14	SA12, SB12	14	2
15	SA13, SB13	15	2
16	SA14, SB14	16	3
17	SA15, SB15	17	3

Table 3. CRAY T932 Port Assignments

Shared Module	Port	Options	Physical CPU	Group
SR00	0	SA0, SB0	0	0
SR00	1	SA1, SB1	1	0
SR00	2	SA2, SB2	2	1
SR00	3	SA3, SB3	3	1
SR00	4	SA4, SB4	4	2
SR00	5	SA5, SB5	5	2
SR00	6	SA6, SB6	6	3
SR00	7	SA7, SB7	7	3
SR00	10	SA8, SB8	10	0
SR00	11	SA9, SB9	11	0
SR00	12	SA10, SB10	12	1
SR00	13	SA11, SB11	13	1
SR00	14	SA12, SB12	14	2
SR00	15	SA13, SB13	15	2
SR00	16	SA14, SB14	16	3
SR00	17	SA15, SB15	17	3
SR01	0	SA0, SB0	20	0
SR01	1	SA1, SB1	21	0
SR01	2	SA2, SB2	22	1
SR01	3	SA3, SB3	23	1
SR01	4	SA4, SB4	24	2
SR01	5	SA5, SB5	25	2
SR01	6	SA6, SB6	26	3
SR01	7	SA7, SB7	27	3
SR01	10	SA8, SB8	30	0
SR01	11	SA9, SB9	31	0
SR01	12	SA10, SB10	32	1
SR01	13	SA11, SB11	33	1
SR01	14	SA12, SB12	34	2
SR01	15	SA13, SB13	35	2
SR01	16	SA14, SB14	36	3
SR01	17	SA15, SB15	37	3

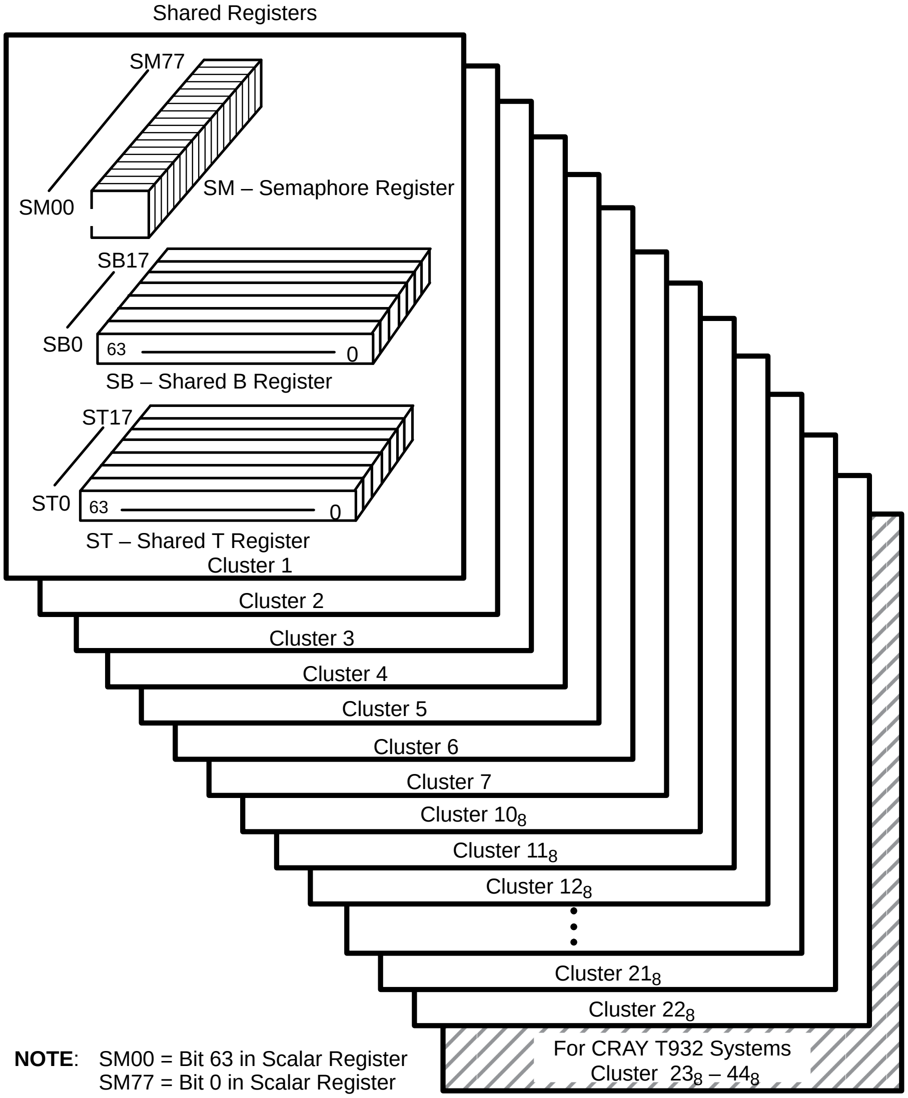
Figure 5. Shared Module Port/Group Designation



Shared Registers

The shared module contains all the logic for the shared registers. The shared registers provide intermediate data storage and control between CPUs. The shared registers are divided into groups called clusters. Each shared module contains 18 clusters (numbered 1_8 through 22_8) of shared registers. Each cluster contains 16 shared B (SB) registers, 16 shared T (ST) registers, and 64 (100_8) semaphore (SM) registers. Figure 6 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.

Figure 6. Shared Registers



NOTE: SM00 = Bit 63 in Scalar Register
SM77 = Bit 0 in Scalar Register

Shared Register Instructions

The shared register instructions are listed below in three groups: instructions that move data between the CPU and shared registers, instructions that utilize the semaphore registers, and instructions that affect cluster attachments.

Table 4. 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 (S*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*)

Table 5. Semaphore Instructions

Instruction	CAL	Description
0034*jk* (*j*2=0)	SM*jk* 1,TS	Test and set semaphore *jk* (*jk* = 0 – 37)
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)
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)
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*

Table 6. CLN Instructions

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

SB and ST Register Operations

The 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.

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 enables some programs to be simplified.

The fetch and increment instructions 026*ij*4 and 026*ij*5 use only 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 is incremented by 1.

In C90 mode, an SB register read operation results in the upper 32 bits of the destination address register being zero filled. This is because the address registers in the CRAY C90™ series systems are only 32 bits long.

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 cleared or set by using instruction 0036*jk* or 0037*jk*, respectively. For these instructions, *j* indicates whether SM registers 0 through 37 or SM registers 40 through 77 are used. If bit 2 of the *j* field is 0, then the *jk* field is used to indicate the SM register (SM 0 through 37). If bit 2 of the *j* field is 1, then (A*k*) indicates the SM register (SM 40 through 77). 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, then (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. If the SM register content is a 0, this instruction sets the SM register, and the CPU continues issuing instructions. If the SM register content is a 1, the CPU holds 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. After the SM register is cleared, it is eventually reset by the CPU holding issue on the test and set instruction, and the CPU continues issuing instructions.

The test and set branch instruction, 0064*jknm*, 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 contents of register A*k* specifies the SM register to test.

The 0064*jknm* instruction operates in a similar manner to the 0034*jk* instruction, except that if the SM register content is a 1, the CPU branches to location *nm*. If the SM register content is a 0, the CPU sets the SM register, the next instruction issues, and no branch is taken.

Cluster Operations

There are two ways in which a CPU becomes attached to a cluster. One way is when a user mode job exchanges into the CPU. The shared register cluster specified in the user job’s exchange package is sent to the shared module, along with the MM/UM (monitor mode or user mode) bit. If the MM/UM bit indicates that the job is a user mode job, then the shared module attaches the CPU to the specified cluster.

When a monitor-mode job exchanges into the CPU, it also has a cluster number specified in its exchange package, which is sent to the shared module. But when the shared module determines that the job is a monitor-mode job (from the MM/UM bit), it does not attach the CPU to the cluster specified.

The second way in which a CPU becomes attached to a cluster is via the 0014*j*3 instruction. This is a monitor-mode instruction that sets the cluster number register on the shared module. It attaches the CPU to that cluster. Valid cluster numbers are 1 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.

A new capability for the CRAY T90 series computers is that of exchanging from one user-mode job to another user-mode job, instead of automatically exchanging from user-mode to monitor-mode. This is possible because of the addition of multiple exit addresses in the exchange package. This capability reduces overhead because the monitor is not required to exchange a new user-mode job into the CPU.

When a user-mode job exchanges out, the hardware keeps track of the CPU number and the cluster number that job was using. If the system is operating in Triton mode, the previous user-mode job's cluster attachment is still active when a new user-mode job enters the CPU. This means that the CPU can be attached to multiple clusters at one time. It is possible for a CPU to be attached to 36 clusters at one time in a CRAY T932 system.

This feature allows user-mode jobs to call other user-mode jobs without losing previous cluster attachments. Keeping cluster attachments is important for avoiding false cluster deadlock conditions.

Broadcast Cluster Detach

The CRAY T90 series has a feature called automatic broadcast cluster detach (AutoBCD). When this feature is active [selectable in System Configuration Environment (SCE)], the CPU's previous cluster attachment is lost when the monitor exchanges a new user-mode job into the CPU.

AutoBCD affects only Triton mode jobs. It releases previous cluster attachments only when it changes from monitor mode to user mode. AutoBCD has no effect on user-to-user exchanges.

The CRAY T90 series instruction set includes a new instruction, 001640, that releases all cluster assignments previously established by user-mode jobs running in the particular CPU. This instruction is normally used only in Triton mode; in C90 mode, cluster attachments are automatically released on exchanges between user-mode programs.

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

Deadlock

A deadlock condition exists when all CPUs attached to the same cluster are holding issue on a test and set instruction (0034*jk*) directed to the same SM register. When this condition exists, a deadlock counter is started for that cluster. Each CPU’s pending test and set request may be for a different SM register, but 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 clock periods before a deadlock condition actually occurs. If the test and set structure is changed in any way, such as by a broadside load of the semaphores, the counter will stop and be reset to 0.

If a deadlock occurs, all pending test and set requests for the cluster are cleared to prevent multiple deadlock interrupts. If a CPU’s Interrupt on Deadlock (IDL) mode bit is set and if the Enable Interrupt Modes (EIM) bit is set, the CPU will perform an exchange.

General Control and Data Flow

The following paragraphs describe the general flow of data and control signals moving between the CPU and the shared register logic. An 11 × 17 block diagram of the shared module is located at the end of this document.

When a CPU issues an instruction that involves the shared module, the instruction is first converted on the HF option to a 6-bit command code. The shared module further differentiates this code into either a CPU command (Ccmd) code or a shared command (cmd) code. Ccmd codes are used for I/O and CPU instructions, and cmd codes are used for shared register instructions.

The SB option receives controls from the HF option on the CPU. This control information includes the command code, and A*k*, A*j*, and *jk*, depending on the instruction issued. The SA option receives data from the HG option on the CPU. The data arrives in two 32-bit transfers. Figure 7 shows these paths into the SA and SB option on the shared module for all four groups.

The SB option verifies that the shared module is receiving sanity code from the CPU module. If the SB option does detect sanity code, it forwards the control information to the SA option. The SA option distributes the data to all four SA options in the logical group. If the SB option does not detect sanity code, then the control information is not forwarded, and the operation stops. Each SA option in the group then sends the data and controls to two SC options.

The eight SC options then forward the data to one of nine SR options. Each SR option contains all the logic for two clusters of shared registers.

A shared register read operation begins the same way as the write operation and goes through most of the same steps as it moves from the CPU through the SA, SC, and SR options. For an SB/ST register read, parity is performed as the data is read from the registers.

The SR option sends 8 bits of data to each of the eight SD options that handle data for the CPU that requested the data. Options SD00 through SD07 handle data for CPUs in groups 0 and 1, and options SD08 through SD15 handle data for CPUs in groups 2 and 3.

The SD options send the read data, 8 bits at a time, to each of the SB options in the group. However, only the SB option for the correct port receives the necessary control signals. The SB option then forwards the shared register data, in two 32-bit transfers, to the HF option in the CPU.

Figure 7. Shared Module Data and Control Paths

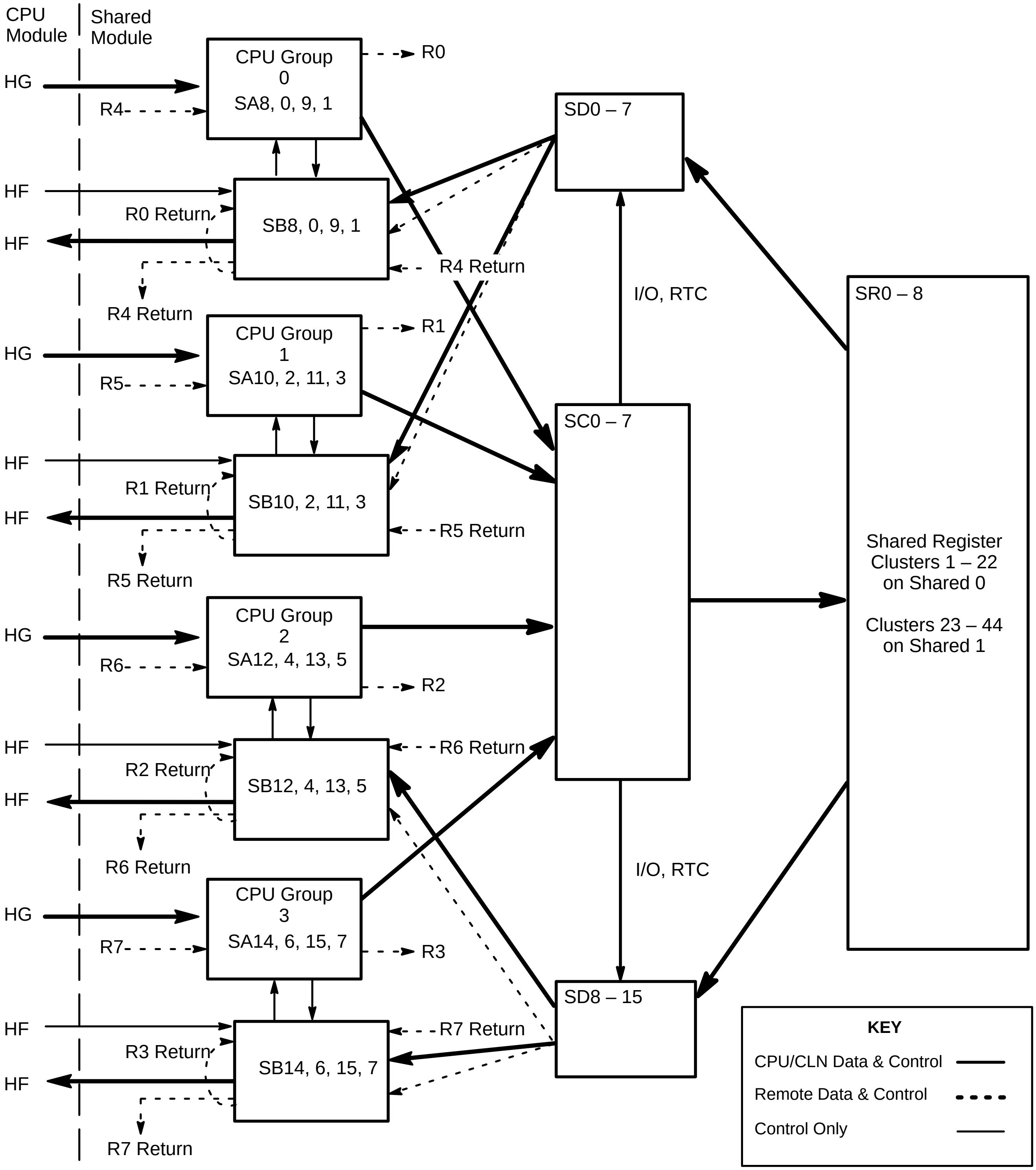
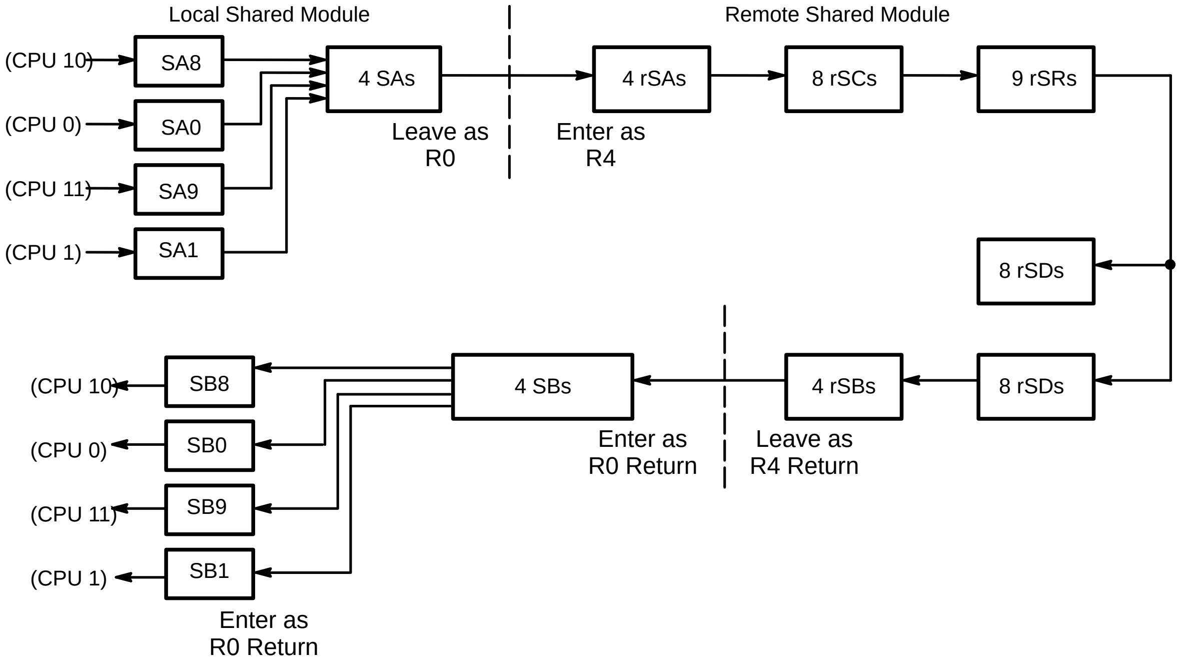


Figure 8 shows the data and control paths for group 0 during remote references. For other groups, the terminology is similar. The letter R is used to indicate **remote** data and control. The number following the letter

R identifies the group of the issuing CPU. Thus, the data and control information leaving group 0 is referred to as R0 data and control, the data and control information leaving group 1 is referred to as R1 data and control, and so on. Data and controls entering the corresponding group on the remote shared module are referred to as R4, R5, R6, or R7 data and control, respectively. When the data and control leave the remote shared module, they are referred to as R4 Return, R5 Return, R6 Return, or R7 Return, depending on the group from which they exit the module. The data and control re-enter the local shared module through the SB options in the same group that initiated the reference; on re-entry, they are referred to as R0 Return, R1 Return, etc. The steering control signals are then used to fan the data and control into the single SB option that will forward the data and control information to the SB option's associated CPU. The data and control enter this single SB option as R0 Return, R1 Return, etc.

Figure 8. Remote Reference Data and Control Paths for CPU Group 0



Shared Register Write Operation

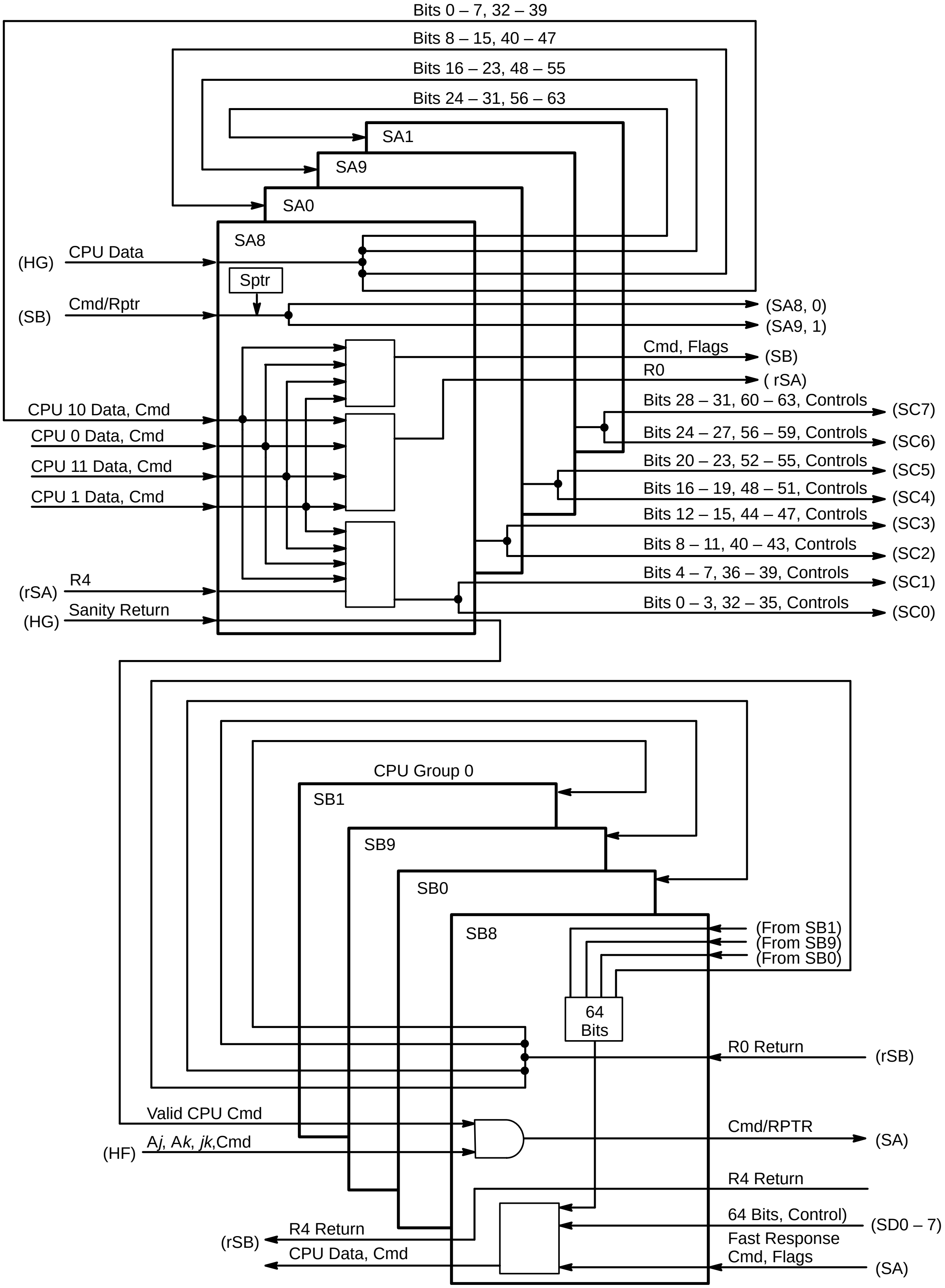
A shared register write operation starts when the command code and the data contained in the A*j* or A*k* register or the *jk* field of the instruction is sent from the HF option on the CPU to the SB option. This example uses CPU 10, port 8. The SB option constructs the destination register pointer (Rptr) signal from the A*j*, A*k*, or *jk* data. If SB8 receives a valid command code from the SA8 option indicating that CPU 10 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 travel the same path to the SA option in two successive, even clock phases, with the cmd signal followed by the Rptr signal.

Table 11 through Table 14, at the end of this document, list the source option, output term, destination option, input term and signal description for group 0.

At the same time that the cmd and Rptr signals enter the SA8 option, the data from the HG option on CPU 10 also enters the SA8 option. The data arrives in two 32-bit transfers. If the command code is valid, this data is then split into 16-bit slices that are sent 8 bits at a time in successive, even clock phases to each of the four SA options in that group; in this case, group 0.

Figure 9 shows the data and control fanout for group 0. The other groups perform similar functions.

Figure 9. Group 0 Data and Control Fanout



Effective CLN

Before the cmd and Rptr signals are sent to the other SA options, the steering pointer (Sptr) 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 10 is first read from a holding register on the option and then added to the cluster offset that was set during system configuration. The resulting CLN is then checked to verify that it is between 1 and 44 inclusive for a system with two shared modules or between 1 and 22 inclusive for a system with one shared module. The logic also verifies that the assigned CLN for CPU 10 is within the cluster range that was set in the system configuration. If either of these tests fails, the eCLN is set to –1 to denote an illegal Sptr signal. If the resulting CLN is not illegal, then a 6-bit eCLN is generated.

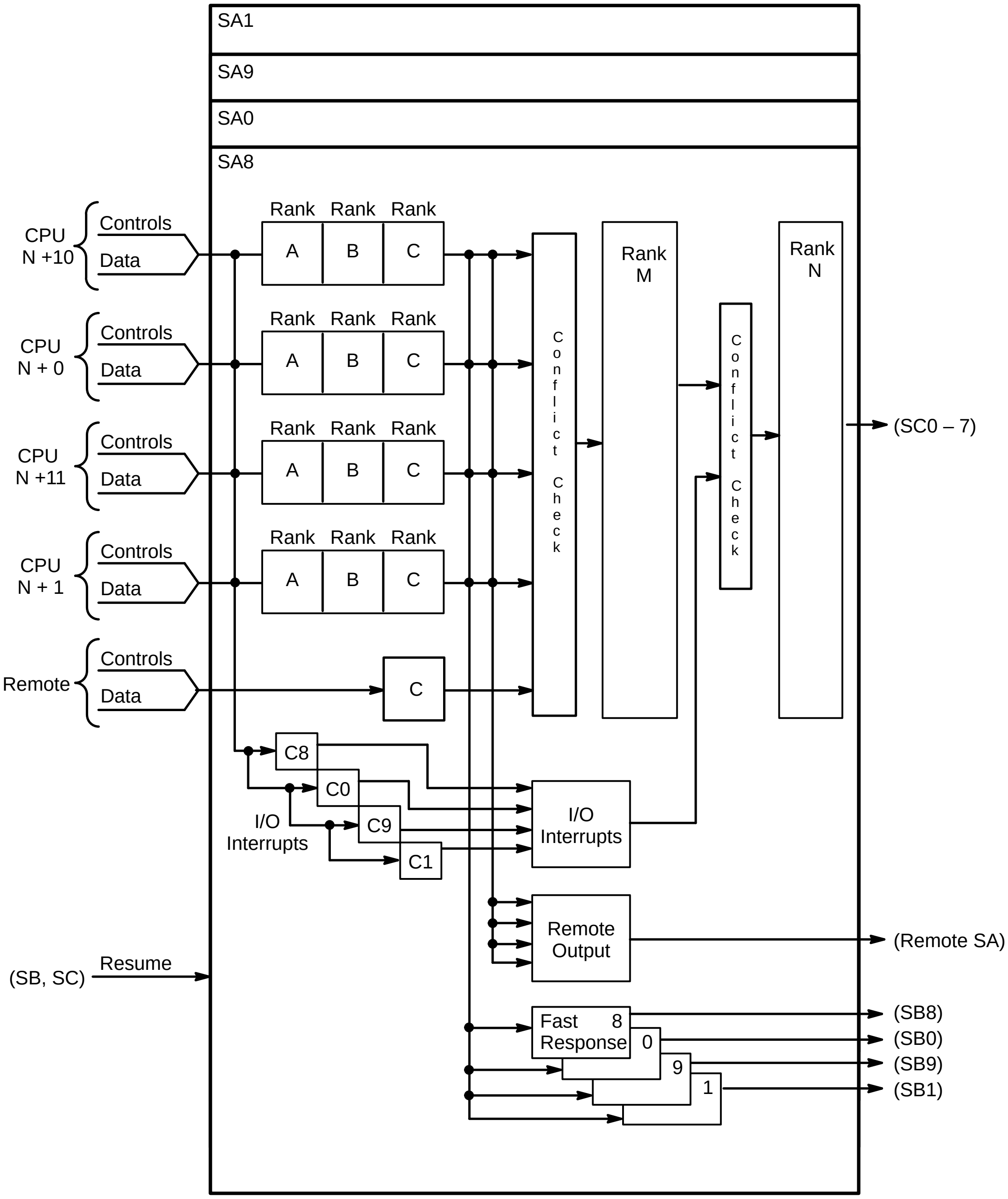
The three control signals (cmd, Rptr, and Sptr) are sent to the other SA options via two paths. One path sends the controls to the SA8 and SA0 options, and the other path sends the controls to the SA9 and SA1 options. Both paths send the controls during two successive, even clock phases. The cmd signal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

SA Option Operation

When the data and controls enter the four SA options, they are placed in separate FIFO (first-in-first-out) queues, as shown in Figure 10. As the data and controls move through 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 use of the output path. Initially, the priority is CPU 10, CPU 0, CPU 11, 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 10 makes a reference, the new order of priority would be CPU 0, CPU 11, CPU 1, and CPU 10.

Figure 10. SA Option Operations



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 10, CPU 0, CPU 11, 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 11, CPU 1, R4, CPU 10, and CPU 0.

Fast Response Data

Before the data and controls move from rank C to rank M, a check is first made to ensure that 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. Table 7 list the functions of the flags. This information is collectively referred to as the *fast response data*. If the fast response output buffer for CPU 10 is busy, it will prevent the data and controls from advancing from rank C to rank M.

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 10. There it is used to release the reservation on the shared register path thus allowing additional references from CPU 10.

Table 7. 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, which have priority.

Source CPU (Scpu)

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 that identifies 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. The SC3 and SC4 options each generate 1 of the remaining 2 bits.

Each SA option sends 8 bits of data to two SC options. The data is sent in two successive clock phases via different paths, with 4 bits sent during each phase. Figure 11 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.

SC Option Operation

The SC options receive data and control information from each of the four CPU groups and route it to one of the nine SR options. Figure 12 shows the internal operations of the SC options. When the data and control information for a reference to a particular cluster enters the SC options, it is 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, the data and control information in buffers B and C move up to buffers A and B. I/O and RTC references are handled in the same manner, but they use only two buffers.

Figure 11. SC Option Data Fanin/Fanout

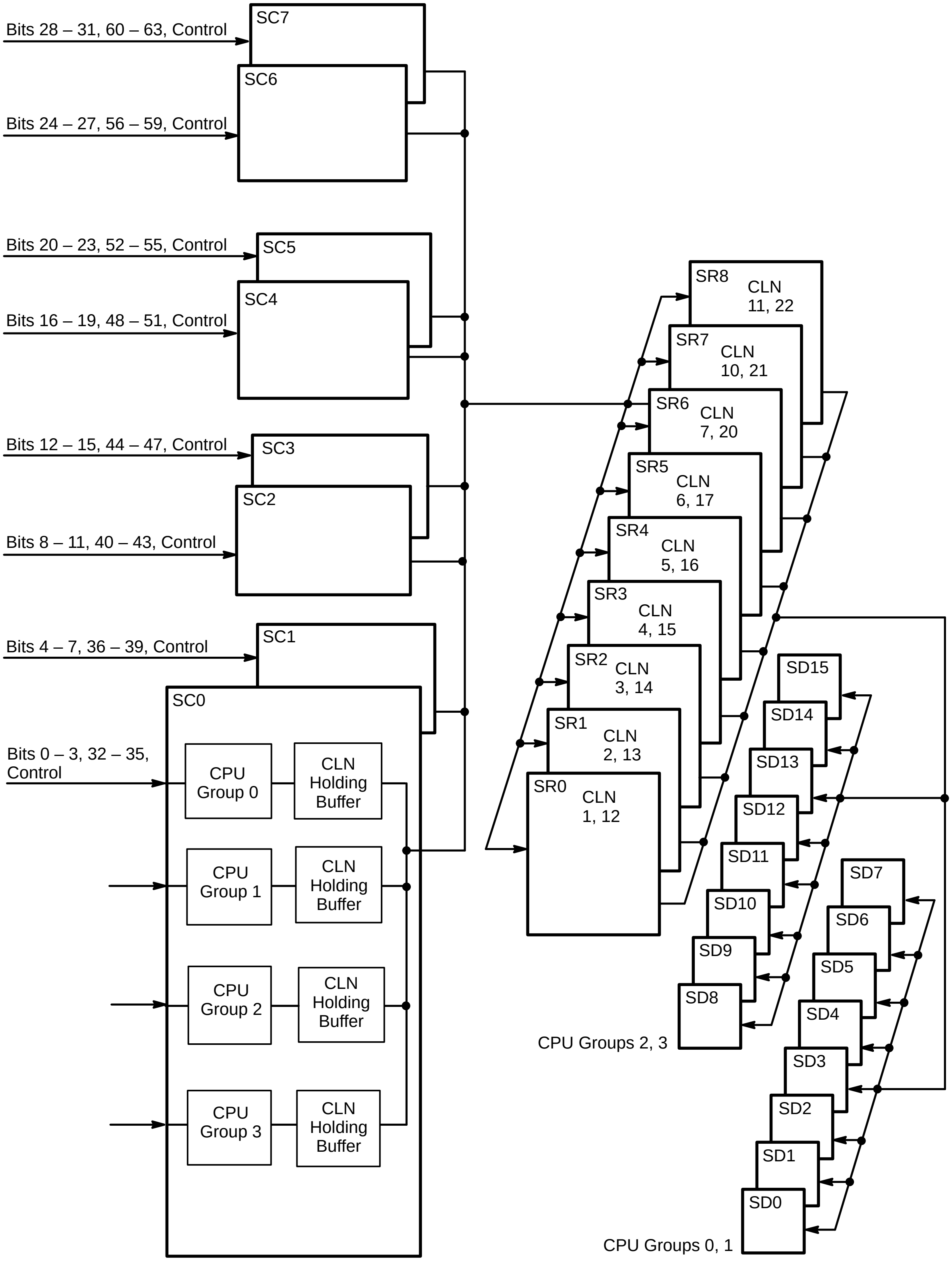
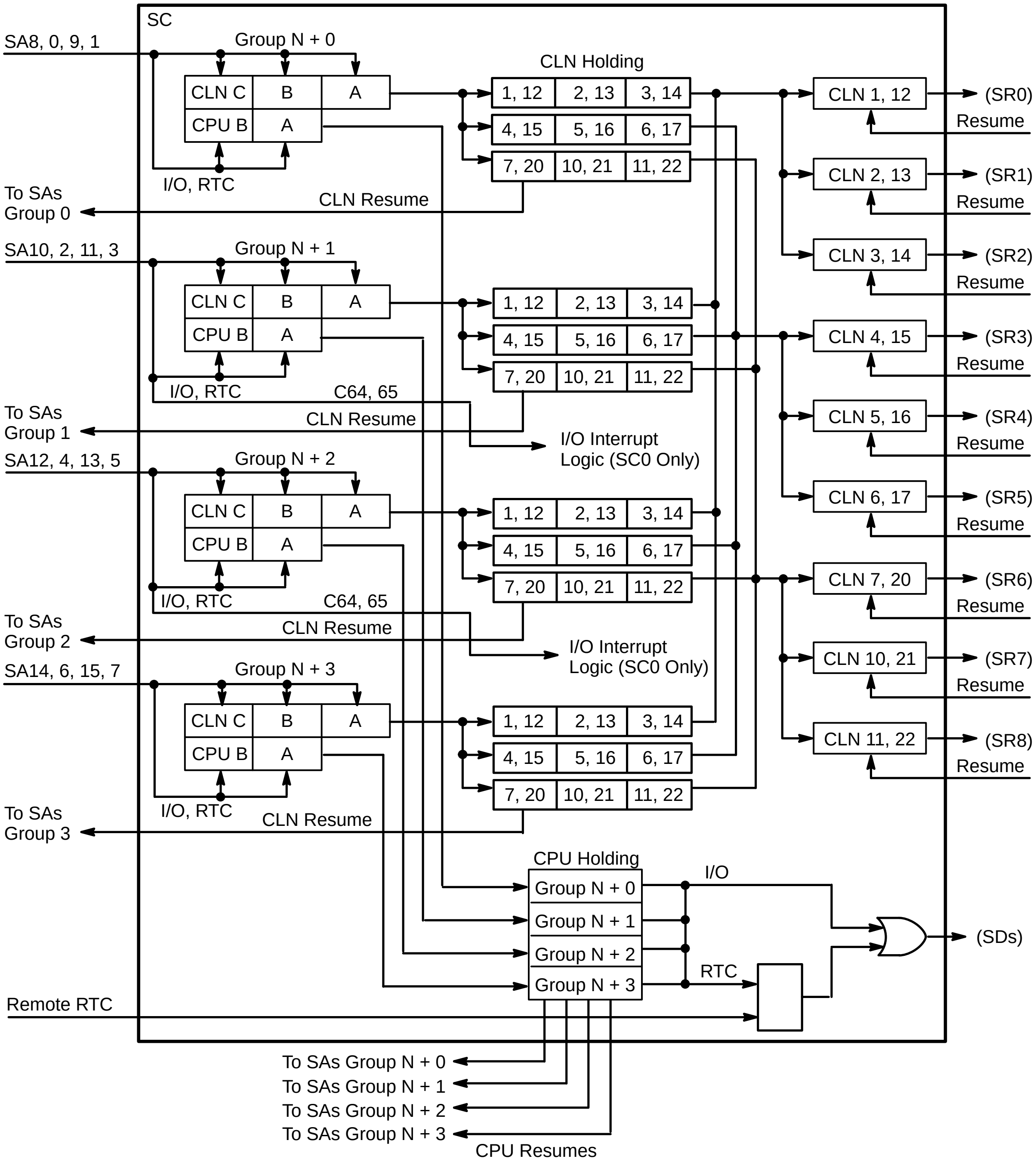


Figure 12. SC Option Operations



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. Each time data is latched into a holding buffer on an option, a Resume signal is sent to the option from which the data came. This enables 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 8 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 8. SR Option Cluster Assignment

Option	Clusters (SR00)	Clusters (SR01)
SR0	1, 12	23, 34
SR1	2, 13	24, 35
SR2	3, 14	25, 36
SR3	4, 15	26, 37
SR4	5, 16	27, 40
SR5	6, 17	30, 41
SR6	7, 20	31, 42
SR7	10, 21	32, 43
SR8	11, 22	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 that operates independently.

SR Option Operation

Each SC option sends its 8 bits of data to the selected SR option in two even clock phases via two separate paths. The control signals are sent one 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. Note that CLN0 and CLN1 are used in reference to the two clusters contained on the particular SR option. Refer to Table 8 for specific SR option locations and cluster assignments.

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 13. 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, and a Resume signal is sent back to the SC option. 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 9 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 either 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 that contain 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.

During 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. During an 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.

Figure 13. SR Option Data Input/Output

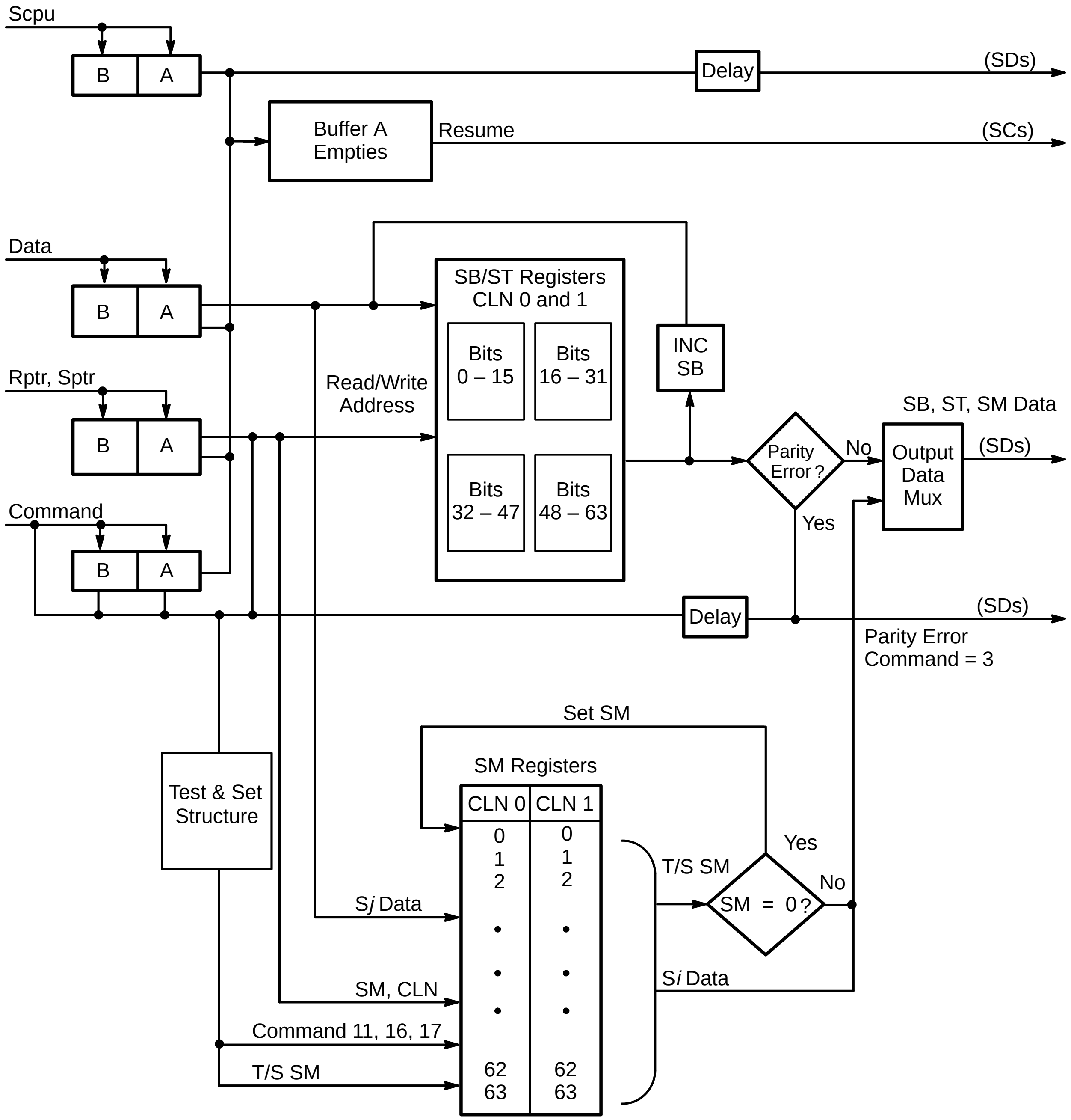


Table 9. SR Option Codes

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*)

Shared Register Read Operation

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.

SR/SD Option Operation

For every SB and 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 parity bits differ from the parity bits 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 43. In either case, the read-out data is sent to the SD options. The data and command code are sent to the issuing CPU. The CPU receives not only the command code of 43 but also a generated command code of 04, parity error.

For a broadside SM register read operation, all 64 SM registers are read simultaneously, and the read-out data is treated as a single word. Test and set operations are discussed at the end of this subsection.

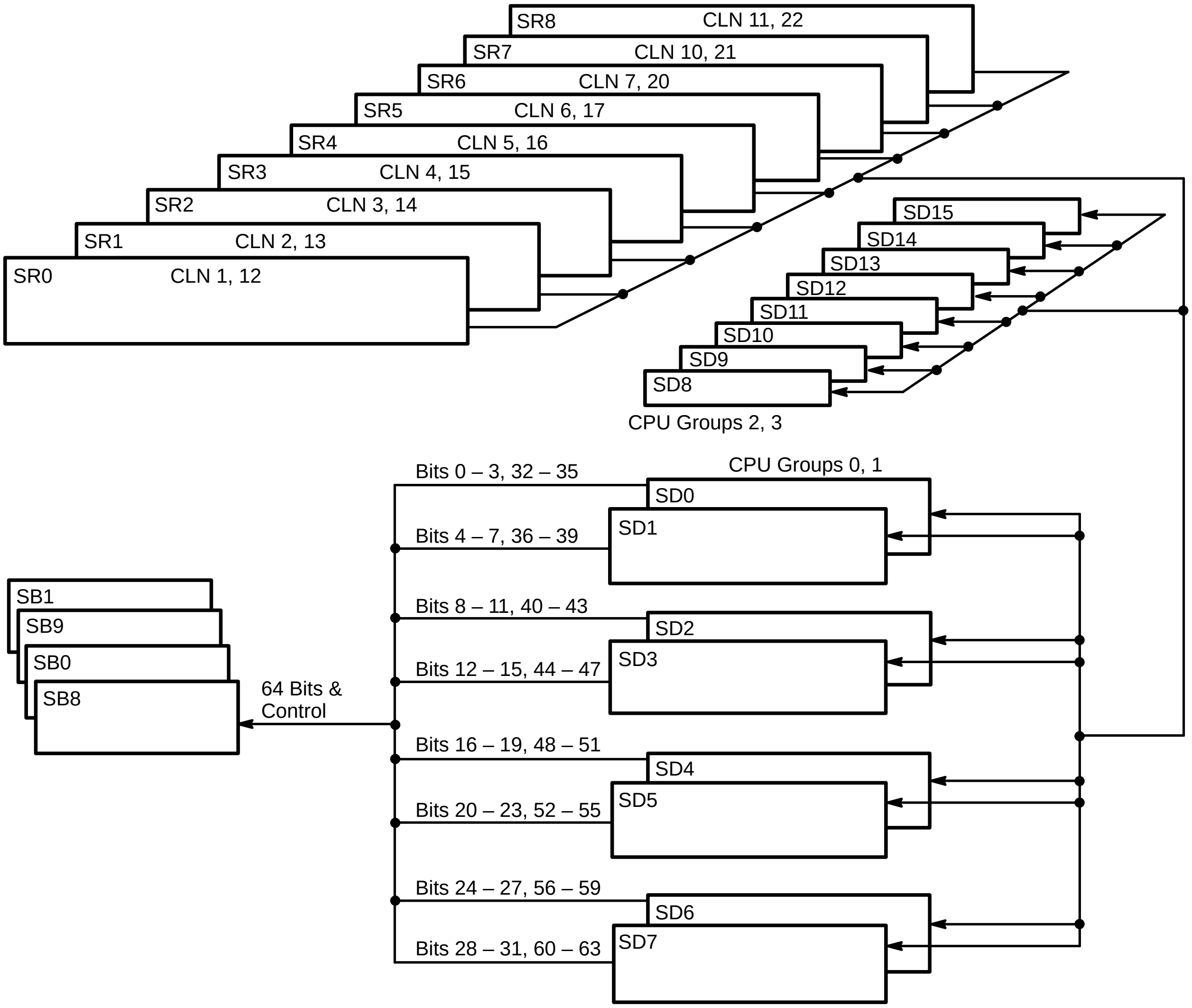
Two control signals accompany the data as it makes its way through the SD and SB options en route to CPU 10. One is the cmd code, and the other is the Scpu signal. Because the Scpu signal is used to steer the data and cmd signal to CPU 10, it is renamed the Sptr(scpu) signal, referred to simply as the Sptr signal. Do not confuse this signal with the Sptr(eCLN) signal that steers write data to the SR option.

The SR option also sends the SR option code, now referred to as the SR response, and Sptr signals to four SD options in CPU groups 0, 1 or to four SD options in CPU groups 2, 3. 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.

The two control signals are immediately fanned out from the four SD options to the remaining SD options. The control signals, however, can be gated into either input buffer A or input buffer B, as shown in Figure 15. 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 10. 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 through 47 or 30 through 37, respectively.

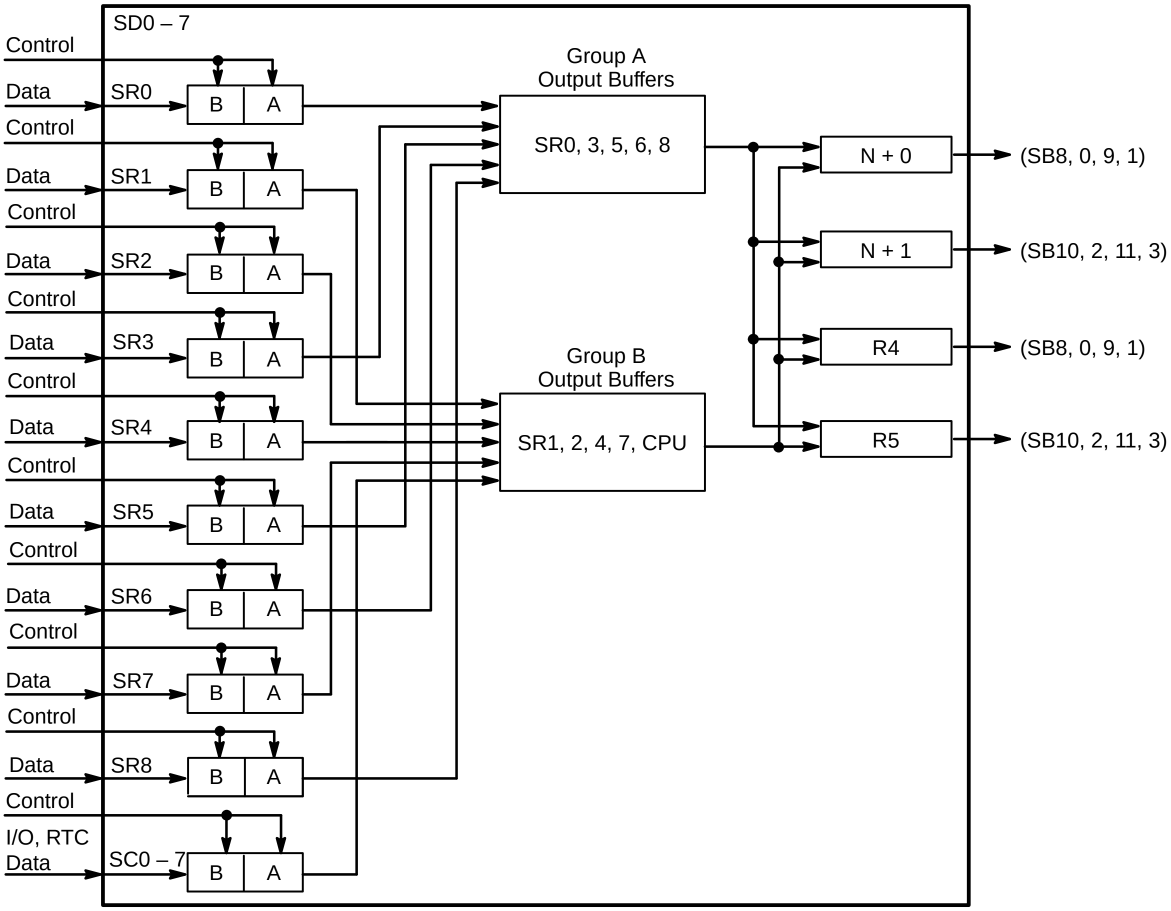
The SR option sends 8 bits of data to each of the eight SD options that handle data for the destination CPU, as shown in Figure 14. Options SD0 through SD7 handle data for CPUs in groups 0 and 1, and options SD8 through SD15 handle data for CPUs in groups 2 and 3. The data is sent 4 bits at a time in two successive, even clock phases. When the data enters the SD options, it is placed in input buffer B; from there it must first move to input buffer A before going further, as shown in Figure 15.

Figure 14. SR to SD Data Distribution



As the data and controls move out of input buffer A, they are gated into one of ten output holding buffers that correspond 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 15. 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 15. 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.

Figure 15. SD Option Operations



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 that the read reference was made from a CPU in group 0 on the other shared module.

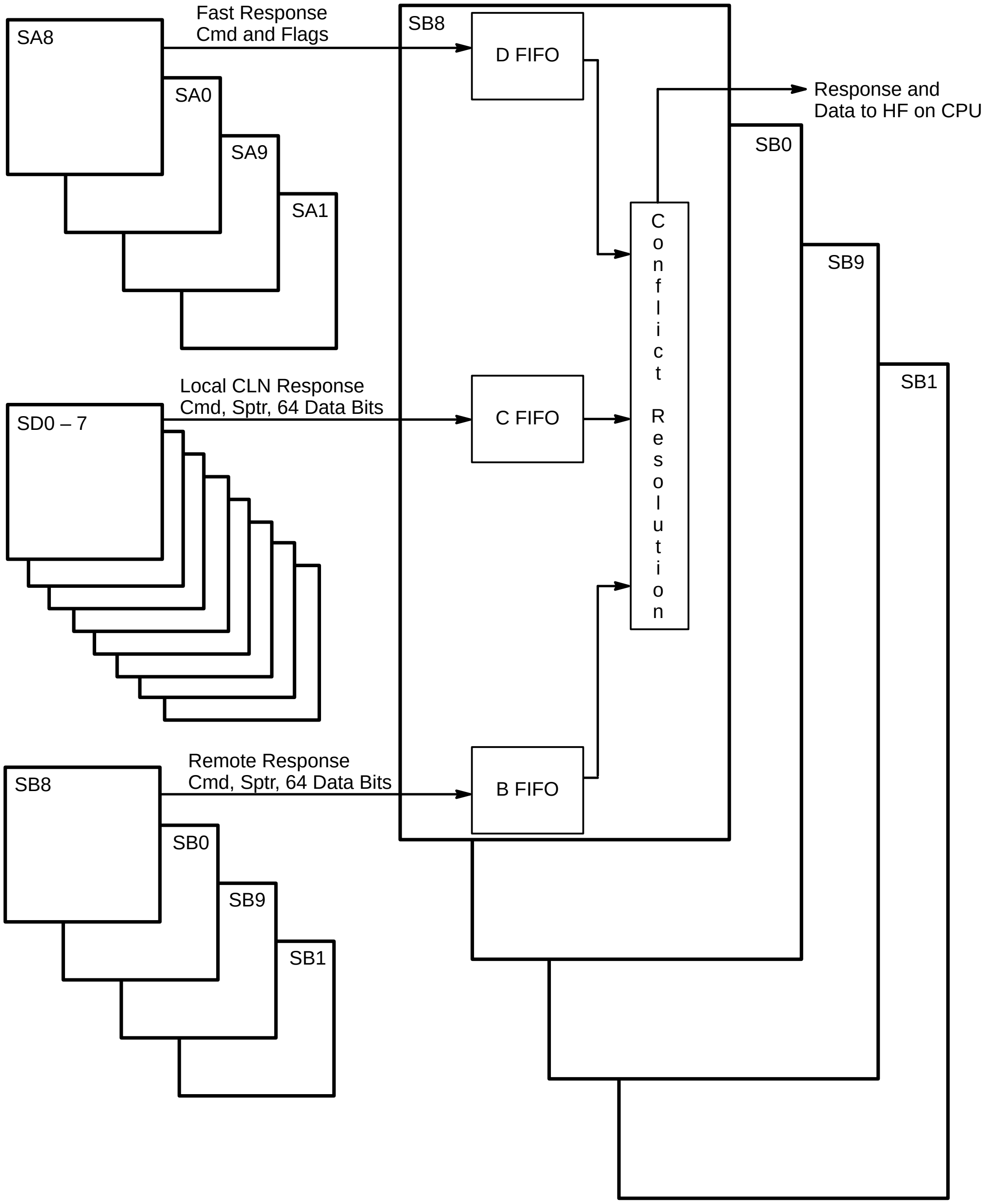
For a local read reference using the N + 0 output path, each of the SD options sends 8 bits of data to each of the SB options. The data is sent 4 bits at a time in two successive, even clock phases via different output paths. Thus, each of the SB options receives all 64 bits of the data. The 6-bit cmd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

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 two 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 two 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 16. 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 two successive, even clock phases via the same output path. The cmd signal is also sent to the HF option to release the reservation on the CPU shared path.

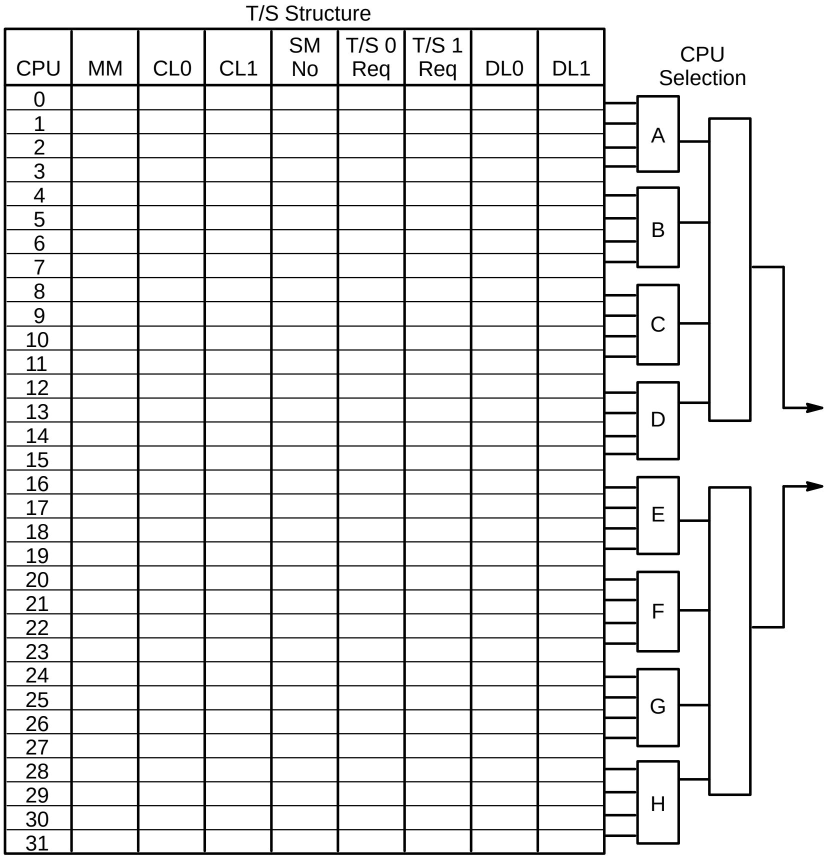
Figure 16. SB Option Operations



Test and Set Structure

The following paragraphs describe the test and set (T/S) structure logic on the SR option. Refer to Figure 17 when reading the following text. The T/S structure on each SR option stores information about each of the 32 CPUs, including the operating mode (MM) of the CPU (monitor mode or user mode), whether it is attached to either or both of the clusters on the SR option (CL0 or CL1), the number of the SM register to be checked (SM No), and whether there is a T/S request currently being processed in the CPU (T/S 0 Req, T/S 1 Req). If a CPU attached to a particular cluster is in monitor mode and has a test and set request pending, the deadlock flag for that CPU and cluster is set (DL0, DL1).

Figure 17. Test and Set Structure



T/S Structure

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								

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 use of 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 17. 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 priorities 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.

Because there is no way to determine which CPU's request will complete 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 34 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.

Real-time Clock Operation

Instruction 0014*j*0 is used to set the RTC by transmitting the value stored in the S*j* register to the real-time clock (HD option in the CPU). This instruction uses the logic on the shared module to synchronize the CPUs so that 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. The letter *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 24. 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 code 6 and checked to verify that it is a valid code for the shared module. The Sptr signal, consisting of the 2 bits that designate 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 that identifies which CPU 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. Refer again to Figure 12, which shows the SC options. 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 conflicts 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.

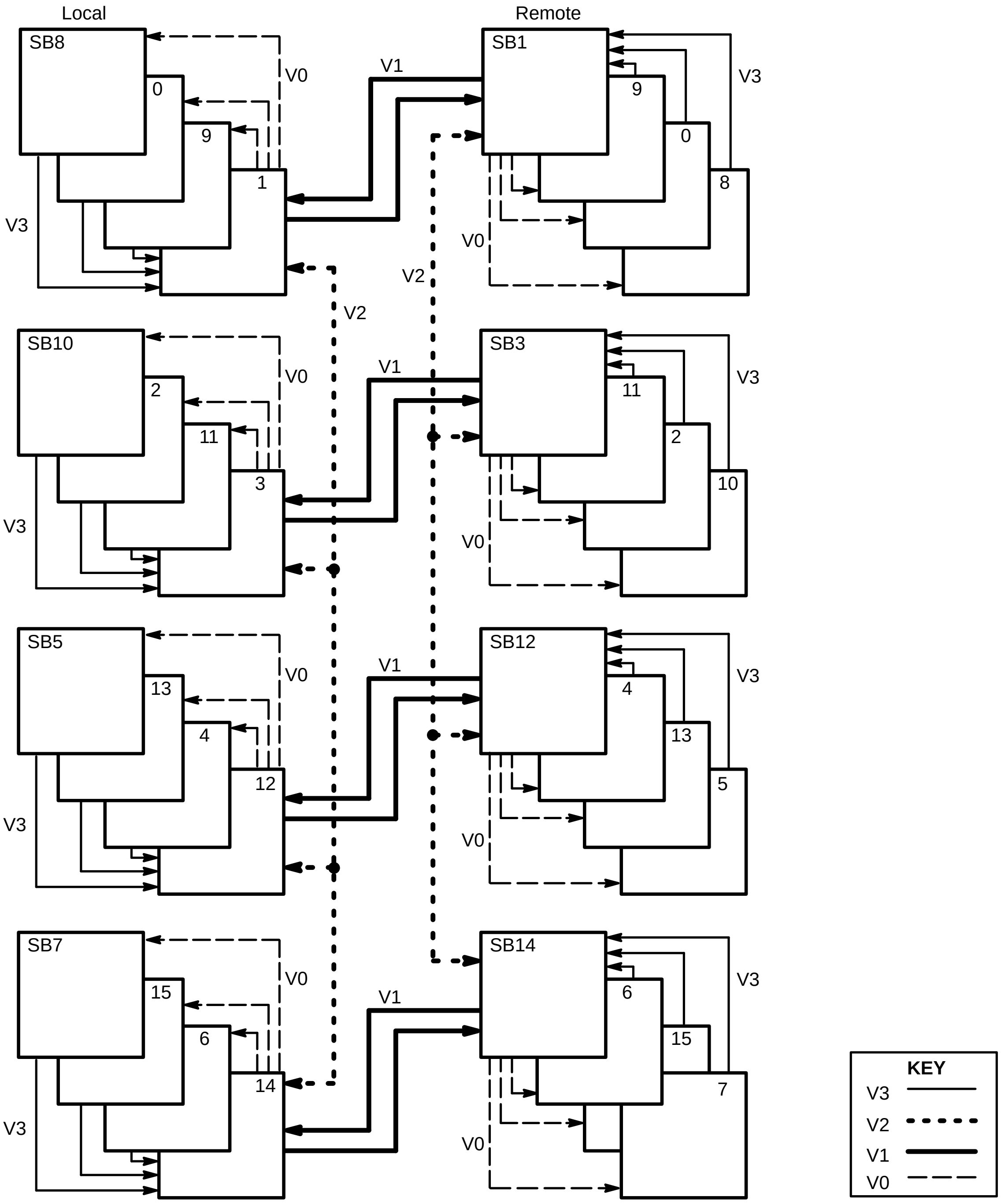
After a CPU group has been selected, a signal is sent to both the local and remote SC options to stop any additional RTC references. The selected group must then compete for use of the output path with an RTC reference from the remote shared module. The priority in this case toggles between the local 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 15 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 cmd code of 6 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 cmd code of 6, it holds the command and data and shuts down all operations on the option. This shut-down prevents all 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 18 shows the voting logic used between the SB options.

One SB option in each CPU group is designated as a controller to monitor SB option shut-down. The controlling options on the local shared module are SB1, SB3, SB12, and SB14. As each SB option detects the cmd code of 6, 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.

Figure 18. RTC Voting Logic



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.

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 cmd code of 6 and the S*j* data to its CPU's HF option. From there, the data and cmd code are 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 shut-down condition and resuming normal option operations. The SB options listed in Table 10 also send the Send Next RTC signal to the SC options, which enables more RTC references. The SB*x* option also converts the cmd code of 6 to the original cmd code of 24 and sends it to CPU*x* to complete the instruction.

Table 10. 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

Set Interprocessor Interrupt (SIPI)

Issuing the 0014*j*1 instruction causes one CPU to interrupt another CPU. This instruction is used at system start-up time when CPU 0 interrupts the other CPUs so they perform an exchange and start processing.

The issuing CPU's HF option decodes the instruction and sends a command code of 25 to the shared module. Additionally, an 8-bit pointer is sent that specifies the CPU to be interrupted.

The SB option receives the command and pointer and forwards them to the SA when it detects sanity code from the CPU. The SB option will also send a fast response of 25 back to the issuing CPU.

The command and 8-bit pointer are sent from the SA option to the SC options. The SC options then send the command and data to the SD options. The SD options broadcast the command and 8-bit pointer to all SB options. Only the port responsible for the CPU to be interrupted will accept the broadcast. That SB option will send the SIPI signal to the CPU, which causes an interrupt.

Exchange Sequence

An exchange sequence that occurs in a CPU generates a command code of 7. The command 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 CLN in bits 0 through 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 (indicating user mode), then the cmd, Scpu, and eCLN (derived from the incoming CLN) signals are sent to the appropriate SR option through the SC options. The SR option uses the Scpu to attach the CPU to the specified cluster.

The data and control information is 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 on 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.

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 1, indicating a multiple CLN flush, is generated. If there is no master clear sequence in progress, a cmd code of 5, indicating a single CLN flush, is generated.

A multiple flush detaches the CPU from all attached clusters. The single flush detaches the CPU from only the cluster specified in the cluster register in the SA option. The flush must complete before the master clear exchange sequence can proceed.

Control Signal Tables

Table 11. Cmd Control Signal Path for CPU 10

Source Option	Output Terms	Destination Option(s)	Input Terms	Description
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 OJA – OJF	SA8, 0 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, OJJ, OKJ, OLJ, OMJ, ONJ, OOJ, OPJ	SR0, 1, SR2, 3 SR4, 5 SR6, 7 SR8	IAA	Bit 0 of cmd signal (Each output term is sent to one SR option.)
SC1	OHJ, OIJ, OJJ, OKJ, OLJ, OMJ, ONJ, OOJ, OPJ	SR0, 1, SR2, 3 SR4, 5 SR6, 7 SR8	IAB	Bit 1 of cmd signal (Each output term is sent to one SR option.)
SC2	OHJ, OIJ, OJJ, OKJ, OLJ, OMJ, ONJ, OOJ, OPJ	SR0, 1, SR2, 3 SR4, 5 SR6, 7 SR8	IAC	Bit 2 of cmd signal (Each output term is sent to one SR option.)
SC3	OHJ, OIJ, OJJ, OKJ, OLJ, OMJ, ONJ, OOJ, OPJ	SR0, 1, SR2, 3 SR4, 5 SR6, 7 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 OEA – OED	SD0, 1 SD2, 3	IEA – IED	4-bit code from CLN M+0, N+0

Source Option	Output Terms	Destination Option(s)	Input Terms	Description
SR1	ODA – ODD OEA – OED	SD0, 1 SD2, 3	IGA – IGD	4-bit code from CLN M+1, N+1
SR2	ODA – ODD OEA – OED	SD0, 1 SD2, 3	IIA – IAD	4-bit code from CLN M+2, N+2
SR3	ODA – ODD OEA – OED	SD4, 5 SD6, 7	IKA – IKD	4-bit code from CLN M+3, N+3
SR4	ODA – ODD OEA – OED	SD4, 5 SD6, 7	IMA – IMD	4-bit code from CLN M+4, N+4
SR5	ODA – ODD OEA – OED	SD4, 5 SD6, 7	IOA – IOD	4-bit code from CLN M+5, N+5
SR6	ODA – ODD OEA – OED	SD4, 5 SD6, 7	IQA – IQD	4-bit code from CLN M+6, N+6
SR7	ODA – ODD OEA – OED	SD4, 5 SD6, 7	ISA – ISD	4-bit code from CLN M+7, N+7
SR8	ODA – ODD OEA – OED	SD4, 5 SD6, 7	IUA – IUD	4-bit code from CLN M+8, N+8
SD1	OQA – OQD OSA – OSD OUA – OUD	SD4, 5	IEA – IED IGA – IGD IIA – IID	4-bit code from CLN M+0, N+0 M+1, N+1 M+2, N+2
SD3	OQA – OQD OSA – OSD OUA – OUD	SD6, 7	IEA – IED IGA – IGD IIA – IID	4-bit code from CLN M+0, N+0 M+1, N+1 M+2, N+2
SD4	OQA – OQD OSA – OSD OUA – OUD	SD0, 1	IQA – IQD ISA – ISD IUA – IUD	4-bit code from CLN M+6, N+6 M+7, N+7 M+8, N+8
SD5	OQA – OQD OSA – OSD OUA – OUD	SD0, 1	IKA – IKD IMA – IMD IOA – IOD	4-bit code from CLN M+3, N+3 M+4, N+4 M+5, N+5
SD6	OQA – OQD OSA – OSD OUA – OUD	SD2, 3	IQA – IQD ISA – ISD IUA – IUD	4-bit code from CLN M+6, N+6 M+7, N+7 M+8, N+8
SD7	OQA – OQD OSA – OSD OUA – OUD	SD2, 3	IKA – IKD IMA – IMD IOA – IOD	4-bit code from CLN M+3, N+3 M+4, N+4 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	”

Source Option	Output Terms	Destination Option(s)	Input Terms	Description
SD7	OAA – OAF	SB11	IMA – IMF	”
SB8	OAA – OAF	HF	ICQ – ICV	6-bit response code

Table 12. Rptr Control Signal Path for CPU 10

Source Option	Output Terms	Destination Option	Input Terms	Description
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 OII – OIJ OJI – OJJ OKI – OKJ OLI – OLJ OMI – OMJ ONI – ONJ OOI – OOJ OPI – OPJ	SR0 SR1 SR2 SR3 SR4 SR5 SR6 SR7 SR8	ICA – ICB	Bits 0 and 1 of Rptr signal
SC6	OHI – OHJ OII – OIJ OJI – OJJ OKI – OKJ OLI – OLJ OMI – OMJ ONI – ONJ OOI – OOJ OPI – OPJ	SR0 SR1 SR2 SR3 SR4 SR5 SR6 SR7 SR8	ICC – ICD	Bits 2 and 3 of Rptr signal
SC7	OHI – OHJ OII – OIJ OJI – OJJ OKI – OKJ OLI – OLJ OMI – OMJ ONI – ONJ OOI – OOJ OPI – OPJ	SR0 SR1 SR2 SR3 SR4 SR5 SR6 SR7 SR8	ICE – ICF	Bits 4 and 5 of Rptr signal

Table 13. Sptr(eCLN) Control Signal Path for CPU 10

Source Option	Output Terms	Destination Option	Input Terms	Description
SA8	OIA – OIF OJA – OJF	SA8, 0 SA9, 1	IIA – IIF	Sptr = eCLN from SA to SR options; CLN is stored on SA options 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 OIJ OJJ OKJ OLJ OMJ ONJ OOJ OPJ	SR0 SR1 SR2 SR3 SR4 SR5 SR6 SR7 SR8	IBA	1-bit cluster pointer; only 1 bit is needed because there are only two clusters on each SR option.

Table 14. Sptr(Scpu) Control Signal Path for CPU 10

Source Option	Output Terms	Destination Option	Input Terms	Description
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 OII OJI OKI OLI OMI ONI OOI OPI	SR0 SR1 SR2 SR3 SR4 SR5 SR6 SR7 SR8	IDA	Bit 0 of Scpu signal
SC1	OHI OII OJI OKI OLI OMI ONI OOI OPI	SR0 SR1 SR2 SR3 SR4 SR5 SR6 SR7 SR8	IDB	Bit 3 of Scpu signal
SC2	OHI OII OJI OKI OLI OMI ONI OOI OPI	SR0 SR1 SR2 SR3 SR4 SR5 SR6 SR7 SR8	IDC	Bit 4 of Scpu signal
SC3	OHI OII OJI OKI OLI OMI ONI OOI OPI	SR0 SR1 SR2 SR3 SR4 SR5 SR6 SR7 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.

Source Option	Output Terms	Destination Option	Input Terms	Description
SC4	OHI OII OJI OKI OLI OMI ONI OOI OPI	SR0 SR1 SR2 SR3 SR4 SR5 SR6 SR7 SR8	IDE	Bit 2 of Scpu signal
SR0	OFA – OFE OGA – OGE	SD0, 1 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 OGA – OGE	SD0, 1 SD2, 3	IGQ – IGU	Bits 0, 1, 3, and 4 of Scpu signal from CLN M+1, N+1
SR2	OFA – OFE OGA – OGE	SD0, 1 SD2, 3	IIQ – IAU	Bits 0, 1, 3, and 4 of Scpu signal from CLN M+2, N+2
SR3	OFA – OFE OGA – OGE	SD4, 5 SD6, 7	IKQ – IKU	Bits 0, 1, 3, and 4 of Scpu signal from CLN M+3, N+3
SR4	OFA – OFE OGA – OGE	SD4, 5 SD6, 7	IMQ – IMU	Bits 0, 1, 3, and 4 of Scpu signal from CLN M+4, N+4
SR5	OFA – OFE OGA – OGE	SD4, 5 SD6, 7	IOQ – IOU	Bits 0, 1, 3, and 4 of Scpu signal from CLN M+5, N+5
SR6	OFA – OFE OGA – OGE	SD4, 5 SD6, 7	IQQ – IQU	Bits 0, 1, 3, and 4 of Scpu signal from CLN M+6, N+6
SR7	OFA – OFE OGA – OGE	SD4, 5 SD6, 7	ISQ – ISU	Bits 0, 1, 3, and 4 of Scpu signal from CLN M+7, N+7
SR8	OFA – OFE OGA – OGE	SD4, 5 SD6, 7	IUQ – IUU	Bits 0, 1, 3, and 4 of Scpu signal from CLN M+8, N+8
SD1	ORA – ORE OTA – OTE OVA – OVE	SD4, 5	IEQ – IEU IGQ – IGU IIQ – IIU	4 bits of Scpu from CLN M+0, N+0 M+1, N+1 M+2, N+2
SD3	ORA – ORE OTA – OTE OVA – OVE	SD6, 7	IEQ – IEU IGQ – IGU IIQ – IIU	4 bits of Scpu from CLN M+0, N+0 M+1, N+1 M+2, N+2
SD4	ORA – ORE OTA – OTE OVA – OVE	SD0, 1	IQQ – IQU ISQ – ISU IUQ – IUU	4 bits of Scpu from CLN M+6, N+6 M+7, N+7 M+8, N+8
SD5	ORA – ORE OTA – OTE OVA – OVE	SD0, 1	IKQ – IKU IMQ – IMU IOQ – IOU	4 bits of Scpu from CLN M+3, N+3 M+4, N+4 M+5, N+5
SD6	ORA – ORE OTA – OTE OVA – OVE	SD2, 3	IQQ – IQU ISQ – ISU IUQ – IUU	4 bits of Scpu from CLN M+6, N+6 M+7, N+7 M+8, N+8

Source Option	Output Terms	Destination Option	Input Terms	Description
SD7	ORA – ORE OTA – OTE OVA – OVE	SD2, 3	IKQ – IKU IMQ – IMU IOQ – IOU	4 bits of Scpu from CLN M+3, N+3 M+4, N+4 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	”