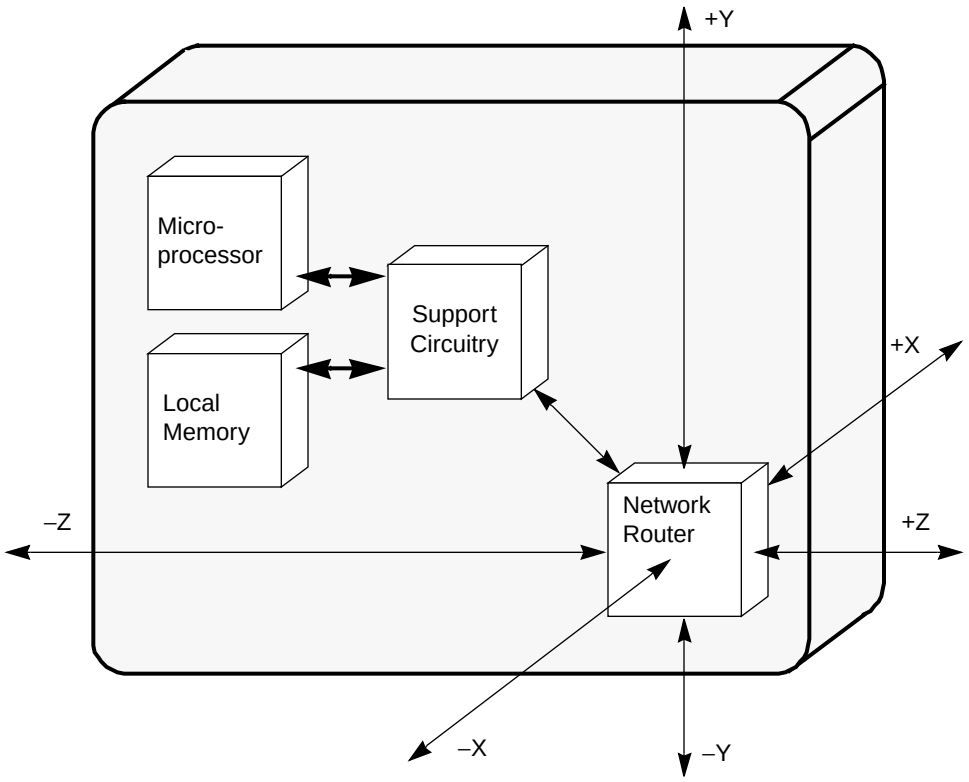
Addressing

HMM-140-0

Addressing Overview

The CRAY T3E system has two categories of memory: local memory and remote memory. With respect to the microprocessor in a processing element (PE), local memory is physically located in the same PE as the microprocessor (refer to Figure 1). Each PE contains a low-latency, high-bandwidth data path between the microprocessor and local memory.

Figure 1. PE and Interconnect Network Components



With respect to the microprocessor in a PE, remote memory is memory that is physically located in another PE. Each PE contains support circuitry that controls the flow of data between the microprocessor and remote memory.

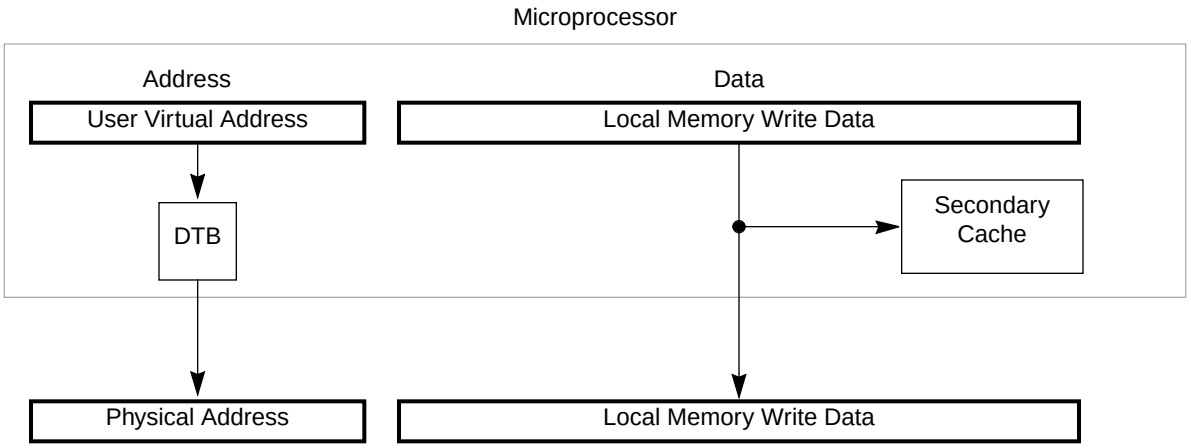
With respect to the microprocessor in a PE, global memory is the sum of local memory and all of remote memory. The microprocessor in each PE can access any location in global memory.

Each microprocessor in the CRAY T3E system accesses the different categories of memory using one of two methods: the direct local memory access method or the E-register access method.

Direct Local Memory Access Method

Software uses the direct local memory access method to transfer data between an internal microprocessor register and a physical address in local memory. As an example, Figure 2 shows the flow of address and data during a write to local memory using the direct local memory access method.

Figure 2. Direct Local Memory Access Method of Transferring Data to Local Memory



The microprocessor converts a user virtual address, which is generated by an application, into a physical address by using a data translation buffer (DTB). When a translation for the given user virtual address is not in the DTB, the microprocessor will give control to system software, which loads the DTB with the appropriate translation parameters.

The microprocessor presents the physical address and the write data for local memory to the support circuitry. The support circuitry then uses the physical address to reference a local memory location and transfers the data into that memory location.

During the data transfer, circuitry in the microprocessor ensures that data in the secondary cache of the microprocessor is consistent with data in local memory.

E-register Access Method

The microprocessor in a PE uses the E-register access method to transfer data to or from any location in global memory or any location in only local memory. E-registers are hardware registers that are located in the support circuitry and are used to hide the latency of global data transfers.

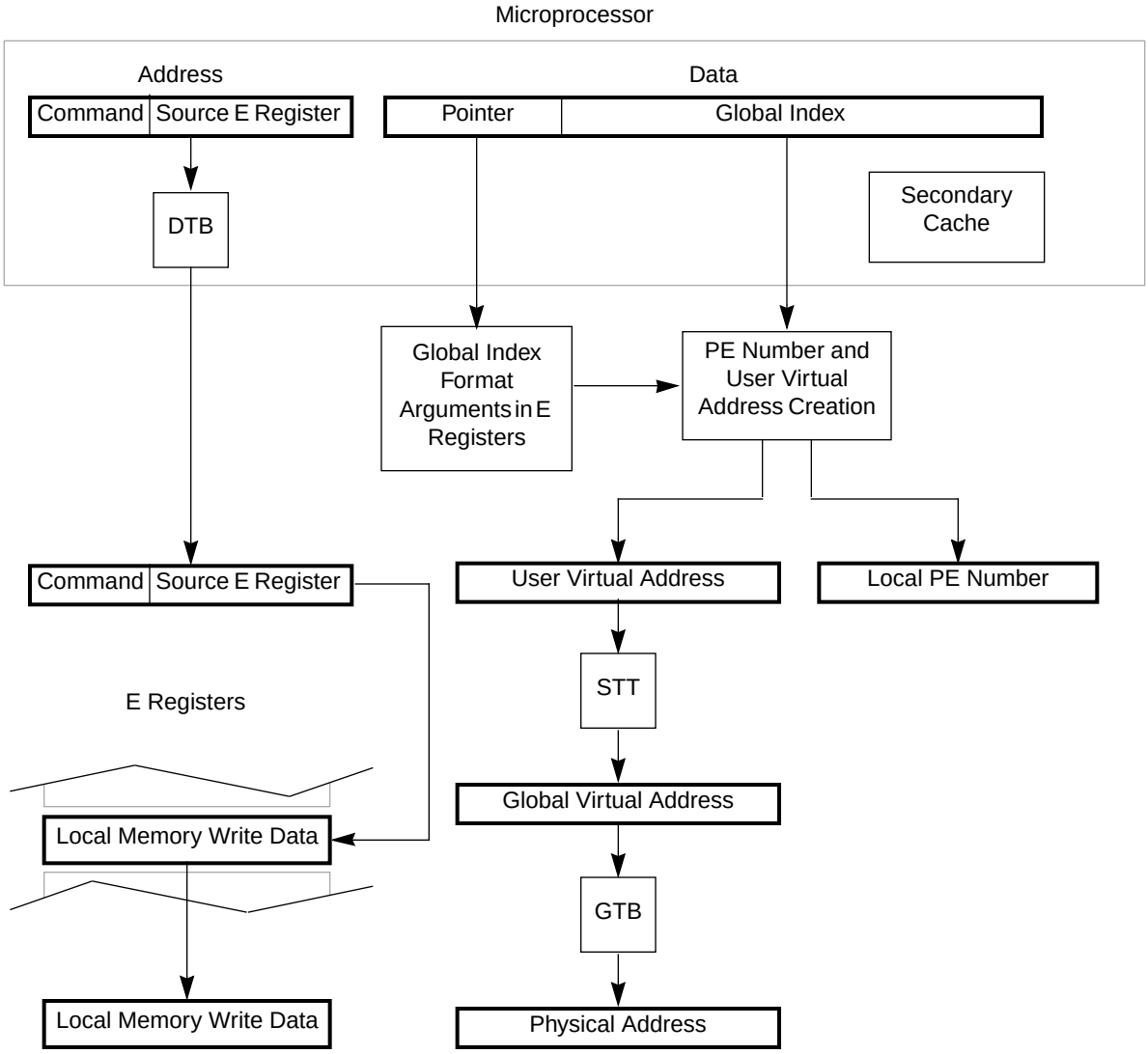
E-register Access to Global Memory

When the microprocessor uses the E-register access method to write data into a location in global memory, the microprocessor first stores write data in a source E register. The support circuitry then transfers the write data from the source E register to a destination global memory location.

When the microprocessor uses the E-register access method to read data from a location in global memory, the support circuitry first transfers read data from a location in global memory to a destination E register. The microprocessor then transfers the read data from the destination E register to a register internal to the microprocessor.

As an example, Figure 3 shows the flow of address and data during a write of data to global memory using the E-register access method. Before initiating the transfer of data, software must have loaded a source E register with the data that will be written into global memory. This data may have originated from the microprocessor or from any location in global memory.

Figure 3. E-register Access Method of Transferring Data to Global Memory



During the transfer, the address generated by the operating system contains command information and a source E-register reference. The command information indicates that data will be transferred from a source E register to a location in global memory. The source E-register reference identifies the E register that contains the write data. Generally, software sets the microprocessor DTB so it does not convert the command and source E-register reference information into another format.

When using the E-register access method to access global memory, the microprocessor data bus does not contain write data. Instead, the data bus contains a global index and a pointer. The global index is a software-formatted

address that includes a PE number and an offset within the user virtual address for that PE. The pointer identifies a group of 4 E registers that contain the software-defined global index format arguments.

By setting the format of the global index, software can interleave the global memory space using different combinations of PE number and offset. For example, Table 1 lists possible formats of a simplified, 4-bit global index.

Table 1. Sample Simplified Global Index Formats

Global Index Bits <3 : 0>	Bits <3 : 2>= PE Number Bits <1 : 0> = Offset		Bits <1 : 0> = PE Number Bits <3 : 2> = Offset		Bits 3 and 1 = PE Number Bits 2 and 0 = Offset	
	PE Number	Offset	PE Number	Offset	PE Number	Offset
0000	0	0	0	0	0	0
0001	0	1	1	0	0	1
0010	0	2	2	0	1	0
0011	0	3	3	0	1	1
0100	1	0	0	1	0	2
0101	1	1	1	1	0	3
0110	1	2	2	1	1	2
0111	1	3	3	1	1	3
1000	2	0	0	2	2	0
1001	2	1	1	2	2	1
1010	2	2	2	2	3	0
1011	2	3	3	2	3	1
1100	3	0	0	3	2	2
1101	3	1	1	3	2	3
1110	3	2	2	3	3	2
1111	3	3	3	3	3	3

After receiving the global index and pointer from the microprocessor, the support circuitry reads the global index format parameters from the E registers identified by the pointer. These parameters indicate which bits of the global index are PE number bits and which bits of the global index are offset bits. In addition, the parameters include a base user virtual address that is used as a reference for the offset.

Using the global index format parameters, the support circuitry separates the global index into a PE number and user virtual address. The PE number may refer to any PE in the system; however, the example shown in Figure 3 assumes that the PE number is equal to the local PE. Because the PE number is the local PE number, the support circuitry transfers data from the specified E register to a location in local memory. When the PE number is a remote PE number, the support circuitry creates a network packet and send the write data, address, and command information to the remote PE through the interconnect network.

After obtaining a user virtual address from the global index, the support circuitry converts the user virtual address into a physical address. This conversion is similar to the user virtual address to physical address conversion that the microprocessor performs (using the DTB) when using the direct local memory access method (refer again to Figure 2).

The support circuitry first converts the user virtual address into a global virtual address. The global virtual address is an intermediate address used by the operating system. To do this conversion, the support circuitry uses a segment translation table (STT). The STT is a software-defined look-up table that is located in the support circuitry. The STT contains the user virtual address to global virtual address translation parameters.

The support circuitry then converts the global virtual address into a physical address. To do the conversion, the support circuitry uses a global translation array and a global translation buffer (GTB). The global translation array is a software-defined look-up table that is stored in local memory. The global translation array contains information that is used to check permissions and translate the global virtual address into a physical address. The GTB is a hardware buffer in the support circuitry that stores frequently or recently accessed global translation array entries.

E-register Access to Local Memory Only

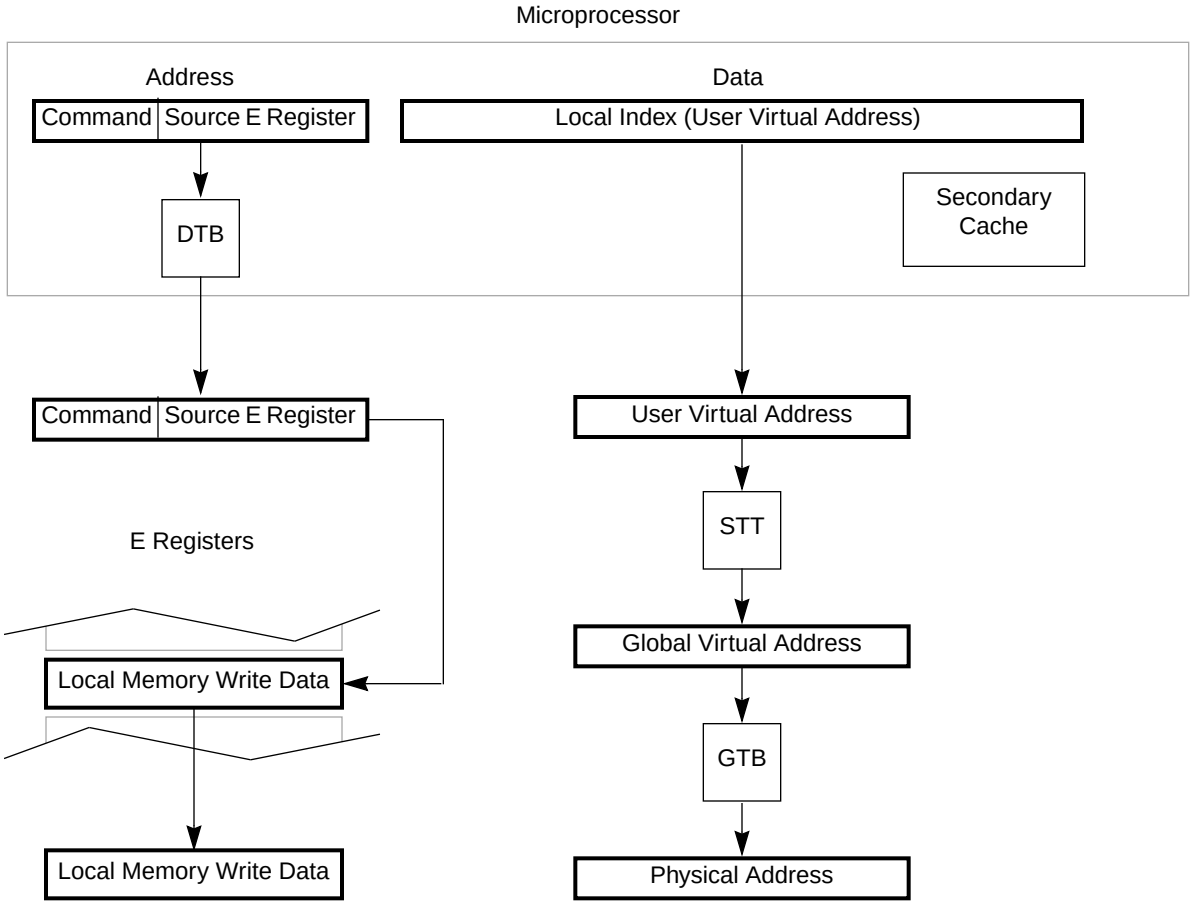
Although software generally uses the E-register access method to access data in global memory (which includes local memory), the E-register access method may also be used to access data in local memory only. This method allows the microprocessor to access local memory using a memory address that is formatted the same as a global address; however, when the microprocessor uses the E-register access method to access local memory only, the support circuitry does not have to perform the complete global address translation.

When the microprocessor uses the E-register access method to write data into a location in local memory only, the microprocessor first stores the write data in a source E register. The support circuitry then transfers the write data from the source E register to a destination local memory location.

When the microprocessor uses the E-register access method to read data from a location in local memory, the support circuitry first transfers the read data from a location in local memory to a destination E register. The microprocessor then transfers the read data from the destination E register to a register internal to the microprocessor.

As an example, Figure 4 shows the flow of address and data during a write of data to local memory using the local-only E-register access method. Before initiating the transfer of data, software must have loaded a source E register with the data that will be written into local memory. This data may have originated from the microprocessor or from any location in global memory.

Figure 4. E-register Access Method of Transferring Data to Local Memory Only



During the transfer, the address generated by the operating system contains command information and a source E-register reference. The command information indicates that data will be transferred from a source E register to local memory. The source E-register reference identifies the E register that contains the local memory write data. Generally, software sets the microprocessor DTB so it does not convert the command and E-register address information into another format.

When using the local-only E-register access method to access local memory, the data bus does not contain local memory write data. Instead, the data bus contains the user virtual address that was generated by an application. The support circuitry converts the user virtual address into a physical address. This conversion is similar to the user virtual address to physical address conversion that the microprocessor performs (using the DTB) when using the direct local memory access method (refer again to Figure 2).

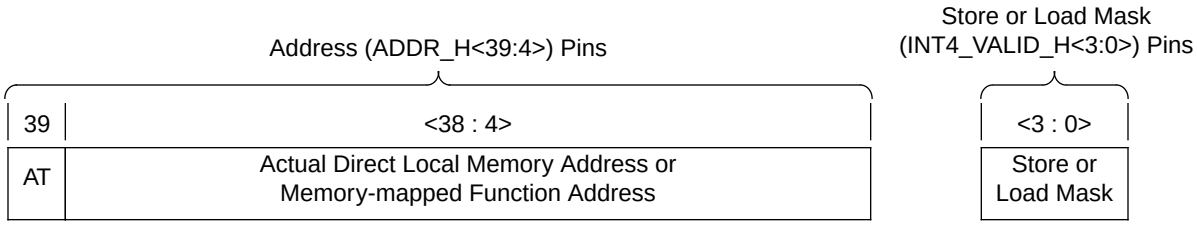
After receiving the user virtual address from the data bus, the support circuitry converts the user virtual address into a global virtual address. The global virtual address is an intermediate address used by the operating system. To do this conversion, the support circuitry uses an STT. The STT is a software-defined look-up table that is located in support circuitry registers. The STT contains the user virtual address to global virtual address translation parameters.

The support circuitry then converts the global virtual address into a physical address. To do this conversion, the support circuitry uses a global translation array and the global translation buffer (GTB). The global translation array is a software-defined look-up table that is stored in local memory. The global translation array contains information that is used to check permissions and translate the global virtual address into a physical address. The GTB is a hardware buffer in the support circuitry that stores frequently or recently accessed global translation array entries.

Microprocessor-generated Addresses

The microprocessor generates an address that it presents to the support circuitry using two sets of hardware pins: the address pins and the store or load mask pins. Figure 5 shows the actual format of the microprocessor-generated address.

Figure 5. Actual Microprocessor-generated Address

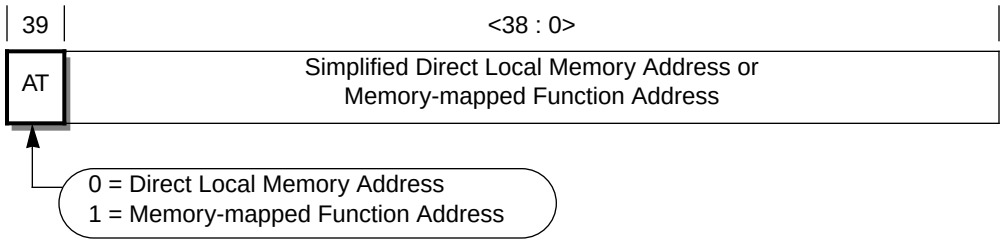


The address pins reference a 128-bit block of data. The microprocessor transfers data to and from the support circuitry over a 128-bit wide data bus. The 36 address pins represent bits <39 : 4> of a byte-oriented address.

The store or load mask pins contain a store mask or a load mask. When the microprocessor stores data, the store mask indicates which 32-bit halfwords transferring over the 128-bit data bus are valid. When the microprocessor loads data, the load mask indicates which 64-bit words in a 256-bit block are requested by the microprocessor (the least significant bit [bit 4] of the address pins is not used for the load request to the support circuitry but is used for the load response from the support circuitry).

To simplify the format of the microprocessor-generated address, it is hereafter represented as a contiguous set of 40 bits that reference a byte of data. There are two types of microprocessor-generated addresses: a direct local memory address or a memory-mapped function address. The type of address is determined by the address type (AT) bit of the address (refer to Figure 6).

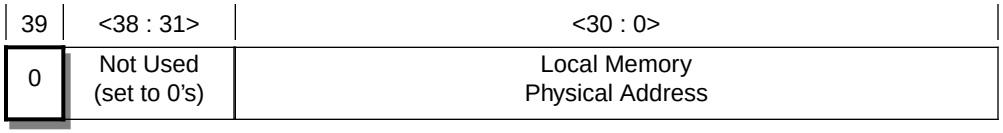
Figure 6. Simplified Microprocessor-generated Address



Direct Local Memory Address

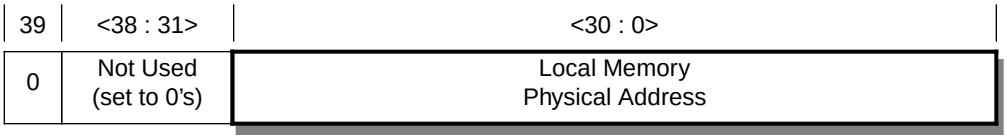
When bit 39 of the microprocessor-generated address is set to 0, the address is a direct local memory address (refer to Figure 7).

Figure 7. Direct Local Memory Address



Direct local memory addresses are used only to reference data in local memory. The support circuitry interprets bits <30 : 0> of a direct local memory address as a byte-oriented address that references a physical location in local memory (refer to Figure 8). The 31 physical address bits provide enough address space to address up to a 2 Gbyte local memory.

Figure 8. Local Memory Physical Address

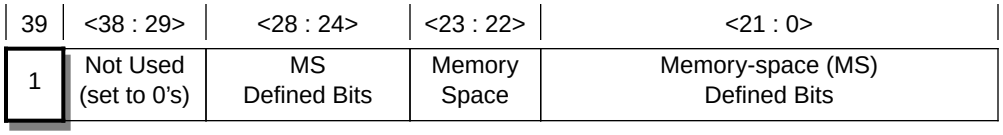


The support circuitry ensures that data stored in the microprocessor secondary cache is consistent with data that is stored in local memory.

Memory-mapped Function Address

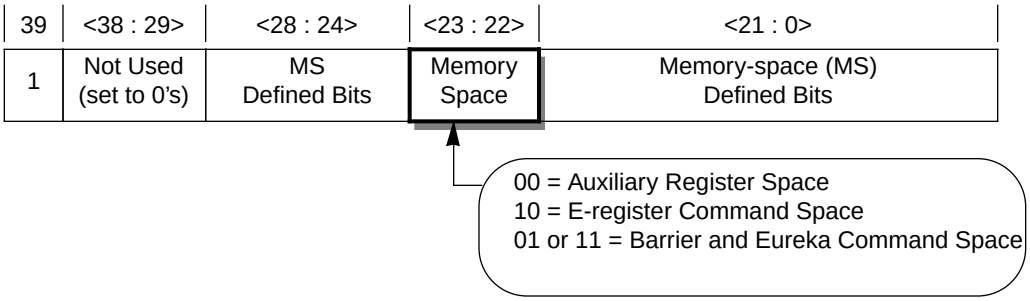
When bit 39 of the microprocessor-generated address is set to 1, the address is a memory-mapped function (MMF) address (refer to Figure 9).

Figure 9. MMF Address



MMF addresses are used to reference data in three memory spaces: auxiliary register space, barrier and eureka command space, and E-register command space. After receiving an MMF address, the support circuitry examines bits <23 : 22> of the address and determines which memory space is referenced (refer to Figure 10).

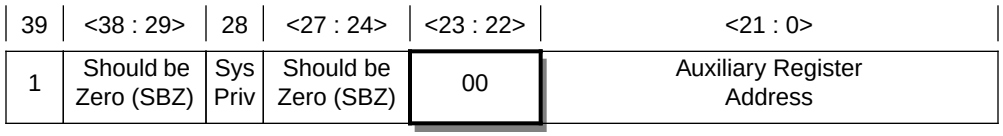
Figure 10. Memory Space Bits



Auxiliary Register Space

When the memory space bits of an MMF address are set to 00, the address references auxiliary register space (refer to Figure 11). Auxiliary register space contains unique addresses for several of the CRAY T3E architecturally defined registers that the microprocessor can read or modify.

Figure 11. Auxiliary Register Space in MMF Address

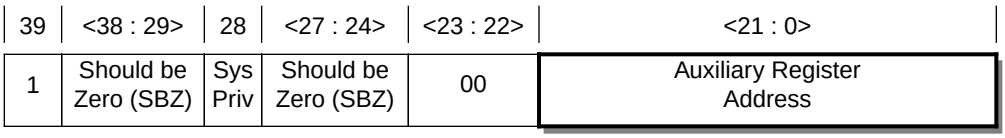


The auxiliary register space address contains two fields: the auxiliary register address field and the system privileged bit.

Auxiliary Register Address

The support circuitry uses the auxiliary register address to reference an auxiliary register in the node hardware (refer to Figure 12).

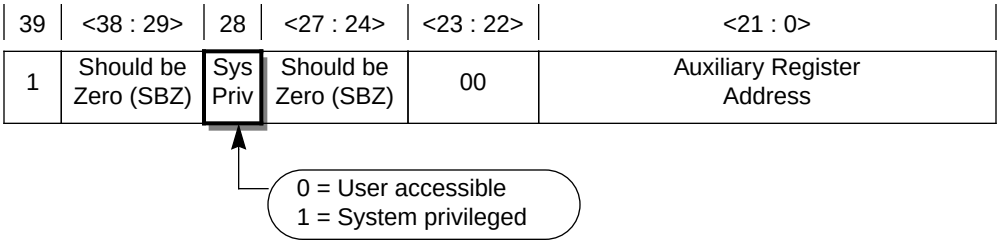
Figure 12. Auxiliary Register Address Field



System-privileged Bit

In the auxiliary register address space, the system-privileged bit divides the address space into two spaces: user accessible space and system-privileged space (refer to Figure 13). In order to address some of the auxiliary registers, the system-privileged bit must be set to a 1. For example, to address the segment translation table entry 1 (STT_ENTRY[1]) register, the system-privileged bit must be set to 1.

Figure 13. System-privileged Bit in Auxiliary Register Space

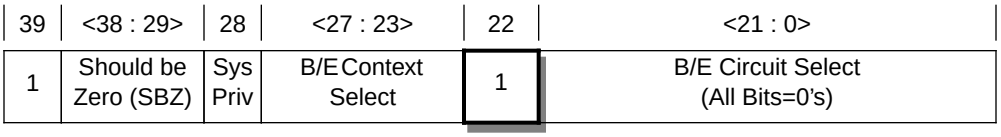


Operating system software may limit the range of user addresses so that user applications can only access auxiliary registers that have an address with the system-privileged bit set to 0. This access limitation prevents users from reading or writing auxiliary registers that should only be read or modified by the operating system. The operating system can access all of the auxiliary registers, regardless of the value of the system-privileged bit..

Barrier and Eureka Command Space

When bit 22 of the memory space bits is set to 1, the address references barrier and eureka (B/E) command space (refer to Figure 14).

Figure 14. B/E Command Space in MMF Address

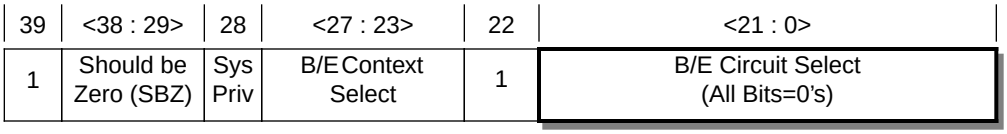


The B/E command space address contains three fields: B/E circuit select field, B/E context select field, and a system-privileged bit.

B/E Circuit Select

The B/E circuit select field (refer to Figure 15) identifies one of the B/E circuits. The support circuitry uses this field to reference a B/E circuit in the node hardware.

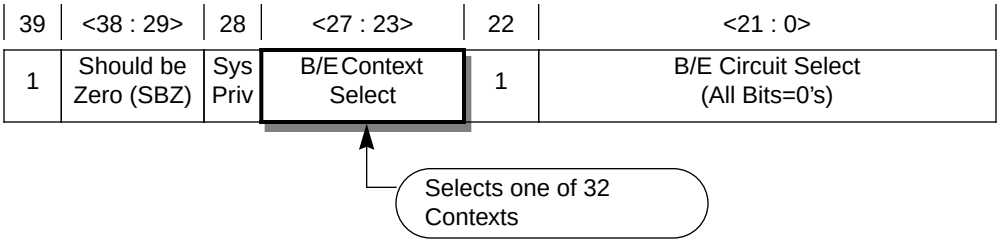
Figure 15. B/E Circuit Select Field



B/E Context Select

The B/E context select field (refer to Figure 16) indicates 1 of 32 possible B/E contexts. Each B/E context contains a set of B/E circuits. Each context may be assigned to a different set of PEs in a partition.

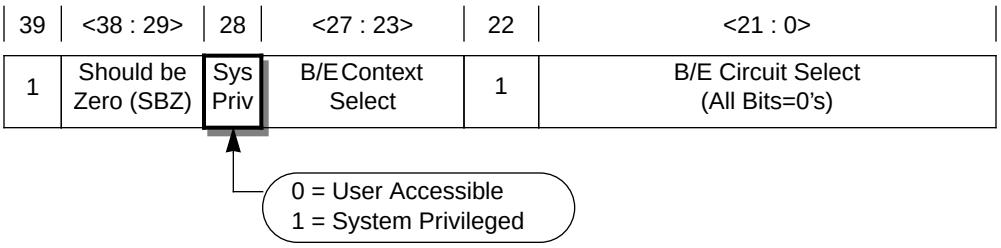
Figure 16. B/E Context Select Bits in B/E Command Space



System-privileged Bit

In B/E command address space, the system-privileged bit (refer to Figure 17) divides the address space into two spaces: user accessible space and system-privileged space. In order to perform some of the B/E commands, the system-privileged bit must be set to a 1.

Figure 17. System-privileged Bit in B/E Command Space

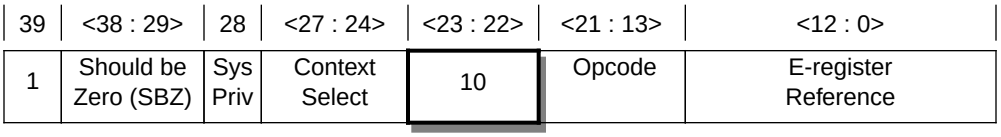


Operating system software may limit the range of user addresses so that user applications can only perform B/E commands that have an address with the system-privileged bit set to 0. This access limitation prevents users from performing the B/E commands that only the operating system should perform. The operating system can perform all of the B/E commands when the operating system sets the system-privileged bit to 1.

E-register Command Space

When the memory space bits of an MMF address are set to 10, the address references E-register command space (refer to Figure 18).

Figure 18. E-register Command Space in MMF Address

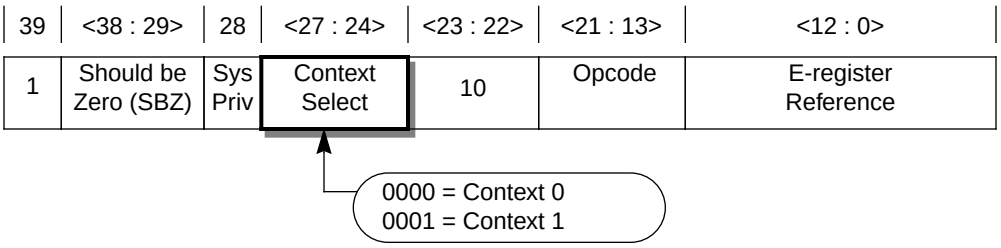


The E-register command space contains four fields: context select field, E-register reference field, a system-privileged bit, and an opcode field.

Context Select

The context select field indicates one of two E-register contexts: E-register context 0 or E-register context 1 (refer to Figure 19). Although the context select field references up to 16 contexts, the CRAY T3E system implements only two E register contexts. The additional contexts are provided for future expansion.

Figure 19. Context Select Field in E-register Command Space



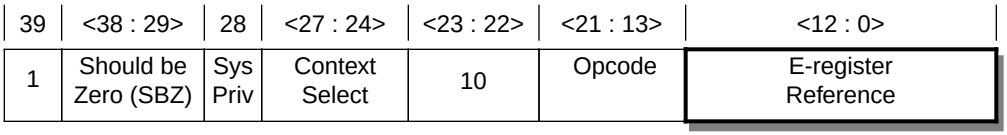
E-register context 0 contains 512 E registers. Generally, software provides access for E-register context 0 address space to user applications and the operating system.

E-register context 1 contains 128 E registers. Generally, software provides access for E-register context 1 address space to the operating system.

E-register Reference

The E-register reference field (refer to Figure 20) identifies one E register or the first E register in a group that is the source or destination of data for an E-register command. Because the microprocessor-generated address is a byte-oriented address, bits <2 : 0> of the E-register reference field must be set to 0 when transferring 64-bit words to or from an E register.

Figure 20. E-register Reference Field



When software transfers a 32-bit halfword to or from an E register, bit 2 of the microprocessor-generated address indicates which halfword in the E register will be transferred. When the microprocessor is in big-endian mode and bit 2 is set to 0, bit 2 indicates that bits <63 : 32> of the E register will be transferred. When the microprocessor is in big-endian mode and bit 2 is set to 1, bit 2 indicates that bits <31 : 0> of the E register will be transferred.

NOTE: Big-endian mode is a software-selectable feature of the microprocessor that indicates whether bit 2 of the microprocessor-generated address is inverted, as described in the previous paragraph, or noninverted.

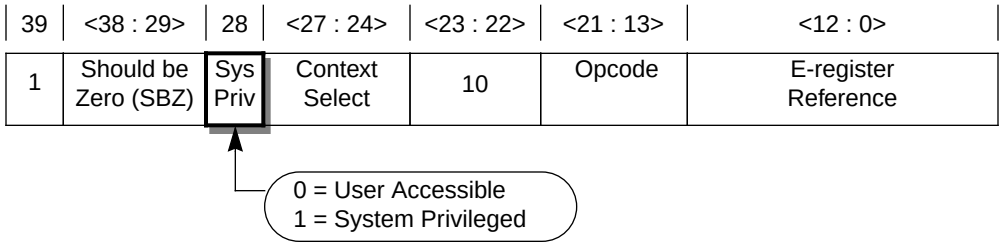
In addition to single 32-bit halfwords or 64-bit words, the E-register reference field may be used to reference a vector. A vector is up to eight 32-bit halfwords or eight 64-bit words. For vectors, bits <9 : 0> of the E-register reference field identify a block of eight E-registers that are the source or destination of data for a vector E-register command. The E-register reference field is right-shifted 3 bits for vector E-register commands to allow merging of the commands in the microprocessor write buffer.

Two of the E-register commands (ERS_READ and ERS_WRITE) reference a block of 32-contiguous E registers. For these E-register commands, bits <7 : 0> of the E-register reference field reference a block of 32-contiguous E registers.

System-privileged Bit

The system-privileged bit divides the E-register commands space into two spaces: user accessible space and system-privileged space (refer to Figure 21). In order to issue some E-register commands, the system-privileged bit must be set to a 1. For example, to issue the special get (SGET) E-register command, bit 28 must be set to 1.

Figure 21. System-privileged Bit in E-register Command Space



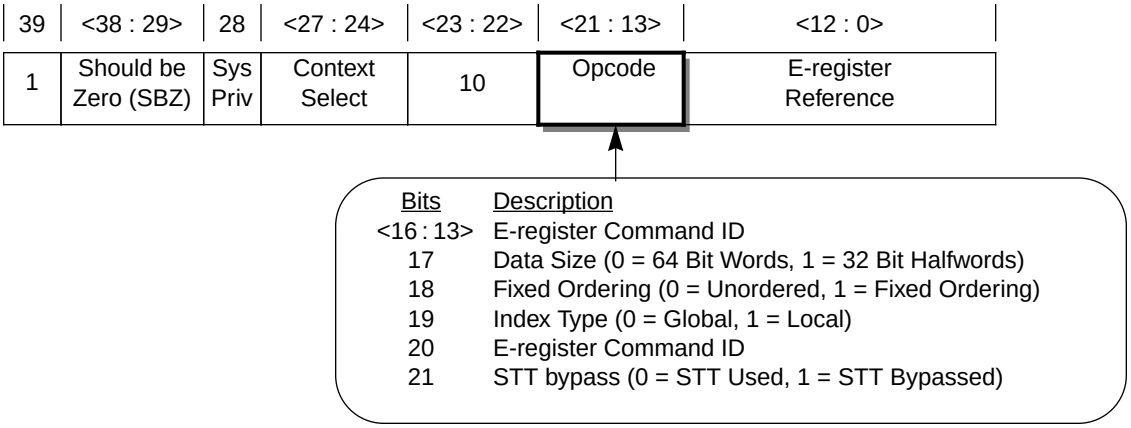
Operating system software may limit the range of user addresses so that user applications can only perform E-register commands that have an address with the system-privileged bit set to 0. This access limitation prevents users from issuing E-register commands that only the operating system should issue. The operating system can issue all of the E-register commands when the operating system sets the system-privileged bit to 1.

Opcode

The opcode field indicates what E-register command is requested (refer to Figure 22). Bits <16 : 13> and 20 identify the E-register command. Bit 17 indicates the size of data that the command will transfer. Bit 18 indicates whether subsequent E-register commands to the same address will be ordered the same as they are received from the microprocessor. Bits 21 and 19 set addressing parameters and are further defined on the following pages.

NOTE: For some memory-mapped commands, bits 21, 19, 18, and 17 are set to specific values and do not represent the format shown in Figure 22.

Figure 22. Opcode

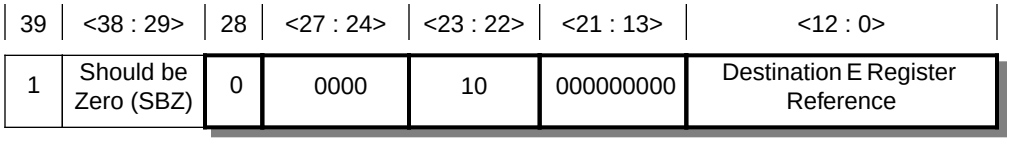


The opcode indicates one of three types of E-register commands: E-register load and store commands, E-register local-memory commands, and E-register global-memory commands.

E-register Load and Store Commands

E-register load and store commands signal the support circuitry to transfer data between registers in the microprocessor and the E registers. For example, an E-register store (STORE) command transfers a 32-bit halfword or a 64-bit word from the microprocessor to a destination E register. Figure 23 shows the format of the microprocessor-generated address for a STORE command.

Figure 23. STORE Address



In this example, bits <28 : 13 > of the microprocessor-generated address are set to specific values. The system-privileged bit (bit 28) is set to 0 to indicate that users can access the E-register command. The context bits (bits <27 : 24>) are set to 0000 to reference E-register context 0. The memory space bits (bits <23 : 22>) are set to 10 to reference E-register command space. The opcode bits (bits <21 : 13>) are set to 000000000 to indicate a STORE command.

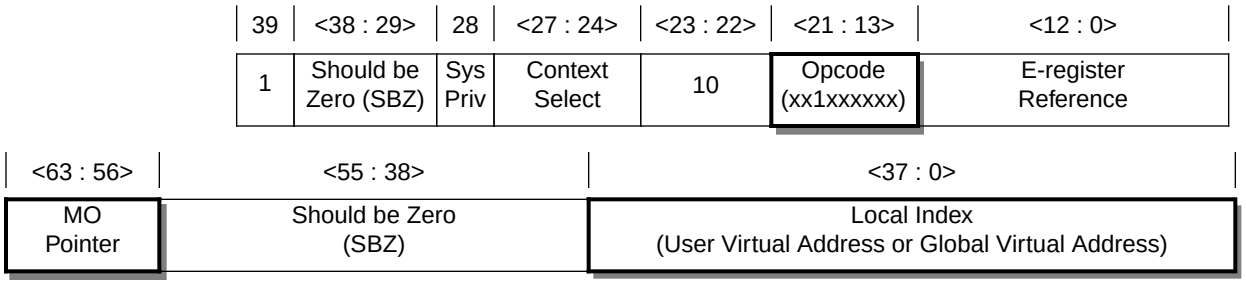
Bits <12 : 0 > contain the destination E-register reference. The support circuitry uses bits <12 : 3> of the destination E register to identify which E register will receive the data. When the E-register command is performed on a 32-bit halfword, the support circuitry uses bit 2 of the destination E register reference to determine which 32-bit halfword in the 64-bit E register will be transferred. The data is transferred from the microprocessor to the support circuitry over the microprocessor data bus. The support circuitry then transfers the data to the destination E register.

E-register Local-memory Commands

When bit 19 of the microprocessor-generated address is set to 1, the E-register command is a local-memory command. Local-memory commands use the E-register access to local memory only method of referencing data (refer again to Figure 4).

When the microprocessor issues an E-register local-memory command, the microprocessor data bus is used to provide addressing parameters to the support circuitry. Figure 24 shows the format of the data bus and the microprocessor-generated address for a local-memory command. The data bus contains two fields: a more operands (MO) pointer field and a local index field.

Figure 24. Microprocessor-generated Address and Data Bus



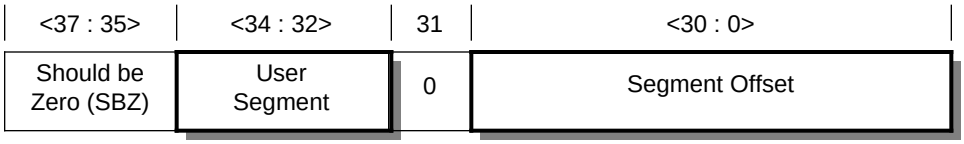
The MO pointer references more E-register command arguments (for example, a swaperand value) that are stored in a group of four E registers. The MO pointer contains bits <12 : 5> of an E register address.

The local index contains a user virtual address or a global virtual address. Bit 21 of the microprocessor-generated address indicates whether the local index is a user virtual address (bit 21 = 0) or a global virtual address (bit 21 = 1).

User Virtual Address

The user virtual address is a byte-oriented address that the program compiler generates and that references different types of data in an application's address space. Figure 25 shows the bit format of the user virtual address. Although the length of the local index is 38 bits, only 34 bits of address are actually used.

Figure 25. User Virtual Address Format



The user virtual address contains two fields: a user segment field and a segment offset field. The user segment field indicates one of eight user segments that the application can access. Each user segment contains up to 2 Gbytes of address space. The segment offset field is a byte-oriented address that indicates which byte in a segment is referenced.

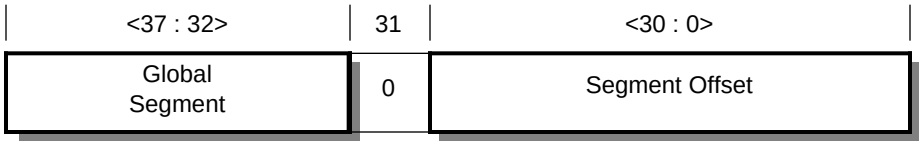
NOTE: Bits 1 and 0 of the user virtual address must be set to 0's because the smallest block of data that can be transferred is a 32-bit halfword. Bit 31 of the user virtual address is not used and must be set to 0.

When the local index contains a user virtual address, the support circuitry converts the user virtual address into a global virtual address.

Global Virtual Address

The global virtual address is an intermediate address used by the operating system to reference different types of data in the CRAY T3E memory space. The global virtual address contains two fields: a global segment field and a segment offset field (refer to Figure 26).

Figure 26. Global Virtual Address Format



The global segment field indicates one of 64 global segments. Each global segment has up to 2 Gbytes of address space. Each user segment maps to one of the global segments. More information on the user segment to global segment mapping process is provided on the following pages.

Software sets the actual size of the global segments by writing a value to bits <5 : 4> of the global translation buffer control (GTB_CTL) register. Generally, software sets the size of the global segments equal to or greater than the size of local memory in each PE. Table 2 lists the possible sizes of the global segments and shows the value of bits <5 : 4> of the GTB_CTL register for each size.

Table 2. . Global Segments

GTB_CTL Bits <5 : 4>	Segment Size	Segment Offset Bits Used	Global Virtual Address Bits That are Not Used
00	256 Mbytes	27 through 0	31 through 28
01	512 Mbytes	28 through 0	31 through 29
10	1 Gbyte	29 through 0	31 through 30
11	2 Gbytes	30 through 0	31

Like the segment offset field in the user virtual address, the segment offset field in the global virtual address is a byte-oriented address that indicates which byte in a segment is referenced.

NOTE: Bits 1 and 0 of the global virtual address must be set to 0's because the smallest block of data that can be transferred is a 32-bit halfword. Also, bit 31 of the global virtual address is not used and must be set to 0.

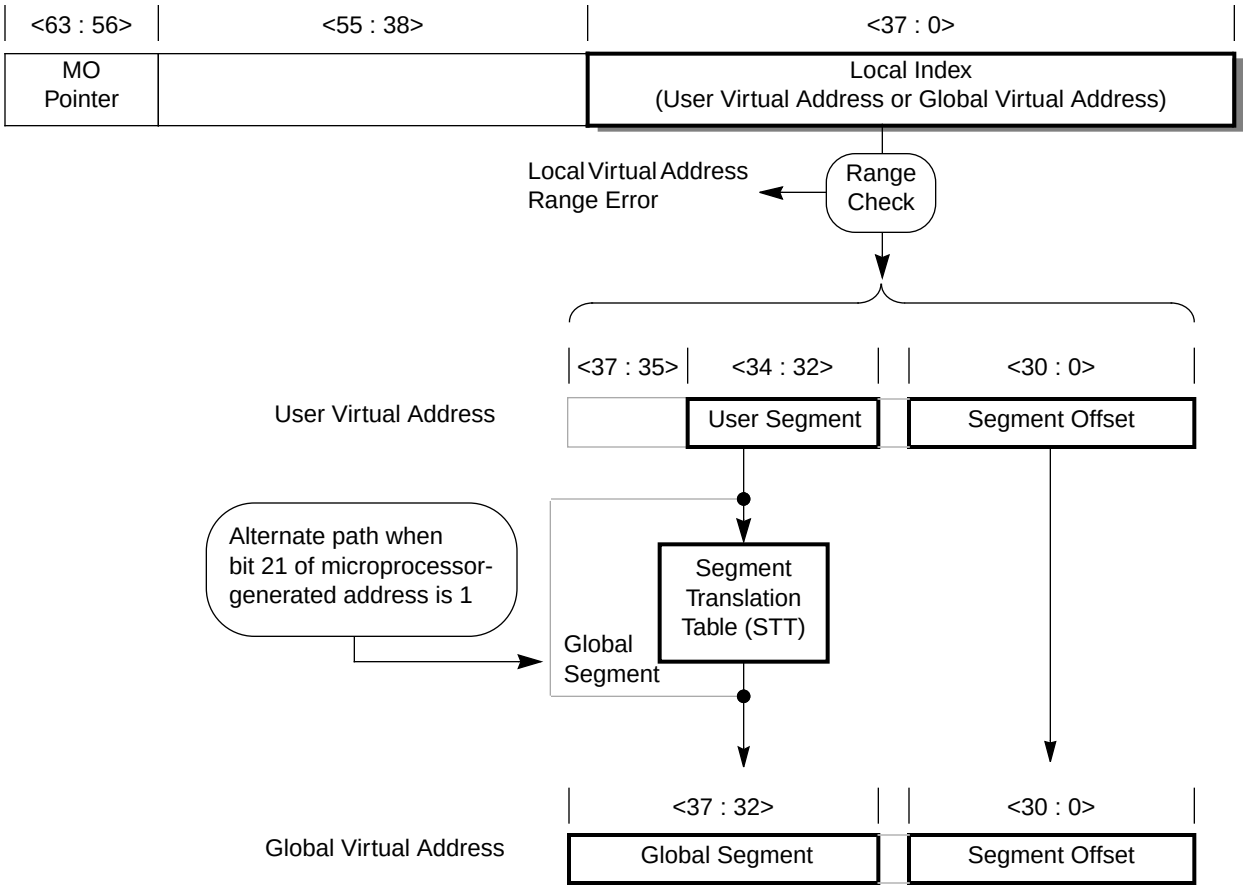
The support circuitry converts a user virtual address into a global virtual address using a segment translation table (STT). The STT is a look-up table in the support circuitry that contains eight lines (one for each user segment). Software references each line in the STT by storing to or loading from the STT entry (STT_ENTRY[7 : 0]) hardware registers. Figure 27 shows the format of the STT; however, only bits <29 : 24> of each line are used for E-register local-memory commands.

Figure 27. STT Format

User Segment	<33 : 30>	<29 : 24>	<23 : 12>	<11 : 0>
0	Protection Bits	Global Segment Number	Virtual PE Limit Number	Base Logical PE Number
1	Protection Bits	Global Segment Number	Virtual PE Limit Number	Base Logical PE Number
2	Protection Bits	Global Segment Number	Virtual PE Limit Number	Base Logical PE Number
3	Protection Bits	Global Segment Number	Virtual PE Limit Number	Base Logical PE Number
4	Protection Bits	Global Segment Number	Virtual PE Limit Number	Base Logical PE Number
5	Protection Bits	Global Segment Number	Virtual PE Limit Number	Base Logical PE Number
6	Protection Bits	Global Segment Number	Virtual PE Limit Number	Base Logical PE Number
7	Protection Bits	Global Segment Number	Virtual PE Limit Number	Base Logical PE Number

Before using the local index, the support circuitry checks the length of the local index. When the local index is a user virtual address and it has more than 35 significant bits or when the local index is a global virtual address and it has more than 38 significant bits, the support circuitry indicates that a local virtual address range error occurred. When the local index is a user virtual address and it has 35 or fewer significant bits or when the local index is a global virtual address and it has 38 or fewer significant bits, the support circuitry reads the global segment number from the line in the STT that is referenced by the user segment (refer to Figure 38).

Figure 28. Local Index to Global Virtual Address



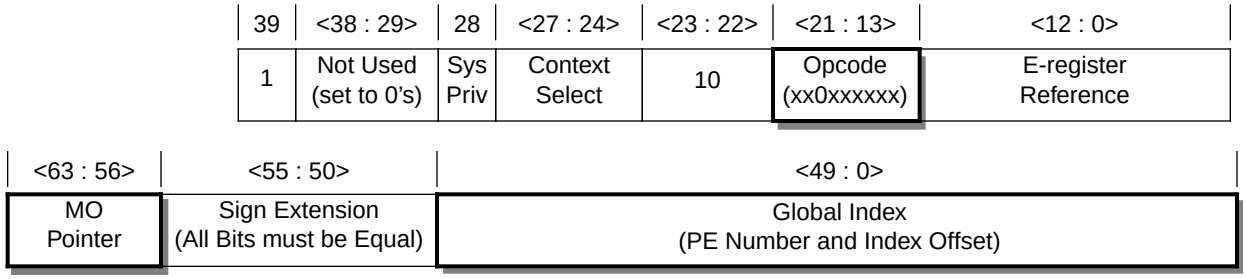
The support circuitry translates the global virtual address into a physical address. More information on this translation is provided in this section.

E-register Global-memory Commands

When bit 19 of the microprocessor-generated address is set to 0, the E-register command is a global-memory command. Global-memory commands use the E-register access to global memory method of accessing data (refer again to Figure 3).

When the microprocessor issues an E-register global-memory command, the microprocessor data bus is used to provide additional addressing parameters to the support circuitry. Figure 29 shows the format of the data bus and the microprocessor-generated address for a global-memory command. The data bus contains two fields: a more operands (MO) pointer field and global index field.

Figure 29. Microprocessor-generated Address and Data Bus



The MO field is used to reference more E-register command and addressing arguments (for example, the centrifuge mask) that are stored in a group of four E registers. The MO pointer contains bits <12 : 5> of an E register reference. More information on these addressing arguments is provided in this subsection.

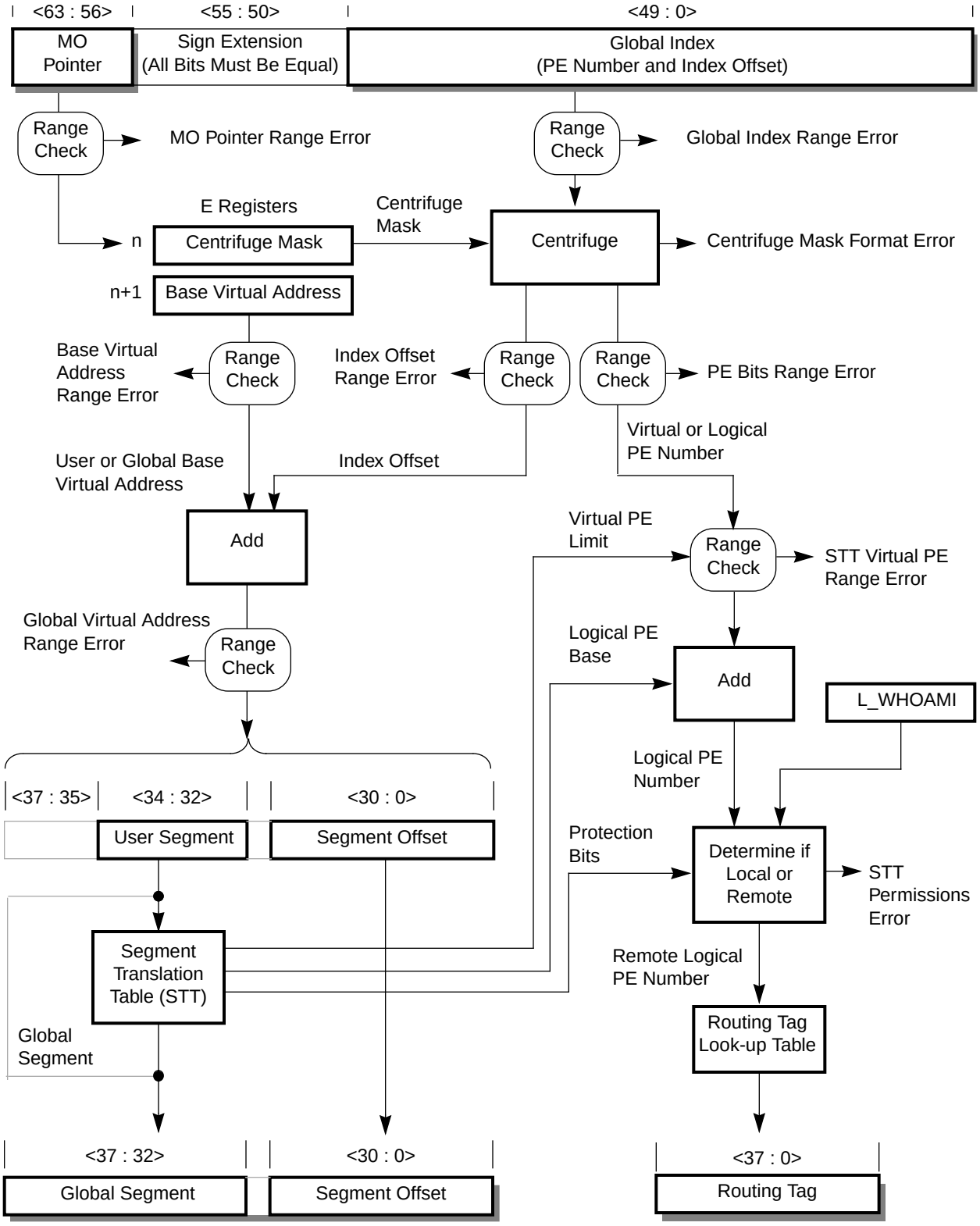
The global index field contains the global index. The global index is a PE number (virtual PE number or a logical PE number) and an index offset in a software-defined format.

NOTE: Bits 1 and 0 of the global index must be set to 0's because the smallest block of data that can be transferred is a 32-bit halfword.

The support circuitry translates the global index into a global virtual address and a routing tag (refer to Figure 30). The global virtual address and routing tag are used in a request packet that is sent to the destination PE. The following subsections describe the process of translating the global index into a global virtual address and a routing tag.

The following subsections describe, in detail, the global address translation process shown in Figure 30. You may want to refer to Figure 30 while reading the following subsections.

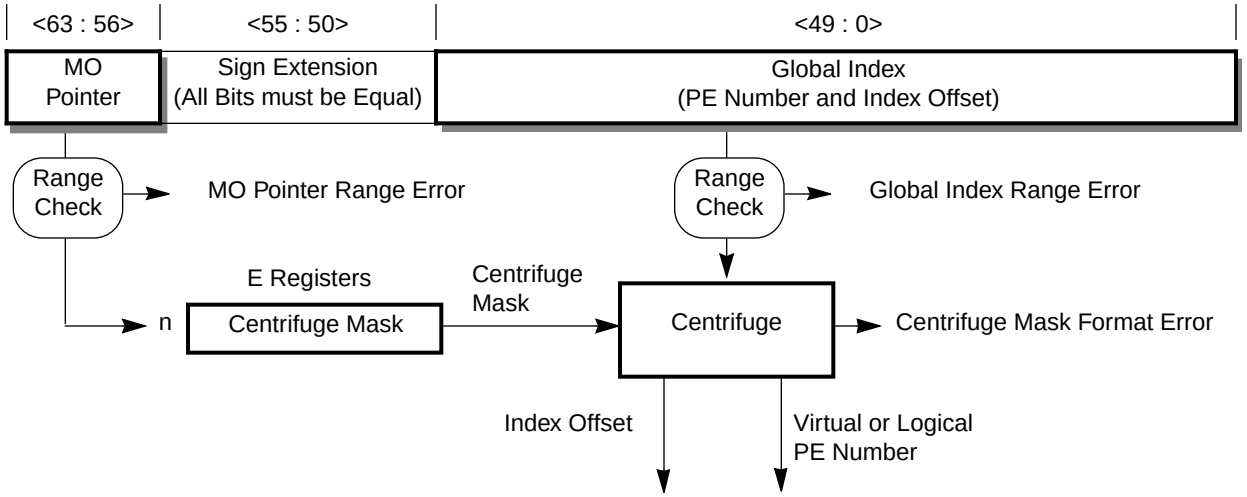
Figure 30. Global Address Translation



Centrifuge

The centrifuge separates the software-defined global index into a PE number and index offset (refer to Figure 31).

Figure 31. Centrifuge



Before using the global index, the support circuitry checks the value of the global index. When the global index is larger than the maximum value allowed for the system configuration, the support circuitry indicates that a global index range error occurred. When the global index is less than or equal to the maximum value for the system configuration, the support circuitry sends the global index to the centrifuge.

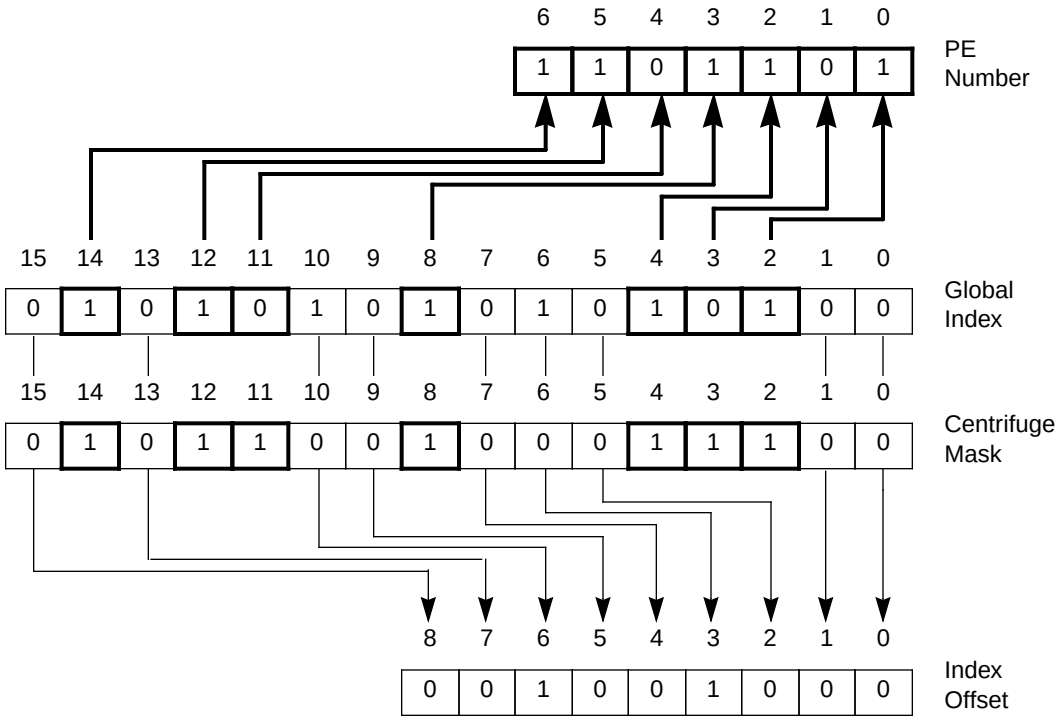
The centrifuge uses a centrifuge mask to determine which bits of the global index are PE number bits and which bits of the global index are index offset bits. Before requesting a global-memory command, software loads an E register with the centrifuge mask. When requesting the global-memory command, software sets bits <63 : 56> of the microprocessor data bus (the MO pointer) equal to bits <12 : 5 > of the E register reference for the E register that contains the centrifuge mask.

Before using the MO pointer, the support circuitry checks the value of the MO pointer. When the value of the MO pointer references a number larger than the maximum number of E registers in the E-register context, the support circuitry indicates that an MO pointer range error occurred. When the value of the MO pointer references a number less than the maximum number of E registers in the E-register context, the support circuitry uses the MO pointer to reference the E register that contains the centrifuge mask and then sends the centrifuge mask to the centrifuge.

The centrifuge mask is a 50-bit value that is stored in bits <49 : 0> of an E register. When a bit of the centrifuge mask is set to 1, the corresponding bit of the global index is part of the PE number. When a bit of the centrifuge mask is set to 0, the corresponding bit of the global index is part of the index offset. Bits 1 and 0 of the centrifuge mask must be set to 0's. (A centrifuge mask of all 0's is valid and results in a PE number equal to 0.)

Figure 32 shows the separation of a sample global index into a PE number and index offset. For clarity, this example shows the process for only the 16 least-significant bits of the global index and centrifuge mask.

Figure 32. Sample Centrifuge Operation



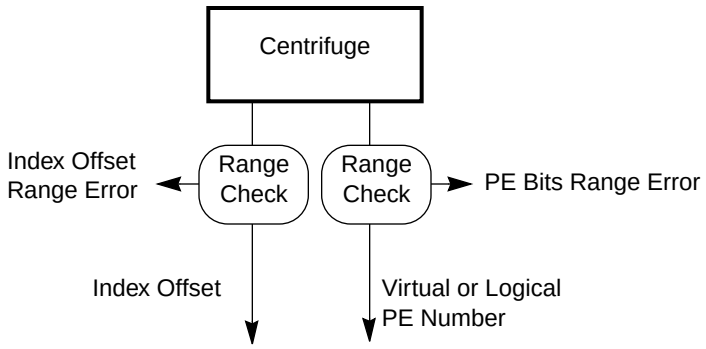
Before using the centrifuge mask, the support circuitry checks the value of the centrifuge mask. When the centrifuge mask contains more than 50 significant bits or when it has more than 20 bits set to 1, the support circuitry indicates that a centrifuge mask format error occurred. When the centrifuge mask contains 50 or fewer significant bits or if it contains 20 or fewer bits set to 1, the support circuitry performs the centrifuge operation.

NOTE: The centrifuge mask format check is for 20 bits that are set to 1 because the STORE_CENT E-register command may be performed without using the global index as an actual data address.

The PE number is a logical PE number or a virtual PE number. When the STT bypass bit (bit 21) of the microprocessor-generated address is set to 1, the support circuitry interprets the PE number as a logical PE number. When the STT bypass bit of the microprocessor-generated address is set to 0, the support circuitry interprets the PE number as a virtual PE number.

After generating the PE number, the support circuitry checks the value of the PE number. When the PE number is larger than 12 bits, the support circuitry indicates that a PE bits range error occurred (refer to Figure 33). When the PE number is less than or equal to 12 bits, the support circuitry continues with the address translation process.

Figure 33. PE Bits Range Error and Index Offset Range Error



The index offset is a byte-oriented address with respect to a base user or global virtual address. The support circuitry adds the index offset to the base user or global virtual address to create the actual user or global virtual address for the E-register command.

The index offset is either a positive or a negative value. Software must use the two's compliment for negative index offsets. Bits <55 : 50> of the global index are the sign extension bits for the index offset (refer again to Figure 31).

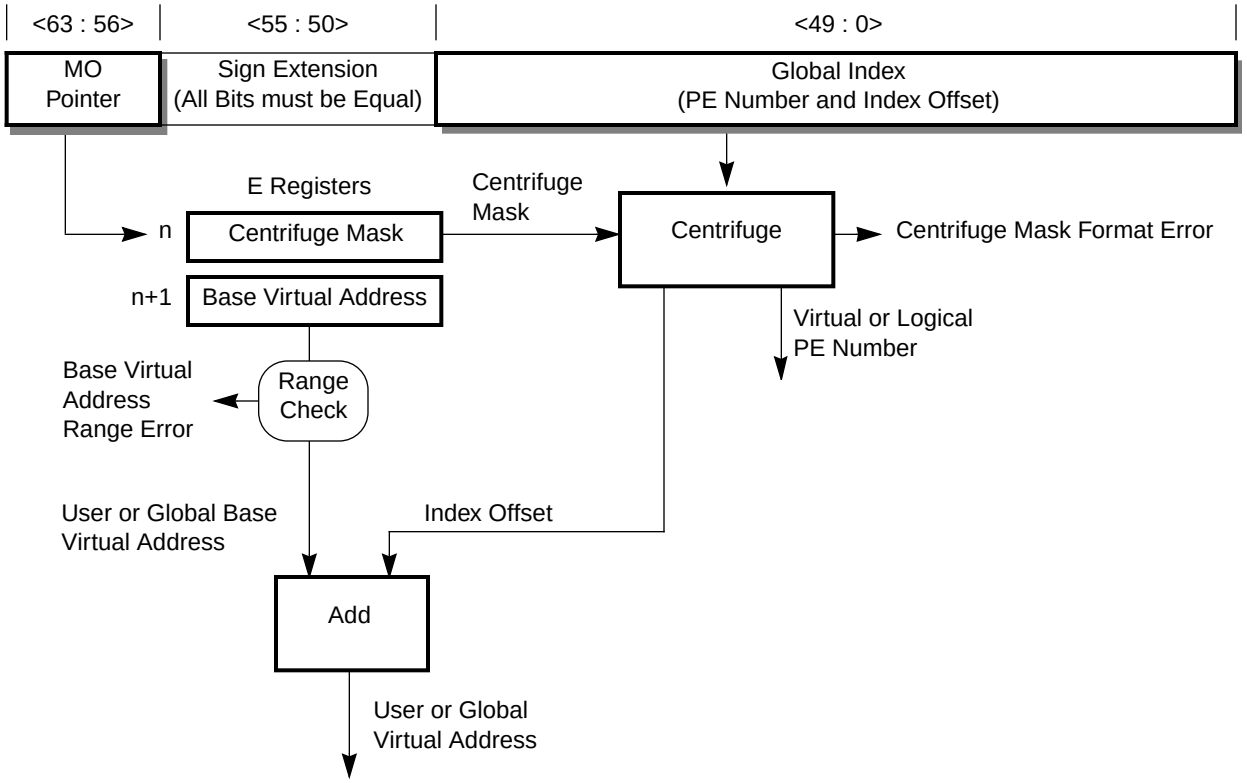
After generating the index offset, the support circuitry checks the value of the index offset. When the index offset contains more than 38 significant bits, the support circuitry indicates that an index offset range error occurred (refer again to Figure 33). When the index offset contains 38 or less significant bits, the support circuitry continues with the address translation process.

Global Virtual Address Generation

When performing an E-register global-memory command, the support circuitry performs one or two steps to convert the index offset into a global virtual address. When performing one step, the support circuitry adds the index offset to a base virtual address. When performing two steps, the support circuitry also converts a user virtual address into a global virtual address.

Before requesting an E-register global-memory command, software stores a base virtual address into the next sequential E register with respect to the E register that contains the centrifuge mask (refer to Figure 34). The base virtual address is a user virtual address or a global virtual address.

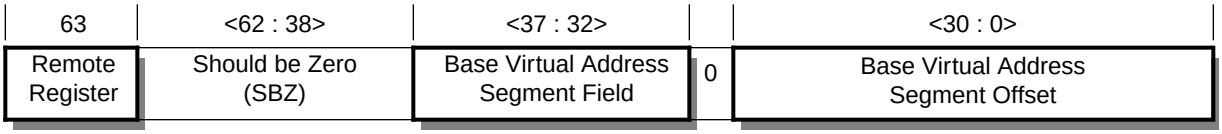
Figure 34. Adding the Index Offset to a Base Virtual Address



When performing the global-memory command, the support circuitry references the base virtual address E register using the MO pointer and obtains the base virtual address. When the STT bypass bit (bit 21) of the microprocessor-generated address is set to 0, the support circuitry interprets the base virtual address as a user virtual address. When the STT bypass bit is set to 1, the support circuitry interprets the base virtual address as a global virtual address.

Figure 35 shows the format of the base virtual address in an E register. When bit 63 of the E register is set to 1, the base virtual address is actually a hardware register address that references a register that is not located in the support circuitry (for example, the I/O registers).

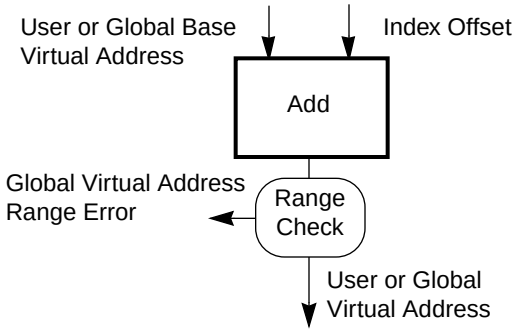
Figure 35. Base Virtual Address Format in an E Register



The support circuitry then checks the value of the base virtual address. When the base virtual address is a user virtual address and it has more than 35 significant bits or when the base virtual address is a global virtual address and it has more than 38 significant bits, the support circuitry indicates that a base virtual address range error occurred. When the base virtual address is a user virtual address and it has 35 or fewer significant bits or when the base virtual address is a global virtual address and it has 38 or fewer significant bits, the support circuitry adds the base virtual address and the index offset together.

Next, the support circuitry checks the value of the sum of the base virtual address and the index offset. When the sum is a user virtual address and it has more than 35 significant bits or when the sum is a global virtual address and it has more than 38 significant bits, the support circuitry indicates that a global virtual address range error occurred (refer to Figure 36). When the sum is a user virtual address and it has 35 or fewer significant bits or when the sum is a global virtual address and it has 38 or fewer significant bits, the support circuitry continues with the address translation.

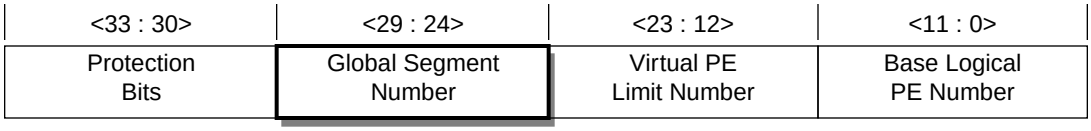
Figure 36. Global Virtual Address Range Check



When the sum of the base virtual address and the index offset is a user virtual address, the support circuitry converts the user virtual address into a global virtual address by using the segment translation table (STT). When the sum of the base virtual address and the index offset is a global virtual address, no conversion is needed and the STT is not used.

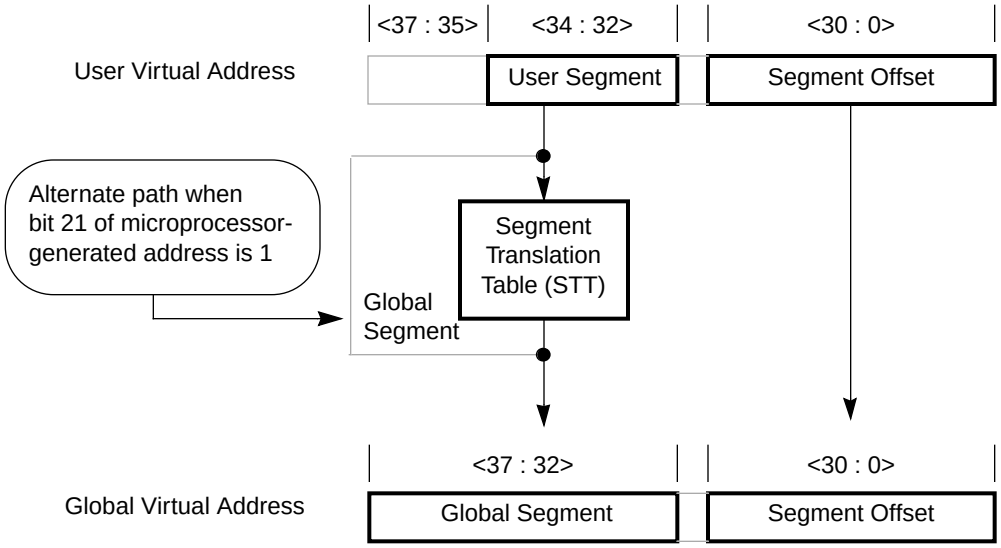
The STT is a look-up table in the support circuitry that contains eight lines (one for each user segment). Software references each line in the STT by storing to or loading from the STT entry (STT_ENTRY[7 : 0]) hardware registers. Figure 37 shows the format of a line in the STT.

Figure 37. Global Segment Number in an STT Line



After generating the user virtual address, the support circuitry uses the user segment number to reference one of the eight lines in the STT. The support circuitry then reads the global segment number from that line and uses it as the global segment of the global virtual address (refer to Figure 38).

Figure 38. User Virtual Address to Global Virtual Address Translation



The support circuitry translates the global virtual address into a physical address. More information on this translation process is provided later in this document.

Virtual PE Range Check

The support circuitry performs a virtual PE range check when the STT bypass bit (bit 21) of the microprocessor-generated address is set to 0. When the STT bypass bit is set to 1, the support circuitry does not perform a virtual PE range check and it continues with the address translation.

To perform a virtual PE range check, the support circuitry compares the virtual PE number from the centrifuge to the virtual PE limit number read from the STT (refer to Figure 39 and Figure 40). When the virtual PE number is greater than the virtual PE limit number, the support circuitry indicates that an STT virtual PE range error occurred. When the virtual PE number is less than or equal to the virtual PE limit, the support circuitry continues with the address translation.

Figure 39. Virtual PE Range Check

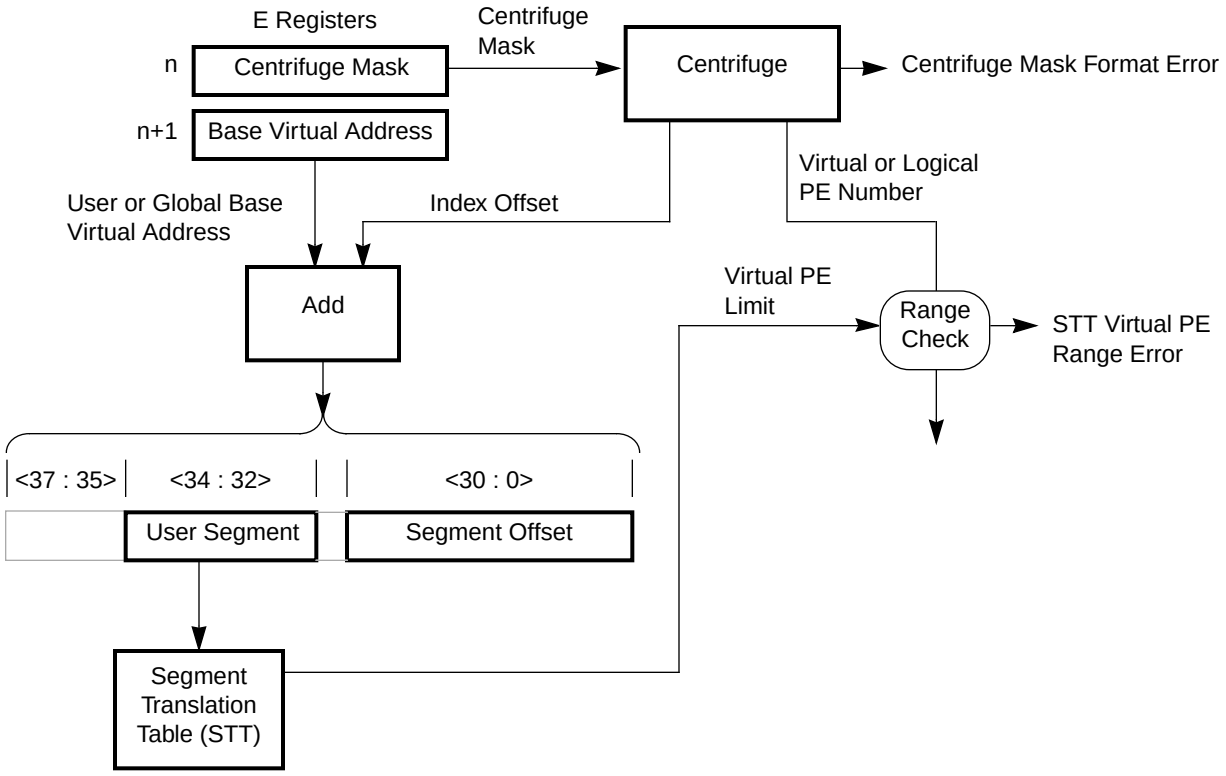
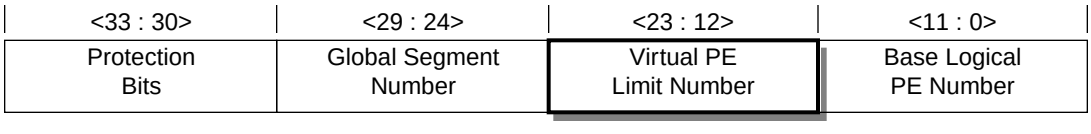


Figure 40. Virtual PE Limit Number in STT Line



Virtual to Logical PE Conversion

The support circuitry converts the virtual PE number received from the centrifuge into a logical PE number when the STT bypass bit (bit 21) of the microprocessor-generated address is set to 0. When the STT bypass bit is set to 1, the support circuitry interprets the PE number from the centrifuge as a logical PE number and a conversion is not performed.

To convert the virtual PE number into a logical PE number, the support circuitry adds the virtual PE number to a base logical PE number read from the STT line (refer to Figure 41 and Figure 42). The base logical PE number is the smallest numbered logical PE in a partition.

Figure 41. Virtual to Logical PE Conversion

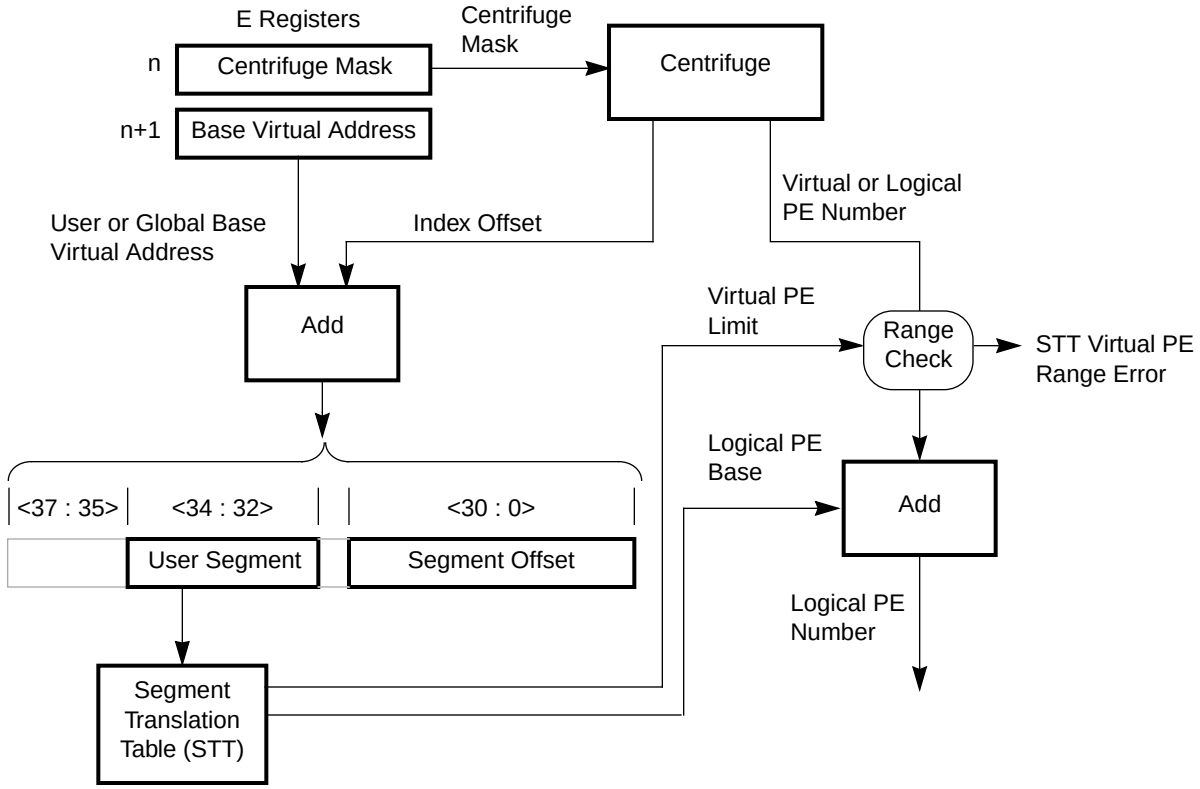
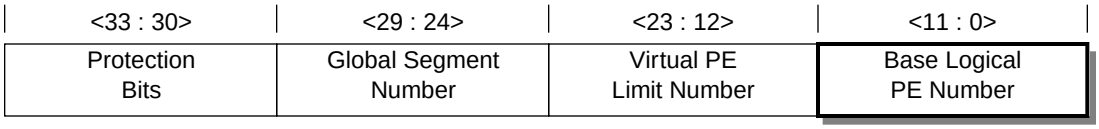


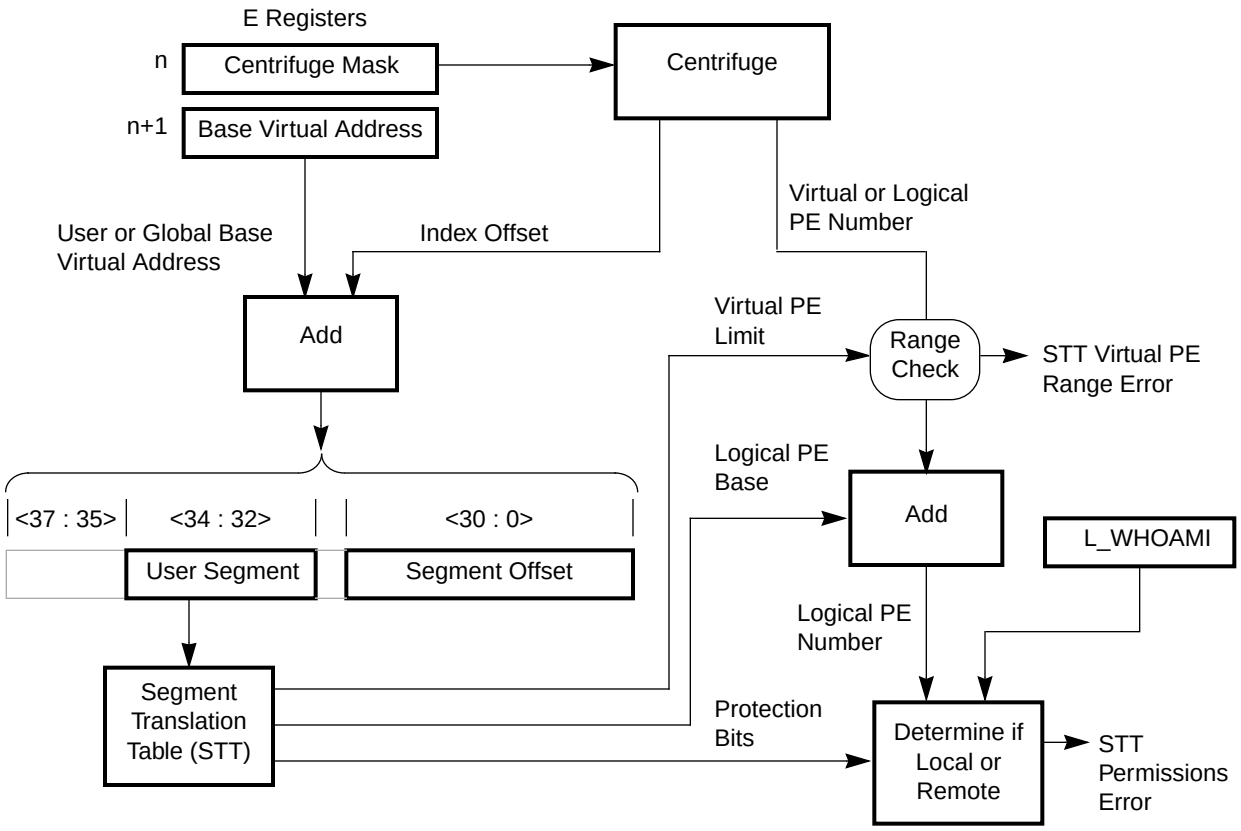
Figure 42. Base Logical PE Number in STT Line



Local or Remote PE Determination

After generating the logical PE number, the support circuitry determines whether the logical PE number references the local PE or a remote PE. To do this, the support circuitry compares the generated logical PE number to the value stored in the logical PE number (L_WHOAMI) register (refer to Figure 43).

Figure 43. Local or Remote PE Determination

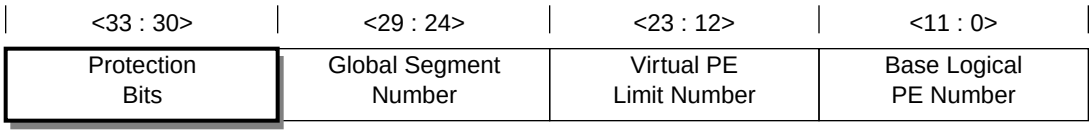


When the logical PE number matches the value stored in the L_WHOAMI register, the PE referenced is the local PE. In this case, the support circuitry does not convert the logical PE number into a routing tag for a request packet. Instead, the support circuitry converts the global virtual address into a physical address for local memory. More information on this translation process is provided later in this document.

When the logical PE number does not match the value stored in the L_WHOAMI register, the PE referenced is a remote PE. In this case, the support circuitry continues to translate the logical PE number into a routing tag. More information on this conversion process is provided on the following pages.

After determining whether the logical PE number is local or remote, the support circuitry checks the permissions for the global segment referenced. To do this, the support circuitry reads the protection bits from the STT line (refer to Figure 44).

Figure 44. Protection Bits in the STT Line



The protection bits indicate the read and write permission for a global segment. Table 3 list the protection bits and describes their corresponding permissions.

Table 3. Protection Bits Values

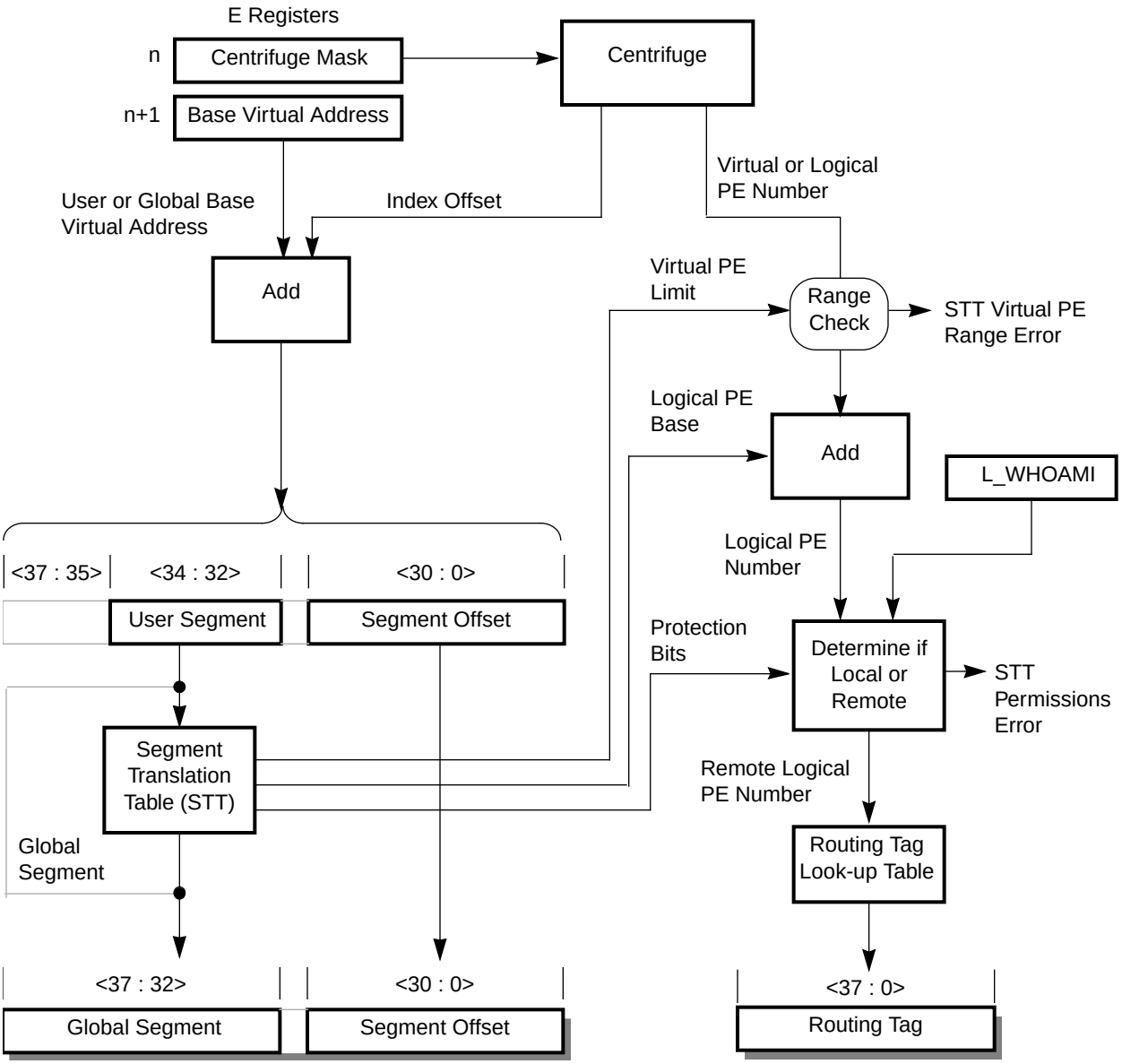
STT Line Bit	Description
30	When set to 1, this bit enables local reads. When set to 0, this bit disables local reads.
31	When set to 1, this bit enables local writes. When set to 0, this bit disables local writes.
32	When set to 1, this bit enables remote reads. When set to 0, this bit disables remote reads.
33	When set to 1, this bit enables remote writes. When set to 0, this bit disables remote writes.

When the global-memory command is a request to read or write data in a global segment and the permissions bits are set so that read or writes are disabled, the support circuitry indicates that an STT permissions error occurred. When the global-memory command is a request to read or write data in a global segment and the permissions bits are set so that reads or writes are enabled, the support circuitry either converts the global virtual address into a physical address (for a local PE number) or continues to convert the logical PE number into a routing tag (for a remote PE number).

Routing Tag Creation

The support circuitry converts the logical PE number into a routing tag when the logical PE number is a remote PE number. To do the conversion, the support circuitry uses a routing tag look-up table (refer to Figure 45).

Figure 45. Routing Tag Creation



The routing tag look-up table is a set of hardware registers called the network look-up table (R_NET_LUT) registers. There are 544 R_NET_LUT registers, which provide enough registers to assign an R_NET_LUT register to each logical PE for systems that contain up to 544 PEs. Each register contains a routing tag that routes a packet from the local PE to a remote PE in the system.

The address for each R_NET_LUT register contains logical PE information that indicates which logical PE the R_NET_LUT register is assigned to. Using the generated logical PE number, the support circuitry retrieves the corresponding routing tag for the destination logical PE from the appropriate R_NET_LUT register.

After translating the global index into a routing tag and global virtual address, the support circuitry generates a request packet that contains the routing tag and global virtual address and sends the packet to the destination PE. After receiving the request packet, the destination PE converts the global virtual address into a physical address that references data in the local memory of the destination PE.

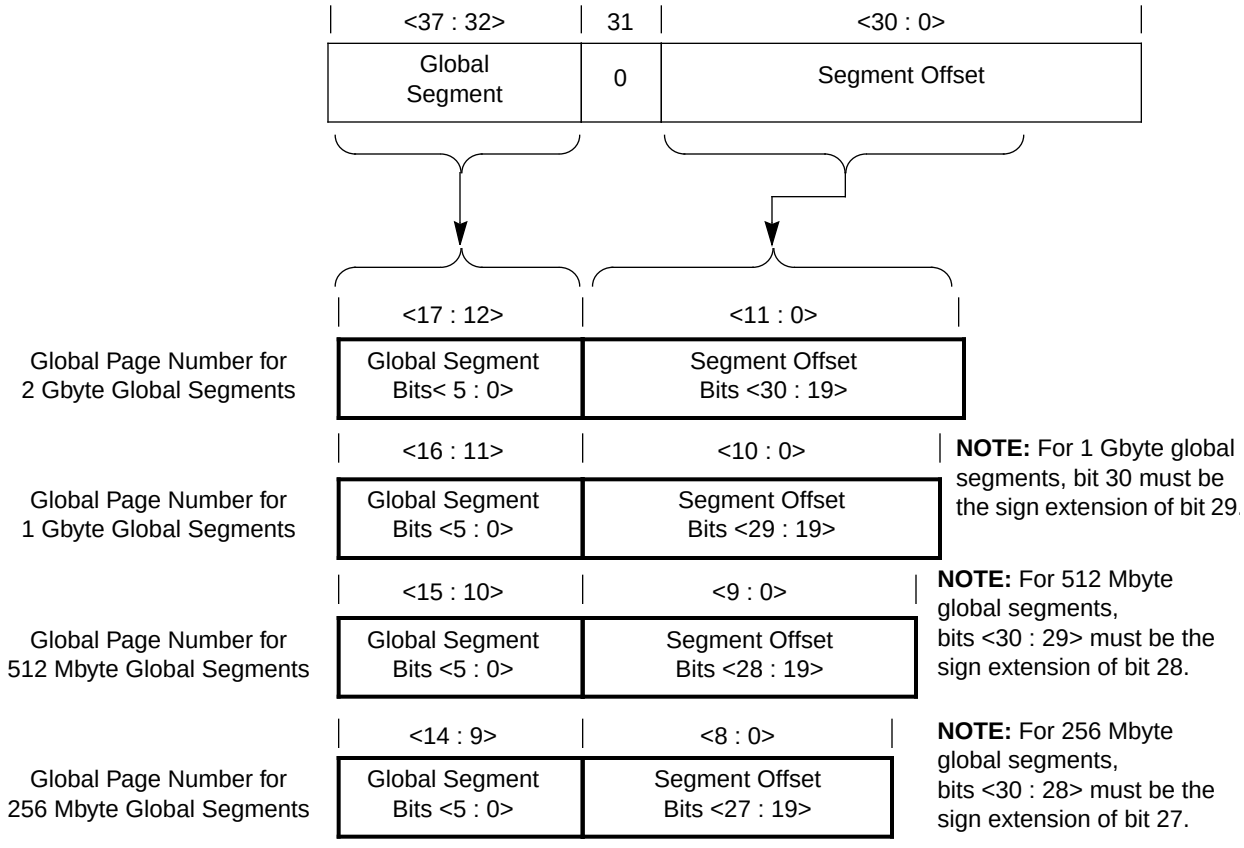
Physical Address

The support circuitry in the destination PE translates the global virtual address into a physical address that references data in local memory. During the translation process, the support circuitry uses two types of address information: the global page number and the physical page address.

Global Page Number

The global page number is a subset of the global virtual address. The global page number references each 512 Kbyte page in the global virtual address space. The format of the global page number depends on the software-selected size of the global segment (refer to Figure 46).

Figure 46. Global Page Numbers



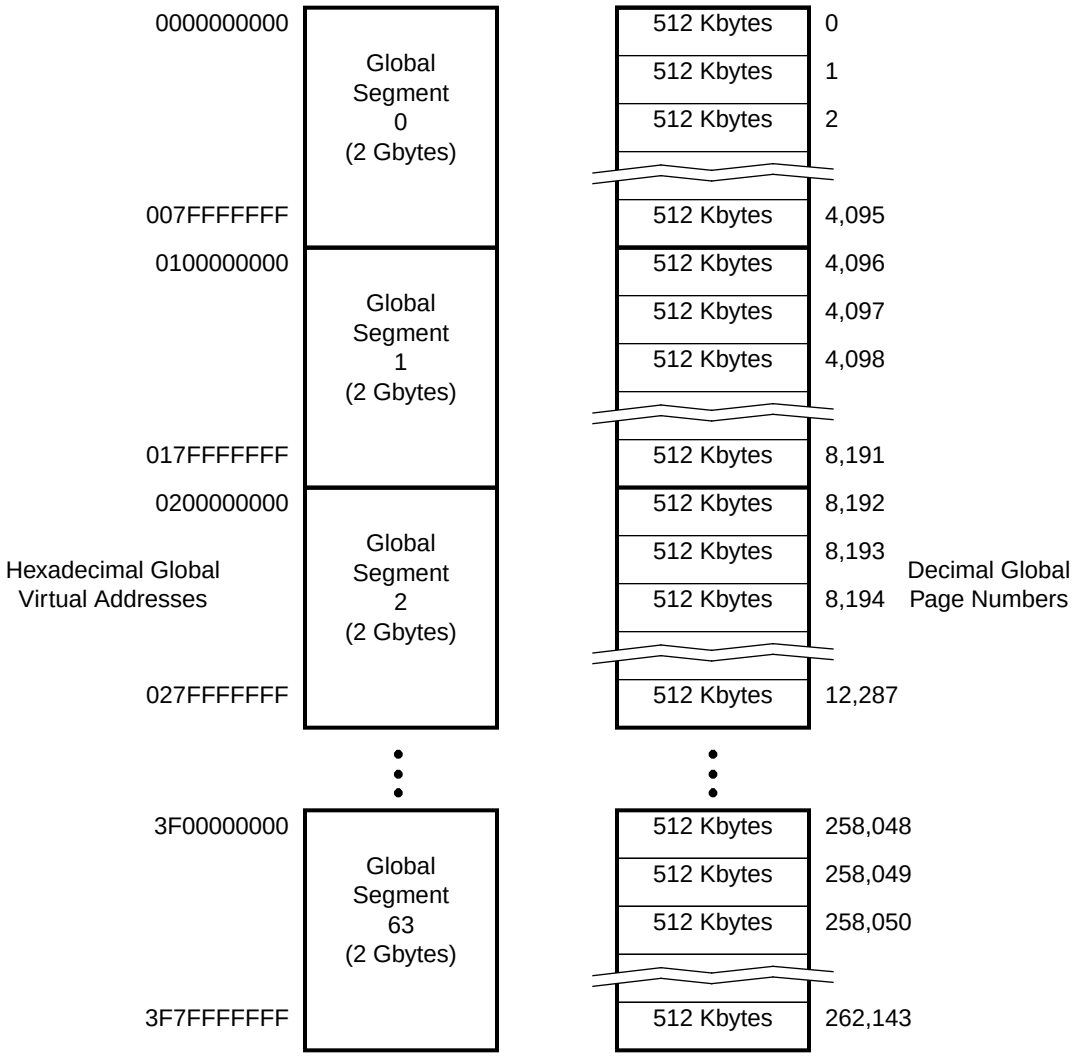
The size of the global segment determines the total number of global pages in the global virtual address. Table 4 lists the segment sizes and shows the total number of global pages for each segment size.

Table 4. Total Global Pages

GTB_CTL Bits <5 : 4>	Global Segment Size	Global Pages per Segment	Total Number of Global Pages
00	256 Mbytes	512	32,768
01	512 Mbytes	1,024	65,536
10	1 Gbyte	2,048	131,072
11	2 Gbytes	4,096	262,144

Figure 47 shows an example of the translation between the global virtual address and global pages for 2 Gbyte segments.

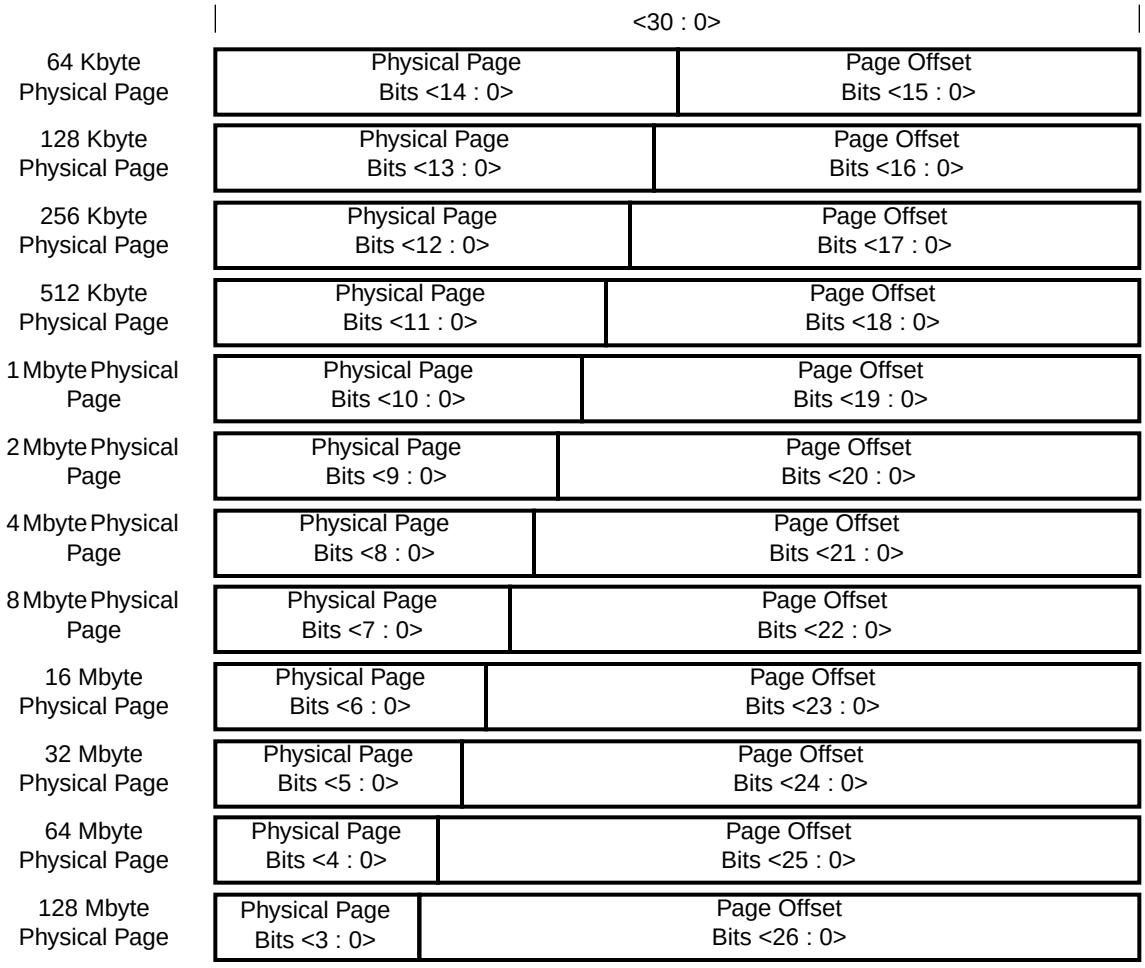
Figure 47. Global Segments and Global Pages



Physical Address Generation

The physical address is a byte-oriented address that references a byte of data in the local memory of a PE. The physical address contains 31 bits that are organized in two fields: physical page field and page offset field. The format of the physical address depends on the physical page size that is referenced (refer to Figure 48).

Figure 48. Physical Addresses

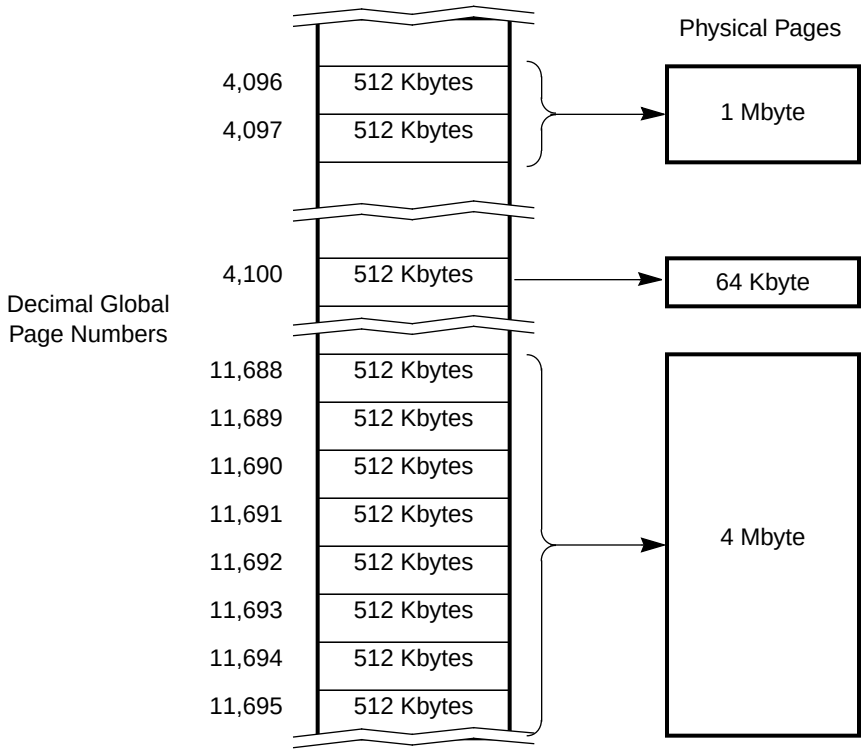


Physical Page

The physical page field uniquely identifies a physical page in the local memory address space. Software assigns each 512 Kbyte global page in the global virtual address space to a physical page. Each physical page ranges in size from 64 Kbytes to 128 Mbytes. Although global pages are aligned on 512 Kbyte boundaries in the global virtual address space, physical pages do not have to be aligned on 512 Kbyte boundaries in the physical memory.

Figure 49 shows sample global page to physical page translations. Physical pages that are smaller than 512 Kbytes are called small pages. Each global page can be assigned to only one small page. Physical pages that are larger than 512 Kbytes are called large pages. Global pages that are assigned to a large physical page must be contiguous global page numbers in the global virtual address.

Figure 49. Sample Global Page to Physical Page Translations



The support circuitry uses a software-defined look-up table and a buffer in the support circuitry when translating a global page number into a physical page number. The look-up table is called the global translation array and the buffer is called the global translation buffer.

Global Translation Array

The global translation array is a software-defined look-up table that is stored in local memory. The global translation array contains an entry for each 512 Kbyte global page. Each entry in the array contains 4 bytes of information.

Software selects the size of the global translation array (and global segments) by writing a value to bits <5 : 4> of the GTB_CTL register. Table 5 lists the global segment sizes and the corresponding sizes of the global translation array.

Table 5. Global Translation Array Sizes

GTB_CTL Bits <5 : 4>	Global Segment Size	Global Translation Array Entries	Global Translation Array Size
00	256 Mbytes	32,768	128 Kbytes
01	512 Mbytes	65,536	256 Kbytes
10	1 Gbyte	131,072	512 Kbytes
11	2 Gbytes	262,144	1 Mbyte

Software references the first entry of the global translation array using a base address. The base address is stored in a hardware register called the global translation buffer entry array base (GTB_EA_BASE) register.

Each entry in the global translation array is referenced by shifting the global page number left two bits and combining it with the base address of the global translation array (refer to Figure 50).

Figure 50. Global Translation Array Addresses

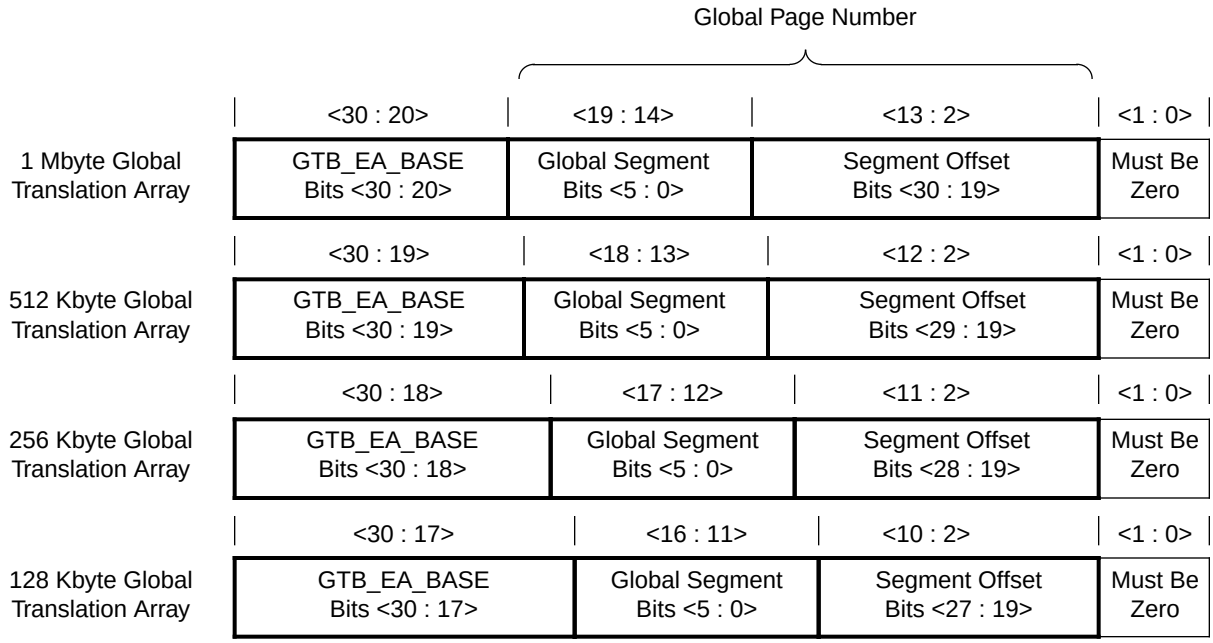
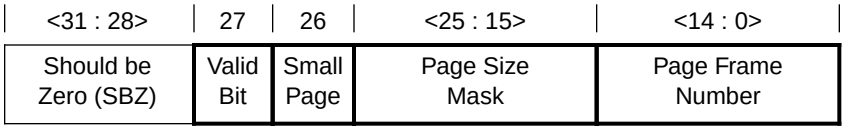


Figure 51 shows the format of an entry in the global translation array. Each entry contains a page frame number field, a page size mask field, a small page location bit and a valid bit.

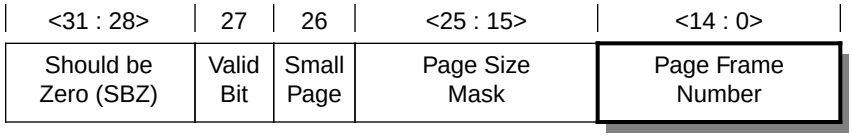
Figure 51. Global Translation Array Entry



Page Frame Number

The page frame number (refer to Figure 52) contains bits <30 : 16> of a physical address; however, not all of the page frame number bits may be used in the final physical address. For example, in a 2 Mbyte physical page address, only bits <14 : 5> of the page frame number bits are used as bits <30 : 21> (the physical page field) of the final physical address.

Figure 52. Page Frame Number Field of Global Translation Array Entry



Software indicates which bits of the page frame number are used as the physical page number by setting the page size mask bits to the appropriate values. More information on the page size mask bits is provided on the next page.

Page Size Mask

The page size mask field (refer to Figure 53) indicates the size of the physical page in the physical address. The physical page size ranges from 64 Kbytes to128 MBytes.

Figure 53. Page Size Mask Field of Global Translation Array Entry

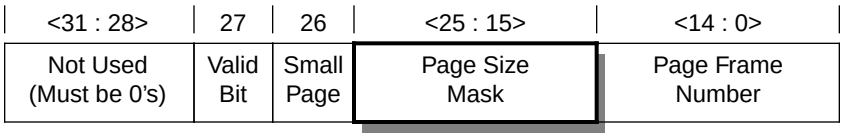


Table 6 shows the page size mask values and lists the physical page size for each value. The most significant bits of the page size mask must be set to 1's and the least significant bits must be set to 0's (unless the mask is set to all 0's or all 1's). Software must ensure that the format of the page size mask is correct. There is no hardware detection of an incorrectly formatted page size mask.

Table 6. Page Size Mask

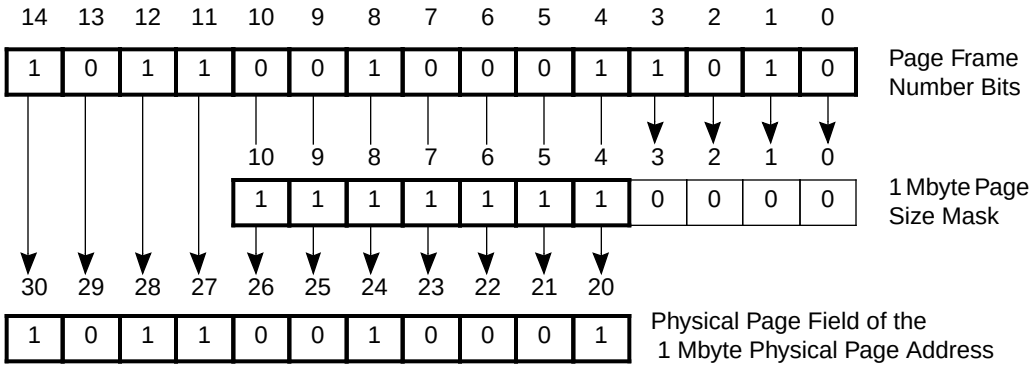
Page Size Mask Bits	Physical Page Size
00000000000	128 Mbytes
10000000000	64 Mbytes
11000000000	32 Mbytes
11100000000	16 Mbytes
11110000000	8 Mbytes
11111000000	4 Mbytes
11111100000	2 Mbytes
11111110000	1 Mbytes
11111111000	512 Kbytes
11111111100	256 Kbytes
11111111110	128 Kbytes
11111111111	64 Kbytes

The support circuitry uses the page frame number and the page size mask to generate the physical page field of the physical address. The following example describes the generation of the physical page field.

Figure 54 shows a sample page frame number and the resulting physical page number. In this example, software has set the page size mask to indicate a 1 Mbyte physical page.

The support circuitry compares the page size mask bits with bits <10 : 0> of the page frame number bits. When a bit of the page size mask is set to 1, the corresponding bit of the page frame number is used in the physical page number. When a bit of the page size mask is set to 0, the corresponding bit of the page frame number is not used in the physical page number.

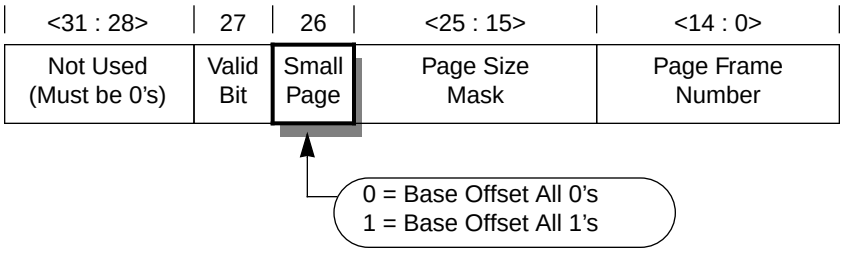
Figure 54. Generating the Physical Page Number Bits



Small Page Location Bit

A 512 Kbyte global page may be assigned to one 64 Kbyte, 128 Kbyte, or 256 Kbyte small physical page. The small page location bit (refer to Figure 55) indicates the value of the base offset for the small page.

Figure 55. Small Page Location Bit of Global Translation Array Entry



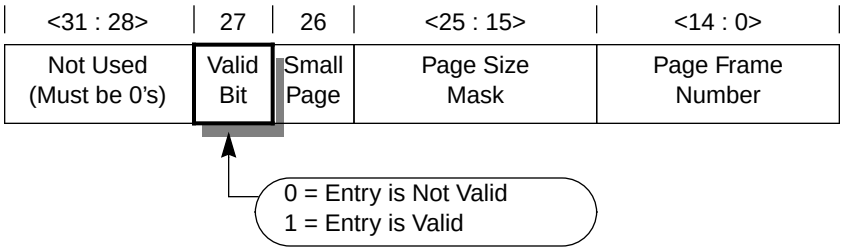
When set to 0, the small page bit indicates that the base offset for the small page is set to 0. When set to 1, the small page bit indicates that the base offset for the small page is set to all 1's.

NOTE: The following two pages provide more information on the base offset.

Valid Bit

The valid bit (refer to Figure 56) indicates the validity of the global translation array entry. When set to 1, the valid bit indicates the entry is a valid entry. When set to 0, the valid bit indicates the entry is not a valid entry.

Figure 56. Valid Bit of Global Translation Array Entry



When the support circuitry reads a global translation array entry from memory and the valid bit is set to 0, the support circuitry indicates that an error occurred.

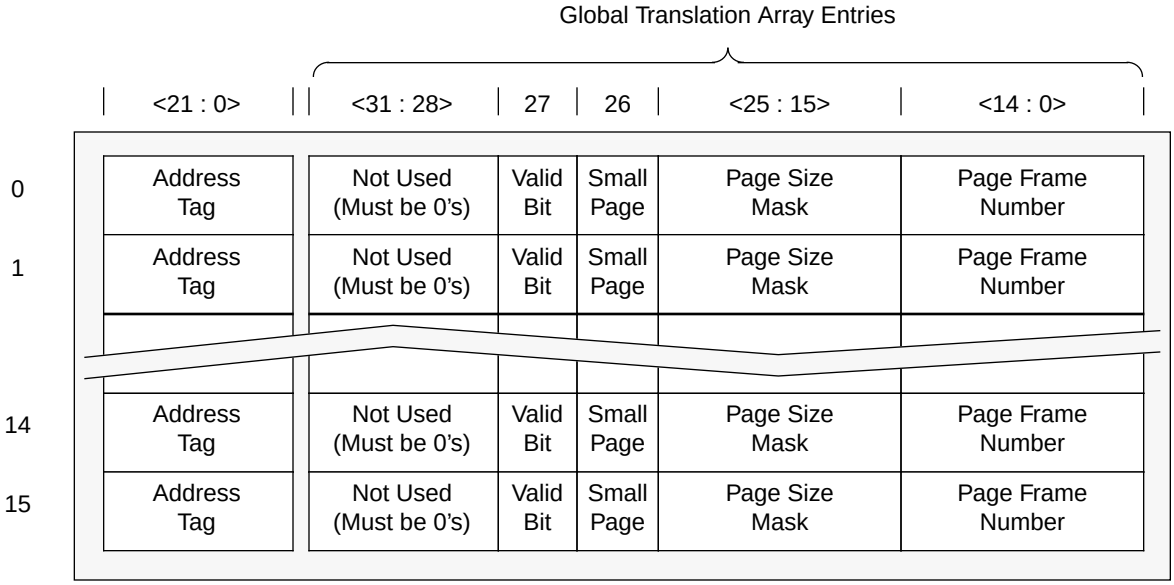
Global Translation Buffer

The global translation buffer (GTB) is a buffer located in the support circuitry that stores frequently or recently accessed global translation array entries. The GTB provides a low-latency method of retrieving global translation array information without having to access memory each time a global virtual address is translated into a physical address.

Global Translation Buffer Format

Figure 57 shows the format of the GTB. Each line in the GTB contains a global translation array entry and contains an address tag.

Figure 57. GTB Format



The address tag contains 22 bits that are compared with bits <37 : 16> of a global virtual address. Bits <21 : 3> of the address tag identify which 512 Kbyte global page corresponds to the global translation array entry stored in the line of the GTB. Bits <2 : 0> of the address tag are set by the small page location bit. When the small page bit is set to 0, bits <2 : 0> of the address tag are set to 0's. When the small page bit is set to 1, bits <2 : 0> of the address tag are set to 1's.

NOTE: The global pages that are part of global segment 0 are not translated into physical page numbers. In this case, the support circuitry uses the global virtual address as a physical page address.

Global Translation Buffer Operation

The support circuitry uses bits <37 : 16> of the global virtual address, the address tag in each line of the GTB, and the page size mask in each line of the GTB to determine whether the global virtual address matches the address tag of a line in the GTB. When the support circuitry receives a global virtual address, it increments through each line of the GTB, examines the page size mask for that line, and compares specified bits of the global virtual address bits <37 : 16> to specified bits of the address tag. Table 7 lists the possible page size mask bit values and shows the specified global virtual address (GVA) bits and address tag bits for each value.

NOTE: Bit 31 of the GVA is not used; however, it is included here to simplify the documentation.

Table 7. Specified GVA Bits and Address Tag Bits

Page Size Mask Bits	Physical Page Size	Specified GVA Bits	Specified Address Tag Bits
00000000000	128 Mbytes	Bits <37 : 27>	Bits <21 : 11>
10000000000	64 Mbytes	Bits <37 : 26>	Bits <21 : 10>
11000000000	32 Mbytes	Bits <37 : 25>	Bits <21 : 9>
11100000000	16 Mbytes	Bits <37 : 24>	Bits <21 : 8>
11110000000	8 Mbytes	Bits <37 : 23>	Bits <21 : 7>
11111000000	4 Mbytes	Bits <37 : 22>	Bits <21 : 6>
11111100000	2 Mbytes	Bits <37 : 21>	Bits <21 : 5>
11111110000	1 Mbytes	Bits <37 : 20>	Bits <21 : 4>
11111111000	512 Kbytes	Bits <37 : 19>	Bits <21 : 3>
11111111100	256 Kbytes	Bits <37 : 18>	Bits <21 : 2>
11111111110	128 Kbytes	Bits <37 : 17> *	Bits <21 : 1>
11111111111	64 Kbytes	Bits <37 : 16> **	Bits <21 : 0>

* Bits <18 : 17> must be set to all 0's or all 1's.

** Bits <18 : 16> must be set to all 0's or all 1's

When the support circuitry finds a match between the specified bits of the address tag and the global virtual address, and the valid bit for that GTB line is set to 1, the support circuitry generates the physical page field using the page frame number and the page size mask bits in that line of the GTB.

When the support circuitry does not find a match between the specified bits of an address tag and the global virtual address or when the valid bit in the GTB line is set to 0, the support circuitry must read a new global translation array entry from memory. When this occurs, it is called a GTB miss.

During a GTB miss, the support circuitry reads the value stored in the GTB_EA_BASE register and combines this value with the global page number to generate a global translation array address (refer again to Figure 50). The support circuitry uses this address to reference the global translation array entry and read the entry out of memory.

When the valid bit in the global translation array entry is set to 1, the support circuitry transfers the entry to the GTB and completes the generation of the physical page field. When the valid bit in the entry is set to 0, the support circuitry indicates that an error occurred.

NOTE: Software must ensure that the address space of a large page in the GTB does not overlap the address space of a smaller page in the GTB. If this occurs, the support circuitry may find a match between the specified bits of the GVA and the specified bits of the address tag for a line in the GTB that was not the intended match. This may cause an incorrect page frame number and page size mask to be used.

Software may perform two commands on the GTB: invalidate all the entries and invalidate a single entry. To invalidate all the entries in the GTB, software must write a 1 to the invalidate all bit (bit 2) of the global translation buffer control (GTB_CTL) register. To invalidate a single entry, software must perform two steps. First, software must write the global virtual address for the corresponding entry into the global translation buffer memory mover global virtual address (GTB_MM_GVA) register. Second, software must write a 1 to the invalidate entry bit (bit 3) of the GTB_CTL register.

Software must use the large physical page sizes to ensure good system performance when running applications such as large scientific programs. When several small pages are used, the support circuitry may have to access memory several times to service GTB misses.

Physical Page Offset

The page offset field of the physical address references a byte of data in a physical page. The page offset bits are obtained from the segment offset bits of the global virtual address; however, not all of the segment offset bits are used as physical page offset bits.

Source of Physical Page Offset

Table 8 lists the possible physical page sizes and shows which bits of the global virtual address are used as the physical page offset field for each size.

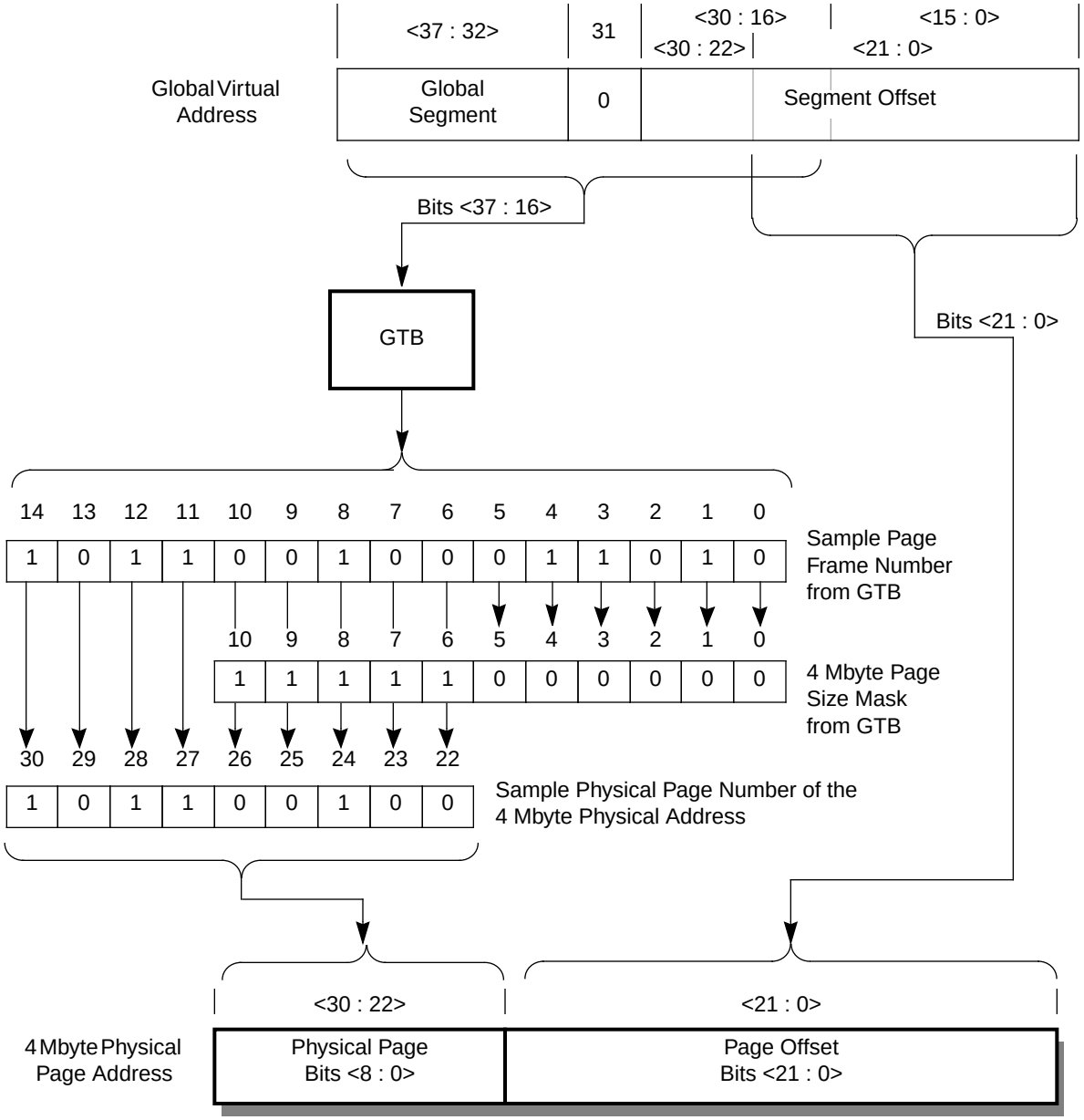
Table 8. Physical Page Offset Source

Physical Page Size	Source of Physical Page Offset Field
128 Mbytes	Bits <26 : 0> of the global virtual address
64 Mbytes	Bits <25 : 0> of the global virtual address
32 Mbytes	Bits <24 : 0> of the global virtual address
16 Mbytes	Bits <23 : 0> of the global virtual address
8 Mbytes	Bits <22 : 0> of the global virtual address
4 Mbytes	Bits <21 : 0> of the global virtual address
2 Mbytes	Bits <20 : 0> of the global virtual address
1 Mbytes	Bits <19 : 0> of the global virtual address
512 Kbytes	Bits <18 : 0> of the global virtual address
256 Kbytes	Bits <17 : 0> of the global virtual address
128 Kbytes	Bits <16 : 0> of the global virtual address
64 Kbytes	Bits <15 : 0> of the global virtual address

Sample Physical Page Offset Generation

Figure 58 shows a sample physical page offset generation for a 4 Mbyte physical page address. In this example, it is assumed that a GTB miss did not occur and the global segment is not 0.

Figure 58. Physical Page Address Generation



Atomic Memory Mover

The atomic memory mover moves data from an old physical page to a new physical page when software remaps a global page to a new physical page. After software stores parameters into the memory-mover registers, the memory mover performs a data move that is transparent to the user applications.

Atomic Memory Mover Operation

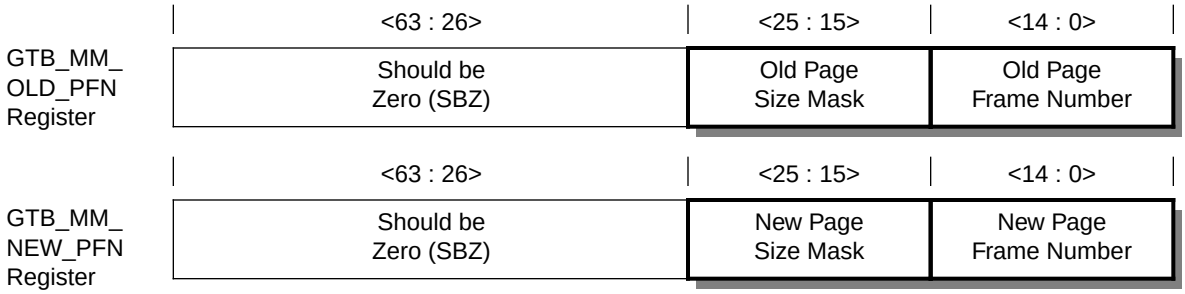
The following steps describe how software remaps a global page to a new physical page and describe the parameters that software must load into the memory-mover registers.

1. Software reads the GTB_CTL register to ensure that bit 0 is set to 0. When set to 0, this bit indicates that the memory mover is idle.

2. Software sets bit 27 (FLUSH_ALL_REFS bit) of the C_ERR1 register to 1.

3. The memory mover contains two registers that store page frame numbers and page size masks: the GTB_MM_OLD_PFN register and the GTB_MM_NEW_PFN register (refer to Figure 59). After ensuring that the memory mover is idle (by reading bit 0 of the GTB_CTL register) and before initiating a new memory move, software stores the old and new page frame numbers and page size masks into these registers.

Figure 59. New and Old Page Frame Number Registers



4. Software halts the operation of the global translation buffer. To do this, software writes a 1 to the hold bit (bit 1) of the GTB_CTL register.

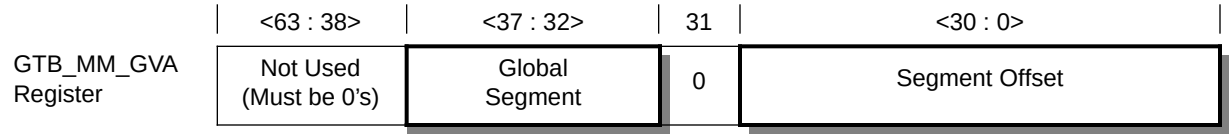
NOTE: If a message SEND command is in progress, the support circuitry does not set the hold bit of the GTB_CTL register to 1 until the message send operation is complete. If software is sending messages and needs to ensure that the GTB hold is in effect, software must read the value of the hold bit of the GTB_CTL register to ensure that it is set to 1.

5. Software issues a memory barrier command.

6. Software reads the GTB_CTL register to ensure that the QUIET bit (bit 6) is set to 1.

7. Software invalidates the GTB entries for the global page that is being remapped. To invalidate a GTB entry, software must perform two steps. First, software must write the global virtual address for the corresponding entry into the GTB_MM_GVA register (refer to Figure 60). Second, software must write a 1 to the invalidate entry bit (bit 3) of the GTB_CTL register.

Figure 60. Memory Mover Global Virtual Address Register



8. Software rewrites the appropriate entries in the global translation array stored in memory with the values for the new page assignments.

9. Software issues a memory barrier command.

10. Software resumes operation of the GTB. To do this, software writes a 1 to the valid bit (bit 0) of the GTB_CTL register and a 0 to the hold bit (bit 1) of the GTB_CTL register.

11. Software reads the GTB_CTL register to ensure that bit 0 is set to 0. When set to 0, this bit indicates that the memory move is complete.

12. After the memory move is complete, software sets bit 27 (FLUSH_ALL_REFS bit) of the C_ERR1 register to 0.

When software sets the valid bit (bit 0) of the GTB_CTL register to 1, the memory-mover starts the data move. During the data move, the support circuitry compares an incoming global virtual address to both the new and old page

translation parameters. Depending on how far the data move has progressed and depending on the value of the global virtual address, the support circuitry transfers data into or out of either the new physical page or the old physical page.

Software may check for the completion of the memory move using a polled method. To do this, software periodically reads the memory-mover valid bit (bit 0) of the GTB_CTL register. When set to 1, this bit indicates that the memory mover is performing a data move. When hardware resets the memory mover valid bit to 0, this bit indicates that the data move is complete.

Software may also check for the completion of the data move using an interrupt method. To do this, software sets bit 35 (MOVER bit) of the IR_ENABLE register to 1, which enables the MOVER interrupt (bit 35) of the IR_STATUS register. When set to 1, the MOVER interrupt indicates that an atomic memory mover operation is complete.

NOTE: Software cannot abort an atomic memory mover operation. The data move is not affected if software sets the valid bit (bit 0) of the GTB_CTL register to 0 during a memory move.

Atomic Memory Mover Characteristics

The atomic memory mover may be used to perform several functions. For example, software may change a small page into a bigger page or may combine multiple small pages into a large page.

Using the memory mover, software can change a small page into a bigger page by using one data move operation. After loading the memory-mover registers with the correct parameters and after changing the appropriate entries in the global translation array, software signals the memory mover to start the data move. When the data move is complete, the small page will be remapped as a bigger page.

Software can also combine multiple small pages into a large page by using multiple data move commands. For all but the last small page, software performs data move operations (as described in the previous paragraph) to align the small pages in memory. Before requesting the data move for the last small page, software loads the memory-mover registers with the correct parameters and changes the appropriate entries in the global translation array to reflect a new large page. After software requests the final data move and the data move is complete, the small pages will be remapped into a large page.

The atomic memory mover keeps data in local memory and data in the secondary cache of the microprocessor consistent during the data move.

Index

F

G

I

L

M

O

P

R

S

U

V