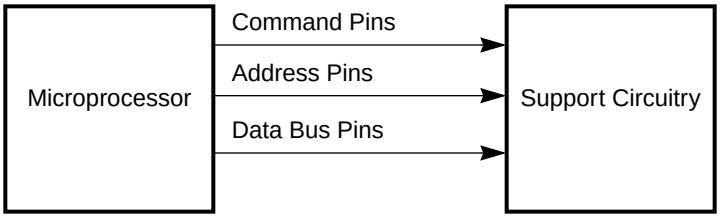
Commands

HMM-142-0

Commands Overview

A command is information generated by the microprocessor that signals the support circuitry to perform functions on system data. The microprocessor sends a command to the support circuitry by placing information on the microprocessor command pins and, in some cases, the microprocessor address pins (refer to Figure 1).

Figure 1. Command and Address Pins



The command pins indicate when the microprocessor is requesting a write-related command (WRITE_BLOCK) or a read-related command (READ_MISS). WRITE_BLOCK commands indicate that the microprocessor will transfer system data or additional address information to the support circuitry over the microprocessor data bus. READ_MISS commands indicate that the microprocessor is requesting data. The microprocessor does not transfer any information to the support circuitry over the microprocessor data bus when issuing a READ_MISS command.

The address pins may contain additional command information when software is using the E-register access method of referencing data.

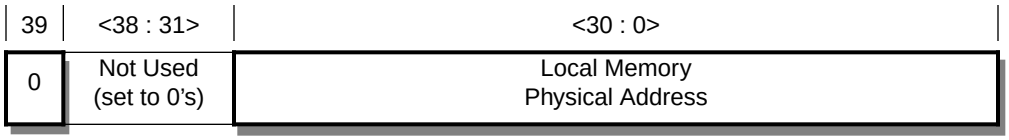
There are two main types of commands: direct local memory commands and memory-mapped commands.

Direct Local Memory Commands

The microprocessor initiates a direct local memory command by setting the microprocessor command pins to signal a WRITE_BLOCK or READ_MISS and by setting bit 39 of the microprocessor-generated address to 0. There are two direct local memory commands: direct local memory store and direct local memory load.

Figure 2 shows the format of the microprocessor-generated address for a direct local memory command. Bits <30 : 0> of the address contain a byte-oriented address that references a physical location in local memory. Bits <38 : 31> are not used and should be set to 0 along with bit 39.

Figure 2. Direct Local Memory Address

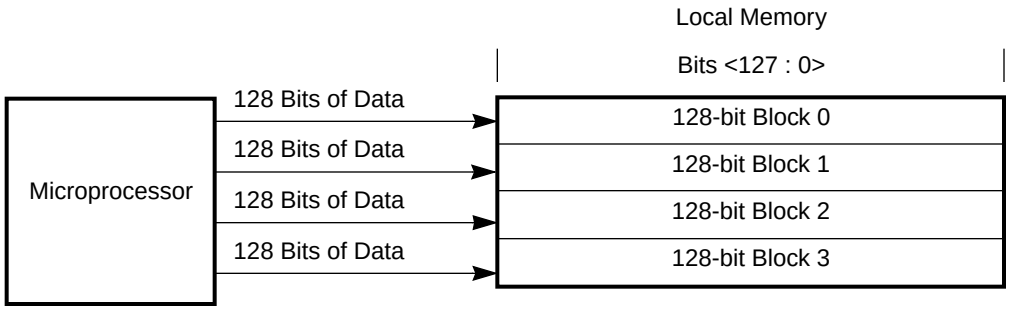


The following subsections describe the direct local memory commands.

Direct Local Memory Store

A direct local memory store command signals the support circuitry to transfer up to 64-bytes of data from the microprocessor to local memory. The support circuitry receives the store data from the microprocessor in four 128-bit transfers over the microprocessor data bus (refer to Figure 3).

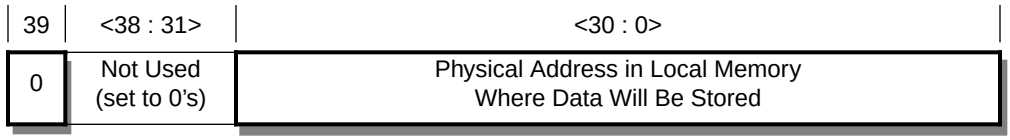
Figure 3. Direct Local Memory Store Command



The microprocessor presents a direct local memory store command to the support circuitry when valid data for a local memory location is removed from the microprocessor secondary cache to make room for new valid data. The data that is removed from the secondary cache is called a victim.

The microprocessor issues a direct local memory store command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the direct local memory store address (refer to Figure 4) on the microprocessor address bus.

Figure 4. Direct Local Memory Store Address

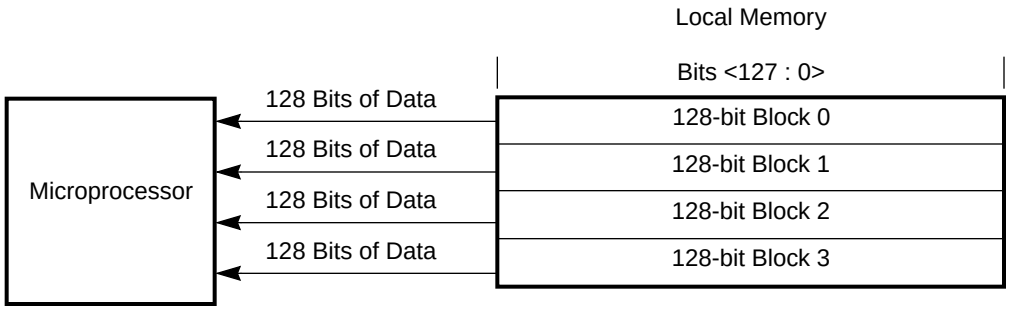


NOTE: Bits <30 : 0> reference a 64-byte block of data; therefore, bits <5 : 0> of the microprocessor-generated address are set to 0.

Direct Local Memory Load

A direct local memory load command signals the support circuitry to transfer 64-bytes of data from local memory to the microprocessor. The support circuitry sends the load data to the microprocessor in four 128-bit transfers over the microprocessor data bus (refer to Figure 5).

Figure 5. Direct Local Memory Load Command

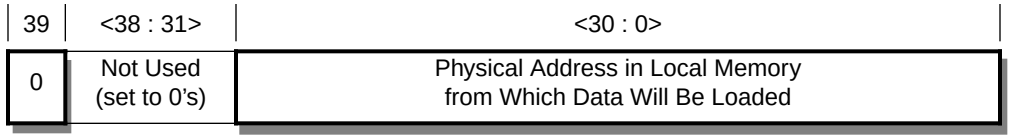


The microprocessor presents a direct local memory load command to the support circuitry when a secondary cache miss occurs. A secondary cache miss occurs when the microprocessor attempts to read data from a 64-byte block in the secondary cache and the data in the block does not have the correct address information or the data in the block is invalid.

Direct Local Memory Load Address

The microprocessor issues a direct local memory load command by placing a READ_MISS command on the microprocessor command pins and by placing the direct local memory load address (refer to Figure 6) on the microprocessor address bus.

Figure 6. Direct Local Memory Load Address



NOTE: Bits <30 : 0> reference a 64-byte block of data; therefore, bits <5 : 0> of the microprocessor-generated address are set to 0.

Streams

A stream is a set of load commands that reference contiguous addresses in local memory. For example, a stream could consist of 50 direct local memory load commands where each load command references the next sequential local memory address. Although a stream must reference contiguous addresses, a stream does not have to emerge from the microprocessor as contiguous commands. Other commands may be intermixed within a stream.

The support circuitry maintains a history of memory references and detects when streams occur. After detecting the presence of a stream, the support circuitry performs read-ahead operations and stores data from memory into stream buffers located in the support circuitry. The stream buffers reduce the latency needed for the support circuitry to supply data for a stream of load commands.

The support circuitry uses four components to perform stream operations. Table 1 lists these components and provides a description for each component.

Table 1. Support Circuitry Stream-control Components

Component	Description
History buffer (HB)	Stores up to 8 addresses. When the support circuitry receives an address from the microprocessor and the address matches an address in the history buffer, the support circuitry assigns a new stream.
Stream buffers	Store data or instructions for up to 6 individual streams. Two or four stream buffers are assigned to each stream. The number depends on the value of bit 4 (DOUBLE_ENABLE bit) of the C_SYS_CTL register.
Initial prefetch (IP) buffers	Store up to 4 initial prefetches from local memory. An initial prefetch is data or instructions read from local memory in anticipation of a possible stream.
Read-ahead buffers	Six 256-byte buffers in the support circuitry (M option) that temporarily store data or instructions from local memory that are destined for the local PE or a remote PE.

Although stream operations are performed by hardware, software can change parameters that affect stream operations. Software changes these parameters by writing to fields of the user control (USER_CTL) register.

Bits <1 : 0> of the USER_CTL register control data streams. Bits <3 : 2> of the USER_CTL register control instruction streams (refer to Table 2 and Table 3).

Table 2. USER_CTL Bits <1 : 0>

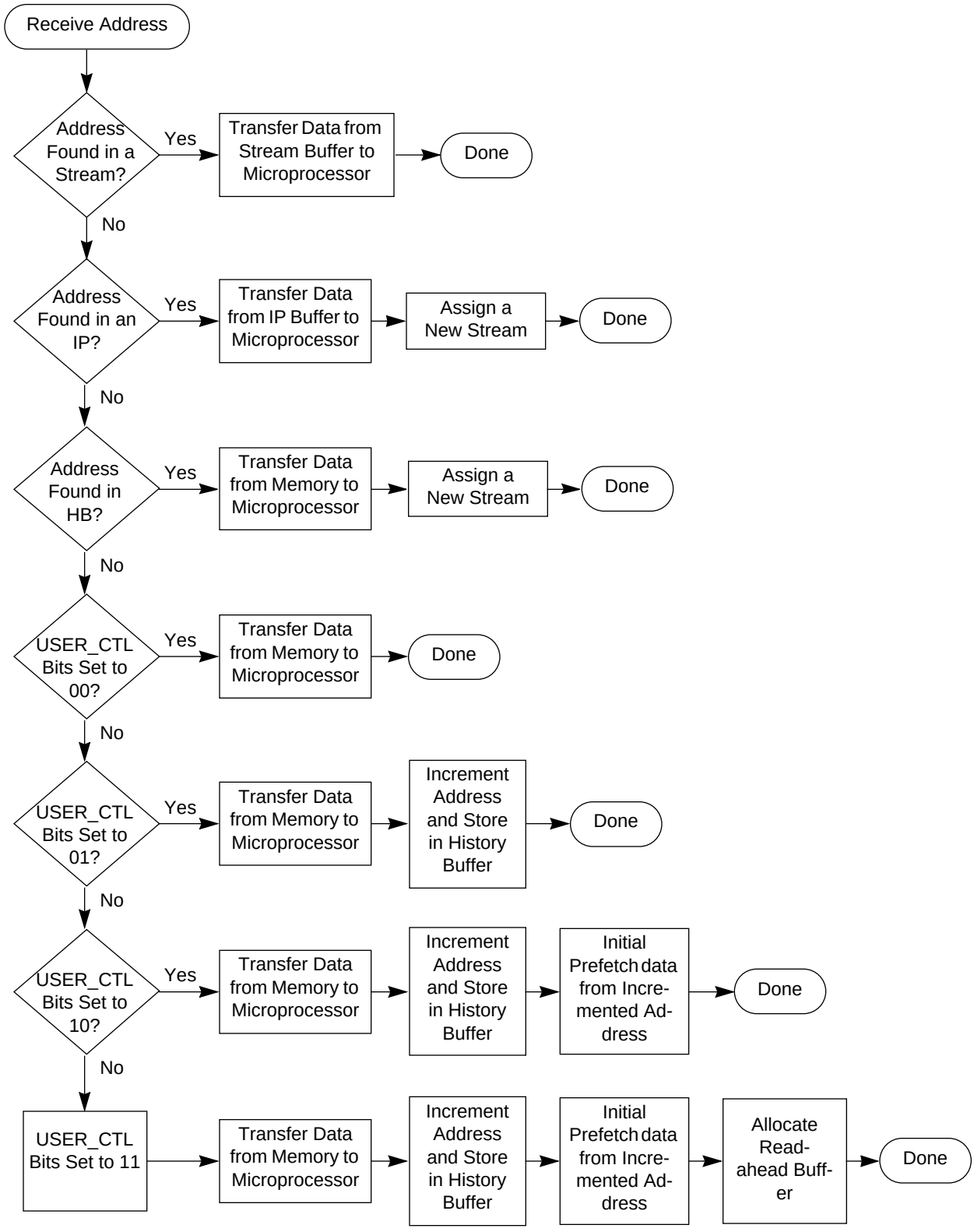
Bits <1 : 0>	Description
00	Disable the detection and assignment of new data streams. Old data streams are not affected.
01	Enable the detection and assignment of new data streams.
10	Enable the detection and assignment of new data streams. Also, perform an initial prefetch on a stream buffer miss.
11	Enable the detection and assignment of new data streams. Also, perform an initial prefetch and allocate a read-ahead buffer on a stream buffer miss.

Table 3. USER_CTL Bits <3 : 2>

Bits <3 : 2>	Description
00	Disable the detection and assignment of new instruction streams. Old instruction streams are not affected.
01	Enable the detection and assignment of new instruction streams.
10	Enable the detection and assignment of new instruction streams. Also, perform an initial prefetch on a stream buffer miss.
11	Enable the detection and assignment of new instruction streams. Also, perform an initial prefetch and allocate a read-ahead buffer on a stream buffer miss.

Figure 7 is a flow chart that illustrates stream operations that the support circuitry performs when it receives a local-memory load command and address from the microprocessor.

Figure 7. Stream Flow Chart

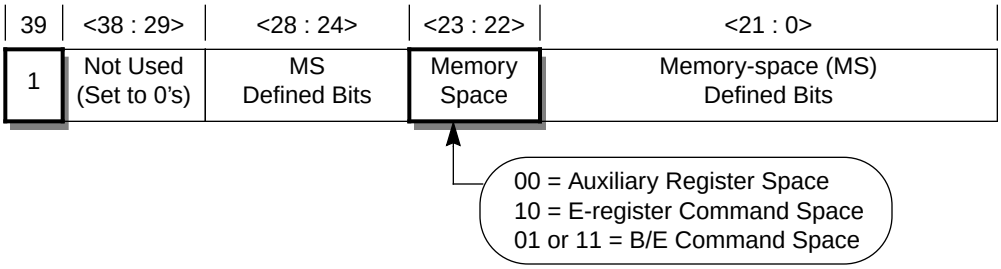


Memory-mapped Commands

The microprocessor initiates a memory-mapped command by setting the microprocessor command pins to signal a WRITE_BLOCK or READ_MISS and by setting bit 39 of the microprocessor-generated address to 1 (to indicate a memory-mapped function address). There are three main types of memory-mapped commands: auxiliary register commands, barrier and eureka (B/E) commands, and E-register commands.

Figure 8 shows the format of the microprocessor-generated address for a memory-mapped function. Bits <23 : 22> of the address indicate which main type of memory-mapped function is being referenced. This, along with the microprocessor command pins, indicates which type of memory-mapped command is being issued.

Figure 8. Memory-mapped Function Address



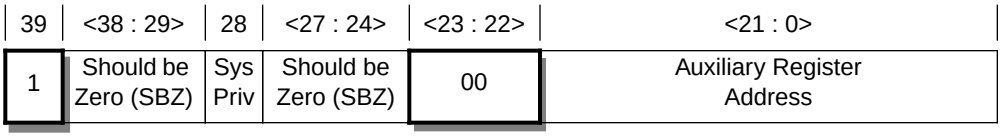
The following subsections describe the main types of memory-mapped commands.

Auxiliary Register Commands

Auxiliary register commands signal the support circuitry to transfer data between the microprocessor and the auxiliary registers. Auxiliary registers are registers located in a PE that the microprocessor can access directly without using the E registers.

The microprocessor indicates an auxiliary register command by setting bit 39 of microprocessor-generated address to 1 and by setting bits <23 : 22> of the address to 00 (refer to Figure 9).

Figure 9. Auxiliary Register Space Address



The microprocessor identifies which auxiliary register is the source or destination register for the command by setting the auxiliary register address field (bits <21 : 0>) of the microprocessor-generated address to the appropriate value.

There are two auxiliary register commands: store and load. When the microprocessor sets the command pins to indicate a WRITE_BLOCK command, the auxiliary register command is a store. This command signals the support circuitry to transfer data from the microprocessor to a destination auxiliary register.

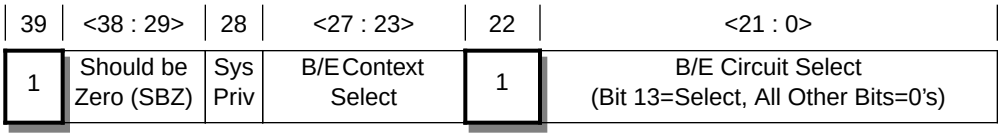
When the microprocessor sets the command pins to indicate a READ_MISS command, the auxiliary register command is a load. This command signals the support circuitry to transfer data from a source auxiliary register to the microprocessor.

Barrier and Eureka Commands

Barrier and eureka (B/E) commands signal the support circuitry to change the state of the barrier and eureka synchronization circuits.

The microprocessor indicates a B/E command by setting bits 39 and 22 of the microprocessor-generated address to 1 (refer to Figure 10).

Figure 10. B/E Command Space Address



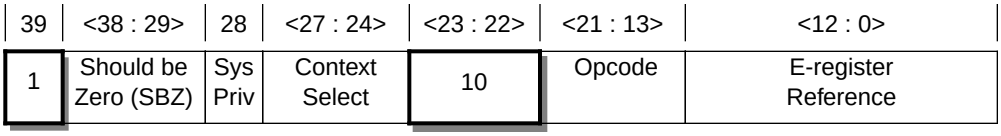
E-register Commands

E-register commands signal the support circuitry to perform several types of functions. These functions include:

- Transferring data between the microprocessor and the E registers
- Transferring data between the E registers and local or global memory
- Performing arithmetic or logical functions on system data

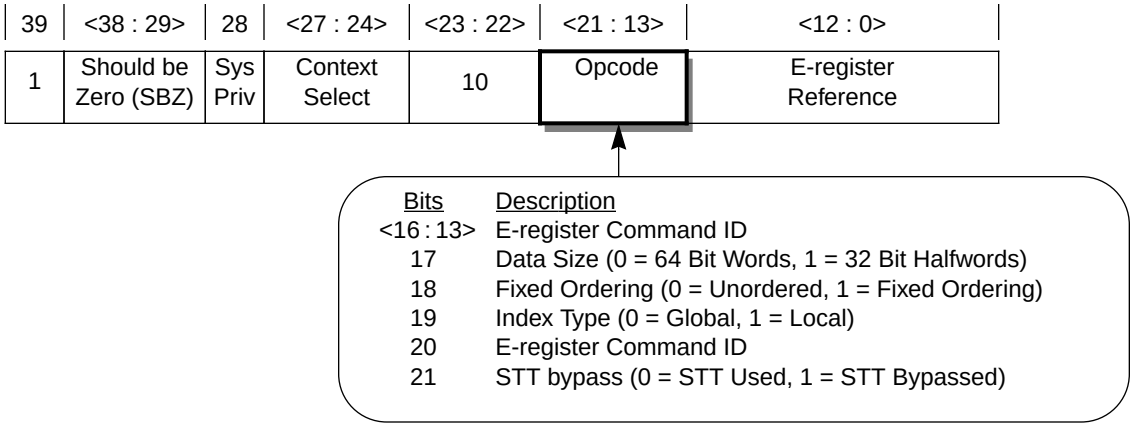
The microprocessor indicates an E-register command by setting bit 39 of the microprocessor-generated address to 1 and by setting bits <23 : 22> of the address to 10 (refer to Figure 11).

Figure 11. E-register Command Space Address



Bits <21 : 13> of the microprocessor-generated address contain an opcode. The opcode indicates which E-register command the microprocessor is issuing. Figure 12 shows the generalized format of the opcode, and Table 4 lists the opcodes.

Figure 12. Opcode



The following subsections describe each of the E-register commands. For faster referencing, the headings are alphabetized by command name.

Table 4. Opcodes

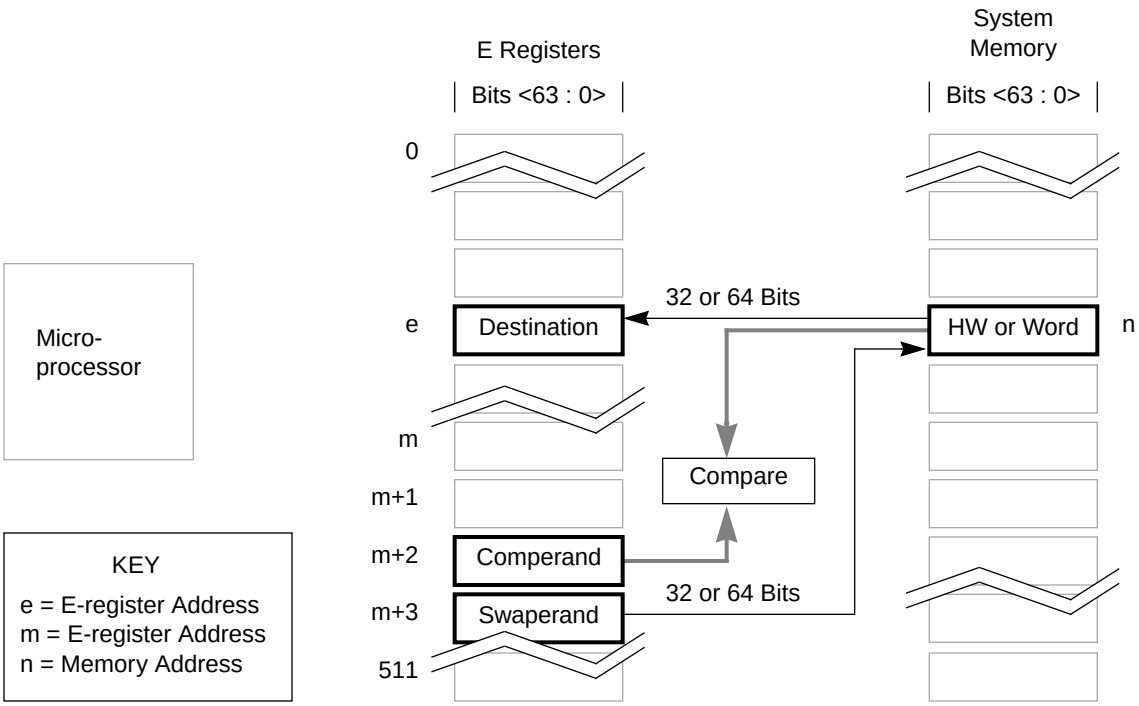
Microprocessor-generated Address Bits									
21	20	19	18	17	16	15	14	13	
STT Use (T)	ID	Index Type (L)	Fixed Ordering (F)	Data Size (S)	Command ID				Command Name
X	0	X	X	X	X	0	0	0	LOAD or STORE
1	0	1	0	0=64 Bits 1=32 Bits	0	0	0	1	STORE_PLZ
0=Use 1=Bypass	0	0=Global 1=Local	0=No 1=Yes	0=64 Bits 1=32 Bits	0	0	1	0	PUT
0=Use 1=Bypass	0	0=Global 1=Local	0=No 1=Yes	0=64 Bits 1=32 Bits	0	0	1	1	GET
0=Use 1=Bypass	0	0=Global 1=Local	0=No 1=Yes	1	0	1	0	0	GET_INC
0=Use 1=Bypass	0	0=Global 1=Local	0=No 1=Yes	0=64 Bits 1=32 Bits	0	1	0	1	MSWAP
1	0	0	0	1	1	0	0	1	STORE_CENT
0=Use 1=Bypass	0	0=Global 1=Local	0=No 1=Yes	0=64 Bits 1=32 Bits	1	0	1	0	PUTV
0=Use 1=Bypass	0	0=Global 1=Local	0=No 1=Yes	0=64 Bits 1=32 Bits	1	0	1	1	GETV
0=Use 1=Bypass	0	0=Global 1=Local	0=No 1=Yes	1	1	1	0	0	GET_ADD
0=Use 1=Bypass	0	0=Global 1=Local	0=No 1=Yes	0=64 Bits 1=32 Bits	1	1	0	1	CSWAP
0=Use 1=Bypass	0	0	0=No 1=Yes	0	1	1	1	1	SEND
0	1	0	0	0	0	0	0	0	ERS_READ or ERS_WRITE
1	1	1	0	0	1	0	0	0	SPUT
1	1	1	0	1	1	0	0	0	SGET
0=Use 1=Bypass	1	0=Global 1=Local	1	0	1	0	0	0	DPUT
0=Use 1=Bypass	1	0=Global 1=Local	1	1	1	0	0	0	DGET

NOTE: X = Value of this bit does not matter

CSWAP

The compare and swap (CSWAP) E-register commands signal the support circuitry to compare a comperand value to data in system memory. If the comperand value and the data in system memory are equal, the support circuitry transfers the data from system memory to a destination E register and transfers a swaperand value from another source E register into the original system memory location (refer to Figure 13). If the comperand and data in system memory are not equal, the support circuitry transfers the data from system memory to a destination E register, but does not perform the swap operation.

Figure 13. CSWAP Commands



There are two CSWAP commands: CSWAP4 and CSWAP8. The CSWAP4 command performs functions on 32-bit (4-byte) halfwords (HW) of data. The CSWAP8 command performs functions on 64-bit (8-byte) words of data.

The write buffer in the microprocessor may merge two, three, or four CSWAP commands into a single operation. The merged CSWAP operation compares and possibly swaps up to four 64-bit words of data. Each of the CSWAP commands in the merged operation may use unique comperand and swaperand values, or two, three, or four CSWAP commands in the merged operation may use the same comperand and swaperand values from the same two source E registers.

CSWAP Address

The microprocessor issues a CSWAP command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the CSWAP address arguments on the microprocessor address bus and data bus. Before requesting a CSWAP command, software must load the more operands (MO) block of E registers with the centrifuge mask, the base virtual address, the comperand value, and the swaperand value.

NOTE: If a parameter in the MO block is not used, the parameter may be set to any value. For example, if the centrifuge mask is not used, it may be set to any value.

CSWAP ERS Codes

If the E-register state (ERS) codes are set to EMPTY for any of the E registers in the MO block, the support circuitry will stall before performing the CSWAP command. When the ERS codes for these E registers are set to FULL, the support circuitry sets the ERS code for the destination E register to EMPTY and performs the CSWAP command. After storing the value read from memory into the destination E register, the support circuitry sets the ERS code for the destination E register to FULL.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

CSWAP Destination E-register Format

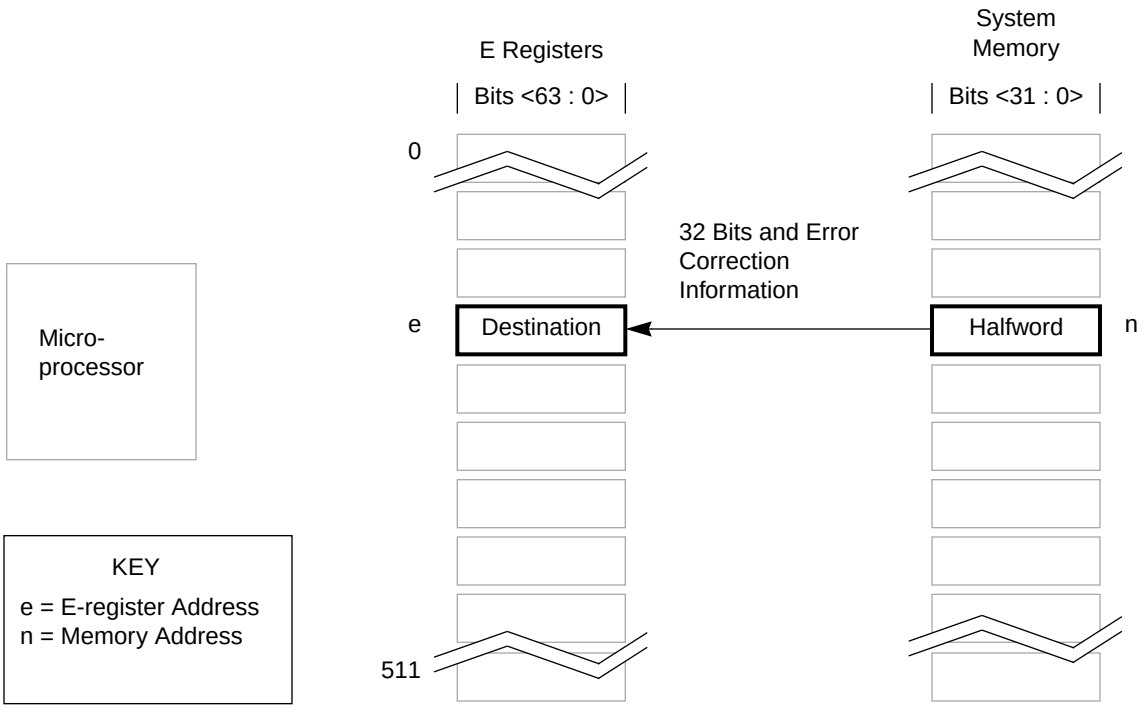
After the support circuitry performs a CSWAP8 command, the destination E register contains the 64-bit word of data read from memory. After the support circuitry performs a CSWAP4 command, bits <63 : 32> of the destination E register contain the 32-bit halfword read from memory and bits <31 : 0> are read as zeros.

NOTE: The CSWAP8 command is issued when the data size bit (S) of the opcode is set to 0. The CSWAP4 command is issued when the data size bit of the opcode is set to 1.

DGET

The diagnostic get (DGET) E-register command signals the support circuitry to transfer a 32-bit halfword, check bits for the halfword, and syndrome bits for the halfword from system memory to a destination E register (refer to Figure 14). If the halfword contains a single-bit error, the support circuitry corrects the error and places the corrected data in the destination E register (unless bit 5 of the M_CFG register is set to 1). If the halfword contains a multiple-bit error, the support circuitry places the uncorrected data in the destination E register. The DGET command is used to test data error correcting circuitry.

Figure 14. DGET Command



The write buffer in the microprocessor may merge two, three, or four DGET commands into a single operation. The merged DGET operation transfers up to four 32-bit halfwords, along with the check bits and syndrome for each halfword, from any four locations in system memory to four contiguous destination E registers.

DGET Address

The microprocessor issues a DGET command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the DGET address arguments on the microprocessor address bus and data bus. Before requesting a DGET command, software must load the MO block of E registers with the centrifuge mask and the base virtual address.

NOTE: If a parameter in the MO block is not used, the parameter may be set to any value. For example, if the centrifuge mask is not used, it may be set to any value.

DGET ERS Codes

If the ERS codes are set to EMPTY for the E registers that contain the centrifuge mask and base virtual address, the support circuitry stalls before performing the DGET command. When the ERS codes for these E registers are set to FULL, the support circuitry sets the ERS code for the destination E register to EMPTY and performs the DGET command. After storing the DGET output into the destination E register, the support circuitry sets the ERS code for the destination E register to FULL.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

DGET Destination E-register Format

The format of the destination E register for a DGET command is different if the 32-bit halfword is bits <31 : 0> of a word or is bits <63 : 32> of a word. Figure 15 shows the format of the destination E register after the support circuitry completes a DGET command for bits <31 : 0> of a word.

Figure 15. DGET Destination E-register Format (Lower 32 Bits)

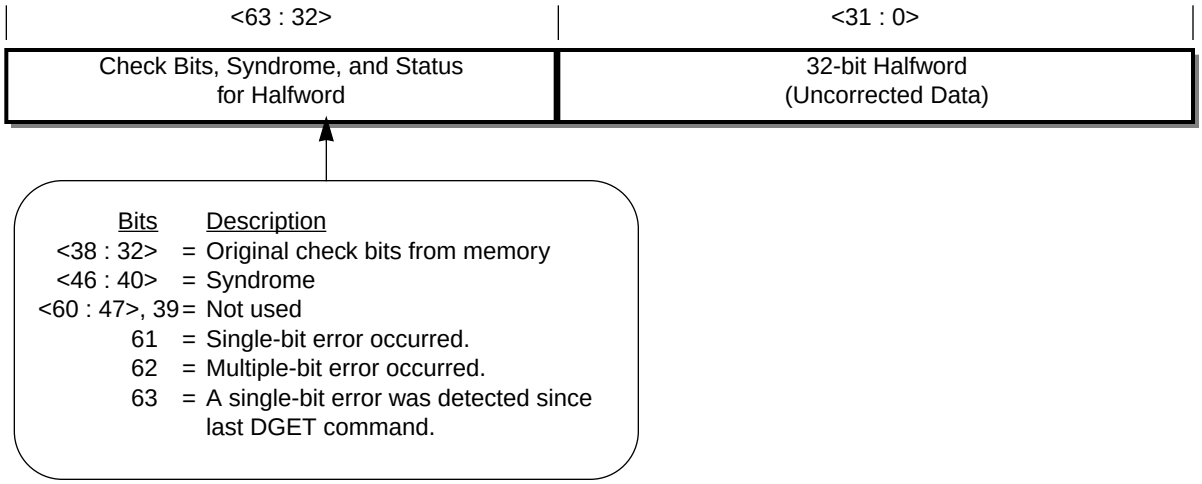
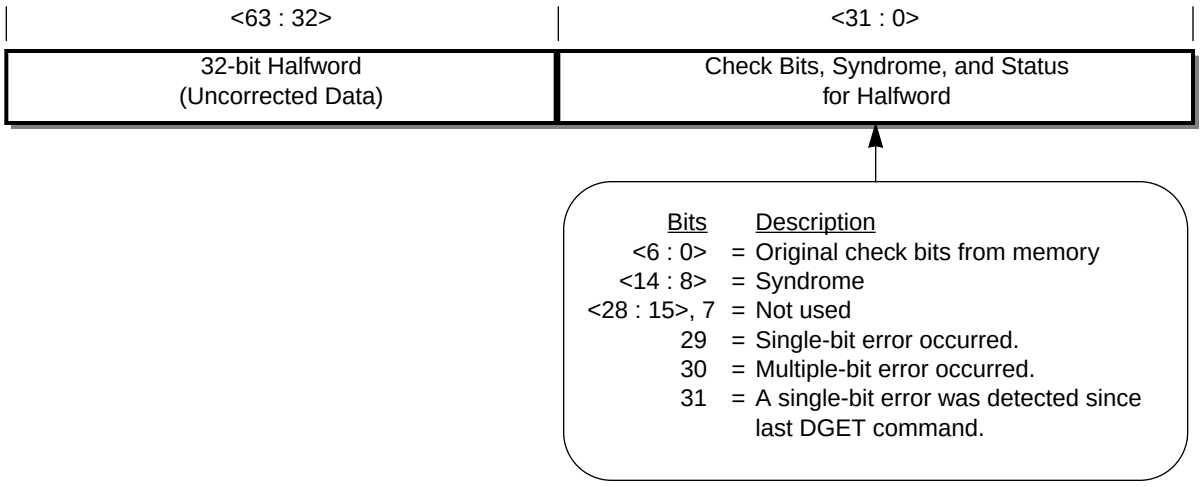


Figure 16 shows the format of the destination E register after the support circuitry completes a DGET command for bits <63 : 32> of a word.

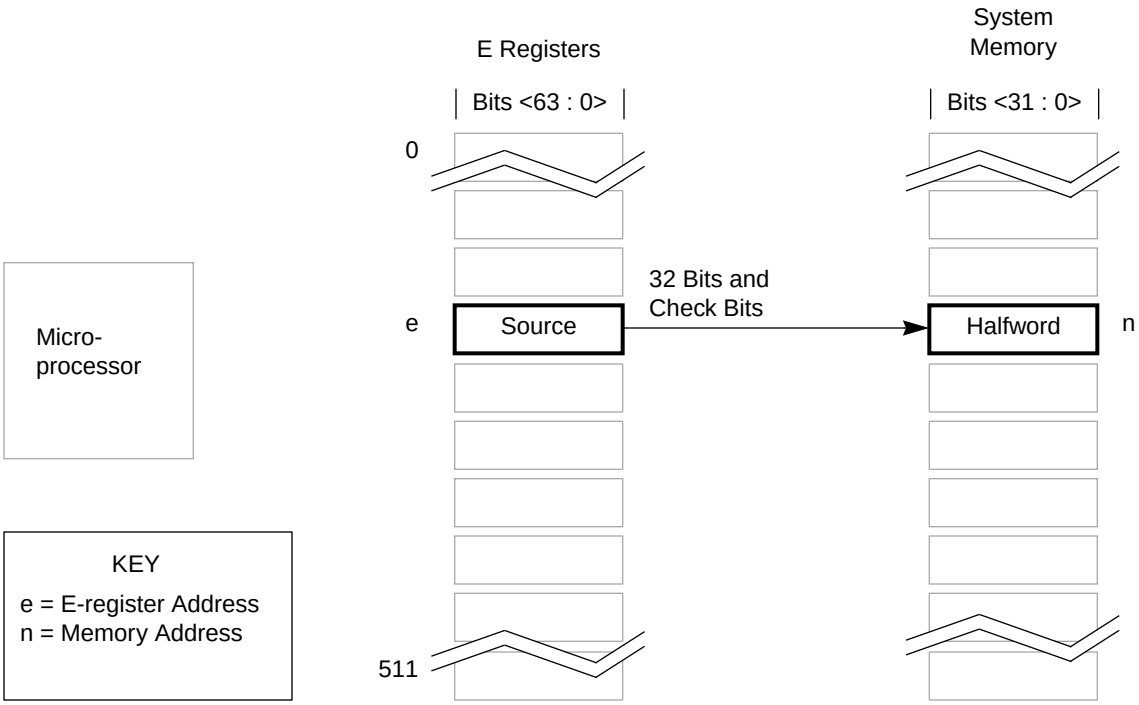
Figure 16. DGET Destination E-register Format (Upper 32 Bits)



DPUT

The diagnostic put (DPUT) E-register command signals the support circuitry to transfer a 32-bit halfword and check bits for the halfword from a source E register to a system memory location (refer to Figure 17). The DPUT command is used to set the check bits for a system memory location into a known state before testing the data error correcting circuitry.

Figure 17. DPUT Command



The write buffer in the microprocessor may merge two, three, or four DPUT commands into a single operation. The merged DPUT operation transfers up to four 32-bit halfwords, along with check bits for each halfword, from four contiguous source E registers to any four locations in system memory.

DPUT Address

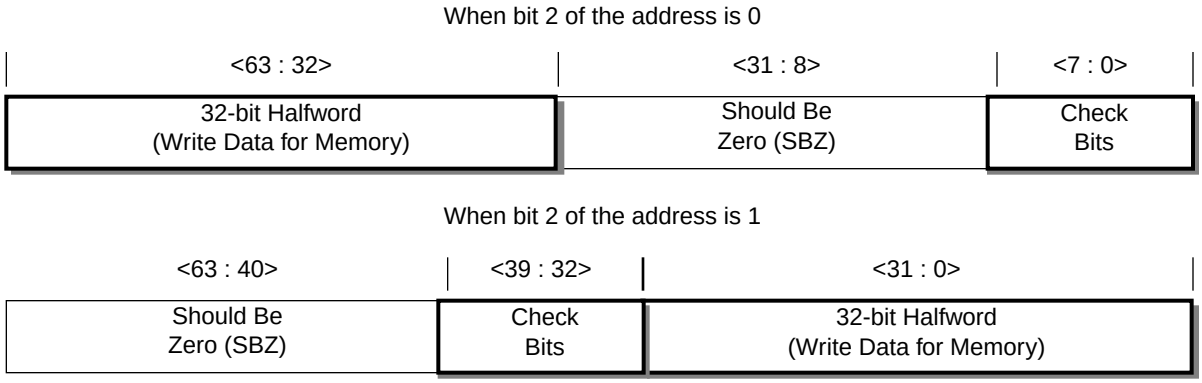
The microprocessor issues a DPUT command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the DPUT address arguments on the microprocessor address bus and data bus. Before requesting a DPUT command, software must load the MO block of E registers with the centrifuge mask and the base virtual address.

NOTE: If a parameter in the MO block is not used, the parameter may be set to any value. For example, if the centrifuge mask is not used, it may be set to any value.

DPUT Source E-register Format

In addition to the MO block of E registers, software must load a source E register with the 32-bit halfword and checkbits for the halfword before requesting a DPUT command. Figure 18 shows the format of the source E register for a DPUT command.

Figure 18. DPUT Source E-register Format



DPUT ERS Codes

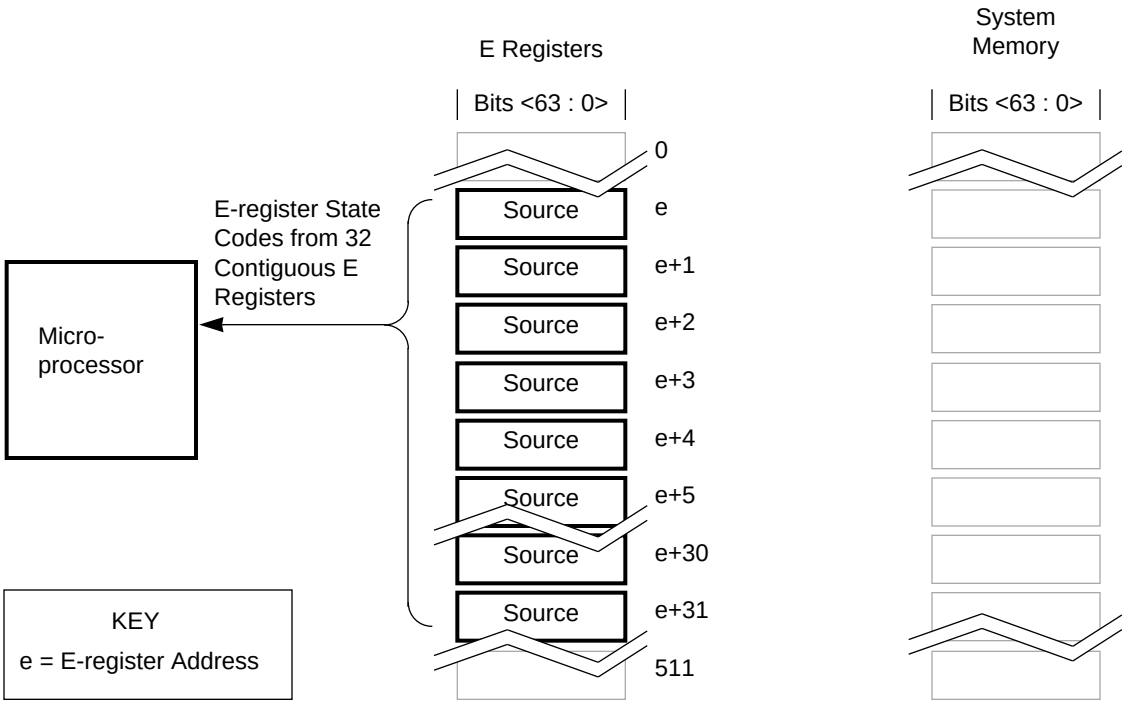
If the ERS codes are set to EMPTY for the E registers that contain the centrifuge mask and base virtual address, the support circuitry stalls before performing the DPUT command. When the ERS codes for these E registers are set to FULL, the support circuitry performs the DPUT command. The DPUT command does not modify the ERS codes for any of the E registers.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

ERS_READ

The E-register state code read (ERS_READ) E-register command signals the support circuitry to transfer the ERS codes from 32 contiguous source E registers to the microprocessor (refer to Figure 19).

Figure 19. ERS_READ Command



The missed-address file in the microprocessor may merge two, three, or four ERS_READ commands into a single operation. The merged ERS_READ operation transfers the ERS code from 128 contiguous source E registers to the microprocessor.

ERS_READ Address

The microprocessor issues an ERS_READ command by placing a READ_MISS command on the microprocessor command pins and by placing the ERS_READ address arguments on the microprocessor address bus. The microprocessor data bus is not used for an ERS_READ command.

NOTE: Bits <7 : 0> of the microprocessor-generated address reference a block of 32 destination E registers.

ERS_READ ERS Codes

The ERS_READ command does not affect the value of any of the ERS codes in the source E registers.

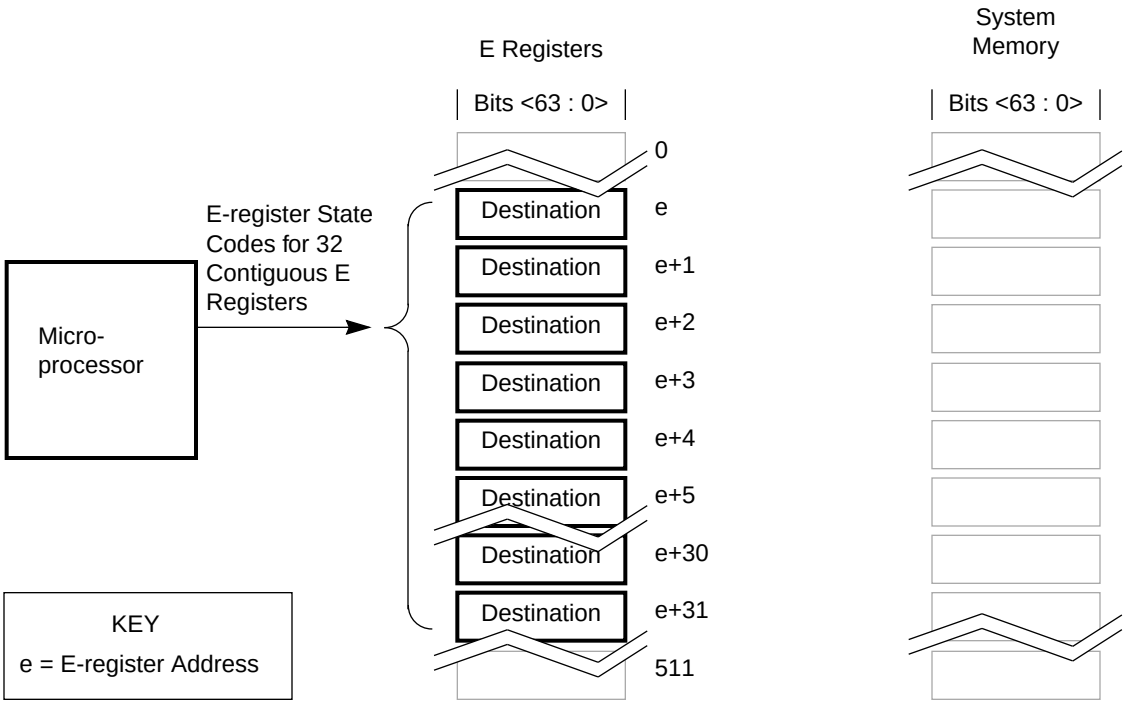
ERS_READ Data Format

The microprocessor data bus contains the ERS codes for 32 contiguous destination E registers. For example, bits <7 : 6> of the data bus contain the ERS code for the fourth E register in a block of 32 destination E registers.

ERS_WRITE

The E-register state code write (ERS_WRITE) E-register command signals the support circuitry to transfer the ERS codes for 32 contiguous E registers from the microprocessor to 32 contiguous destination E registers (refer to Figure 20).

Figure 20. ERS_WRITE Command



The write buffer in the microprocessor may merge two, three, or four ERS_WRITE commands into a single operation. The merged ERS_WRITE operation transfers the ERS code for 128 contiguous destination E registers from the microprocessor to the destination E registers.

ERS_WRITE Address

The microprocessor issues an ERS_WRITE command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the ERS_WRITE address arguments on the microprocessor address bus and data bus.

NOTE: Bits <7 : 0> of the microprocessor-generated address reference a block of 32 destination E registers.

ERS_WRITE Data Format

The microprocessor data bus contains the ERS codes for 32 contiguous destination E registers. For example, bits <7 : 6> of the data bus contain the ERS code for the fourth E register in a block of 32 destination E registers.

ERS_WRITE ERS Codes

When the support circuitry performs a ERS_WRITE command, the ERS codes for the destination E registers are changed to the values received from the microprocessor. The support circuitry does not change the ERS codes until all outstanding E-register references are complete.

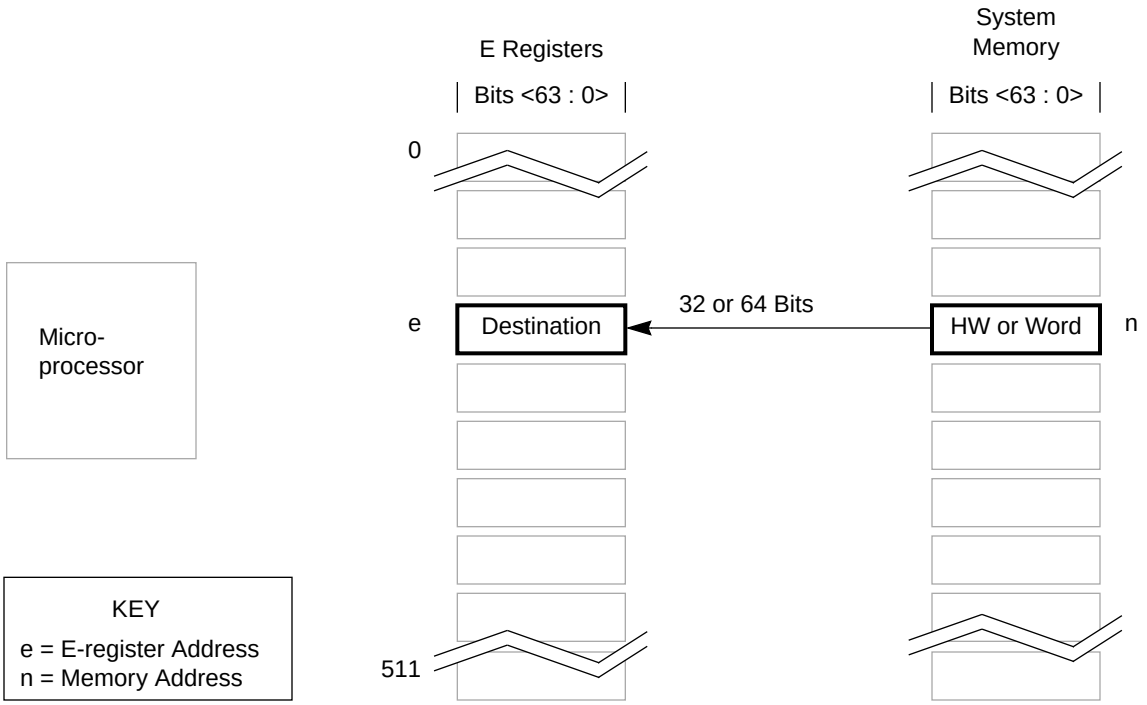
A nonprivileged user cannot set the ERS codes for the destination E registers to EMPTY. If a nonprivileged user attempts to do this, en error occurs.

The support circuitry does not complete an ERS_WRITE operation if the EREG_PENDING state is non-zero for the specified context. The support circuitry stalls the ERS_WRITE operation until all outstanding GET- or PUT-related commands for the specified context have completed. To avoid a stall or hung condition, software must ensure that all outstanding GET- or PUT-related commands are complete and then reset the EREG_PENDING counters (if they are not zero) before issuing the ERS_WRITE command.

GET

The get (GET) E-register commands signal the support circuitry to transfer data from a location in system memory to a destination E register (refer to Figure 21).

Figure 21. GET Commands



There are two GET commands: GET4 and GET8. The GET4 command transfers a 32-bit (4-byte) halfword (HW) of data. The GET8 command transfers a 64-bit (8-byte) word of data.

The write buffer in the microprocessor may merge two, three, or four GET commands into a single operation. The merged GET operation transfers up to four 64-bit words of data from any four locations in system memory to four contiguous destination E registers.

In addition to transferring data from system memory to the E registers, the GET command may also transfer data from a remote hardware register to a destination E register.

GET Address

The microprocessor issues a GET command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the GET address arguments on the microprocessor address bus and data bus. Before requesting a GET command, software must load the MO block of E registers with the centrifuge mask and the base virtual address (BVA) or a base remote register address.

When bit 63 of the E register that contains the BVA is set to 1, the base virtual address is actually a base remote register address. The final register address is created by adding the base remote register address and the index offset.

NOTE: If a parameter in the MO block is not used, the parameter may be set to any value. For example, if the centrifuge mask is not used, it may be set to any value.

GET ERS Codes

If the ERS codes are set to EMPTY for the E registers in the MO that contain the centrifuge mask and BVA, the support circuitry stalls before performing the GET command. When the ERS codes for these E registers are set to FULL, the support circuitry sets the ERS code for the destination E register to EMPTY and performs the GET command. After storing the value read from memory into the destination E register, the support circuitry sets the ERS code for the destination E register to FULL.

NOTE: If the ERS code for any of the E registers listed above is set to FULL_F or FULL_SR, an error occurs.

GET Destination E-register Format

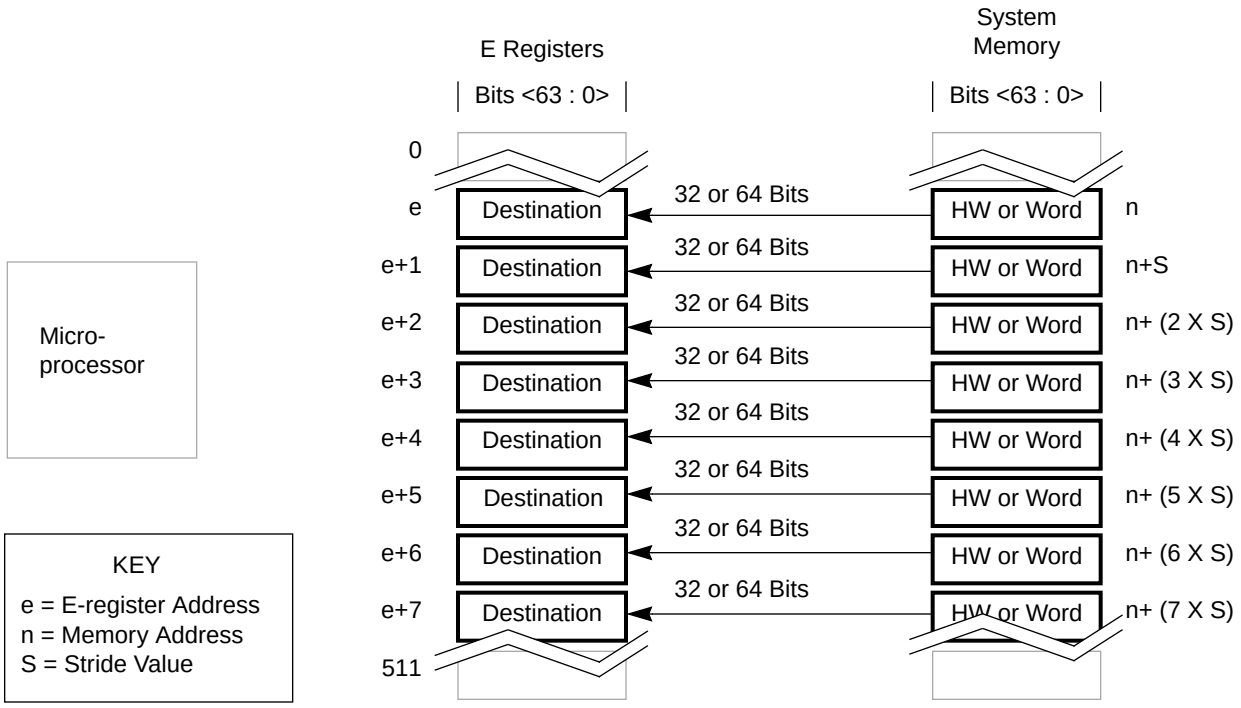
After the support circuitry performs a GET8 command, the destination E register contains the 64-bit word of data read from memory. After the support circuitry performs a GET4 command, bits <63 : 32> of the destination E register contain the 32-bit halfword read from memory and bits <31 : 0> are read as 0's.

NOTE: The GET8 command is issued when the data size bit (S) of the opcode is set to 0. The GET4 command is issued when the data size bit of the opcode is set to 1.

GETV

The get vector (GETV) E-register commands signal the support circuitry to transfer data from eight locations in system memory to eight contiguous destination E registers (refer to Figure 22). The support circuitry references each of the eight locations in system memory by adding a stride value to a base memory address.

Figure 22. GETV Commands



There are two GETV commands: GETV4 and GETV8. The GETV4 command transfers eight 32-bit (4-byte) halfwords (HW) of data. The GETV8 command transfers eight 64-bit (8-byte) words of data.

The write buffer in the microprocessor may merge two, three, or four GETV commands into a single operation. The merged GETV operation transfers up to 32 64-bit words of data from system memory to 32 contiguous destination E registers. The base system memory addresses and the stride values for each of the GETV commands in the merged operation may differ for each GETV command.

NOTE: The GETV command transfers data from system memory to destination E registers. The GETV command does not transfer data from remote registers to E registers.

GETV Address

The microprocessor issues a GETV command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the GETV address arguments on the microprocessor address bus and data bus. Before requesting a GETV command, software must load the MO block of E registers with the centrifuge mask, the BVA, and a stride.

The stride is a positive or negative value. Bits <49 : 0> of the E register contain the value of the stride. Bits <63 : 50> of the E register contain the sign-extension for the stride. Software must use a two's compliment value for a negative stride.

The support circuitry does not check the value of the stride. If software sets the value of the stride to zero, the support circuitry repeatedly references the same location in system memory. If software sets the value of the stride to a value that results in an address larger than the maximum address allowed in the system, the support circuitry uses only the least-significant bits of the resulting address.

GETV ERS Codes

If the ERS codes are set to EMPTY for the E registers in the MO block that contain the centrifuge mask, base virtual address, and stride, the support circuitry stalls before performing the GETV command. When the ERS codes for these E registers are set to FULL, the support circuitry sets the ERS codes for the destination E registers to EMPTY and performs the GETV command. After storing the values read from memory into the destination E registers, the support circuitry sets the ERS codes for the E registers to FULL.

NOTE: If the ERS code for any of the E registers listed above is set to FULL_F or FULL_SR, an error occurs.

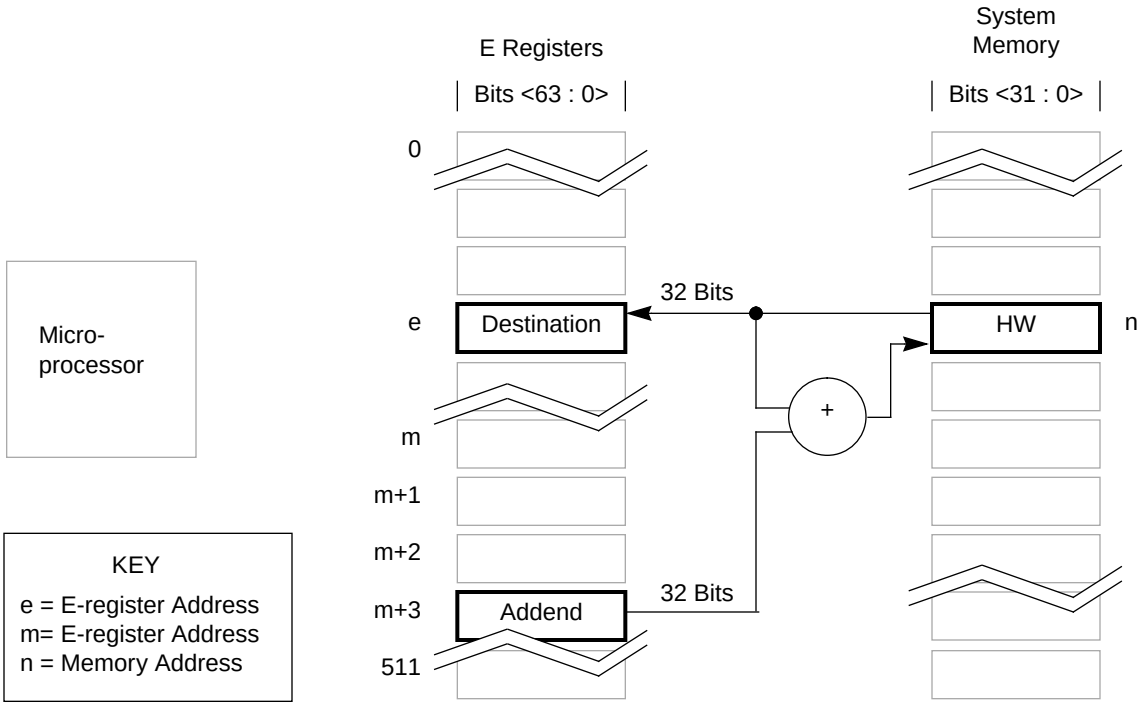
GETV Destination E-register Format

After the support circuitry performs a GETV8 command, the destination E registers contain the 64-bit words of data read from memory. After the support circuitry performs a GETV4 command, bits <63 : 32> of the destination E registers contain the 32-bit halfwords read from memory, and bits <31 : 0> of each E register are read as 0's.

GET_ADD

The get-and-add (GET_ADD) E-register command signals the support circuitry to transfer data a 32-bit halfword from system memory to a destination E register, add a 32-bit addend value to the data read from system memory, and store the sum into the original system memory location (refer to Figure 23).

Figure 23. GET_ADD Commands



The write buffer in the microprocessor may merge two, three, or four GET_ADD commands into a single operation. Each of the GET_ADD commands in the merged GET_ADD operation may use the same addend value (from the same source E register) or may use unique addend values (from different source E registers).

GET_ADD Address

The microprocessor issues a GET_ADD command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the GET_ADD address arguments on the microprocessor address bus and data bus. Before requesting a GET_ADD command, software must load the MO block of E registers with the centrifuge mask, the BVA, and an addend value.

NOTE: If a parameter in the MO block is not used, the parameter may be set to any value. For example, if the centrifuge mask is not used, it may be set to any value.

GET_ADD ERS Codes

If the ERS codes are set to EMPTY for the E registers in the MO block that contain the centrifuge mask, base virtual address, and addend, the support circuitry stalls before performing the GET_ADD command. When the ERS codes for these E registers are set to FULL, the support circuitry sets the ERS code for the destination E register to EMPTY and performs the GET_ADD command. After storing the value read from memory into the destination E register, the support circuitry sets the ERS code for the destination E register to FULL.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

GET_ADD Destination E-register Format

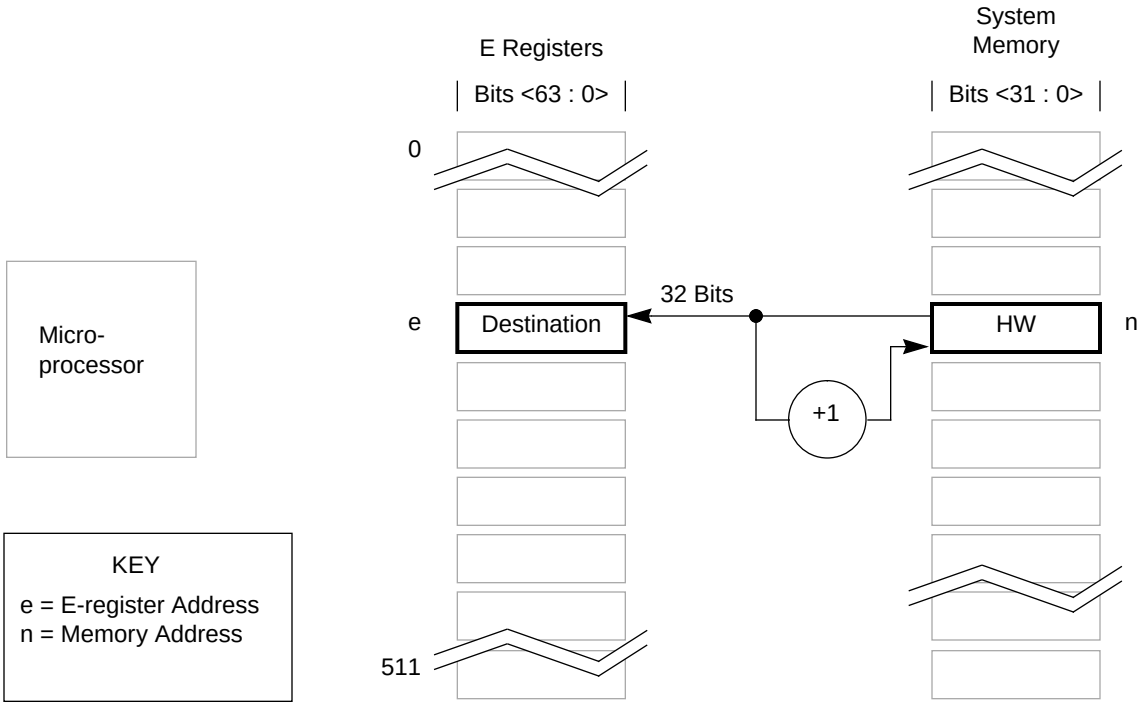
After the support circuitry performs a GET_ADD command, bits <63 : 32> of the destination E register contain the 32-bit halfword read from memory, and bits <31 : 0> of the E register are read as 0's.

NOTE: The GET_ADD command must be issued with the data size bit of the opcode set to 1.

GET_INC

The get-and-increment (GET_INC) E-register command signals the support circuitry to transfer a 32-bit halfword of data from system memory to a destination E register, add 1 to the data, and then store the sum into the original system memory location (refer to Figure 24).

Figure 24. GET_INC Command



The write buffer in the microprocessor may merge two, three, or four GET_INC commands into a single operation. The merged GET_INC operation transfers and adds as many as four 32-bit halfwords of system data.

GET_INC Address

The microprocessor issues a GET_INC command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the GET_INC address arguments on the microprocessor address bus and data bus. Before requesting a GET_INC command, software must load the MO block of E registers with the centrifuge mask and the base virtual address.

NOTE: If a parameter in the MO block is not used, the parameter may be set to any value. For example, if the centrifuge mask is not used, it may be set to any value.

GET_INC ERS Codes

If the ERS codes are set to EMPTY for the E registers in the MO block that contain the centrifuge mask and base virtual address, the support circuitry stalls before performing the GET_INC command. When the ERS codes for these E registers are set to FULL, the support circuitry sets the ERS code for the destination E register to EMPTY and performs the GET_INC command. After storing the value read from memory into the destination E register, the support circuitry sets the ERS code for the destination E register to FULL.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

GET_INC Destination E-register Format

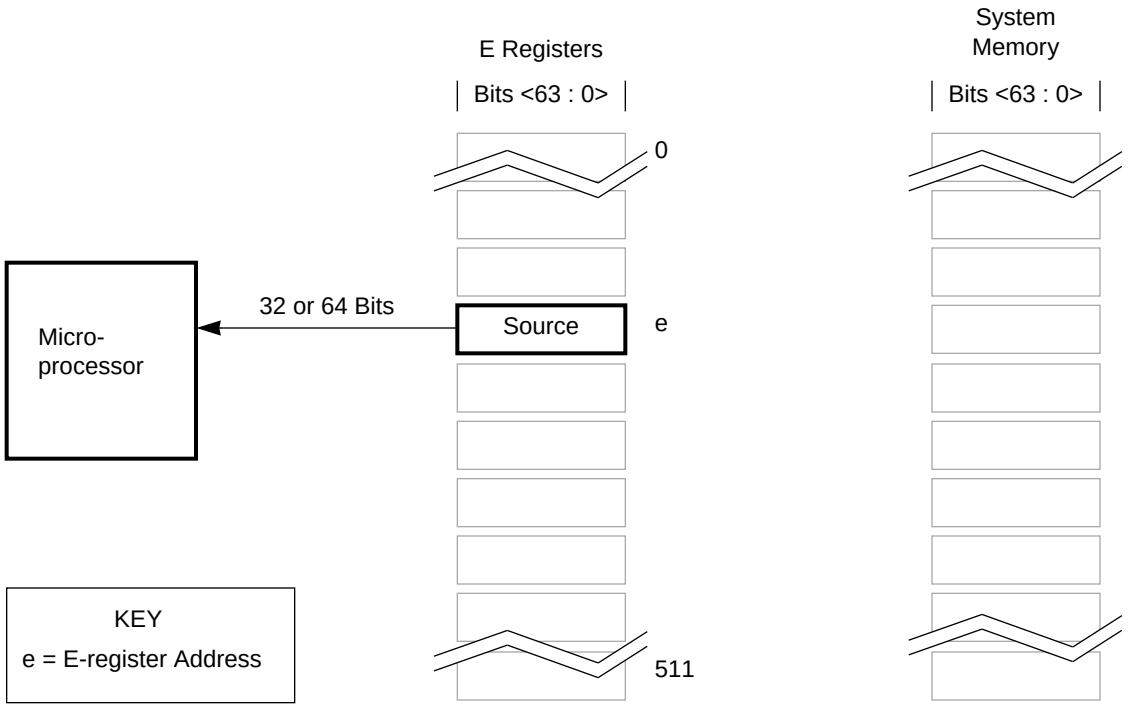
After the support circuitry performs a GET_INC command, bits <63 : 32> of the destination E register contain the 32-bit halfword read from memory and bits <31 : 0> of the E register are read as 0's.

NOTE: The GET_INC command must be issued with the data size bit of the opcode set to 1.

LOAD

The load (LOAD) E-register commands signal the support circuitry to transfer system data from a source E register to the microprocessor (refer to Figure 25).

Figure 25. LOAD Commands



There are two LOAD commands: LOAD4 and LOAD8. The LOAD4 command transfers a 32-bit (4-byte) halfword of data from a source E register to the microprocessor. The LOAD8 command transfers a 64-bit (8-byte) word of data from a source E register to the microprocessor.

The missed-address file in the microprocessor may merge two, three, or four LOAD commands into a single operation. The merged LOAD operation transfers up to four 64-bit words from four contiguous source E registers to the microprocessor.

LOAD Address

The microprocessor issues a LOAD command by placing a READ_MISS command on the microprocessor command pins and by placing the LOAD address arguments on the microprocessor address bus. The microprocessor data bus is not used for a LOAD command.

LOAD Source E-register Format

The source E register contains 32 or 64 bits of system data. The LOAD4 command transfers the 32 most-significant bits or the 32 least-significant bits of the source E register to the microprocessor (as indicated by the microprocessor-generated address). The LOAD8 command transfers all 64-bits of the source E register to the microprocessor.

NOTE: The LOAD8 command is issued when the microprocessor requests a load of 8 bytes of data. The LOAD4 command is issued when the microprocessor requests a load of 4 bytes of data.

LOAD ERS Codes

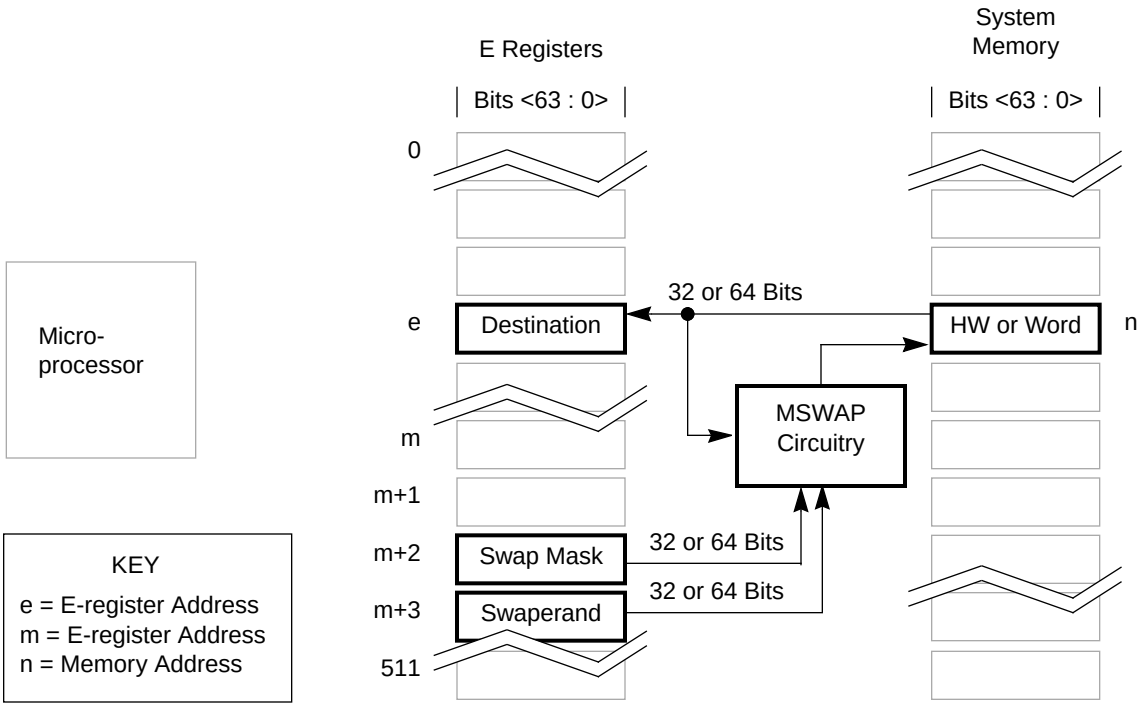
If the ERS code is set to EMPTY for the source E register, the support circuitry stalls before performing the LOAD command. When the ERS code for the source E registers is set to FULL, the support circuitry performs the LOAD command.

NOTE: If the ERS code for the E register mentioned above is set to FULL_F or FULL_SR, an error occurs.

MSWAP

The maskable atomic swap (MSWAP) E-register commands signal the support circuitry to transfer data from system memory to a destination E register and transfer specified bits of a swaperand value from another source E register into the corresponding bits of the original system memory location (refer to Figure 26). Software specifies the bits of the swaperand value that are transferred by setting the corresponding bits of the swap mask to 1.

Figure 26. MSWAP Command



There are two types of MSWAP commands: MSWAP4 and MSWAP8. The MSWAP4 command performs a maskable swap on 32-bit (4-byte) halfwords (HW) of data. The MSWAP8 command performs a maskable atomic swap on 64-bit (8-byte) words of data.

The write buffer in the microprocessor may merge two, three, or four MSWAP commands into a single operation. The merged MSWAP operation swaps up to four 64-bit words of data. Each of the MSWAP commands in the merged operation may use unique swap mask and swaperand values or use the same swap mask and swaperand values from the same source E registers.

MSWAP Functions

Software may use MSWAP commands to perform several types of functions which include an atomic swap function, a get-and-set function, or a get-and-clear function.

An atomic swap function transfers data from system memory to a destination E register and transfers a swaperand value from another source E register into the original system memory location. To issue an atomic swap function, software stores a swap mask of all 1’s into a source E register, stores the swaperand value into another source E register, and issues the MSWAP command (refer again to Figure 26).

A get-and-set function transfers data from system memory to a destination E register and then sets specified bits in the original system memory location to 1. To issue a get-and-set function, software stores a swap mask with specified bits set to 1 into another source E register, stores a swaperand value of all 1’s into a source E register, and issues the MSWAP command (refer again to Figure 26).

A get-and-clear function transfers data from system memory to a destination E register and then resets specified bits in the original system memory location to 0. To issue a get-and-clear function, software stores a swap mask with specified bits set to 1 into a source E register, stores a swaperand value of all 0’s into another source E register, and issues the MSWAP command (refer again to Figure 26).

MSWAP Address

The microprocessor issues an MSWAP command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the MSWAP address arguments on the microprocessor address bus and data bus. Before requesting an MSWAP command, software must load the MO block of E registers with the centrifuge mask, the base virtual address, the swap mask, and the swaperand value.

MSWAP ERS Codes

If the ERS codes are set to EMPTY for the E registers in the MO block that contain the centrifuge mask, base virtual address, swap mask, and swaperand value, the support circuitry stalls before performing the MSWAP command. When the ERS codes for these E registers are set to FULL, the support circuitry sets the ERS code for the destination E register to EMPTY and performs the MSWAP command. After storing the value read from memory into the destination E register, the support circuitry sets the ERS code for the destination E register to FULL.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

MSWAP Destination E-register Format

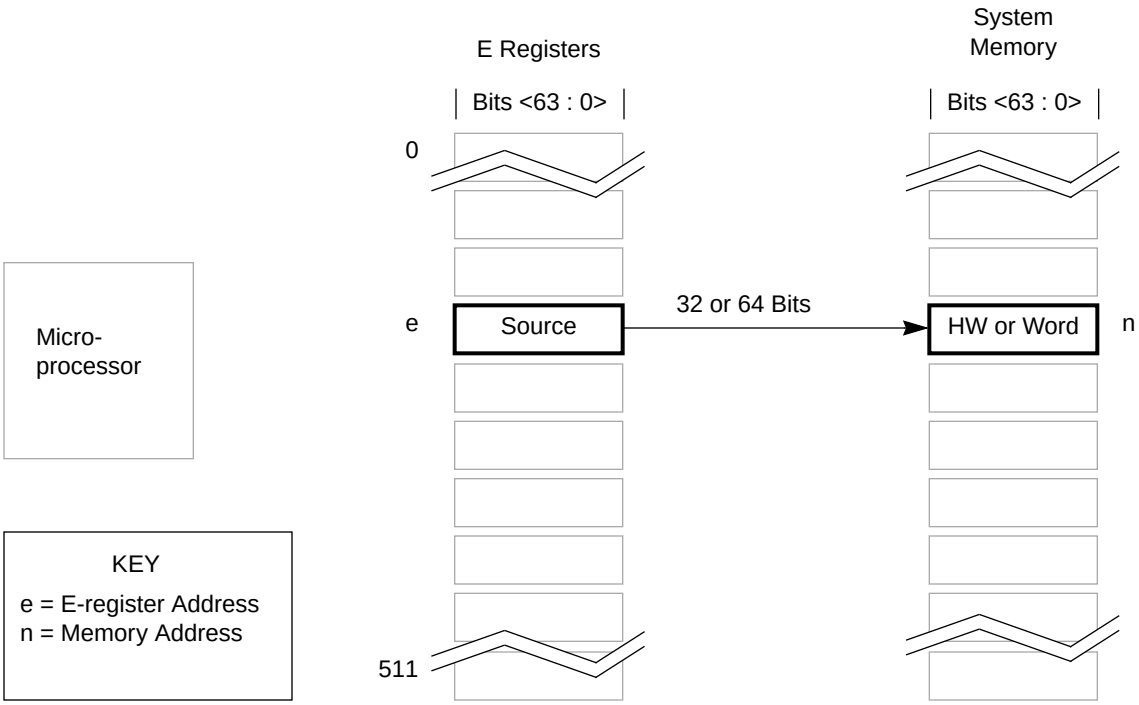
After the support circuitry performs an MSWAP8 command, the destination E register contains the 64-bit word of data read from memory. After the support circuitry performs an MSWAP4 command, bits <63 : 32> of the destination E register contain the 32-bit halfword read from memory and bits <31 : 0> are read as 0’s.

NOTE: The MSWAP8 command is issued when the data size bit (S) of the opcode is set to 0. The MSWAP4 command is issued when the data size bit of the opcode is set to 1.

PUT

The put (PUT) E-register commands signal the support circuitry to transfer data from a source E register to a location in system memory (refer to Figure 27).

Figure 27. PUT Command



There are two PUT commands: PUT4 and PUT8. The PUT4 command transfers a 32-bit (4-byte) halfword (HW) of data. The PUT8 command transfers a 64-bit (8-byte) word of data.

The write buffer in the microprocessor may merge two, three, or four PUT commands into a single operation. The merged PUT operation transfers up to four 64-bit words of data from four contiguous source E registers to any four locations in system memory.

In addition to transferring data from an E register to system memory, the PUT command may also transfer data from an E register to a remote hardware register.

PUT Address

The microprocessor issues a PUT command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the PUT address arguments on the microprocessor address bus and data bus. Before requesting a PUT command, software must load the MO block of E registers with the centrifuge mask and either the base virtual address (BVA) or a base remote register address.

When bit 63 of the E register that contains the BVA is set to 1, the BVA is actually a base remote register address. The final register address is created by adding the base remote register address and the index offset.

NOTE: If a parameter in the MO block is not used, the parameter may be set to any value. For example, if the centrifuge mask is not used, it may be set to any value.

PUT Source E-register Format

The source E register contains 32 or 64 bits of system data. The PUT4 command transfers the 32 most-significant bits of the source E register to a location in system memory. The PUT8 command transfers all 64-bits of the source E register to a location in system memory.

NOTE: The PUT8 command is issued when the data size bit (S) of the opcode is set to 0. The PUT4 command is issued when the data size bit of the opcode is set to 1.

PUT ERS Codes

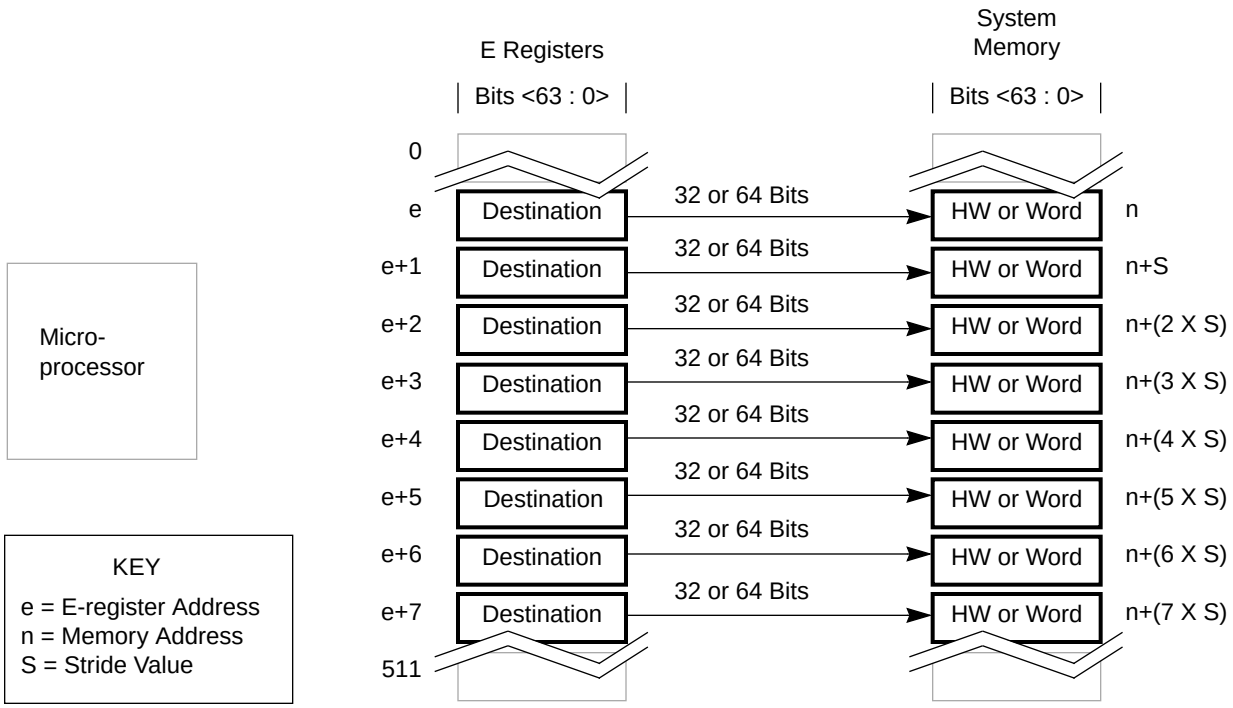
If the ERS codes are set to EMPTY for the source E register or the E registers in the MO block that contain the centrifuge mask and BVA, the support circuitry stalls before performing the PUT command. When the ERS codes for these E registers are set to FULL, the support circuitry performs the PUT command.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

PUTV

The put vector (PUTV) E-register commands signal the support circuitry to transfer data from eight contiguous E registers to eight locations in system memory (refer to Figure 28). Each of the eight locations in system memory are referenced by adding a stride value to a base memory address.

Figure 28. PUTV Command



There are two PUTV commands: PUTV4 and PUTV8. The PUTV4 command transfers eight 32-bit (4-byte) halfwords (HW) of data. The PUTV8 command transfers eight 64-bit (8-byte) words of data.

The write buffer in the microprocessor may merge two, three, or four PUTV commands into a single operation. The merged PUTV operation transfers up to 32 64-bit words of data from 32 contiguous source E registers to 32 locations in system memory. The base system memory addresses and the stride values for each of the PUTV commands in the merged operation may differ for each PUTV command.

NOTE: The PUTV command transfers data from a source E register to system memory. The PUTV command does not transfer data from source E registers to remote registers.

PUTV Address

The microprocessor issues a PUTV command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the PUTV address arguments on the microprocessor address bus and data bus. Before requesting a PUTV command, software must load the MO block of E registers with the centrifuge mask, the base virtual address (BVA), and a stride.

The stride is a positive or negative value. Bits <49 : 0> of the E register contain the value of the stride. Bits <63 : 50> of the E register contain the sign-extension for the stride. Software must use a two's compliment value for a negative stride.

The support circuitry does not check the value of the stride. If software sets the value of the stride to zero, the support circuitry repeatedly references the same location in system memory. If software sets the value of the stride to a value that results in an address larger than the maximum address allowed in the system, the support circuitry uses only the least-significant bits of the resulting address. Because of this characteristic, the support circuitry references an actual location in memory instead of indicating that an error occurred.

NOTE: If a parameter in the MO block is not used, the parameter may be set to any value. For example, if the centrifuge mask is not used, it may be set to any value.

PUTV Source E-register Format

Each source E register contains 32 or 64 bits of system data. The PUTV4 command transfers the 32 most-significant bits of the source E register to a 32-bit location in system memory. The PUTV8 command transfers all 64-bits of the source E register to a location in system memory.

PUTV ERS Codes

If the ERS codes are set to EMPTY for the source E registers or the E registers in the MO block that contain the centrifuge mask, BVA, and stride, the support circuitry stalls before performing the PUTV command. When the ERS codes for these E registers are set to FULL, the support circuitry performs the PUTV command.

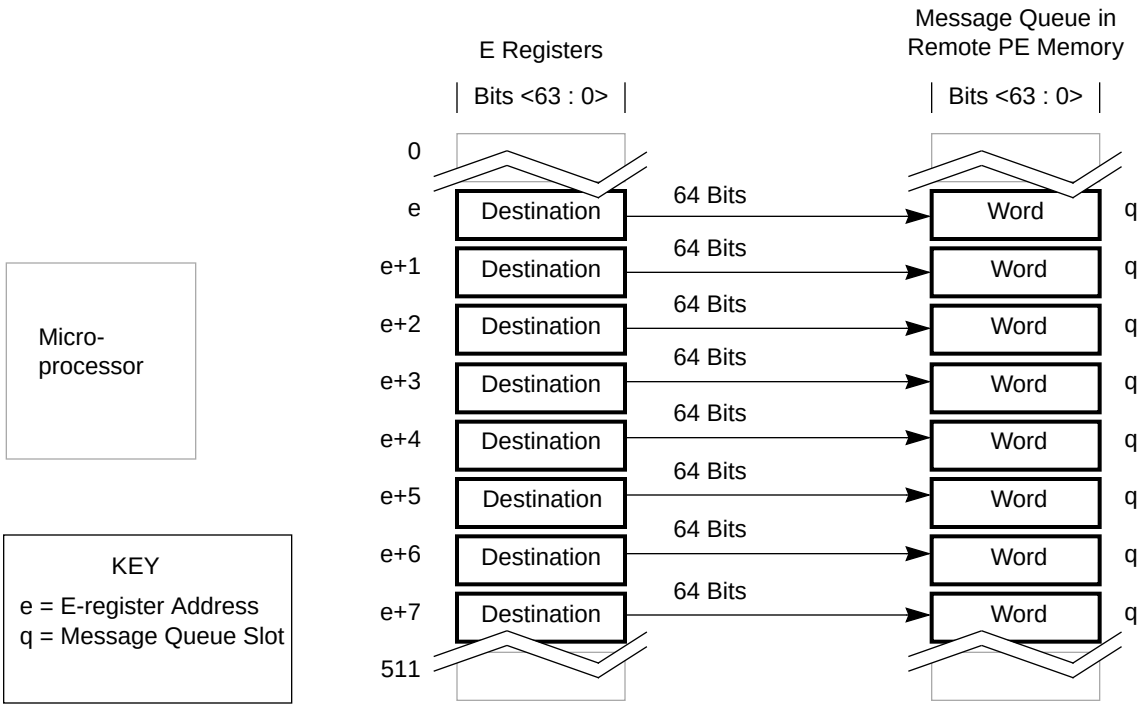
NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

SEND

The message send (SEND) E-register command signals the support circuitry to transfer a 64-byte (8-word) message from eight contiguous source E registers to a 64-byte slot in a remote PE message queue (refer to Figure 29). A message queue is a software-defined region in the memory of a PE. The maximum size of a message queue is 64 Mbytes, and a PE may have more than one message queue.

The support circuitry in the remote PE sends an interrupt to the microprocessor when a specified number (or more than the specified number) of messages have been stored in a message queue (the specified number is a software-defined value). The microprocessor in the remote PE then reads the messages from the message queue.

Figure 29. SEND Command



The write buffer in the microprocessor may merge two, three, or four SEND commands into a single operation. The merged SEND operation transfers up to four 64-byte messages from 32 contiguous source E registers to four 64-byte slots in the same or different message queues.

Software may also use the SEND command to access remote hardware registers.

SEND Address

The microprocessor issues a SEND command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the SEND address arguments on the microprocessor address bus and data bus. Before requesting a SEND command, software must load the MO block of E registers with the centrifuge mask and either the base virtual address (BVA) or a base remote register address.

When bit 63 of the E register that contains the BVA is set to 0, the sum of the BVA and the index offset must be a valid message queue control word (MQCW) register address.

When bit 63 of the E register that contains the BVA is set to 1, the sum of the BVA and the index offset must be a register in the I/O controller.

SEND Source E-register Format

Each source E register contains a 64-bit word that is part of the 64-byte message.

SEND ERS Codes

If the ERS codes are set to EMPTY for any of the source E registers or the E registers in the MO block that contain the centrifuge mask and BVA, the support circuitry stalls before performing the SEND command. When the ERS codes for these E registers are set to FULL, the support circuitry performs the SEND command.

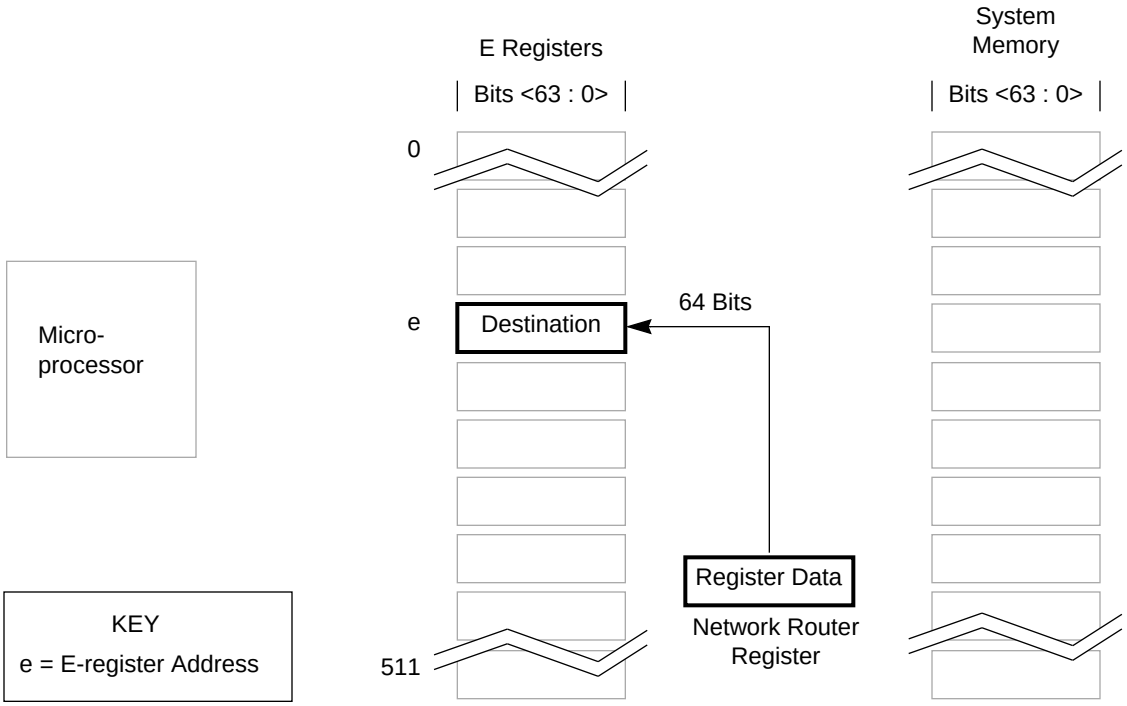
NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

More information on messaging is provided in the *CRAY T3E System Messaging* document.

SGET

The special get (SGET) E-register command signals the support circuitry to transfer a 64-bit word of data from a register in any network router (called the special registers) to a destination E register (refer to Figure 30). Network packets created by the SGET command travel through a nonblocking network.

Figure 30. SGET Command



The write buffer in the microprocessor may merge two, three, or four SGET commands into a single operation. The merged SGET operation transfers up to four 64-bit words of data from any four network router registers to four contiguous destination E registers.

SGET Address

The microprocessor issues an SGET command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the SGET address arguments on the microprocessor address bus and data bus.

SGET ERS Codes

When the support circuitry performs an SGET command, the support circuitry sets the ERS code for the destination E register to EMPTY. (If the ERS code for the destination E register was set to EMPTY, an error occurs). After storing the value read from memory into the destination E register, the support circuitry sets the ERS code for the destination E register to FULL.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

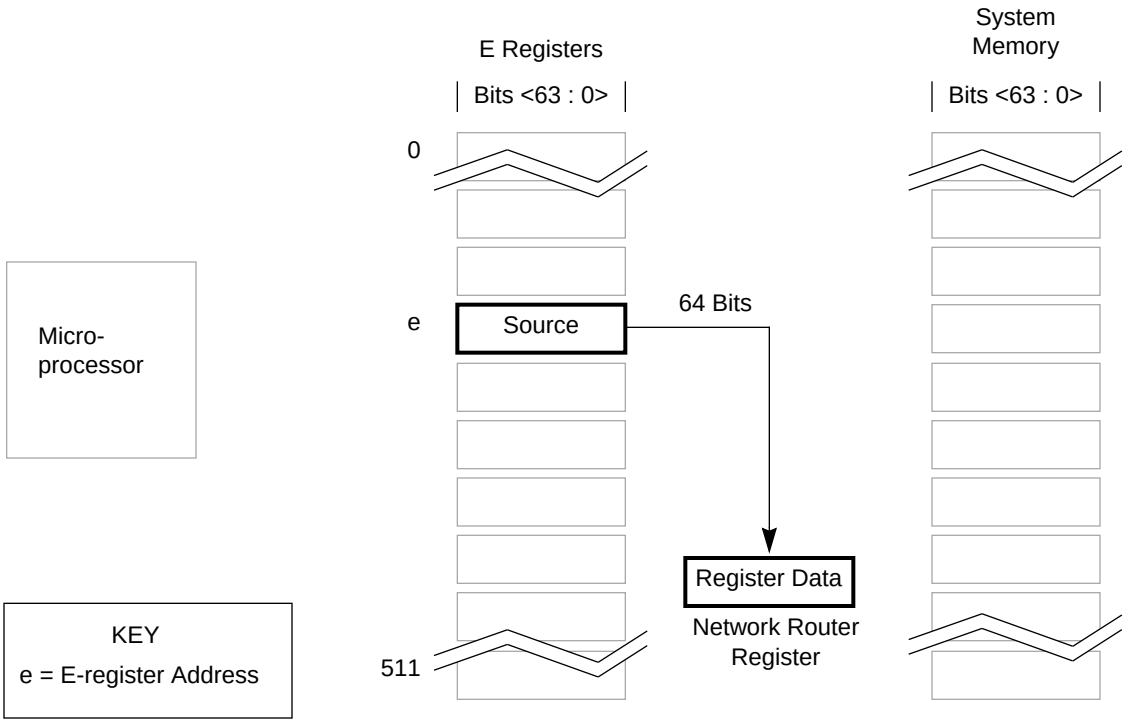
SGET Destination E-register Format

After the support circuitry performs an SGET command, the destination E register contains the 64-bit word of data read from a network router register.

SPUT

The special put (SPUT) E-register command signals the support circuitry to transfer a 64-bit word of data from a source E register to a register in any network router (called the special registers). Refer to Figure 31. Network packets created by the SPUT command travel through a nonblocking network. Because of this characteristic, software uses the SPUT command to initialize network routing information when the CRAY T3E system is deadstarted.

Figure 31. SPUT Command



The write buffer in the microprocessor may merge two, three, or four SPUT commands into a single operation. The merged SPUT operation transfers up to four 64-bit words of data from four contiguous E registers to any four network router registers in the system.

SPUT Address

The microprocessor issues an SPUT command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the SPUT address arguments on the microprocessor address bus and data bus.

SPUT Source E-register Format

The source E register contains a 64-bit word of data that will be stored in the network router register.

SPUT ERS Codes

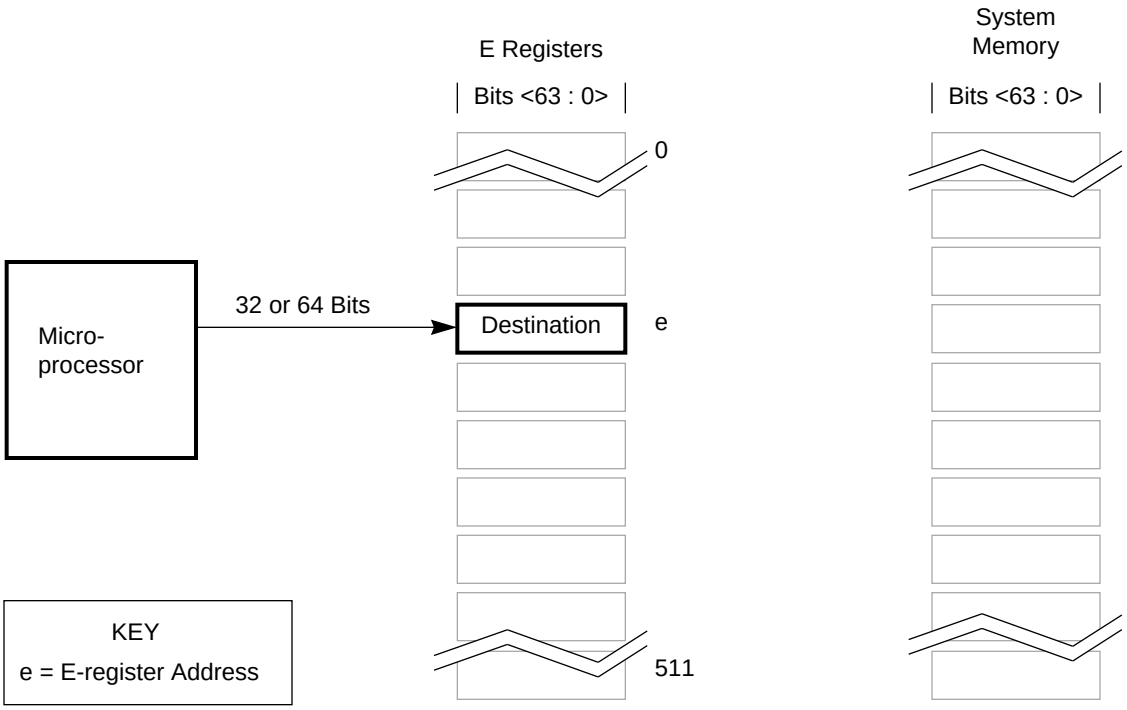
If the ERS code is set to EMPTY for the source E register, the support circuitry stalls before performing the SPUT command. When the ERS code for the source E register is set to FULL, the support circuitry performs the SPUT command.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

STORE

The store (STORE) E-register commands signal the support circuitry to transfer system data from the microprocessor to a destination E register (refer to Figure 32).

Figure 32. STORE Commands



There are two STORE commands: STORE4 and STORE8. The STORE4 command transfers a 32-bit halfword from the microprocessor to the 32 most-significant or 32 least-significant bits of the destination E register (as indicated by the microprocessor-generated address). The STORE8 command transfers a 64-bit word from the microprocessor to the destination E register.

The write buffer in the microprocessor may merge two, three, or four STORE commands into a single operation. The merged STORE operation transfers up to four 64-bit words from the microprocessor to four contiguous destination E registers.

STORE Address

The microprocessor issues a STORE command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the STORE address arguments on the microprocessor address bus and data bus.

STORE ERS Codes

If the ERS code is set to EMPTY for the destination E register, the support circuitry stalls before performing the STORE command. In this case, the EMPTY ERS code indicates that the support circuitry is performing a separate E-register command that will write data into the destination E register.

When the ERS code for the destination E register is set to FULL, the support circuitry sets the ERS code for the destination E register to EMPTY, retrieves and stores the data into the destination E register, and then sets the ERS code for the destination E register to FULL.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

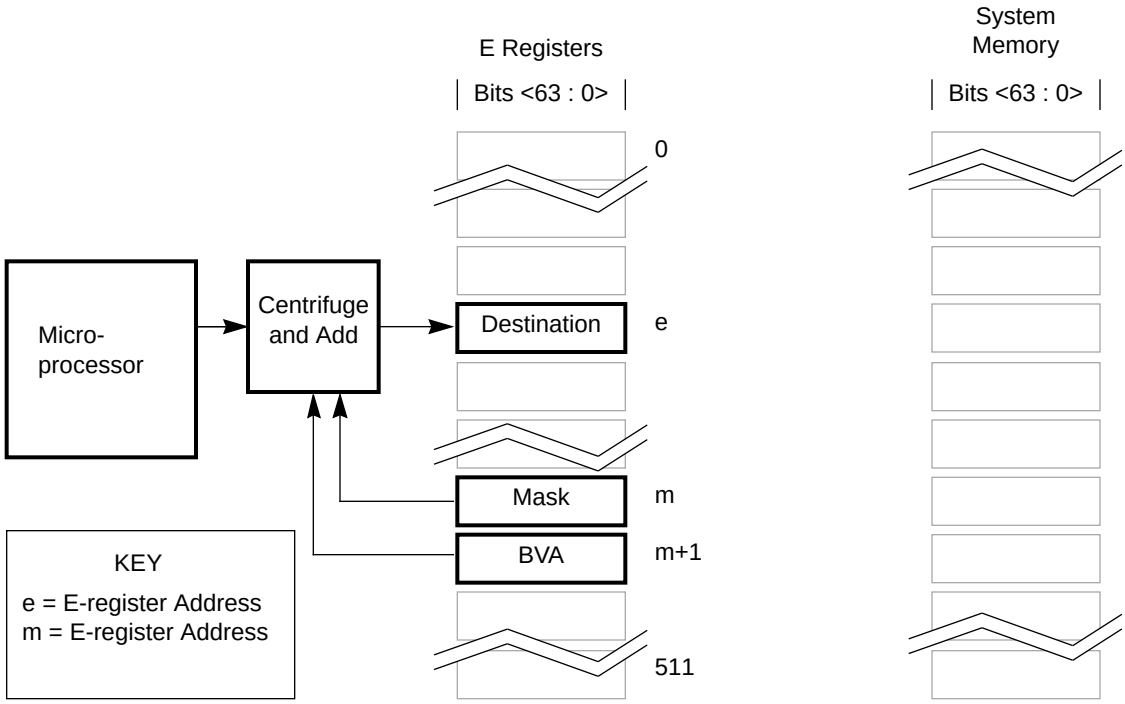
STORE Destination E Register Format

The destination E register contains the 32-bit halfword or 64-bit word of data from the microprocessor. When the data is a 32-bit halfword, the microprocessor-generated address indicates whether the halfword is the most-significant or least-significant bits of the E register.

STORE_CENT

The store centrifuge result (STORE_CENT) E-register command signals the support circuitry to separate the PE number from a global index (using the centrifuge mask), add the index offset to the base virtual address (BVA), and store the PE number and sum in a destination E register (refer to Figure 33).

Figure 33. STORE_CENT Command



The write buffer in the microprocessor may merge two, three, or four STORE_CENT commands into a single operation. The merged STORE_CENT operation separates up to four global indexes into PE numbers and global virtual addresses.

STORE_CENT Address

The microprocessor issues a STORE_CENT command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the STORE_CENT address arguments on the microprocessor address bus and data bus. Before requesting a STORE_CENT command, software must load the MO block of E registers with the centrifuge mask and the BVA.

STORE_CENT ERS Codes

If the ERS codes are set to EMPTY for the E registers that contain the centrifuge mask and BVA, the support circuitry stalls before performing the STORE_CENT command. Also, if the ERS code for the destination E register is set to EMPTY, the support circuitry stalls before performing the STORE_CENT command. In this case, the EMPTY ERS code indicates that the support circuitry is in the process of performing a separate E-register command that will write data into the destination E register.

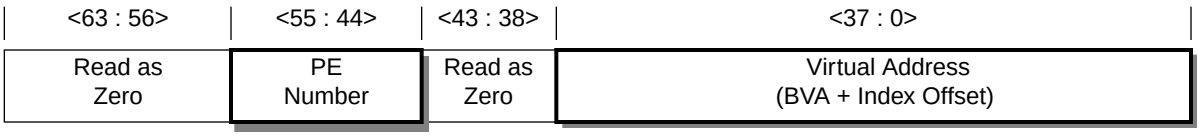
When the ERS codes for these E registers are set to FULL, the support circuitry sets the ERS code for the destination E register to EMPTY and performs the STORE_CENT command. After storing the STORE_CENT output into the destination E register, the support circuitry sets the ERS code for the destination E register to FULL.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

STORE_CENT Destination E-register Format

Figure 34 shows the format of the destination E register after the support circuitry completes a STORE_CENT command.

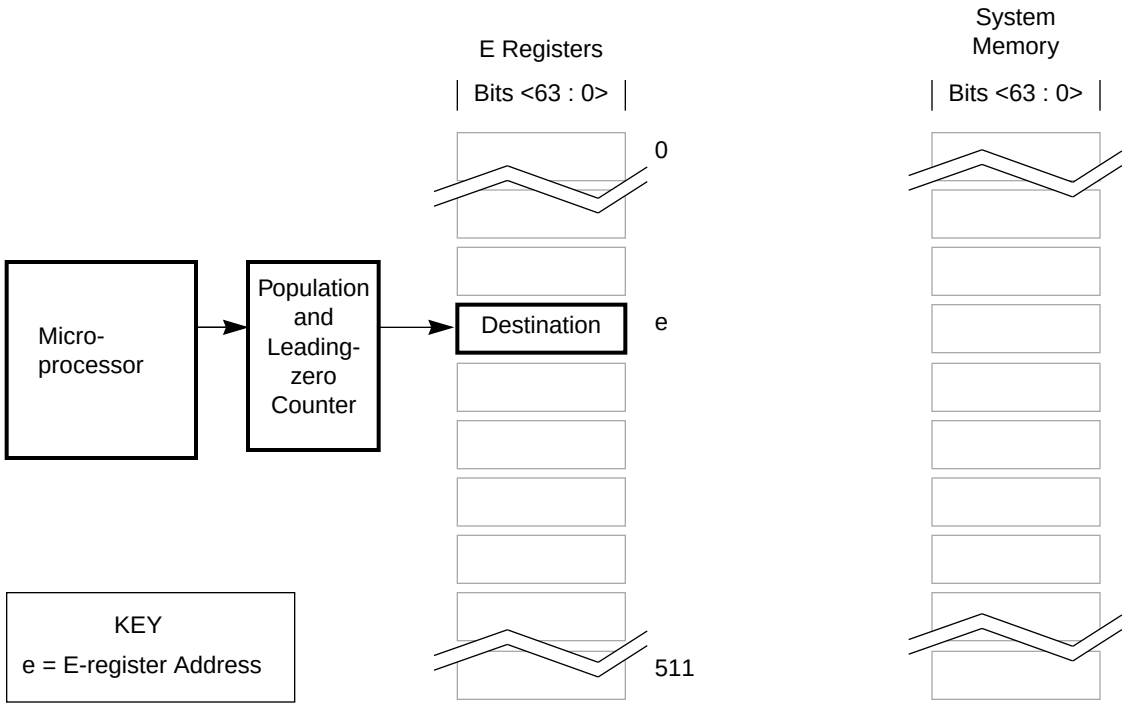
Figure 34. STORE_CENT Data in Destination E Register



STORE_PLZ

The store population and leading-zero count (STORE_PLZ) E-register command signals the support circuitry to perform a population and leading-zero count on data received from the microprocessor and to store the result into a destination E register (refer to Figure 35). The population count is the number of bits set to 1 in the data. The leading-zero count is the number of bits set to 0 from the most-significant bit of the data to the first bit that is set to 1.

Figure 35. STORE_PLZ Command



There are two STORE_PLZ commands: STORE_PLZ4 and STORE_PLZ8. The STORE_PLZ4 performs a PLZ command on a 32-bit halfword of data. The STORE_PLZ8 performs a PLZ command on a 64-bit word of data.

The write buffer in the microprocessor may merge two, three, or four STORE_PLZ commands into a single operation. The merged STORE_PLZ operation performs population and leading-zero counts on up to four 64-bit words of data.

STORE_PLZ Address

The microprocessor issues a STORE_PLZ command by placing a WRITE_BLOCK command on the microprocessor command pins and by placing the STORE_PLZ address arguments on the microprocessor address bus and data bus.

STORE_PLZ ERS Codes

If the ERS code is set to EMPTY for the destination E register, the support circuitry stalls before performing the STORE_PLZ command. In this case, the EMPTY ERS code indicates that the support circuitry is in the process of performing a separate E-register command that will write data into the destination E register.

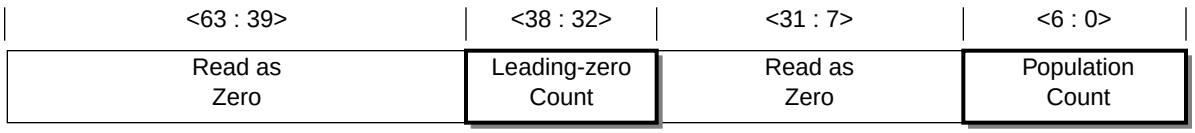
When the ERS code for the destination E register is set to FULL, the support circuitry sets the ERS code for the destination E register to EMPTY, does a population and leading-zero count on the data, stores the data into the destination E register, and then sets the ERS code for the destination E register to FULL.

NOTE: If the ERS code for any of the E registers mentioned above is set to FULL_F or FULL_SR, an error occurs.

STORE_PLZ Destination E-register Format

Figure 36 shows the format of the destination E register after the support circuitry completes a STORE_PLZ command.

Figure 36. STORE_PLZ Data in Destination E Register



Index

I

L

M

O

P

R

S

U

W