Memory Manager

Component Design Document

1 Description

The component manages access to a single memory location through a single pointer. When requested, the component loans out access to the pointer if it is available. The length of the pointer will always be the entire length of the memory region. The component will reject any requests to access the pointer again until the pointer is returned from the requestor. Request/release memory transactions are each provided a unique ID. To release the memory, the same ID must be provided that was issues upon request. This mechanism reduces the risk of an inadvertant call to release from causing an unintended release of the memory. The component includes a data product relating whether the memory is currently allocated or not. The component responds to commands to CRC, dump, write, and force-release the memory region. Note that this component is active only to provide a seperate thread of execution on which to execute the CRC command and the memory write command, each which could take a long time to execute.

2 Requirements

The requirements for the Memory Manager component are specified below.

1. The component shall manage access to a single contiguous memory region specified during initialization.
2. The component shall ensure that only one component has access to the memory region at any given time.
3. The component shall include a command to dump parts or all of the memory region.
4. The component shall include a command to CRC parts or all of the memory region.
5. The component shall include a command to write to the memory region.
6. The component shall report the last calculated CRC as a data product.
7. The component shall report the Available/In Use state of the memory as a data product.
8. The component shall include a command to free the memory region for access even if the memory region is currently in use.

3 Design

3.1 At a Glance

Below is a list of useful parameters and statistics that give a quick look into the makeup of the component.

- **Execution** - *active*
- **Number of Connectors** - 8

- **Number of Invokee Connectors** - 3
- **Number of Invoker Connectors** - 5
- **Number of Generic Connectors** - *None*
- **Number of Generic Types** - *None*
- **Number of Unconstrained Arrayed Connectors** - *None*
- **Number of Commands** - 5
- **Number of Parameters** - *None*
- **Number of Events** - 12
- **Number of Faults** - *None*
- **Number of Data Products** - 3
- **Number of Data Dependencies** - *None*
- **Number of Packets** - 1

3.2 Diagram

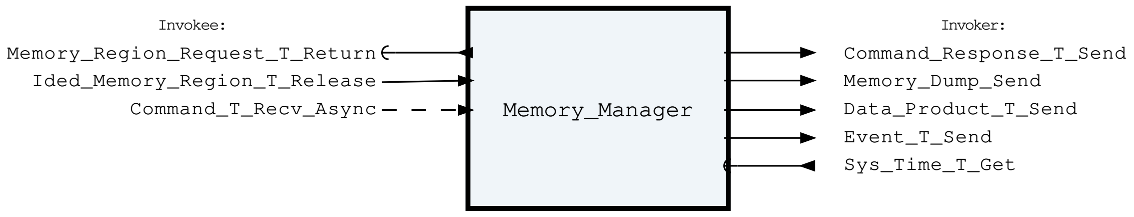

Figure 1: Memory Manager component diagram.

3.3 Connectors

Below are tables listing the component's connectors.

3.3.1 Invokee Connectors

The following is a list of the component's *invokee* connectors:

Table 1: Memory Manager Invokee Connectors

Name	Kind	Type	Return_Type	Count
Memory_Region_Request_T_Return	return	-	Memory_Region_Request.T	1
Ided_Memory_Region_T_Release	recv_sync	Ided_Memory_Region.T	-	1
Command_T_Recv_Async	recv_async	Command.T	-	1

Connector Descriptions:

- **`Memory_Region_Request_T_Return`** - The memory region is requested on this connector.
- **`Ided_Memory_Region_T_Release`** - The memory region is released (returned) on this connector.

- **Command_T_Recv_Async** - This is the command receive connector.

3.3.2 Internal Queue

This component contains an internal first-in-first-out (FIFO) queue to handle asynchronous messages. This queue is sized at initialization as a configurable number of bytes. Determining the size of the component queue can be difficult. The following table lists the connectors that will put asynchronous messages onto the queue, and the maximum sizes of each of those messages on the queue. Note that each message put onto the queue also incurs an overhead on the queue of 5 additional bytes, which is included in the max message size below:

Table 2: Memory Manager Asynchronous Connectors

Name	Type	Max Size (bytes)
Command_T_Recv_Async	Command.T	106

If you are unsure how to size the queue of this component, it is recommended that you make the queue size a multiple of the largest size found above.

3.3.3 Invoker Connectors

The following is a list of the component's *invoker* connectors:

Table 3: Memory Manager Invoker Connectors

Name	Kind	Type	Return_Type	Count
Command_Response_T_Send	send	Command_Response.T	-	1
Memory_Dump_Send	send	Memory_Packetizer_Types.Memory_Dump	-	1
Data_Product_T_Send	send	Data_Product.T	-	1
Event_T_Send	send	Event.T	-	1
Sys_Time_T_Get	get	-	Sys_Time.T	1

Connector Descriptions:

- **Command_Response_T_Send** - This connector is used to register and respond to the component's commands.
- **Memory_Dump_Send** - The memory dump connector.
- **Data_Product_T_Send** - Data products are sent out of this connector.
- **Event_T_Send** - Events are sent out of this connector.
- **Sys_Time_T_Get** - The system time is retrieved via this connector.

3.4 Interrupts

This component contains no interrupts.

3.5 Initialization

Below are details on how the component should be initialized in an assembly.

3.5.1 Component Instantiation

This component contains no instantiation parameters in its discriminant.

3.5.2 Component Base Initialization

This component achieves base class initialization using the `init_Base` subprogram. This subprogram requires the following parameters:

Table 4: Memory Manager Base Initialization Parameters

Name	Type
`Queue_Size`	`Natural`

Parameter Descriptions:

- **`Queue_Size`** - The number of bytes that can be stored in the component's internal queue.

3.5.3 Component Set ID Bases

This component contains commands, events, packets, faults, or data products that require a base identifier to be set at initialization. The `set_Id_Bases` procedure must be called with the following parameters:

Table 5: Memory Manager Set Id Bases Parameters

Name	Type
`Event_Id_Base`	`Event_Types.Event_Id_Base`
`Data_Product_Id_Base`	`Data_Product_Types.Data_Product_Id_Base`
`Command_Id_Base`	`Command_Types.Command_Id_Base`
`Packet_Id_Base`	`Packet_Types.Packet_Id_Base`

Parameter Descriptions:

- **`Event_Id_Base`** - The value at which the component's event identifiers begin.
- **`Data_Product_Id_Base`** - The value at which the component's data product identifiers begin.
- **`Command_Id_Base`** - The value at which the component's command identifiers begin.
- **`Packet_Id_Base`** - The value at which the component's unresolved packet identifiers begin.

3.5.4 Component Map Data Dependencies

This component contains no data dependencies.

3.5.5 Component Implementation Initialization

The calling of this implementation class initialization procedure is mandatory. This init function provides memory allocation for the managers internal memory region. Preallocated memory can be provided via the "bytes" access type, in which case "size" must be negative and will be ignored. If you would like to allocate the internal memory on the heap then "bytes" must be set to null, and "size" must be a positive number representing the number of bytes you would like to allocate. The `init` subprogram requires the following parameters:

Table 6: Memory Manager Implementation Initialization Parameters

Name	Type	Default Value

Bytes	Basic_Types.Byte_Array_Access	null
Size	Integer	-1

Parameter Descriptions:

- **Bytes** - A pointer to an allocation of bytes to be used for the memory region. If this is set to null, then memory will be allocated on the heap using the "size" parameter instead. Note: This must be set to null if the "size" parameter is positive below.
- **Size** - The number of bytes to allocate on the heap for the memory region. Note: This must be set to a negative value if the "bytes" parameters is not null.

3.6 Commands

These are the commands for the Memory Manager component.

Table 7: Memory Manager Commands

Local ID	Command Name	Argument Type
0	Dump_Memory_Region	-
1	Dump_Memory_Region_Bytes	Virtual_Memory_Region_Positive.T
2	Crc_Memory_Region_Bytes	Virtual_Memory_Region_Positive.T
3	Write_Memory_Region	Virtual_Memory_Region_Write.T
4	Force_Release	-

Command Descriptions:

- **Dump_Memory_Region** - Dump the entire memory region.
- **Dump_Memory_Region_Bytes** - Dump the memory region at the provided virtual address and length.
- **Crc_Memory_Region_Bytes** - Perform a CRC on the region with the provided virtual address and length. The CRC will be reported via event and data product, if those connectors are connected.
- **Write_Memory_Region** - Perform a write to the memory region at the provided address. If the memory is not available an error event will be produced.
- **Force_Release** - Forces the release of the memory region if it is currently allocated. This command can be used to recover from an anomolous condition.

3.7 Parameters

The Memory Manager component has no parameters.

3.8 Events

Below is a list of the events for the Memory Manager component.

Table 8: Memory Manager Events

Local ID	Event Name	Parameter Type
0	Memory_Unavailable	-
1	Unexpected_Memory_Id	Ided_Memory_Region.T
2	Memory_Already_Released	Ided_Memory_Region.T

3	Dumping_Memory	Virtual_Memory_Region_Positive.T
4	Invalid_Memory_Region	Invalid_Virtual_Memory_Region.T
5	Crcing_Memory	Virtual_Memory_Region_Positive.T
6	Memory_Crc	Virtual_Memory_Region_Crc.T
7	Writing_Memory	Virtual_Memory_Region.T
8	Memory_Written	Virtual_Memory_Region.T
9	Memory_Force_Released	-
10	Invalid_Command_Received	Invalid_Command_Info.T
11	Dropped_Command	Command_Header.T

Event Descriptions:

- **`Memory_Unavailable`** - The memory region was requested, but the memory is currently in use.
- **`Unexpected_Memory_Id`** - Cannot release a memory region with an unexpected ID.
- **`Memory_Already_Released`** - Cannot release a memory region when the memory region is currently available (ie. already released).
- **`Dumping_Memory`** - The component is currently dumping the virtual memory location for the following region.
- **`Invalid_Memory_Region`** - The operation could not be performed on the requested virtual memory region, since the address and length fall outside the memory region managed by the component.
- **`Crcing_Memory`** - The component is currently CRCing the virtual memory location for the following region.
- **`Memory_Crc`** - The virtual memory region CRC has been calculated.
- **`Writing_Memory`** - The component is currently writing to the virtual memory location for the following region.
- **`Memory_Written`** - The virtual memory region has been written.
- **`Memory_Force_Released`** - The virtual memory region was force released.
- **`Invalid_Command_Received`** - A command was received with invalid parameters.
- **`Dropped_Command`** - A command was dropped because the component queue overflowed.

3.9 Data Products

Data products for the Memory Manager component.

Table 9: Memory Manager Data Products

Local ID	Data Product Name	Type
0x0000 (0)	Crc_Report	Virtual_Memory_Region_Crc.T
0x0001 (1)	Memory_Region_Status	Memory_Manager_State.T
0x0002 (2)	Memory_Location	Memory_Region.T

Data Product Descriptions:

- **`Crc_Report`** - The last computed CRC by the memory manager component.
- **`Memory_Region_Status`** - Status relating whether the memory region is currently allocated or not.
- **`Memory_Location`** - Reports the physical start address and length of the virtual memory

region allocated to this component.

3.10 Data Dependencies

The Memory Manager component has no data dependencies.

3.11 Packets

Packets for the Memory Manager.

Table 10: Memory Manager Packets

Local ID	**Packet Name**	**Type**
`0x0000 (0)`	`Memory_Region_Packet`	*Undefined*

Packet Descriptions:

- **`Memory_Region_Packet`** - This packet contains memory region data.

3.12 Faults

The Memory Manager component has no faults.

4 Unit Tests

The following section describes the unit test suites written to test the component.

4.1 *Memory_ Manager_ Tests* Test Suite

This is a unit test suite for the Memory Manager component

Test Descriptions:

- **`Test_Init`** - This unit test excersizes all possible combinations of initializing the component to make sure only valid initializations succeed, and others throw an assertion.
- **`Test_Nominal_Request_Release`** - This unit test excersizes the nominal request/release of the component's memory region.
- **`Test_Off_Nominal_Request_Release`** - This unit test excersizes the error conditions that can arise when requesting/releasing the component's memory region.
- **`Test_Nominal_Memory_Dump`** - This unit test excersizes the memory dump commands.
- **`Test_Nominal_Memory_Crc`** - This unit test excersizes the memory crc commands.
- **`Test_Nominal_Memory_Write`** - This unit test excersizes the memory write command.
- **`Test_Write_Unreleased_Region`** - This unit test excersizes the write command when the memory is not available.
- **`Test_Dump_Invalid_Region`** - This unit test excersizes the memory dump command with invalid memory regions.
- **`Test_Crc_Invalid_Region`** - This unit test excersizes the crc command with invalid memory regions.
- **`Test_Write_Invalid_Region`** - This unit test excersizes the write command with invalid memory regions.

- **`Test_Force_Release_Command`** - This unit test excersizes the force release command.
- **`Test_Command_Dropped`** - This unit test excersizes the behavior when the internal queue overflows.
- **`Test_Invalid_Command`** - This unit test makes sure an invalid command is reported and ignored.

5 Appendix

5.1 Preamble

This component contains no preamble code.

5.2 Packed Types

The following section outlines any complex data types used in the component in alphabetical order. This includes packed records and packed arrays that might be used as connector types, command arguments, event parameters, etc..

Command.T:

Generic command packet for holding arbitrary commands

Table 11: Command Packed Record : 808 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
`Header`	`Command_Header.T`	-	40	0	39	-
`Arg_Buffer`	`Command_Types.Command_Arg_Buffer_Type`	-	768	40	807	`Header.Arg_Buffer_Length`

Field Descriptions:

- **`Header`** - The command header
- **`Arg_Buffer`** - A buffer to that contains the command arguments

Command_Header.T:

Generic command header for holding arbitrary commands

Table 12: Command_Header Packed Record : 40 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
`Source_Id`	`Command_Types.Command_Source_Id`	0 to 65535	16	0	15
`Id`	`Command_Types.Command_Id`	0 to 65535	16	16	31
`Arg_Buffer_Length`	`Command_Types.Command_Arg_Buffer_Length_Type`	0 to 96	8	32	39

Field Descriptions:

- **`Source_Id`** - The source ID. An ID assigned to a command sending component.
- **`Id`** - The command identifier
- **`Arg_Buffer_Length`** - The number of bytes used in the command argument buffer

Command_Response.T:

Record for holding command response data.

Table 13: Command_Response Packed Record : 56 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Source_Id	Command_ Types.Command_ Source_Id	0 to 65535	16	0	15
Registration_ Id	Command_ Types.Command_ Registration_ Id	0 to 65535	16	16	31
Command_Id	Command_Types. Command_Id	0 to 65535	16	32	47
Status	Command_Enums. Command_ Response_ Status.E	0 => Success 1 => Failure 2 => Id_Error 3 => Validation_Error 4 => Length_Error 5 => Dropped 6 => Register 7 => Register_Source	8	48	55

Field Descriptions:

- **`Source_Id`** - The source ID. An ID assigned to a command sending component.
- **`Registration_Id`** - The registration ID. An ID assigned to each registered component at initialization.
- **`Command_Id`** - The command ID for the command response.
- **`Status`** - The command execution status.

Data_Product.T:

Generic data product packet for holding arbitrary data types

Table 14: Data_Product Packed Record : 344 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Data_Product_ Header.T	-	88	0	87	-
Buffer	Data_Product_ Types.Data_ Product_ Buffer_Type	-	256	88	343	Header.Buffer_ Length

Field Descriptions:

- **`Header`** - The data product header

- **Buffer** - A buffer that contains the data product type

Data_Product_Header.T:

Generic data_product packet for holding arbitrary data_product types

Table 15: Data_Product_Header Packed Record : 88 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Data_Product_Types.Data_Product_Id	0 to 65535	16	64	79
Buffer_Length	Data_Product_Types.Data_Product_Buffer_Length_Type	0 to 32	8	80	87

Field Descriptions:

- **Time** - The timestamp for the data product item.
- **Id** - The data product identifier
- **Buffer_Length** - The number of bytes used in the data product buffer

Event.T:

Generic event packet for holding arbitrary events

Table 16: Event Packed Record : 344 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Event_Header.T	-	88	0	87	–
Param_Buffer	Event_Types.Parameter_Buffer_Type	-	256	88	343	Header.Param_Buffer_Length

Field Descriptions:

- **Header** - The event header
- **Param_Buffer** - A buffer that contains the event parameters

Event_Header.T:

Generic event packet for holding arbitrary events

Table 17: Event_Header Packed Record : 88 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Event_Types.Event_Id	0 to 65535	16	64	79
Param_Buffer_Length	Event_Types.Parameter_Buffer_Length_Type	0 to 32	8	80	87

Field Descriptions:

- **Time** - The timestamp for the event.
- **Id** - The event identifier
- **Param_Buffer_Length** - The number of bytes used in the param buffer

Ided_Memory_Region.T:

A memory region that has a unique identifier associated with it.

Table 18: Ided_Memory_Region Packed Record : 112 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Id	Interfaces. Unsigned_16	0 to 65535	16	0	15
Region	Memory_Region.T	-	96	16	111

Field Descriptions:

- **Id** - Unique identifier for the memory region.
- **Region** - The source address and length to copy from.

Invalid_Command_Info.T:

Record for holding information about an invalid command

Table 19: Invalid_Command_Info Packed Record : 112 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Id	Command_Types. Command_Id	0 to 65535	16	0	15
Errant_Field_ Number	Interfaces. Unsigned_32	0 to 4294967295	32	16	47
Errant_Field	Basic_Types.Poly_ Type	-	64	48	111

Field Descriptions:

- **Id** - The command Id received.
- **Errant_Field_Number** - The field that was invalid. 1 is the first field, 0 means unkown field, 2**32 means that the length field of the command was invalid.
- **Errant_Field** - A polymorphic type containing the bad field data, or length when Errant_Field_Number is 2**32.

Invalid_Virtual_Memory_Region.T:

Packed record which holds information about an invalid virtual memory region.

Table 20: Invalid_Virtual_Memory_Region Packed Record : 128 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Invalid_Region	Virtual_Memory_Region. T	-	64	0	63

Managed_Region	Virtual_Memory_Region_ Positive.T	-	64	64	127

Field Descriptions:

- **Invalid_Region** - The memory region that was deemed invalid to operate on.
- **Managed_Region** - The memory region managed by the component that the invalid region was compared against.

Memory_Manager_State.T:

Packed record which holds the memory state of the memory manager.

Table 21: Memory_Manager_State Packed Record : 8 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
State	Memory_Manager_ Enums.Memory_ State.E	0 => Available 1 => In_Use	8	0	7

Field Descriptions:

- **State** - The memory state.

Memory_Region.T:

A memory region described by a system address and length (in bytes).

Table 22: Memory_Region Packed Record : 96 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Address	System.Address	-	64	0	63
Length	Natural	0 to 2147483647	32	64	95

Field Descriptions:

- **Address** - The starting address of the memory region.
- **Length** - The number of bytes at the given address to associate with this memory region.

Memory_Region_Request.T:

A requested memory region that has a unique identifier and return status associated with it.

Table 23: Memory_Region_Request Packed Record : 120 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Ided_Region	Ided_Memory_ Region.T	-	112	0	111
Status	Memory_Manager_ Enums.Memory_ Request_Status.E	0 => Success 1 => Failure	8	112	119

Field Descriptions:

- **Ided_Region** - Unique identifier for the memory region and the region itself.
- **Status** - The return status of the memory request.

Sys_Time.T:

A record which holds a time stamp using GPS format including seconds and subseconds since epoch (1-5-1980 to 1-6-1980 midnight).

Table 24: Sys_Time Packed Record : 64 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Seconds	Interfaces. Unsigned_32	0 to 4294967295	32	0	31
Subseconds	Interfaces. Unsigned_32	0 to 4294967295	32	32	63

Field Descriptions:

- **Seconds** - The number of seconds elapsed since epoch.
- **Subseconds** - The number of 1/(2^32) sub-seconds.

Virtual_Memory_Region.T:

A memory region described by a virtual memory address (an index into a zero-addressed memory region) and length (in bytes).

Table 25: Virtual_Memory_Region Packed Record : 64 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Address	Natural	0 to 2147483647	32	0	31
Length	Natural	0 to 2147483647	32	32	63

Field Descriptions:

- **Address** - The virtual memory address (an index into a zero-addressed memory region).
- **Length** - The number of bytes at the given address to associate with this memory region.

Virtual_Memory_Region_Crc.T:

A virtual memory region CRC report.

Table 26: Virtual_Memory_Region_Crc Packed Record : 80 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Region	Virtual_Memory_Region. T	-	64	0	63
Crc	Crc_16.Crc_16_Type	-	16	64	79

Field Descriptions:

- **Region** - The virtual memory region that was CRCed
- **Crc** - The computed CRC

Virtual_Memory_Region_Positive.T:

A memory region described by a virtual memory address (an index into a zero-addressed memory region) and length (in bytes) that cannot be less than 1.

Table 27: Virtual_Memory_Region_Positive Packed Record : 64 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Address	Natural	0 to 2147483647	32	0	31
Length	Positive	1 to 2147483647	32	32	63

Field Descriptions:

- **Address** - The virtual memory address (an index into a zero-addressed memory region).
- **Length** - The number of bytes at the given address to associate with this memory region.

Virtual_Memory_Region_Write.T:

A virtual memory region write type that fits within a command.

Preamble (inline Ada definitions):

```ada
use Command_Types;
subtype Byte_Buffer_Index_Type is Command_Arg_Buffer_Index_Type range
↪  Command_Arg_Buffer_Index_Type'First .. Command_Arg_Buffer_Index_Type'Last -
↪  4 - 2;
subtype Byte_Buffer_Type is Basic_Types.Byte_Array (Byte_Buffer_Index_Type);
subtype Region_Length_Type is Natural range 0 .. Byte_Buffer_Type'Length;
```

Table 28: Virtual_Memory_Region_Write Packed Record : 768 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Address	Natural	0 to 2147483647	32	0	31	-
Length	Region_Length_Type	0 to 90	16	32	47	-
Data	Byte_Buffer_Type	-	720	48	767	Length

Field Descriptions:

- **Address** - The virtual memory address (an index into a zero-addressed memory region).
- **Length** - The number of bytes at the given address to associate with this memory region.
- **Data** - The bytes to write to the memory region

5.3 Enumerations

The following section outlines any enumerations used in the component.

Command_Enums.Command_Response_Status.E:

This status enumerations provides information on the success/failure of a command through the command response connector.

Table 29: Command_Response_Status Literals:

Name	Value	Description
Success	0	Command was passed to the handler and successfully executed.
Failure	1	Command was passed to the handler not successfully executed.
Id_Error	2	Command id was not valid.
Validation_Error	3	Command parameters were not successfully validated.
Length_Error	4	Command length was not correct.
Dropped	5	Command overflowed a component queue and was dropped.
Register	6	This status is used to register a command with the command routing system.
Register_Source	7	This status is used to register command sender's source id with the command router for command response forwarding.

Memory_Manager_Enums.Memory_Request_Status.E:

This status relates whether or not the memory request succeeded or failed.

Table 30: Memory_Request_Status Literals:

Name	Value	Description
Success	0	The memory request succeeded.
Failure	1	The memory request failed.

Memory_Manager_Enums.Memory_State.E:

This status relates whether or not the memory manager memory region is currently available or if it is in use.

Table 31: Memory_State Literals:

Name	Value	Description
Available	0	The memory region is available for request.
In_Use	1	The memory region is NOT available for request, as it is currently in use by another requestor.