Memory Copier

Component Design Document

1 Description

This component services a command to copy from one memory region to another. The to/from destination of the copy command is determined by how it is connected in the assembly. The component will wait a configurable timeout for the copy command to complete before failing the command and reporting a timeout error.

2 Requirements

The requirements for the Memory Copier component are specified below.

1. The component shall copy a memory region from one memory store to another on command.

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** - 9
- **Number of Invokee Connectors** - 3
- **Number of Invoker Connectors** - 6
- **Number of Generic Connectors** - *None*
- **Number of Generic Types** - *None*
- **Number of Unconstrained Arrayed Connectors** - *None*
- **Number of Commands** - 1
- **Number of Parameters** - *None*
- **Number of Events** - 8
- **Number of Faults** - *None*
- **Number of Data Products** - *None*
- **Number of Data Dependencies** - *None*
- **Number of Packets** - *None*

3.2 Diagram

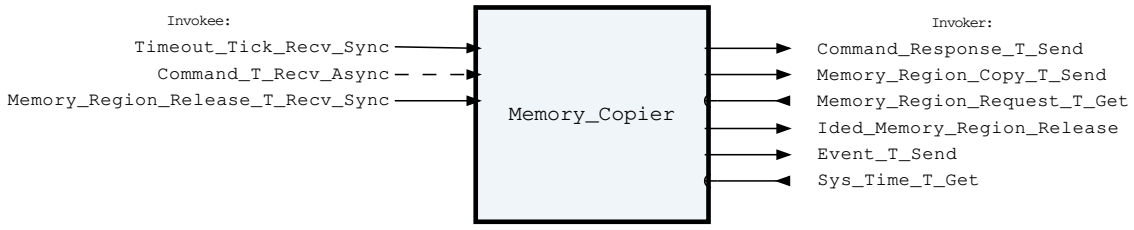

Figure 1: Memory Copier component diagram.

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 Copier Invokee Connectors

Name	Kind	Type	Return_Type	Count
Timeout_Tick_Recv_Sync	recv_sync	Tick.T	-	1
Command_T_Recv_Async	recv_async	Command.T	-	1
Memory_Region_Release_T_Recv_Sync	recv_sync	Memory_Region_Release.T	-	1

Connector Descriptions:

- **Timeout_Tick_Recv_Sync** - The component should be attached to a periodic tick that is used to timeout waiting for a memory region copy response. See the ticks_Until_Timeout initialization parameter.
- **Command_T_Recv_Async** - The command receive connector.
- **Memory_Region_Release_T_Recv_Sync** - The memory region is returned synchronously on this connector. The component waits internally for this response, or times out if the response is not received in time.

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 Copier 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 Copier Invoker Connectors

Name	Kind	Type	Return_Type	Count
Command_Response_T_Send	send	Command_Response.T	-	1
Memory_Region_Copy_T_Send	send	Memory_Region_Copy.T	-	1
Memory_Region_Request_T_Get	get	-	Memory_Region_Request.T	1
Ided_Memory_Region_Release	send	Ided_Memory_Region.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 send the command response back to the command router.
- **Memory_Region_Copy_T_Send** - A memory region is sent on this connector for copy.
- **Memory_Region_Request_T_Get** - The scratch memory region is requested on this connector.
- **Ided_Memory_Region_Release** - The memory region is released (returned) to scratch on this connector.
- **Event_T_Send** - The event send 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 Copier 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 Copier Set Id Bases Parameters

Name	Type
`Event_Id_Base`	`Event_Types.Event_Id_Base`
`Command_Id_Base`	`Command_Types.Command_Id_Base`

Parameter Descriptions:

- **`Event_Id_Base`** - The value at which the component's event identifiers begin.
- **`Command_Id_Base`** - The value at which the component's command 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. Initialization parameters for the Memory Copier. The `init` subprogram requires the following parameters:

Table 6: Memory Copier Implementation Initialization Parameters

Name	Type	Default Value
`Ticks_Until_Timeout`	`Natural`	*None provided*

Parameter Descriptions:

- **`Ticks_Until_Timeout`** - The component will wait until it has received at least this many ticks before reporting a timeout error while waiting for a memory copy to complete. For example, if the component is attached to a 10Hz rate group and this value is set to 7, then the component will wait between 700 and 800 ms before declaring a timeout error from an unresponsive downstream component.

3.6 Commands

These are the commands for the Memory Copier component.

Table 7: Memory Copier Commands

Local ID	Command Name	Argument Type
0	`Copy_Memory_Region`	`Virtual_Memory_Region_Copy.T`

Command Descriptions:

- **Copy_Memory_Region** - Copy Length bytes of memory from the source virtual memory address to the destination Address.

3.7 Parameters

The Memory Copier component has no parameters.

3.8 Events

Events for the Memory Copier component.

Table 8: Memory Copier Events

Local ID	Event Name	Parameter Type
0	Memory_Region_Length_Mismatch	Invalid_Memory_Region_Length.T
1	Memory_Region_Unavailable	-
2	Starting_Copy	Virtual_Memory_Region_Copy.T
3	Finished_Copy	Virtual_Memory_Region_Copy.T
4	Invalid_Command_Received	Invalid_Command_Info.T
5	Copy_Timeout	-
6	Copy_Failure	Memory_Region_Release.T
7	Command_Dropped	Command_Header.T

Event Descriptions:

- **Memory_Region_Length_Mismatch** - A memory region was received with a length less than that specified in the copy command. The length of the region requested must be the same size or greater than the length specified in the copy command.
- **Memory_Region_Unavailable** - Requesting the memory region was denied because it is currently in use by another component.
- **Starting_Copy** - Starting copy from source to destination.
- **Finished_Copy** - Finished copy from source to destination, without errors.
- **Invalid_Command_Received** - A command was received with invalid parameters.
- **Copy_Timeout** - A timeout occured while waiting for a copy operation to complete.
- **Copy_Failure** - A copy failed.
- **Command_Dropped** - A command was dropped due to a full queue.

3.9 Data Products

The Memory Copier component has no data products.

3.10 Packets

The Memory Copier component has no packets.

4 Unit Tests

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

4.1 *Memory_ Copier_ Tests* Test Suite

This is a unit test suite for the Memory Copier component.

Test Descriptions:

- **`Test_Nominal_Copy`** - This unit test tests the nominal copy command from the source to the destination.
- **`Test_Copy_Failure`** - This unit test tests the component's response to a failed copy.
- **`Test_Copy_Timeout`** - This unit test tests the component's response when the destination component does not respond to a copy command before a timeout occurs.
- **`Test_Memory_Unavailable`** - This unit test tests the component's response when the source component is unavailable for copy.
- **`Test_Length_Mismatch`** - This unit test tests the component's response when source component returns a region that is too small.
- **`Test_Full_Queue`** - This unit test tests a command being dropped due to a full queue.
- **`Test_Invalid_Command`** - This unit test exercises that an invalid command throws the appropriate event.

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 9: 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 10: 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 11: 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.

Event.T:

Generic event packet for holding arbitrary events

Table 12: 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 13: 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 14: 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 15: 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_Memory_Region_Length.T:

A packed record which holds data related to an invalid memory region length.

Table 16: Invalid_Memory_Region_Length Packed Record : 128 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Region	Memory_Region.T	-	96	0	95
Expected_Length	Natural	0 to 2147483647	32	96	127

Field Descriptions:

- **Region** - The memory region and operation.
- **Expected_Length** - The length bound that the memory region failed to meet.

Memory_Region.T:

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

Table 17: 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_Copy.T:

A memory region copy record.

Table 18: Memory_Region_Copy Packed Record : 160 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Source_Region	Memory_Region.T	-	96	0	95
Destination_Address	System.Address	-	64	96	159

Field Descriptions:

- **Source_Region** - The source address and length to copy from.
- **Destination_Address** - The destination address of the memory region copy.

Memory_Region_Release.T:

A memory region copy release record. This is returned from a component who has completed a copy operation.

Table 19: Memory_Region_Release Packed Record : 104 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Region	Memory_Region.T	-	96	0	95
Status	Memory_Enums. Memory_Copy_ Status.E	0 => Success 1 => Failure	8	96	103

Field Descriptions:

- **Region** - The address and length to release.
- **Status** - The return status from the operation.

Memory_Region_Request.T:

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

Table 20: 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 21: 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.

Tick.T:

The tick datatype used for periodic scheduling. Included in this type is the Time associated with a tick and a count.

Table 22: Tick Packed Record : 96 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Count	Interfaces. Unsigned_32	0 to 4294967295	32	64	95

Field Descriptions:

- **Time** - The timestamp associated with the tick.
- **Count** - The cycle number of the tick.

Virtual_Memory_Region_Copy.T:

A virtual memory region copy to a physical memory address.

Table 23: Virtual_Memory_Region_Copy Packed Record : 128 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Source_Address	Natural	0 to 2147483647	32	0	31
Source_Length	Natural	0 to 2147483647	32	32	63
Destination_ Address	System.Address	-	64	64	127

Field Descriptions:

- **Source_Address** - The virtual memory address (an index into a zero-addressed memory region).
- **Source_Length** - The number of bytes at the given address to associate with this memory region.
- **Destination_Address** - The starting address of the destination 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 24: 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_Enums.Memory_Copy_Status.E:

This status enumerations provides information on the success/failure of a memory copy.

Table 25: Memory_Copy_Status Literals:

Name	Value	Description
Success	0	The copy command succeeded.
Failure	1	The copy command failed.

Memory_Manager_Enums.Memory_Request_Status.E:

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

Table 26: Memory_Request_Status Literals:

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