Parameter Manager

Component Design Document

1 Description

This component is responsible for managing a working and default parameter table. Its sole responsibility is to respond to commands to copy parameter tables from one region to another.

2 Requirements

The requirements for the Parameter Manager component are specified below.

1. The component shall copy a parameter table from a command payload to default and working regions, 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** - 10
- **Number of Invokee Connectors** - 3
- **Number of Invoker Connectors** - 7
- **Number of Generic Connectors** - *None*
- **Number of Generic Types** - *None*
- **Number of Unconstrained Arrayed Connectors** - *None*
- **Number of Commands** - 2
- **Number of Parameters** - *None*
- **Number of Events** - 10
- **Number of Faults** - *None*
- **Number of Data Products** - 1
- **Number of Data Dependencies** - *None*
- **Number of Packets** - *None*

3.2 Diagram

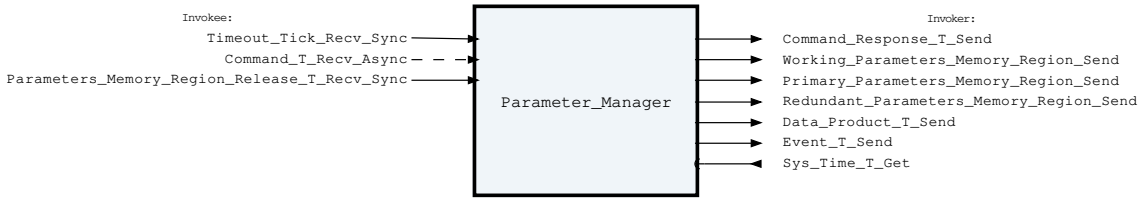

Figure 1: Parameter Manager component diagram.

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: Parameter Manager 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
Parameters_Memory_Region_Release_T_Recv_Sync	recv_sync	Parameters_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 parameter update/fetch response. See the ticks_Until_Timeout initialization parameter.
- **Command_T_Recv_Async** - The command receive connector.
- **Parameters_Memory_Region_Release_T_Recv_Sync** - Parameter update/fetch responses are 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: Parameter Manager Asynchronous Connectors

Name	Type	Max Size (bytes)
Command_T_Recv_Async	Command.T	265

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: Parameter Manager Invoker Connectors

Name	Kind	Type	Return_Type	Count
Command_Response_T_Send	send	Command_Response.T	-	1
Working_Parameters_Memory_Region_Send	send	Parameters_Memory_Region.T	-	1
Primary_Parameters_Memory_Region_Send	send	Parameters_Memory_Region.T	-	1
Redundant_Parameters_Memory_Region_Send	send	Parameters_Memory_Region.T	-	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 send the command response back to the command router.
- **Working_Parameters_Memory_Region_Send** - Requests to update/fetch the working parameters are made on this connector.
- **Primary_Parameters_Memory_Region_Send** - Requests to update/fetch the default parameters are made on this connector.
- **Redundant_Parameters_Memory_Region_Send** - Requests to update/fetch the default parameters are made on this connector.
- **Data_Product_T_Send** - The destination for fetched data products to be sent to.
- **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: Parameter 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: Parameter Manager Set Id Bases Parameters

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

Parameter Descriptions:

- **`Command_Id_Base`** - The value at which the component's command identifiers begin.
- **`Data_Product_Id_Base`** - The value at which the component's data product identifiers begin.
- **`Event_Id_Base`** - The value at which the component's event 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 Parameter Manager. The `init` subprogram requires the following parameters:

Table 6: Parameter Manager 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 parameter update/fetch response from either the working or default parameter components. 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 Parameter Manager component.

Table 7: Parameter Manager Commands

Local ID	Command Name	Argument Type
0	Update_Parameter_Table	Packed_Parameter_Table.T
1	Validate_Parameter_Table	Packed_Parameter_Table.T

Command Descriptions:

- **`Update_Parameter_Table`** - Send received parameter table to default and working regions.
- **`Validate_Parameter_Table`** - Validate a received parameter table.

3.7 Parameters

The Parameter Manager component has no parameters.

3.8 Events

Events for the Parameter Manager component.

Table 8: Parameter Manager Events

Local ID	Event Name	Parameter Type
0	Starting_Parameter_Table_Copy	Parameter_Manager_Table_ Header.T
1	Finished_Parameter_Table_Copy	Parameter_Manager_Table_ Header.T
2	Invalid_Command_Received	Invalid_Command_Info.T
3	Parameter_Table_Copy_Timeout	-
4	Parameter_Table_Copy_Failure	Parameters_Memory_Region_ Release.T
5	Working_Table_Update_Failure	Packed_Validation_Header.T
6	Primary_Table_Update_Failure	Packed_Validation_Header.T
7	Command_Dropped	Command_Header.T
8	Table_Validation_Failure	Packed_Validation_Header.T
9	Table_Validation_Success	Packed_Validation_Header.T

Event Descriptions:

- **`Starting_Parameter_Table_Copy`** - Starting parameter table copy from source to destination.
- **`Finished_Parameter_Table_Copy`** - Finished parameter table copy from source to destination, without errors.

- **Invalid_Command_Received** - A command was received with invalid parameters.
- **Parameter_Table_Copy_Timeout** - A timeout occurred while waiting for a parameter table copy operation to complete.
- **Parameter_Table_Copy_Failure** - A parameter table copy failed.
- **Working_Table_Update_Failure** - A parameter table copy to the working table failed.
- **Primary_Table_Update_Failure** - A parameter table copy to the primary table failed.
- **Command_Dropped** - A command was dropped due to a full queue.
- **Table_Validation_Failure** - A parameter table validation failed.
- **Table_Validation_Success** - A parameter table validation was successful.

3.9 Data Products

Data products for the Parameter Manager component

Table 9: Parameter Manager Data Products

Local ID	Data Product Name	Type
0x0000 (0)	Validation_Status	Packed_Validation.T

Data Product Descriptions:
- **Validation_Status** - The validation status with timestamp and last table ID/version.

3.10 Data Dependencies

The Parameter Manager component has no data dependencies.

3.11 Packets

The Parameter Manager component has no packets.

3.12 Faults

The Parameter Manager component has no faults.

4 Unit Tests

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

4.1 *Parameter_Manager_Tests* Test Suite

This is a unit test suite for the Parameter Manager component.

Test Descriptions:
- **Test_Nominal_Validation** - This unit test tests the nominal validation command.
- **Test_Validation_Failure** - This unit test tests the component's response to a failed validation.
- **Test_Validation_Timeout** - This unit test tests the component's response when the destination component does not respond to a validation command before a timeout occurs.

- **Test_Nominal_Copy** - This unit test tests the nominal copy command.
- **Test_Copy_Failure** - This unit test tests the component's response to a failed parameter table 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_Full_Queue** - This unit test tests a command or memory region 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 10: Command Packed Record : 2080 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	-	2040	40	2079	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 11: 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 255	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 12: 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 13: Data_Product Packed Record : 1112 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	-	1024	88	1111	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 14: 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 128	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 15: 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 16: 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

Invalid_Command_Info.T:

Record for holding information about an invalid command

Table 17: 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 unknown 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.

Memory_Region.T:

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

Table 18: 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.

Packed_Parameter_Table.T:

Generic parameter packet for holding a parameter table

Table 19: Packed_Parameter_Table Packed Record : 2040 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Parameter_Manager_Table_Header.T	-	64	0	63	-
Table_Buffer	Parameter_Manager_Types.Parameter_Manager_Buffer_Type	-	1976	64	2039	Header.Table_Buffer_Length

Field Descriptions:

- **Header** - The parameter table header
- **Table_Buffer** - A buffer that contains the parameter table

Packed_Validation.T:

A packed validation record

Table 20: Packed_Validation Packed Record : 56 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Last_Validation_Version	Short_Float	-3.40282e+38 to 3.40282e+38	32	0	31
Crc_Table	Crc_16.Crc_16_Type	-	16	32	47
Last_Validation_Status	Parameter_Enums.Parameter_Table_Update_Status.E	0 => Success 1 => Length_Error 2 => Crc_Error 3 => Parameter_Error 4 => Dropped	8	48	55

Field Descriptions:

- **Last_Validation_Version** - The version of the most recently validated table.
- **Crc_Table** - The CRC of the most recently validated table.
- **Last_Validation_Status** - The status of the most recent table validation operation.

Packed_Validation_Header.T:

A packed record which holds the most recently validated parameter table header with its validation status.

Table 21: Packed_Validation_Header Packed Record : 72 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Last_ Validation_ Header	Parameter_ Manager_Table_ Header.T	-	64	0	63
Last_ Validation_ Status	Parameter_ Enums. Parameter_ Table_Update_ Status.E	0 => Success 1 => Length_Error 2 => Crc_Error 3 => Parameter_Error 4 => Dropped	8	64	71

Field Descriptions:

- **Last_Validation_Header** - The header of the most recently validated parameter table.
- **Last_Validation_Status** - The status of the most recent table validation operation.

Parameter_Manager_Table_Header.T:

A packed record which holds parameter table header data. This data will be prepended to the table data upon upload. *Preamble (inline Ada definitions):*

```
-- Declare the start index at which to begin calculating the CRC. The
-- start index is dependent on this type, and so is declared here so that
-- it is easier to keep in sync.
Crc_Section_Length : constant Natural := Crc_16.Crc_16_Type'Length;
Version_Length : constant Natural := 4;
```

Table 22: Parameter_Manager_Table_Header Packed Record : 64 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Table_ Buffer_ Length	Parameter_ Manager_ Types. Parameter_ Manager_ Buffer_ Length_Type	0 to 247	16	0	15
Crc_Table	Crc_16.Crc_ 16_Type	-	16	16	31
Version	Short_Float	-3.40282e+38 to 3.40282e+38	32	32	63

Field Descriptions:

- **Table_Buffer_Length** - The length of the parameter table buffer.
- **Crc_Table** - The CRC of the parameter table, as computed by a ground system, and uplinked with the table.
- **Version** - The current version of the parameter table.

Parameters_Memory_Region.T:

A packed record which holds the parameter memory region to operate on as well as an enumeration specifying the operation to perform.

Table 23: Parameters_Memory_Region Packed Record : 104 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Region	Memory_Region.T	-	96	0	95
Operation	Parameter_Enums. Parameter_Table_ Operation_Type.E	0 => Get 1 => Set 2 => Validate	8	96	103

Field Descriptions:

- **Region** - The memory region.
- **Operation** - The parameter table operation to perform.

Parameters_Memory_Region_Release.T:

A packed record which holds the parameter memory region to release as well as the status returned from the parameter update operation.

Table 24: Parameters_Memory_Region_Release Packed Record : 104 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Region	Memory_Region. T	-	96	0	95
Status	Parameter_ Enums. Parameter_ Table_Update_ Status.E	0 => Success 1 => Length_Error 2 => Crc_Error 3 => Parameter_Error 4 => Dropped	8	96	103

Field Descriptions:

- **Region** - The memory region.
- **Status** - The return status from the parameter update operation.

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 25: 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 26: 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.

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 27: 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.

Parameter_Enums.Parameter_Table_Operation_Type.E:

This enumeration lists the different parameter table operations that can be performed.

Table 28: Parameter_Table_Operation_Type Literals:

Name	Value	Description
Get	0	Retrieve the current values of the parameters.
Set	1	Set the current values of the parameters.
Validate	2	Validate the current values of the parameters.

Parameter_Enums.Parameter_Table_Update_Status.E:

This status enumeration provides information on the success/failure of a parameter table update.

Table 29: Parameter_Table_Update_Status Literals:

Name	Value	Description
Success	0	Parameter was successfully staged.
Length_Error	1	Parameter table length was not correct.
Crc_Error	2	The computed CRC of the table does not match the stored CRC.
Parameter_Error	3	An individual parameter was found invalid due to a constraint error within a component, or failing component-specific validation.
Dropped	4	The operation could not be performed because it was dropped from a full queue.