Science Assembly

Assembly Design Document

1 Description

This is the example assembly.

2 Design

2.1 At a Glance

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

- **Number of Components** - 8
- **Number of Component Types** - 8
- **Number of Active Components** - 3
- **Number of Passive Components** - 5
- **Number of Components with Queue** - 4
- **Number of Components without Queue** - 4
- **Number of Components with Events** - 3
- **Number of Components with Data Products** - 2
- **Number of Components with Data Dependencies** - 1
- **Number of Components with Packets** - *None*
- **Number of Components with Commands** - 2
- **Number of Components with Parameters** - 1
- **Number of Components with Faults** - *None*
- **Number of Connections** - 11
- **Number of Events** - 7
- **Number of Data Products** - 4
- **Number of Data Dependencies** - 2
- **Number of Packets** - *None*
- **Number of Commands** - 3
- **Number of Parameters** - 2
- **Number of Faults** - *None*

2.2 Components

Table 1: Science Assembly Components

Name	Type	Has Queue	Execution
Rate_Group_Instance	Example_Rate_Group	yes	active
Command_Router_Instance	Example_Command_Router	yes	active
Science_Instance	Example_Science	yes	passive
Time_Instance	Example_Time	no	passive
Parameters_Instance	Example_Parameters	no	passive
Logger_Instance	Example_Logger	yes	active
Data_Collector_Instance	Example_Data_Collector	no	passive
Database_Instance	Example_Database	no	passive

Table 2: Science Assembly Component Item Counts

Component Name	Connectors	Commands	Events	Data Products	Data Dependencies	Parameters	Packets	Faults
Rate_Group_Instance	2	0	2	0	0	0	0	0
Command_Router_Instance	2	1	3	0	0	0	0	0
Science_Instance	6	2	2	2	2	2	0	0
Time_Instance	1	0	0	0	0	0	0	0
Parameters_Instance	2	0	0	0	0	0	0	0
Logger_Instance	2	0	0	0	0	0	0	0
Data_Collector_Instance	3	0	0	2	0	0	0	0
Database_Instance	2	0	0	0	0	0	0	0

Component Descriptions:

- **Rate_Group_Instance** - *No description provided.*
- **Command_Router_Instance** - *No description provided.*
- **Science_Instance** - *No description provided.*
- **Time_Instance** - *No description provided.*
- **Parameters_Instance** - *No description provided.*

- **`Logger_Instance`** - *No description provided.*
- **`Data_Collector_Instance`** - *No description provided.*
- **`Database_Instance`** - *No description provided.*

2.3 Views

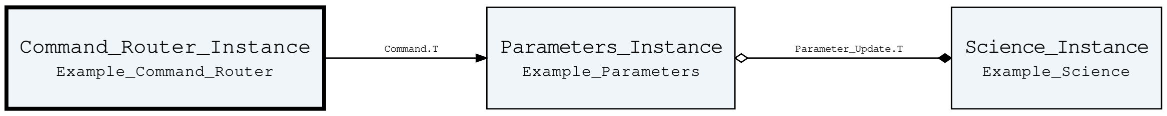

Figure 1: **Parameters View2 View:** This is also a parameters view.

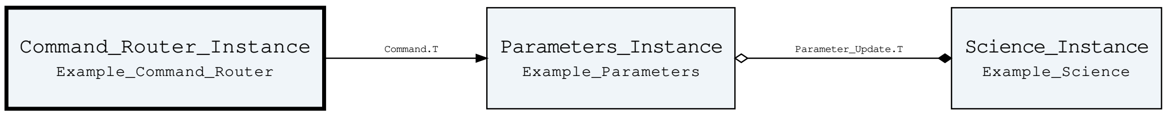

Figure 2: **Parameters View View:** This is a parameters view.

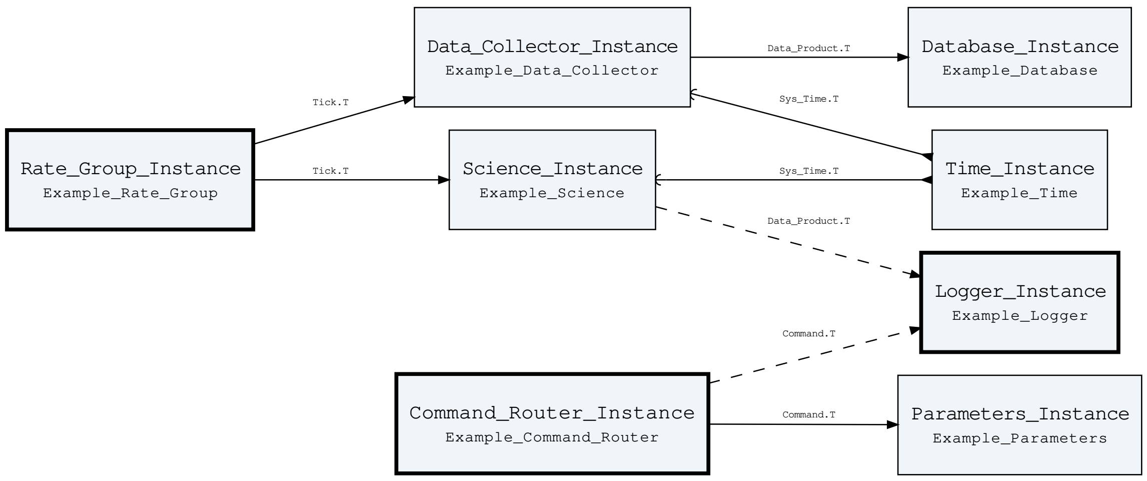

Figure 3: **Grouped View View:** This is a grouped view.

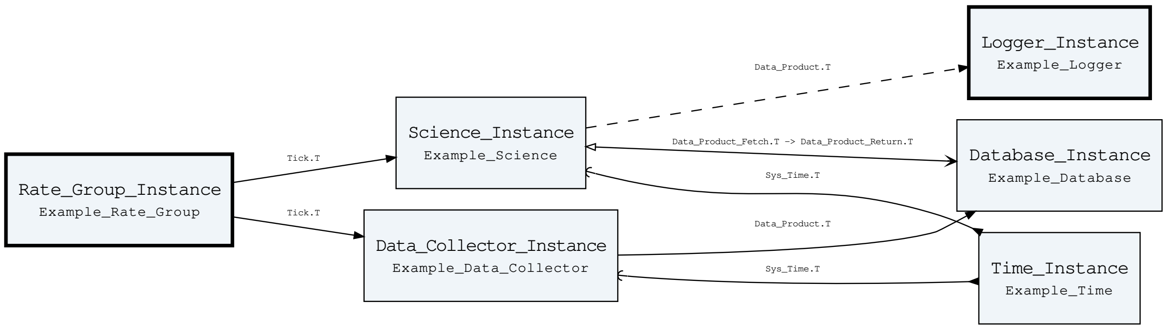

Figure 4: **No Command Params View:** This is also a view without commands or parameters.

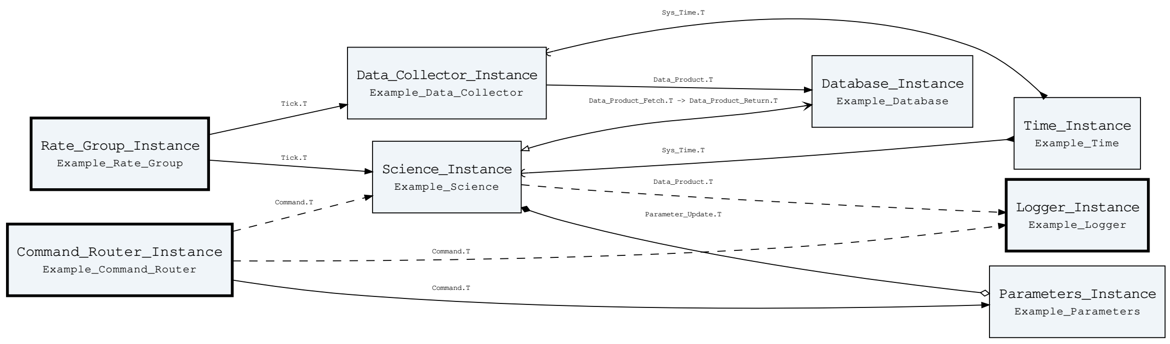

Figure 5: **Science Assembly View View:** This is the assembly view.

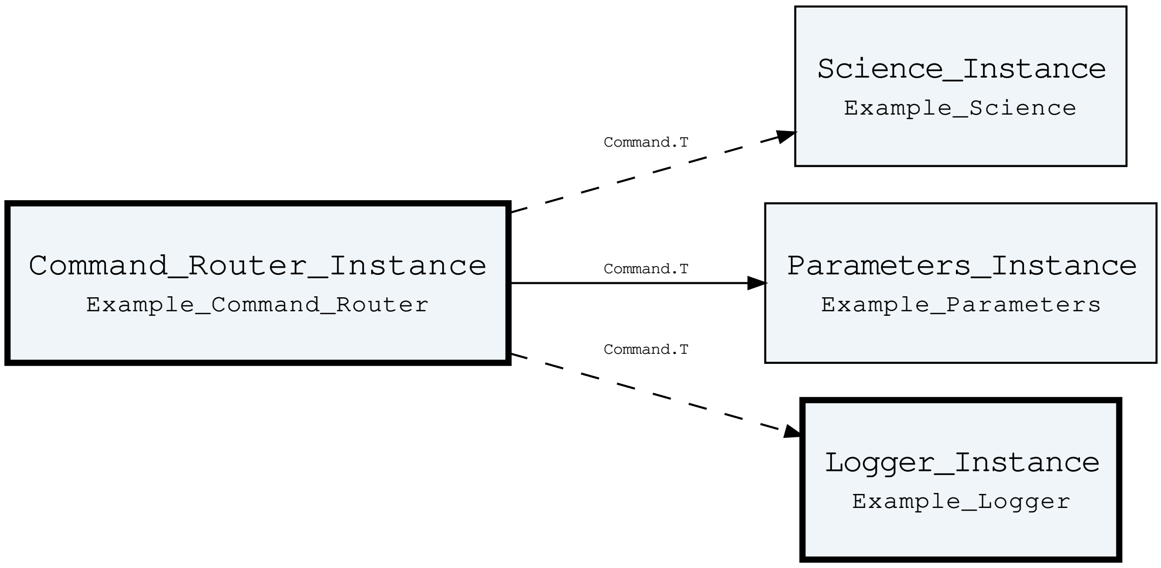

Figure 6: **Command View View:** This is the command view.

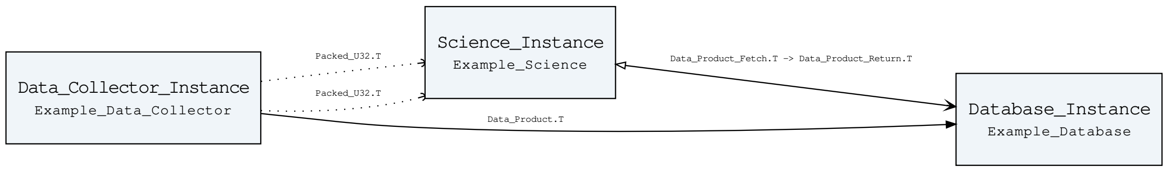

Figure 7: **Data Collector View View:** This is view showing data dependencies.

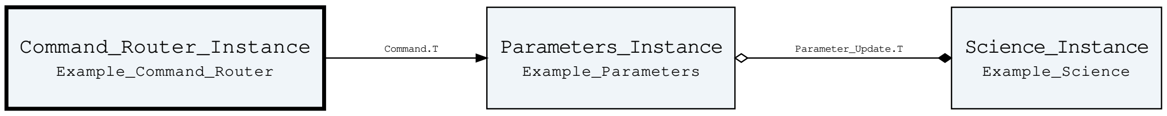

Figure 8: **Parameters Context View View:** This is a parameters view.

2.4 Task Priorities

The table below outlines the system tasks for the Science Assembly assembly. Task names are in the form *component_ name.task_ name*. The priority *rank* is a number from 1 to *n* denoting how the priority of a component's task compares to others in the system. A rank of 1 is the highest priority in the system. The *priority value* is the actual priority number provided to the system scheduler. A larger *priority value* value signifies a higher priority task.

Table 3: Science Assembly Component Task Priorities

Task Number	**Task Name**	**Priority Rank**	**Priority Value**
0	`Rate_Group_Instance.Active_Task`	1	3
1	`Command_Router_Instance.Active_Task`	2	2
2	`Logger_Instance.Active_Task`	3	1

2.5 Commands

The table below shows the commands for the Science Assembly assembly.

Table 4: Science Assembly Commands

Command ID	**Command Name**	**Argument Type**
`0x0001 (1)`	`Command_Router_Instance. Noop`	-
`0x0002 (2)`	`Science_Instance.Enable_ Science`	-
`0x0003 (3)`	`Science_Instance.Disable_ Science`	-

Command Descriptions:

- **`Command_Router_Instance.Noop`** - Simple NOOP command which produces an event saying that it was triggered.
- **`Science_Instance.Enable_Science`** - Start collecting science.
- **`Science_Instance.Disable_Science`** - Stop collecting science.

2.6 Events

The table below shows the events for the Science Assembly assembly.

Table 5: Science Assembly Events

Event ID	**Event Name**	**Parameter Type**
`0x0001 (1)`	`Rate_Group_Instance.Cycle_ Slip`	`Cycle_Slip_Param.T`

0x0002 (2)	Rate_Group_Instance. Incoming_Tick_Dropped	Tick.T
0x0003 (3)	Command_Router_Instance. Command_Received	Command_Header.T
0x0004 (4)	Command_Router_Instance. Noop_Received	-
0x0005 (5)	Command_Router_Instance. Invalid_Command_Received	Invalid_Command_Info.T
0x0006 (6)	Science_Instance.Science_ Started	-
0x0007 (7)	Science_Instance.Science_ Stopped	-

Event Descriptions:

- **`Rate_Group_Instance.Cycle_Slip`** - Execution ran long on this cycle.
- **`Rate_Group_Instance.Incoming_Tick_Dropped`** - The rate group component's queue is full, so it cannot store the tick coming in. This usually means the rate group is cycle slipping and not running as fast as it needs to.
- **`Command_Router_Instance.Command_Received`** - A command was received by the command router to be routed.
- **`Command_Router_Instance.Noop_Received`** - A Noop command was received.
- **`Command_Router_Instance.Invalid_Command_Received`** - A command was received with invalid parameters.
- **`Science_Instance.Science_Started`** - Science collection has started.
- **`Science_Instance.Science_Stopped`** - Science collection has stopped.

2.7 Data Products

The table below shows the data products for the Science Assembly assembly.

Table 6: Science Assembly Data Products

Data Product ID	**Data Product Name**	**Type**
0x0001 (1)	Data_Collector_Instance. Sensor_1_Data	Packed_U32.T
0x0002 (2)	Data_Collector_Instance. Sensor_2_Data	Packed_U32.T
0x0064 (100)	Science_Instance.Science_ 1_Data	Packed_F32.T

`0x0065 (101)`	`Science_Instance.Science_2_Data`	`Packed_F32.T`

Data Product Descriptions:

- **`Data_Collector_Instance.Sensor_1_Data`** - Sensor data value 1.
- **`Data_Collector_Instance.Sensor_2_Data`** - Sensor data value 2.
- **`Science_Instance.Science_1_Data`** - Science data value 1.
- **`Science_Instance.Science_2_Data`** - Science data value 2.

3 Appendix

3.1 Connections

Table 7: Science Assembly Connections

Number	From	To	Kind
1	`Rate_Group_Instance.Tick_T_Send [1]`	`Data_Collector_Instance.Tick_T_Recv_Sync`	`send-recv_sync`
2	`Rate_Group_Instance.Tick_T_Send [2]`	`Science_Instance.Tick_T_Recv_Sync`	`send-recv_sync`
3	`Command_Router_Instance.Command_T_Send [1]`	`Science_Instance.Command_T_Recv_Async`	`send-recv_async`
4	`Science_Instance.Sys_Time_T_Get`	`Time_Instance.Sys_Time_T_Return`	`get-return`
5	`Parameters_Instance.Parameter_Update_T_Provide`	`Science_Instance.Parameter_Update_T_Modify`	`provide-modify`
6	`Science_Instance.Data_Product_T_Send`	`Logger_Instance.Data_Product_T_Recv_Async`	`send-recv_async`
7	`Science_Instance.Data_Product_Fetch_T_Request`	`Database_Instance.Data_Product_Fetch_T_Service`	`request-service`
8	`Command_Router_Instance.Command_T_Send [2]`	`Logger_Instance.Command_T_Recv_Async`	`send-recv_async`

9	Command_Router_ Instance.Command_ T_Send [3]	Parameters_Instance. Command_T_Recv_Sync	send-recv_sync
10	Data_Collector_ Instance.Sys_Time_ T_Get	Time_Instance.Sys_ Time_T_Return	get-return
11	Data_Collector_ Instance.Data_ Product_T_Send	Database_Instance. Data_Product_T_Recv_ Sync	send-recv_sync

Connection Descriptions:

- **`Rate_Group_Instance.Tick_T_Send[1]-Data_Collector_Instance.Tick_T_Recv_Sync`** - This is the first connection in the model
- **`Rate_Group_Instance.Tick_T_Send[2]-Science_Instance.Tick_T_Recv_Sync`** - This is the first connection in the model
- **`Command_Router_Instance.Command_T_Send[1]-Science_Instance.Command_T_Recv_Async`** - *No description provided.*
- **`Science_Instance.Sys_Time_T_Get-Time_Instance.Sys_Time_T_Return`** - *No description provided.*
- **`Parameters_Instance.Parameter_Update_T_Provide-Science_Instance.Parameter_Update_T_Modify`** - *No description provided.*
- **`Science_Instance.Data_Product_T_Send-Logger_Instance.Data_Product_T_Recv_Async`** - *No description provided.*
- **`Science_Instance.Data_Product_Fetch_T_Request-Database_Instance.Data_Product_Fetch_T_Service`** - *No description provided.*
- **`Command_Router_Instance.Command_T_Send[2]-Logger_Instance.Command_T_Recv_Async`** - *No description provided.*
- **`Command_Router_Instance.Command_T_Send[3]-Parameters_Instance.Command_T_Recv_Sync`** - *No description provided.*
- **`Data_Collector_Instance.Sys_Time_T_Get-Time_Instance.Sys_Time_T_Return`** - *No description provided.*
- **`Data_Collector_Instance.Data_Product_T_Send-Database_Instance.Data_Product_T_Recv_Sync`** - *No description provided.*

3.2 Packed Types

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

Command.T:

Generic command packet for holding arbitrary commands

Table 8: 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 that contains the command arguments

Command_Header.T:

Generic command header for holding arbitrary commands

Table 9: 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

Cycle_Slip_Param.T:

This is a type that contains useful information about a cycle slip.

Table 10: Cycle_Slip_Param Packed Record : 112 bits

Name	**Type**	**Range**	**Size (Bits)**	**Start Bit**	**End Bit**
`Slipped_Tick`	`Tick.T`	-	96	0	95
`Num_Slips`	`Interfaces.Unsigned_16`	0 to 65535	16	96	111

Field Descriptions:

- **`Slipped_Tick`** - The tick during which the cycle slip occurred.
- **`Num_Slips`** - The number of cycle slips that have occurred.

Data_Product.T:

Generic data product packet for holding arbitrary data types

Table 11: 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_Fetch.T:

A packed record which holds information for a data product request.

Table 12: Data_Product_Fetch Packed Record : 16 bits

Name	**Type**	**Range**	**Size (Bits)**	**Start Bit**	**End Bit**
`Id`	`Data_Product_Types.Data_Product_Id`	`0 to 65535`	16	0	15

Field Descriptions:

- **`Id`** - The data product identifier

Data_Product_Header.T:

Generic data_product packet for holding arbitrary data_product types

Table 13: 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

Data_Product_Return.T:

This record holds data returned from a data product fetch request.

Table 14: Data_Product_Return Packed Record : 352 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
The_ Status	Data_ Product_ Enums. Fetch_ Status.E	0 => Success 1 => Not_Available 2 => Id_Out_Of_Range	8	0	7	-
The_Data_ Product	Data_ Product.T	-	344	8	351	-

Field Descriptions:

- **The_Status** - A status relating whether or not the data product fetch was successful or not.
- **The_Data_Product** - The data product item returned.

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 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.

Packed_F32.T:

Single component record for holding packed 32-bit floating point number.

Table 16: Packed_F32 Packed Record : 32 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Value	Short_Float	-3.40282e+38 to 3.40282e+38	32	0	31

Field Descriptions:

- **Value** - The 32-bit floating point number.

Packed_U32.T:

Single component record for holding packed unsigned 32-bit value.

Table 17: Packed_U32 Packed Record : 32 bits

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

Field Descriptions:

- **Value** - The 32-bit unsigned integer.

Parameter.T:

Generic parameter packet for holding a generic parameter

Table 18: Parameter Packed Record : 280 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Parameter_ Header.T	-	24	0	23	-
Buffer	Parameter_ Types. Parameter_ Buffer_Type	-	256	24	279	Header.Buffer_ Length

Field Descriptions:

- **Header** - The parameter header
- **Buffer** - A buffer that contains the parameter type

Parameter_Header.T:

Generic parameter header for holding arbitrary parameters

Table 19: Parameter_Header Packed Record : 24 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Id	Parameter_Types. Parameter_Id	0 to 65535	16	0	15
Buffer_Length	Parameter_Types. Parameter_Buffer_ Length_Type	0 to 32	8	16	23

Field Descriptions:

- **Id** - The parameter identifier
- **Buffer_Length** - The number of bytes used in the parameter type buffer

Parameter_Update.T:

A record intended to be used as a provide/modify connector type for updating/fetching parameters.

Table 20: Parameter_Update Packed Record : 312 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
`Table_Id`	`Parameter_Types.Parameter_Table_Id`	`0 to 65535`	16	0	15	-

`Operation`	`Parameter_Enums.Parameter_Operation_Type.E`	`0 => Stage`
		`1 => Update`
		`2 => Fetch`
		`3 => Validate`
			8	16	23	-

`Status`	`Parameter_Enums.Parameter_Update_Status.E`	`0 => Success`
		`1 => Id_Error`
		`2 => Validation_Error`
		`3 => Length_Error`
			8	24	31	-

`Param`	`Parameter.T`	-	280	32	311	-

Field Descriptions:

- **`Table_Id`** - The ID for the table that contains this parameter
- **`Operation`** - The parameter operation to perform.
- **`Status`** - The parameter return status.
- **`Param`** - The parameter that has been updated or fetched.

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.

3.3 Enumerations

The following section outlines any enumerations used in the assembly.

Data_Product_Enums.Fetch_Status.E:

This status denotes whether a data product fetch was successful.

Table 23: Fetch_Status Literals:

Name	**Value**	**Description**
Success	0	The data product was returned successfully.
Not_Available	1	No data product is yet available for the provided id.
Id_Out_Of_Range	2	The data product id was out of range.

Parameter_Enums.Parameter_Operation_Type.E:

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

Table 24: Parameter_Operation_Type Literals:

Name	**Value**	**Description**
Stage	0	Stage the parameter.
Update	1	All parameters are staged, it is ok to update all parameters now.
Fetch	2	Fetch the parameter.
Validate	3	Validate all the parameters.

Parameter_Enums.Parameter_Update_Status.E:

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

Table 25: Parameter_Update_Status Literals:

Name	**Value**	**Description**
Success	0	Parameter was successfully staged.
Id_Error	1	Parameter id was not valid.
Validation_Error	2	Parameter values were not successfully validated.
Length_Error	3	Parameter length was not correct.