Ccsds Downsampler

Component Design Document

1 Description

The CCSDS downsampler is a component that is intended to filter down packets that are listed in the downsample list. The input list has two items, one is the APID, and the other is the filter factor. The filter factor is used to know the cadence of filtering and sending packets. This is maintained by a protected binary tree object which takes the APID of the packets from the input list, and adding them to a binary tree with the filter factor. When the packet is received, the APID is checked for filtering and then the filter factor to determine if we send them on or not. Packets that are not in the input list will not be filtered and sent as normal. As a note, the larger that the downsampled list is, the more there is to check in the supporting binary tree. It's recommended that the downsampled list contain less than a couple hundred items.

2 Requirements

The requirements for the CCSDS downsampler component are specified below.

1. The component shall receive ccsds packets and forward them on if not in the APID list.
2. The component shall use an initial list of APIDs and associated filter factors and is configured at initialization.
3. The component shall filter out packets when the APID matches and the filter factor indicates as such.
4. The component shall be able to receive commands that can change the filter factor cadence.

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** - *passive*
- **Number of Connectors** - 7
- **Number of Invokee Connectors** - 2
- **Number of Invoker Connectors** - 5
- **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** - 3
- **Number of Faults** - *None*
- **Number of Data Products** - 2
- **Number of Data Dependencies** - *None*
- **Number of Packets** - *None*

3.2 Diagram

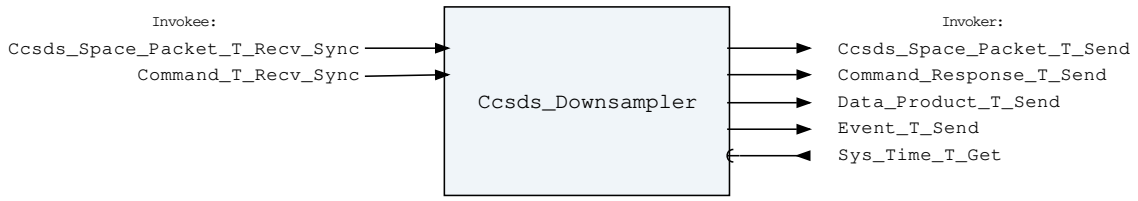

Figure 1: Ccsds Downsampler component diagram.

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: Ccsds Downsampler Invokee Connectors

Name	Kind	Type	Return_Type	Count
Ccsds_Space_Packet_T_Recv_Sync	recv_sync	Ccsds_Space_Packet.T	-	1
Command_T_Recv_Sync	recv_sync	Command.T	-	1

Connector Descriptions:

- **Ccsds_Space_Packet_T_Recv_Sync** - This connector is the input connector for the packets that are coming in. This is where packets are checked for filtering.
- **Command_T_Recv_Sync** - This is the command receive connector.

3.3.2 Invoker Connectors

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

Table 2: Ccsds Downsampler Invoker Connectors

Name	Kind	Type	Return_Type	Count
Ccsds_Space_Packet_T_Send	send	Ccsds_Space_Packet.T	-	1
Command_Response_T_Send	send	Command_Response.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:

- **Ccsds_Space_Packet_T_Send** - The connector that will forward on unfiltered packets.
- **Command_Response_T_Send** - The connector that sends a command response when received.
- **Data_Product_T_Send** - The connector for data products
- **Event_T_Send** - The Event connector to send the events specific to the component.
- **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 contains no base class initialization, meaning there is no `init_Base` subprogram for this component.

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 3: Ccsds Downsampler 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. The component achieves implementation class initialization using the `init` subprogram. The `init` subprogram requires the following parameters:

Table 4: Ccsds Downsampler Implementation Initialization Parameters

Name	Type	Default Value
`Downsample_List`	`Ccsds_Downsampler_Types.Ccsds_Downsample_Packet_List_Access`	*None provided*

Parameter Descriptions:

- **`Downsample_List`** - The list of APIDs that are to be downsampled and the initial filter factor associated with those APIDs.

3.6 Commands

These are the commands for the ccsds downsampler component.

Table 5: Ccsds Downsampler Commands

Local ID	Command Name	Argument Type
0	`Modify_Filter_Factor`	`Filter_Factor_Cmd_Type.T`

Command Descriptions:

- **`Modify_Filter_Factor`** - Modify the filter factor of a specified APID. A value of 0 will filter all packets of that ID.

3.7 Parameters

The Ccsds Downsampler component has no parameters.

3.8 Events

Below is a list of the events for the Ccsds Downsampler component.

Table 6: Ccsds Downsampler Events

Local ID	Event Name	Parameter Type
0	`Invalid_Command_Received`	`Invalid_Command_Info.T`
1	`Modified_Factor_Filter`	`Filter_Factor_Cmd_Type.T`
2	`Factor_Filter_Change_Failed_Invalid_Apid`	`Filter_Factor_Cmd_Type.T`

Event Descriptions:

- **`Invalid_Command_Received`** - A command was received with invalid parameters.
- **`Modified_Factor_Filter`** - This event indicates that the filter factor for a particular id has been set based on what was commanded.

- **Factor_Filter_Change_Failed_Invalid_Apid** - This event indicates that the command received an APID it could not find so it fails since it cannot find the ID.

3.9 Data Products

Data products for the ccsds downsampler component.

Table 7: Ccsds Downsampler Data Products

Local ID	Data Product Name	Type
0x0000 (0)	Total_Packets_Filtered	Packed_U16.T
0x0001 (1)	Total_Packets_Passed	Packed_U16.T

Data Product Descriptions:

- **Total_Packets_Filtered** - The total number of packets that have been filtered and not passed on.
- **Total_Packets_Passed** - The total number of packets that were not filtered and passed on.

3.10 Packets

The Ccsds Downsampler component has no packets.

4 Unit Tests

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

4.1 *Ccsds_Downsampler_Tests* Test Suite

This is a unit test suite for the CCSDS Downsampler component

Test Descriptions:

- **Test_Downsample_Packet** - This unit test is to test the cases when the component will downsample an APID and permutations related to the filter factor.
- **Test_Modify_Filter_Factor** - This unit test sends the command to modify a filter factor for both valid ids and invalid ids.
- **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..

Ccsds_Primary_Header.T:

Record for the CCSDS Packet Primary Header *Preamble (inline Ada definitions):*

```
subtype Three_Bit_Version_Type is Interfaces.Unsigned_8 range 0 .. 7;
type Ccsds_Apid_Type is mod 2**11;
type Ccsds_Sequence_Count_Type is mod 2**14;
```

Table 8: Ccsds_Primary_Header Packed Record : 48 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Version	Three_Bit_Version_Type	0 to 7	3	0	2
Packet_Type	Ccsds_Enums.Ccsds_Packet_Type.E	0 => Telemetry 1 => Telecommand	1	3	3
Secondary_Header	Ccsds_Enums.Ccsds_Secondary_Header_Indicator.E	0 => Secondary_Header_Not_Present 1 => Secondary_Header_Present	1	4	4
Apid	Ccsds_Apid_Type	0 to 2047	11	5	15
Sequence_Flag	Ccsds_Enums.Ccsds_Sequence_Flag.E	0 => Continuationsegment 1 => Firstsegment 2 => Lastsegment 3 => Unsegmented	2	16	17
Sequence_Count	Ccsds_Sequence_Count_Type	0 to 16383	14	18	31
Packet_Length	Interfaces.Unsigned_16	0 to 65535	16	32	47

Field Descriptions:

- **Version** - Packet Version Number
- **Packet_Type** - Packet Type
- **Secondary_Header** - Does packet have CCSDS secondary header
- **Apid** - Application process identifier
- **Sequence_Flag** - Sequence Flag

- **Sequence_Count** - Packet Sequence Count
- **Packet_Length** - This is the packet data length. One added to this number corresponds to the number of bytes included in the data section of the CCSDS Space Packet.

Ccsds_Space_Packet.T:

Record for the CCSDS Space Packet *Preamble (inline Ada definitions):*

```
use Basic_Types;
subtype Ccsds_Data_Type is Byte_Array (0 ..
↪  Configuration.Ccsds_Packet_Buffer_Size - 1);
```

Table 9: Ccsds_Space_Packet Packed Record : 10240 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Ccsds_ Primary_ Header.T	-	48	0	47	-
Data	Ccsds_Data_ Type	-	10192	48	10239	Header. Packet_Length

Field Descriptions:

- **Header** - The CCSDS Primary Header
- **Data** - User Data Field

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 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 : 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 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 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 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

Filter_Factor_Cmd_Type.T:

Defines the packed apid type the ccsds downsampler uses for the command to modify the filter factor

Table 17: Filter_Factor_Cmd_Type Packed Record : 32 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Apid	Ccsds_Primary_Header.Ccsds_Apid_Type	0 to 2047	16	0	15
Filter_Factor	Interfaces.Unsigned_16	0 to 65535	16	16	31

Field Descriptions:

- **Apid** - *No description provided.*
- **Filter_Factor** - *No description provided.*

Invalid_Command_Info.T:

Record for holding information about an invalid command

Table 18: 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_U16.T:

Single component record for holding packed unsigned 16-bit value.

Table 19: Packed_U16 Packed Record : 16 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Value	Interfaces. Unsigned_16	0 to 65535	16	0	15

Field Descriptions:
- **Value** - The 16-bit unsigned integer.

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 20: 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.

5.3 Enumerations

The following section outlines any enumerations used in the component.

Ccsds_Enums.Ccsds_Packet_Type.E:

This single bit is used to identify that this is a Telecommand Packet or a Telemetry Packet. A Telemetry Packet has this bit set to value 0; therefore, for all Telecommand Packets Bit 3 shall be set to value 1.

Table 21: Ccsds_Packet_Type Literals:

Name	Value	Description
Telemetry	0	Indicates a telemetry packet
Telecommand	1	Indicates a telecommand packet

Ccsds_Enums.Ccsds_Secondary_Header_Indicator.E:

This one bit flag signals the presence (Bit 4 = 1) or absence (Bit 4 = 0) of a Secondary Header data structure within the packet.

Table 22: Ccsds_Secondary_Header_Indicator Literals:

Name	Value	Description
Secondary_Header_Not_Present	0	Indicates that the secondary header is not present within the packet
Secondary_Header_Present	1	Indicates that the secondary header is present within the packet

Ccsds_Enums.Ccsds_Sequence_Flag.E:

This flag provides a method for defining whether this packet is a first, last, or intermediate component of a higher layer data structure.

Table 23: Ccsds_Sequence_Flag Literals:

Name	Value	Description
Continuationsegment	0	Continuation component of higher data structure
Firstsegment	1	First component of higher data structure
Lastsegment	2	Last component of higher data structure
Unsegmented	3	Standalone packet

Command_Enums.Command_Response_Status.E:

This status enumeration 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.