Ccsds Packetizer

Component Design Document

1 Description

This component converts Adamant packets into CCSDS packets, synchronously, and sends them out. Using the CCSDS standard is not required by Adamant. The primary purpose of this component is to add CCSDS packet downlink within an assembly. The conversion process includes calculating a 16-bit CRC which is appended to the end of the CCSDS packet data. Emitted CCSDS packets also include a secondary header with an 8 byte timestamp.

2 Requirements

The requirements for the CCSDS Packetizer component are specified below.

1. The component shall convert Adamant packets to CCSDS telemetry packets.
2. The component shall produce CCSDS packets that include a secondary header with an 8 byte timestamp.
3. The component shall produce CCSDS packets that include a 16-bit CRC-CCITT ($x^{16} + x^{12} + x^5 + 1$) as the last two bytes of the data section.

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** - 2
- **Number of Invokee Connectors** - 1
- **Number of Invoker Connectors** - 1
- **Number of Generic Connectors** - *None*
- **Number of Generic Types** - *None*
- **Number of Unconstrained Arrayed Connectors** - *None*
- **Number of Commands** - *None*
- **Number of Parameters** - *None*
- **Number of Events** - *None*
- **Number of Faults** - *None*
- **Number of Data Products** - *None*

- **Number of Data Dependencies** - *None*
- **Number of Packets** - *None*

3.2 Diagram

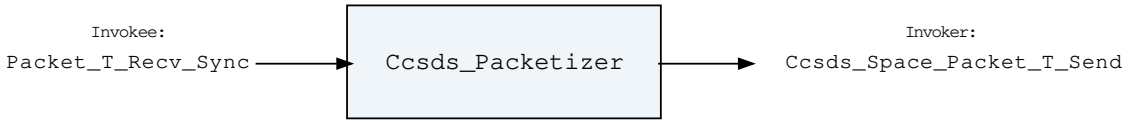

Figure 1: Ccsds Packetizer component diagram.

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 Packetizer Invokee Connectors

Name	Kind	Type	Return_Type	Count
Packet_T_Recv_Sync	recv_sync	Packet.T	-	1

Connector Descriptions:
- **Packet_T_Recv_Sync** - The packet receive connector.

3.3.2 Invoker Connectors

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

Table 2: Ccsds Packetizer Invoker Connectors

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

Connector Descriptions:
- **Ccsds_Space_Packet_T_Send** - The ccsds send connector.

3.4 Initialization

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

3.4.1 Component Instantiation

This component contains no instantiation parameters in its discriminant.

3.4.2 Component Base Initialization

This component contains no base class initialization, meaning there is no `init_Base` subprogram for this component.

3.4.3 Component Set ID Bases

This component contains no commands, events, packets, faults or data products that need base indentifiers.

3.4.4 Component Map Data Dependencies

This component contains no data dependencies.

3.4.5 Component Implementation Initialization

This component contains no implementation class initialization, meaning there is no `init` subprogram for this component.

4 Unit Tests

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

4.1 *Ccsds_Packetizer_Tests* Test Suite

This is a unit test suite for the CCSDS Packetizer component

Test Descriptions:

- **`Test_Nominal_Packetization`** - This unit test excersizes the nominal behavior of the CCSDS Packetizer, checking all emitted CCSDS packets for correctness.
- **`Test_Max_Size_Packetization`** - This unit test excersizes the packetization of a maximum sized Adamant packet into a CCSDS packet, which should succeed without issue.
- **`Test_Min_Size_Packetization`** - This unit test excersizes the packetization of a minimum sized Adamant packet into a CCSDS packet, which should succeed without issue.

5 Appendix

5.1 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 3: 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 4: 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

Packet.T:

Generic packet for holding arbitrary data

Table 5: Packet Packed Record : 10080 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Packet_ Header.T	-	112	0	111	-
Buffer	Packet_ Types.Packet_ Buffer_Type	-	9968	112	10079	Header. Buffer_Length

Field Descriptions:

- **Header** - The packet header
- **Buffer** - A buffer that contains the packet data

Packet_Header.T:

Generic packet header for holding arbitrary data

Table 6: Packet_Header Packed Record : 112 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Packet_Types. Packet_Id	0 to 65535	16	64	79
Sequence_Count	Packet_Types. Sequence_Count_Mod_ Type	0 to 16383	16	80	95
Buffer_Length	Packet_Types. Packet_Buffer_ Length_Type	0 to 1246	16	96	111

Field Descriptions:

- **`Time`** - The timestamp for the packet item.
- **`Id`** - The packet identifier
- **`Sequence_Count`** - Packet Sequence Count
- **`Buffer_Length`** - The number of bytes used in the packet buffer

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 7: 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.

5.2 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 8: 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 9: 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 10: 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