Ccsds Socket Interface

Component Design Document

1 Description

The Socket Interface Component is an interface component which connects the rest of the assembly to an outside entity (usually the ground system) via a TCP/IP socket. It spawns an internal task to listen to the socket for incoming data. It also provides an asynchronous receive connector which it services on it's task, sending any data it receives out of the socket. The data send and recieve connectors are CCSDS.

2 Requirements

No requirements have been specified for this component.

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** - 4
- **Number of Invokee Connectors** - 1
- **Number of Invoker Connectors** - 3
- **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** - 4
- **Number of Faults** - *None*
- **Number of Data Products** - *None*
- **Number of Data Dependencies** - *None*
- **Number of Packets** - *None*

3.2 Diagram

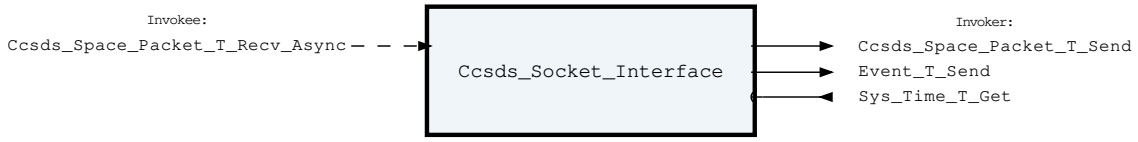

Figure 1: Ccsds Socket Interface component diagram.

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 Socket Interface Invokee Connectors

Name	Kind	Type	Return_Type	Count
Ccsds_Space_Packet_T_Recv_Async	recv_async	Ccsds_Space_Packet.T	-	1

Connector Descriptions:

- **Ccsds_Space_Packet_T_Recv_Async** - On this connector the Socket Interface Component receives data and sends it out of the socket.

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: Ccsds Socket Interface Asynchronous Connectors

Name	Type	Max Size (bytes)
Ccsds_Space_Packet_T_Recv_Async	Ccsds_Space_Packet.T	1285

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: Ccsds Socket Interface Invoker Connectors

Name	Kind	Type	Return_Type	Count

Ccsds_Space_ Packet_T_Send	send	Ccsds_Space_ Packet.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** - On this connector the Socket Interface Component sends any data it received from the socket.
- **Event_T_Send** - Events are sent out of this connector.
- **Sys_Time_T_Get** - The system time is retrieved via this connector.

3.4 Initialization

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

3.4.1 Component Subtask Instantiation

This component contains subtasks. Subtasks are distinct from the component's standard active or passive configuration. Subtasks must be initialized with their own stack, secondary stack, and execution priority during initialization. This component contains the following subtasks.

Component Subtasks:

- **Listener** - This internal task is used to listen on the socket for incomming packets.

3.4.2 Component Instantiation

This component contains no instantiation parameters in its discriminant.

3.4.3 Component Base Initialization

This component achieves base class initialization using the `init_Base` subprogram. This subprogram requires the following parameters:

Table 4: Ccsds Socket Interface 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.

3.4.4 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: Ccsds Socket Interface Set Id Bases Parameters

Name	Type
Event_Id_Base	Event_Types.Event_Id_Base

Parameter Descriptions:

- **Event_Id_Base** - The value at which the component's event identifiers begin.

3.4.5 Component Map Data Dependencies

This component contains no data dependencies.

3.4.6 Component Implementation Initialization

The calling of this implementation class initialization procedure is mandatory. This initialization subprogram connects the component to a TCP socket on the given address and port. The `init` subprogram requires the following parameters:

Table 6: Ccsds Socket Interface Implementation Initialization Parameters

Name	Type	Default Value
Addr	String	"127.0.0.1"
Port	Natural	2001

Parameter Descriptions:

- **Addr** - The IP address or hostname that the component should connect to. This could be something like 127.0.0.1 or www.google.com.
- **Port** - The port that the component should connect to.

3.5 Events

Below is a list of the events for the Ccsds Socket Interface component.

Table 7: Ccsds Socket Interface Events

Local ID	Event Name	Parameter Type
0	Socket_Connected	Socket_Address.T
1	Socket_Not_Connected	Socket_Address.T
2	Packet_Send_Failed	Ccsds_Primary_Header.T
3	Packet_Recv_Failed	Ccsds_Primary_Header.T

Event Descriptions:

- **Socket_Connected** - The socket was successfully connected on the host and port provided.
- **Socket_Not_Connected** - The socket connection failed on the host and port provided
- **Packet_Send_Failed** - Failed to send a packet over the socket because it has an invalid CCSDS header.
- **Packet_Recv_Failed** - Failed to receive a packet over the socket because it has an invalid CCSDS header.

4 Unit Tests

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

4.1 *Tests* Test Suite

This is the packet send unit test suite for the Socket Interface Component

Test Descriptions:

- **`Test_Packet_Send`** - This unit makes sure that packets sent through the component's queue are fowarded through the socket.

4.2 *Tests* Test Suite

This is the packet send unit test suite for the Socket Interface Component

Test Descriptions:

- **`Test_Packet_Receive`** - This unit test makes sure that packets received through the socket are fowarded through the send connector. This test excersizes the additional internal task of the Socket Interface Component.

4.3 *Tests* Test Suite

This is the packet send unit test suite for the Socket Interface Component

Test Descriptions:

- **`Test_Packet_Send`** - This unit makes sure that packets sent through the component's queue are fowarded through the socket.

4.4 *Tests* Test Suite

This is the packet send unit test suite for the Socket Interface Component

Test Descriptions:

- **`Test_Packet_Receive`** - This unit test makes sure that packets received through the socket are fowarded through the send connector. This test excersizes the additional internal task of the Socket Interface Component.

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 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

Event.T:

Generic event packet for holding arbitrary events

Table 10: 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 11: 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

Socket_Address.T:

This is a type that contains the IP address (host) and port number of a socket connection.

Table 12: Socket_Address Packed Record : 64 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Ip_Address	Gnat.Sockets.Inet_ Addr_V4_Type	-	32	0	31
Port	Gnat.Sockets.Port_ Type	0 to 65535	32	32	63

Field Descriptions:

- **Ip_Address** - The host or IP address number.
- **Port** - The port number for the connection.

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 13: 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.