Fault Producer

Component Design Document

1 Description

This is the fault producer component. It allows you to simulate a fault being triggered in the system by throwing a fault upon command.

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** - *passive*
- **Number of Connectors** - 5
- **Number of Invokee Connectors** - 1
- **Number of Invoker Connectors** - 4
- **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** - 3
- **Number of Faults** - 2
- **Number of Data Products** - *None*
- **Number of Data Dependencies** - *None*
- **Number of Packets** - *None*

3.2 Diagram

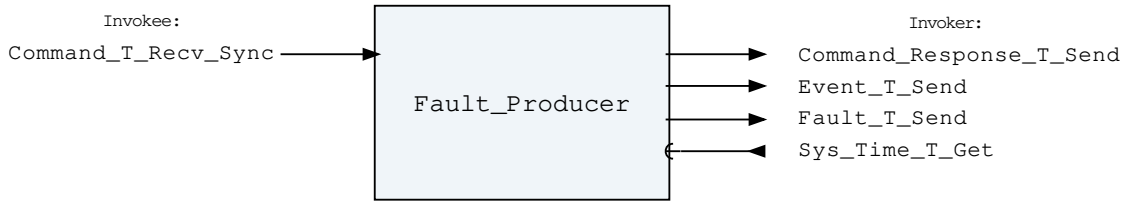

Figure 1: Fault Producer component diagram.

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: Fault Producer Invokee Connectors

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

Connector Descriptions:

- **Command_T_Recv_Sync** - The command receive connector.

3.3.2 Invoker Connectors

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

Table 2: Fault Producer Invoker Connectors

Name	Kind	Type	Return_Type	Count
Command_Response_T_Send	send	Command_Response.T	-	1
Event_T_Send	send	Event.T	-	1
Fault_T_Send	send	Fault.T	-	1
Sys_Time_T_Get	get	-	Sys_Time.T	1

Connector Descriptions:

- **Command_Response_T_Send** - This connector is used to register the components commands with the command router component.
- **Event_T_Send** - The event send connector
- **Fault_T_Send** - The fault 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 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: Fault Producer Set Id Bases Parameters

Name	Type
`Command_Id_Base`	`Command_Types.Command_Id_Base`
`Fault_Id_Base`	`Fault_Types.Fault_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.
- **`Fault_Id_Base`** - The value at which the component's fault 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

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

3.6 Commands

Commands for the fault producer component

Table 4: Fault Producer Commands

Local ID	Command Name	Argument Type
0	`Throw_Fault_1`	-
1	`Throw_Fault_2`	-

Command Descriptions:

- **`Throw_Fault_1`** - Throw the first fault.
- **`Throw_Fault_2`** - Throw the second fault.

3.7 Parameters

The Fault Producer component has no parameters.

3.8 Events

Events for the fault producer component

Table 5: Fault Producer Events

Local ID	Event Name	Parameter Type
0	Sending_Fault_1	-
1	Sending_Fault_2	-
2	Invalid_Command_Received	Invalid_Command_Info.T

Event Descriptions:

- **Sending_Fault_1** - The component received a command to send out fault 1.
- **Sending_Fault_2** - The component received a command to send out fault 2.
- **Invalid_Command_Received** - A command was received with invalid parameters.

3.9 Data Products

The Fault Producer component has no data products.

3.10 Data Dependencies

The Fault Producer component has no data dependencies.

3.11 Packets

The Fault Producer component has no packets.

3.12 Faults

Faults for the fault producer component

Table 6: Fault Producer Fault

Local ID	Fault Name	Parameter Type
0x0000 (0)	Fault_1	-
0x0001 (1)	Fault_2	Packed_Natural.T

Fault Descriptions:

- **Fault_1** - First fault that the component can send.
- **Fault_2** - Second fault that the component can send.

4 Unit Tests

None

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 7: 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 8: 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 9: 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.

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

Fault.T:

Generic fault packet for holding arbitrary faults.

Table 12: Fault Packed Record : 152 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Fault_Header.T	-	88	0	87	-
Param_Buffer	Fault_Types.Parameter_Buffer_Type	-	64	88	151	Header.Param_Buffer_Length

Field Descriptions:

- **Header** - The fault header
- **Param_Buffer** - A buffer that contains the fault parameters

Fault_Header.T:

Generic fault header.

Table 13: Fault_Header Packed Record : 88 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Fault_Types.Fault_Id	0 to 65535	16	64	79
Param_Buffer_Length	Fault_Types.Parameter_Buffer_Length_Type	0 to 8	8	80	87

Field Descriptions:

- **Time** - The timestamp for the fault.
- **Id** - The fault 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 14: 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 unkown 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_Natural.T:

Single component record for holding packed Natural value.

Table 15: Packed_Natural Packed Record : 32 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Value	Natural	0 to 2147483647	32	0	31

Field Descriptions:

- **Value** - The 32-bit Natural 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 16: 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.

Command_Enums.Command_Response_Status.E:

This status enumerations provides information on the success/failure of a command through the command response connector.

Table 17: 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.