Interrupt Listener

Component Design Document

1 Description

The Interrupt Listener allows an attached component to ask if an interrupt has occurred and receive any data that was gathered during the interrupt handling. This component contains an internal piece of data (of generic type) which is set in a custom interrupt procedure passed in at instantiation. External components can request the latest version of this data at any time. A common use for this component might be to manage a counter, where the custom procedure increments the count with each interrupt, and the requester of the data uses the count to determine if an interrupt has been received.

2 Requirements

The requirements for the Interrupt Listener component are specified below.

1. The component shall attach to a single, user-defined interrupt.
2. When an interrupt is received, the component shall pass a user-defined data type to a user-defined custom interrupt handler.
3. The component shall contain a connector which returns the calling component the latest user-defined interrupt data.

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** - 1
- **Number of Invokee Connectors** - 1
- **Number of Invoker Connectors** - *None*
- **Number of Generic Connectors** - *None*
- **Number of Generic Types** - 1
- **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

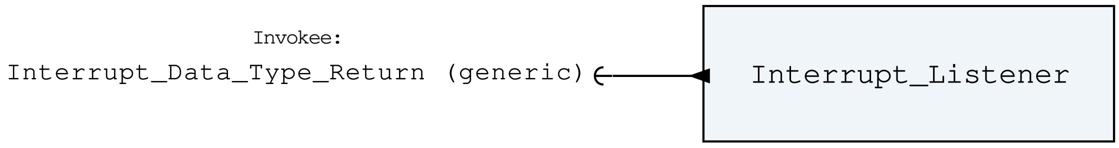

Figure 1: Interrupt Listener component diagram.

The Interrupt Listener has two functions: 1) to run a custom interrupt handler every time an interrupt is received and 2) to pass data captured during that interrupt to an external component(s) when requested. The component has a single connector, `Interrupt_Data_Type_Return` which returns data captured from the most recent interrupt to the caller.

The following diagram shows an example of how the Interrupt Listener component might operate in the context of an assembly.

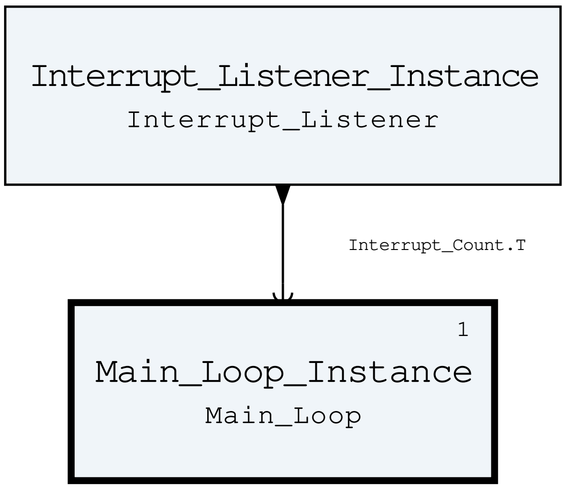

Figure 2: Example usage of the Interrupt Listener in an assembly. The Main Loop component asks the Interrupt Listener if an interrupt has been received before continuing execution.

In this example, a Main Loop component is executing and periodically asks the Interrupt Listener if an interrupt has occurred via the `Interrupt_Count.T` data type. If an interrupt has occurred, then the Main Loop component may perform a different function that iteration.

This use of the Interrupt Listener is ideal for letting connected active components know if an interrupt has occurred during their normal execution without a context switch. This makes the Interrupt Listener most useful in cases where the timing of response to an interrupt is not critical. The Interrupt Listener is also useful in single threaded systems where the Main Loop may poll on the Interrupt Listener continuously, waiting for an interrupt, before continuing execution.

Note, it is acceptable to connect many components up to the same Interrupt Listener `Interrupt_Data_Type_Return` connector, as this is a thread safe operation.

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: Interrupt Listener Invokee Connectors

Name	Kind	Type	Return_Type	Count
Interrupt_Data_ Type_Return	return	-	Interrupt_Data_ Type (generic)	1

Connector Descriptions:

- **Interrupt_Data_Type_Return** - Interrupt data generated by the custom procedure is sent back via the return type.

3.3.2 Invoker Connectors

None

3.4 Interrupts

Below is a list of the interrupts for the Interrupt Listener component.
Interrupt Descriptions:

- **Interrupt** - This component counts the number of times this interrupt occurs.

3.5 Initialization

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

3.5.1 Generic Component Instantiation

The Interrupt Listener is parameterized by the Interrupt_Data_Type. The instantiation of this type is often determined by what data needs to be collected during an interrupt and how a connected component intends to process that data. This type is a user defined type that is passed into the user's custom interrupt handler as an in-out parameter and then passed to a downstream components out of the Interrupt_Data_Type_Return connector. Usually, this type will be used to capture data related to an interrupt in the interrupt handler. This component contains generic formal types. These generic formal types must be instantiated with a valid actual type prior to component initialization. This is done by specifying types for the following generic formal parameters:

Table 2: Interrupt Listener Generic Formal Types

Name	Formal Type Definition
Interrupt_Data_Type	type Interrupt_Data_Type is private;

Generic Formal Type Descriptions:

- **Interrupt_Data_Type** - The user's custom datatype that is set in the custom interrupt handler and then passed to downstream components. If you do not foresee needing to collect specific data in the interrupt handler for the downstream component, consider using the Tick.T type, which includes a timestamp and a count.

3.5.2 Component Instantiation

This component contains the following instantiation parameters in its discriminant:

Table 3: Interrupt Listener Instantiation Parameters

Name	**Type**
`Custom_Interrupt_Procedure`	`Custom_Interrupt_Handler_Package.Interrupt_Procedure_Type`
`Interrupt_Id`	`Ada.Interrupts.Interrupt_Id`
`Interrupt_Priority`	`System.Interrupt_Priority`

Parameter Descriptions:

- **`Custom_Interrupt_Procedure`** - A custom procedure to be called within the interrupt handler. The null procedure can be used here if no specific behavior is desired.
- **`Interrupt_Id`** - Interrupt identifier number for Interrupt
- **`Interrupt_Priority`** - Interrupt priority for Interrupt

3.5.3 Component Base Initialization

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

3.5.4 Component Set ID Bases

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

3.5.5 Component Map Data Dependencies

This component contains no data dependencies.

3.5.6 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 *Tests* Test Suite

This is a unit test suite for the Interrupt Listener component. It tests the component in a Linux environment, responding to signals.

Test Descriptions:

- **`Test_Interrupt_Handling`** - This unit test sends many interrupts to the Interrupt Listener component and expects it count them correctly.

5 Appendix

5.1 Preamble

This component contains the following preamble code. This is inline Ada code included in the component model that is usually used to define types or instantiate generic packages used by the component. Preamble code is inserted as the top line of the component base package specification.

```
-- Define the custom interrupt handling package type:
package Custom_Interrupt_Handler_Package is new Interrupt_Handlers
 ↪  (Interrupt_Data_Type);
```

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..

No complex types found in component.