Tick Divider

Component Design Document

1 Description

The Tick Divider has an invokee connector meant to be called at a periodic rate. This invokee connector will usually be connected to a component which services a periodic hardware tick, or the Ticker component, which simulates a hardware tick. The Tick Divider takes this periodic rate and divides it into subrates which are divisions of the original rate. These divisors are provided via an init routine. Ticks are forwarded out the Tick_T_Send arrayed output connector according to the provided divisors. The priority of the output ticks is determined by the order of the connections in the connector array. The first connector in the array has the highest priority, and should be connected to the output that needs to run first.

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** - 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** - 1
- **Number of Commands** - *None*
- **Number of Parameters** - *None*
- **Number of Events** - 1
- **Number of Faults** - *None*
- **Number of Data Products** - *None*
- **Number of Data Dependencies** - *None*
- **Number of Packets** - *None*

3.2 Diagram

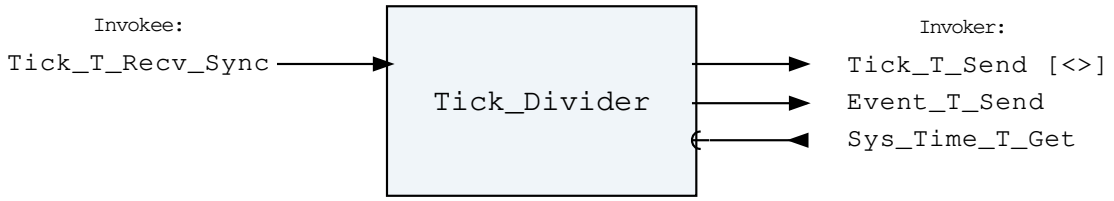

Figure 1: Tick Divider component diagram.

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: Tick Divider Invokee Connectors

Name	Kind	Type	Return_Type	Count
Tick_T_Recv_Sync	recv_sync	Tick.T	-	1

Connector Descriptions:

- **Tick_T_Recv_Sync** - This connector receives a periodic Tick from an external component.

3.3.2 Invoker Connectors

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

Table 2: Tick Divider Invoker Connectors

Name	Kind	Type	Return_Type	Count
Tick_T_Send	send	Tick.T	-	<>
Event_T_Send	send	Event.T	-	1
Sys_Time_T_Get	get	-	Sys_Time.T	1

Connector Descriptions:

- **Tick_T_Send** - This unconstrained arrayed connector is connected to downstream components which require a Tick to be scheduled. Each index of the array will be called at a rate equal to the rate at which the Tick_Recv_Sync connector is called, divided by the divisor provided during initialization.
- **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 Instantiation

This component contains no instantiation parameters in its discriminant.

3.4.2 Component Base Initialization

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

Table 3: Tick Divider Base Initialization Parameters

Name	Type
`Tick_T_Send_Count`	`Connector_Count_Type`

Parameter Descriptions:

- **`Tick_T_Send_Count`** - The size of the Tick_T_Send invoker connector array.

3.4.3 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 4: Tick Divider 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.

3.4.4 Component Map Data Dependencies

This component contains no data dependencies.

3.4.5 Component Implementation Initialization

The calling of this implementation class initialization procedure is mandatory. This initialization function is used to set the divider values for the component. An entry of 0 disables that connector from ever being invoked. The `init` subprogram requires the following parameters:

Table 5: Tick Divider Implementation Initialization Parameters

Name	Type	Default Value
`Dividers`	`Divider_Array_Type_Access`	*None provided*

Parameter Descriptions:

- **`Dividers`** - An access to an array of dividers used to determine the tick rate of each Tick_T_Send connector.

3.5 Events

Below is a list of the events for the Tick Divider component.

Table 6: Tick Divider Events

Local ID	Event Name	Parameter Type
0	Component_Has_Full_Queue	Td_Full_Queue_Param.T

Event Descriptions:

- **Component_Has_Full_Queue** - The tick divider tried to put a Tick on a component's queue, but the queue was full, so the Tick was dropped.

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 Tick Divider component.

Test Descriptions:

- **Nominal** - This unit test exercizes tick dividing with a variety of divisors, including disabled divisors. It runs the component through enough iterations to exercize the count rollover, and checks to make sure this happens.
- **Bad_Setup** - This unit test configures the component with a variety of bad input parameters and checks to make sure exceptions are thrown as expected.
- **Full_Queue** - This unit test makes sure that when a component has a full queue during scheduling the correct event is thrown.

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.

```
1 type Divider_Array_Type is array (Connector_Types.Connector_Index_Type range
   ↪  <>) of Natural;
2 type Divider_Array_Type_Access is access all Divider_Array_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..

Event.T:

Generic event packet for holding arbitrary events

Table 7: 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 8: 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

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 9: 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.

Td_Full_Queue_Param.T:

This is a type that contains useful information about a component full queue.

Table 10: Td_Full_Queue_Param Packed Record : 112 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Dropped_Tick	Tick.T	-	96	0	95
Index	Connector_Types. Connector_Index_ Type	1 to 65535	16	96	111

Field Descriptions:

- **Dropped_Tick** - The tick during which the component's queue was found to be full.
- **Index** - The tick divider index number of the component that had a full queue.

Tick.T:

The tick datatype used for periodic scheduling. Included in this type is the Time associated with a tick and a count.

Table 11: Tick Packed Record : 96 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Count	Interfaces. Unsigned_32	0 to 4294967295	32	64	95

Field Descriptions:

- **Time** - The timestamp associated with the tick.
- **Count** - The cycle number of the tick.