Rate Group

Component Design Document

1 Description

The Rate Group component is a queued component which invokes Tick connectors attached to it whenever it receives a Tick in. The tick in is intended to be periodic, allowing the component to control the execution of other components at a periodic rate. All components attached to the invoker connector of this component are said to be in a rate group, since they all execute at the same rate. Components are executed in the order they are attached to the components invoker connector. The execution of all attached connectors is expected to complete before another incoming Tick is put on the Rate Group component's queue. If the execution runs long, a cycle slip event is reported. The component also includes a connector to service (ie. "pet") a downstream software watchdog during each execution cycle. This connection is optional and may be left unconnected if not used.
Note that this component is designed to be Active in an assembly. In this way the Rate Group will provide a task on which Passive components can execute.

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** - 6
- **Number of Invokee Connectors** - 1
- **Number of Invoker Connectors** - 5
- **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** - 5
- **Number of Faults** - *None*
- **Number of Data Products** - 1
- **Number of Data Dependencies** - *None*
- **Number of Packets** - *None*

3.2 Diagram

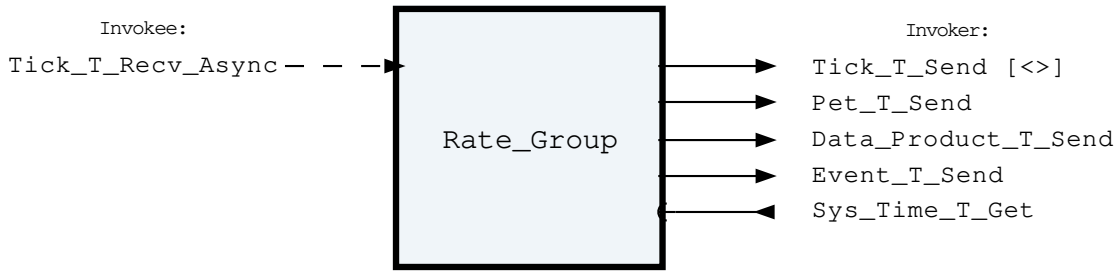

Figure 1: Rate Group component diagram.

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: Rate Group Invokee Connectors

Name	Kind	Type	Return_Type	Count
Tick_T_Recv_Async	recv_async	Tick.T	-	1

Connector Descriptions:

- **Tick_T_Recv_Async** - This connector receives a periodic Tick from an external component, usually a Ticker or Tick_Divider component.

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: Rate Group Asynchronous Connectors

Name	Type	Max Size (bytes)
Tick_T_Recv_Async	Tick.T	17

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: Rate Group Invoker Connectors

Name	Kind	Type	Return_Type	Count
Tick_T_Send	send	Tick.T	-	<>
Pet_T_Send	send	Pet.T	-	1
Data_Product_T_Send	send	Data_Product.T	-	1
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 in sequence until all connected components have been called for that cycle.
- **Pet_T_Send** - The pet send connector. This is used to service a software watchdog component.
- **Data_Product_T_Send** - Data products are sent out of this connector.
- **Event_T_Send** - Events are sent out of this 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 achieves base class initialization using the `init_Base` subprogram. This subprogram requires the following parameters:

Table 4: Rate Group Base Initialization Parameters

Name	Type
Queue_Size	Natural
Tick_T_Send_Count	Connector_Count_Type

Parameter Descriptions:

- **Queue_Size** - The number of bytes that can be stored in the component's internal queue.
- **Tick_T_Send_Count** - The size of the Tick_T_Send invoker connector array.

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 5: Rate Group Set Id Bases Parameters

Name	Type
Data_Product_Id_Base	Data_Product_Types.Data_Product_Id_Base
Event_Id_Base	Event_Types.Event_Id_Base

Parameter Descriptions:

- **Data_Product_Id_Base** - The value at which the component's data product 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

The calling of this implementation class initialization procedure is mandatory. This initialization function is used to set the number of ticks that the component should wait before producing the timing report data product. The `init` subprogram requires the following parameters:

Table 6: Rate Group Implementation Initialization Parameters

Name	Type	Default Value
Ticks_Per_Timing_Report	Interfaces.Unsigned_16	1
Timing_Report_Delay_Ticks	Interfaces.Unsigned_16	3
Issue_Time_Exceeded_Events	Boolean	False

Parameter Descriptions:

- **Ticks_Per_Timing_Report** - The period (in ticks) that the component should wait before sending a timing report data product. A value of zero prevents the component from sending the data product.
- **Timing_Report_Delay_Ticks** - The number of ticks the component waits before calculating and sending a timing report data product. It is common for the first few executions of a rate group to have execution times longer than normal due to startup logic. In this case, it is often desirable to ignore these cycles in the timing report, especially for the maximum values.
- **Issue_Time_Exceeded_Events** - If set to True, an event will be issued any time the maximum execution or wall clock time of the component is exceeded. If set to False, these events will never be issued. The same information is accessible via the component's data products, so enabling the event may become a redundant annoyance.

3.6 Commands

The Rate Group component has no commands.

3.7 Parameters

The Rate Group component has no parameters.

3.8 Events

Below is a list of the events for the Rate Group component.

Table 7: Rate Group Events

Local ID	Event Name	Parameter Type
0	Cycle_Slip	Cycle_Slip_Param.T
1	Max_Cycle_Time_Exceeded	Time_Exceeded.T
2	Max_Execution_Time_Exceeded	Time_Exceeded.T
3	Component_Has_Full_Queue	Full_Queue_Param.T
4	Incoming_Tick_Dropped	Tick.T

Event Descriptions:

- **Cycle_Slip** - Execution ran long on this cycle.
- **Max_Cycle_Time_Exceeded** - A new maximum cycle time was reached. The event parameter is a Tick type with the maximum cycle time as the Time and the cycle count where the maximum cycle was achieved as the Count.
- **Max_Execution_Time_Exceeded** - A new maximum execution time was reached. The event parameter is a Tick type with the maximum cycle time as the Time and the cycle count where the maximum cycle was achieved as the Count.
- **Component_Has_Full_Queue** - The rate group tried to put a Tick on a component's queue, but the queue was full, so the Tick was dropped.
- **Incoming_Tick_Dropped** - The rate group component's queue is full, so it cannot store the tick coming in. This usually means the rate group is cycle slipping and not running as fast as it needs to.

3.9 Data Products

Data products for the Rate Group component.

Table 8: Rate Group Data Products

Local ID	Data Product Name	Type
0x0000 (0)	Timing_Report	Task_Timing_Report.T

Data Product Descriptions:

- **Timing_Report** - Data relating to timing performance of the component.

3.10 Data Dependencies

The Rate Group component has no data dependencies.

3.11 Packets

The Rate Group component has no packets.

3.12 Faults

The Rate Group component has no faults.

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 Rate Group component.

Test Descriptions:

- **`Nominal`** - This unit test exercises Rate Group component through a nominal scenario.
- **`Cycle_Slip_Trigger`** - This unit test triggers a cycle slip and makes sure that the Rate Group component reports it.
- **`Time_Reporting`** - This unit test makes sure that the max execution and max cycle time are reported correctly.
- **`Full_Queue`** - This unit test makes sure that when a component has a full queue during scheduling the correct event is thrown.
- **`Test_Dropped_Tick`** - This unit test makes sure the proper error is thrown when the rate group component's queue gets full.

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..

Cycle_Slip_Param.T:

This is a type that contains useful information about a cycle slip.

Table 9: Cycle_Slip_Param Packed Record : 112 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
`Slipped_Tick`	`Tick.T`	-	96	0	95
`Num_Slips`	`Interfaces. Unsigned_16`	`0 to 65535`	16	96	111

Field Descriptions:

- **`Slipped_Tick`** - The tick during which the cycle slip occurred.
- **`Num_Slips`** - The number of cycle slips that have occurred.

Data_Product.T:

Generic data product packet for holding arbitrary data types

Table 10: Data_Product Packed Record : 344 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length

Header	Data_Product_ Header.T	-	88	0	87	-
Buffer	Data_Product_ Types.Data_ Product_ Buffer_Type	-	256	88	343	Header.Buffer_ Length

Field Descriptions:

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

Data_Product_Header.T:

Generic data_product packet for holding arbitrary data_product types

Table 11: Data_Product_Header Packed Record : 88 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Data_Product_Types. Data_Product_Id	0 to 65535	16	64	79
Buffer_Length	Data_Product_ Types.Data_Product_ Buffer_Length_Type	0 to 32	8	80	87

Field Descriptions:

- **Time** - The timestamp for the data product item.
- **Id** - The data product identifier
- **Buffer_Length** - The number of bytes used in the data product buffer

Delta_Time.T:

A record which holds a time difference using GPS format including seconds and subseconds.

Table 12: Delta_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.

Event.T:

Generic event packet for holding arbitrary events

Table 13: 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 14: 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

Full_Queue_Param.T:

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

Table 15: 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 rate group index number of the component that had a full queue.

Pet.T:

The pet datatype is used for servicing a watchdog. Included in this type is a count.

Table 16: Pet Packed Record : 32 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Count	Interfaces. Unsigned_32	0 to 4294967295	32	0	31

Field Descriptions:

- **Count** - The cycle number of the pet.

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 17: 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.

Task_Timing_Report.T:

Record which holds timing reports for the all-time maximum and recent maximum of an executing task

Table 18: Task_Timing_Report Packed Record : 256 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Max	Timing_Report.T	-	128	0	127
Recent_Max	Timing_Report.T	-	128	128	255

Field Descriptions:

- **Max** - The maximum recorded timing report since start up.
- **Recent_Max** - The maximum recorded timing report during some recent time interval.

Tick.T:

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

Table 19: 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.

Time_Exceeded.T:

Datatype used for reporting time deltas at a particular count number.

Table 20: Time_Exceeded Packed Record : 96 bits

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

Field Descriptions:

- **Time_Delta** - The time delta.
- **Count** - The cycle number of the tick.

Timing_Report.T:

Record which holds wall time and execution time to describe the runtime performance of some piece of code.

Table 21: Timing_Report Packed Record : 128 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Wall_Time	Delta_Time.T	-	64	0	63
Execution_Time	Delta_Time.T	-	64	64	127

Field Descriptions:

- **Wall_Time** - The wall time associated with the measurement.
- **Execution_Time** - The time the task spent executing on the CPU.

5.3 Enumerations

No enumerations found in component.