Cpu Monitor

Component Design Document

1 Description

This component produces a packet holding the CPU execution time for all tasks and interrupts configured for a particular assembly. It is provided an autocoded data structure upon initialization that contains the tasks and interrupts which it is to monitor. The packet produced contains 3 CPU execution numbers (1 bytes in size ranging from 0 - 100) for each task/interrupt, corresponding to different length time periods. The length of these time periods is also specified at initialization as multiples of the master tick driving the component.
Note that this component monitors CPU utilization by calling the Ada runtime Ada.Execution_Time.Clock subprogram which returns the amount of time since startup that a task or interrupt has been running on the CPU. The input to this subprogram is a Ada.Task_Identification id, which is provided by Adamant in an autocoded global variable for every modeled task which is passed into this component upon initialization. This interface is nonstandard, in that it exchanges information without the use of a connector. However, the use of this nonstandard interface improves efficiency and avoids having to include task identification connectors for every active component, which would be overly cumbersome.

2 Requirements

The requirements for the CPU Monitor component are specified below.

1. The component shall produce data reporting the CPU usage of every active component in an assembly.
2. The component shall produce data reporting the CPU usage of every interrupt handler in an assembly.
3. The component shall produce CPU usage data at a periodic rate that is configurable by command.

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** - 7
- **Number of Invokee Connectors** - 2
- **Number of Invoker Connectors** - 5
- **Number of Generic Connectors** - *None*

- **Number of Generic Types** - *None*
- **Number of Unconstrained Arrayed Connectors** - *None*
- **Number of Commands** - 1
- **Number of Parameters** - *None*
- **Number of Events** - 2
- **Number of Faults** - *None*
- **Number of Data Products** - 1
- **Number of Data Dependencies** - *None*
- **Number of Packets** - 1

3.2 Diagram

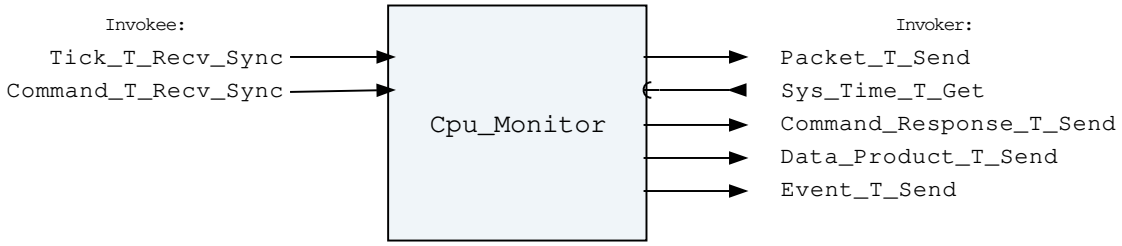

Figure 1: Cpu Monitor component diagram.

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: Cpu Monitor Invokee Connectors

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

Connector Descriptions:

- **`Tick_T_Recv_Sync`** - This is the base tick for the component.
- **`Command_T_Recv_Sync`** - This is the command receive connector.

3.3.2 Invoker Connectors

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

Table 2: Cpu Monitor Invoker Connectors

Name	Kind	Type	Return_Type	Count
Packet_T_Send	send	Packet.T	-	1

Sys_Time_T_Get	get	-	Sys_Time.T	1
Command_Response_T_Send	send	Command_Response.T	-	1
Data_Product_T_Send	send	Data_Product.T	-	1
Event_T_Send	send	Event.T	-	1

Connector Descriptions:

- **Packet_T_Send** - Send a packet of cpu execution times.
- **Sys_Time_T_Get** - The system time is retrieved via this connector.
- **Command_Response_T_Send** - This connector is used to register and respond to the component's commands.
- **Data_Product_T_Send** - Data products are sent out of this connector.
- **Event_T_Send** - Events are sent out of 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 contains no base class initialization, meaning there is no `init_Base` subprogram for this component.

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 3: Cpu Monitor Set Id Bases Parameters

Name	Type
Command_Id_Base	Command_Types.Command_Id_Base
Data_Product_Id_Base	Data_Product_Types.Data_Product_Id_Base
Event_Id_Base	Event_Types.Event_Id_Base
Packet_Id_Base	Packet_Types.Packet_Id_Base

Parameter Descriptions:

- **Command_Id_Base** - The value at which the component's command identifiers begin.
- **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.
- **Packet_Id_Base** - The value at which the component's unresolved packet 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 component requires a list of interrupts and tasks ids to monitor. The `init` subprogram requires the following parameters:

Table 4: Cpu Monitor Implementation Initialization Parameters

Name	Type	Default Value
`Task_List`	`Task_Types.Task_Info_List_Access`	*None provided*
`Interrupt_List`	`Interrupt_Types.Interrupt_Id_List_Access`	*None provided*
`Execution_Periods`	`Execution_Periods_Type`	`[1, 6, 30]`
`Packet_Period`	`Interfaces.Unsigned_16`	`1`

Parameter Descriptions:

- **`Task_List`** - A list of task info records to monitor.
- **`Interrupt_List`** - A list of interrupt ids to monitor.
- **`Execution_Periods`** - The period (in ticks) that specify the duration of time that each CPU measurement is taken over.
- **`Packet_Period`** - The period (in ticks) of how often to send out the cpu execution packet. A value of zero disables sending of the packet.

3.5 Commands

These are the commands for the CPU Monitor component.

Table 5: Cpu Monitor Commands

Local ID	Command Name	Argument Type
0	`Set_Packet_Period`	`Packed_U16.T`

Command Descriptions:

- **`Set_Packet_Period`** - Set the period of the packet. A period of zero disables the sending of the packet.

3.6 Events

Below is a list of the events for the Cpu Monitor component.

Table 6: Cpu Monitor Events

Local ID	Event Name	Parameter Type
0	`Packet_Period_Set`	`Packed_U16.T`
1	`Invalid_Command_Received`	`Invalid_Command_Info.T`

Event Descriptions:

- **`Packet_Period_Set`** - A command was received to change the packet period.
- **`Invalid_Command_Received`** - A command was received with invalid parameters.

3.7 Packets

Packets for the cpu monitor.

Table 7: Cpu Monitor Packets

Local ID	Packet Name	Type
0x0000 (0)	Cpu_Usage_Packet	*Undefined*

Packet Descriptions:

- **Cpu_Usage_Packet** - This packet contains cpu usage numbers for tasks and interrupts in the system.

4 Unit Tests

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

4.1 *Cpu_Monitor_Tests* Test Suite

This is a unit test suite for the CPU Monitor component. Testing the actual correctness of the produced CPU monitor packet is not easy to do at the unit test level. This will be done at an integrated test level. The unit tests below make sure commanding and packet generation of the component works as expected.

Test Descriptions:

- **Test_Packet_Period** - This unit test exercises the command to change the packet creation rate.
- **Test_Invalid_Command** - This unit test makes sure an invalid command is reported and ignored.

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 -- This type holds the number of ticks
2 type Num_Measurement_Periods is range 0 .. 2;
3 type Execution_Periods_Type is array (Num_Measurement_Periods) of Positive;
```

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 8: 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 that contains the command arguments

Command_Header.T:

Generic command header for holding arbitrary commands

Table 9: 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 10: 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.

Data_Product.T:

Generic data product packet for holding arbitrary data types

Table 11: 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 12: 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

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

Invalid_Command_Info.T:

Record for holding information about an invalid command

Table 15: 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 unknown 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_U16.T:

Single component record for holding packed unsigned 16-bit value.

Table 16: Packed_U16 Packed Record : 16 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Value	Interfaces. Unsigned_16	0 to 65535	16	0	15

Field Descriptions:

- **`Value`** - The 16-bit unsigned integer.

Packet.T:

Generic packet for holding arbitrary data

Table 17: Packet Packed Record : 10080 bits *(maximum)*

Name	Type	Range	Size (Bits)	Start Bit	End Bit	Variable Length
Header	Packet_ Header.T	-	112	0	111	-
Buffer	Packet_ Types.Packet_ Buffer_Type	-	9968	112	10079	Header. Buffer_Length

Field Descriptions:

- **`Header`** - The packet header
- **`Buffer`** - A buffer that contains the packet data

Packet_Header.T:

Generic packet header for holding arbitrary data

Table 18: Packet_Header Packed Record : 112 bits

Name	Type	Range	Size (Bits)	Start Bit	End Bit
Time	Sys_Time.T	-	64	0	63
Id	Packet_Types. Packet_Id	0 to 65535	16	64	79

Sequence_Count	Packet_Types. Sequence_Count_Mod_ Type	0 to 16383	16	80	95
Buffer_Length	Packet_Types. Packet_Buffer_ Length_Type	0 to 1246	16	96	111

Field Descriptions:

- **Time** - The timestamp for the packet item.
- **Id** - The packet identifier
- **Sequence_Count** - Packet Sequence Count
- **Buffer_Length** - The number of bytes used in the packet 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 19: 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.

Tick.T:

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

Table 20: 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.

5.3 Enumerations

The following section outlines any enumerations used in the component.

Command_Enums.Command_Response_Status.E:

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

Table 21: 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.