Real Time Programming - Assignment 1

Project Proposal

Authors: Matthew Hannah (6963838), Rhys Hill (6964060)
Reviewed by: Matthew Hannah, Rhys Hill
Date: 5/04/2016
Last Update: 14/04/2016

Proposed Project Title

Train Project

Scope of Works

Our student team will complete a number of activities in accordance to the 2015 Train Project. This will involve the implementation of a number of software functions. Some of these will create an interface to work with the hardware, namely the DIO192, DDA06/Jr, DAS08/Jr, and INT32. These software functions will provide a framework in which trains can be driven in real-time without fear of crashing. This will be done by means of real-time scheduling; handling sensor events and acting accordingly. However, as the system is required to be ‘crash free’ or in better terms, implement a hard real-time methodology, the system must be designed to run within the bounds of deadlines and timing constraints; implemented programmatically but with the assistance of state transition diagrams. The train system should also have have the capability of running 4 trains at once. This means that concurrent processes must exhibit mutual exclusion as to not cause any system errors. Furthermore, the use of barriers, semaphores and protected objects, will assist in the concurrent aim.

Relevance to RTP learning outcomes

This is a table which maps the learning outcomes to the activities and/or achievements involved with the train project. This is to clearly demonstrate that the project completion will meet the requirements of the unit.

Learning Outcomes	**Activities/Achievements**
1. Describe the behaviour of concurrent processes including the major states and the characteristics of real-time systems.	This project works with sporadics and will therefore enhance our learning of concurrent processes and the major states involved in real-time systems.
2. Describe the implementation of real-time systems on single processor systems, including interrupts, real-time clock, pre-emption and time-slicing.	As this project is the implementation of a real-time system it will increase our understanding of them. We will make use of interrupts to detect sensor changes and a real-clock to monitor deadlines.
3. Determine sample actual response times and predict worst-case response times using relevant graphical representations and formulae.	This project contains multiple measurable response times including; user input command response for commands like dac, block, polarity, sensor change detection.
4. Design hard real-time software using a suitable design methodology.	The timestamps used in this project will be used to ensure the meeting of all deadlines.
5. Analyse a specification of a complex hardware system requiring real-time control and identify required functionality and parameters.	The project builds on a partially completed real-time system. This forces us to analyze the existing system so that we can extend its functionality. We will also identify other required functionality and need to be able to fit this within the existing framework
6. Construct control software for a complex real-time system in a suitable high-level language.	Ada will be used as it is a high level language which has the appropriate real-time functionality
7. Describe in detail the features of a high-level language for programming real-time systems.	The use of Ada for such a complex system will force us to learn its features. We will therefore be able to describe them
8. Compare two programming languages &/or operating systems with respect to programming real-time systems.	Both team members have previous experience with concurrency in java. Learning Ada through this project will give us a basis for comparison

Software Requirements Specification

This is a list of each software function we are implementing into the final product, each will include where applicable; a description, trigger, precondition, postcondition, dependencies, inputs, outputs, and exceptions.

Set_Voltage

Description
Used to interface with the DDA06/Jr digital to analogue card, the set_voltage function sets the reference voltage for the CABs

Trigger
- Dac_Command

Precondition
- The system is active
- The hardware is active
- The input data is valid

Postconditions
- The voltage value assigned to the correct dac and is displayed on the software screen

Dependencies
- Railsdefs
- Dda06defs
- Unsigned_Types
- IO_Ports

Inputs
- Dac (Dac_Id) - Id of the digital to analogue converter
- Voltage_Value (Unsigned_8) - Voltage value

Outputs
- IO Write of the voltage value to the specified dac (port) on DDA06/Jr card

Exceptions
N/A

b_command

Description
Used to process block commands entered into the system or by the system; the function will validate the data entered

Trigger
- User enters ‘d’ during a dialogue loop
- System calls b_command

Preconditions
- The system is active

Postconditions
- The software functions for interfacing with the DIO192 board are run

Dependencies
- Ada.Text_IO
- block_driver

Inputs
- Block (Integer) - the block number
- Polarity (Character) - polarity of the block; positive or negative
- Cab (Character) - the cab number

Outputs
- Set_Cab_And_Polarity

Exceptions
- Block value entered is not in range of 1..24 => output “command ignored”
- Polarity entered is not characters ‘+’ or ‘-’ => output “command ignored”
- Cab entered is not characters in range of 1..4 => output “command ignored”

Set_Cab

Description
Used to interface with the DIO192 board, the function will set the cab number for a specific block.

Trigger
- Set_Cab_And_Polarity

Preconditions
- The system is active
- The hardware is active
- The input data is valid

Postconditions
- The cab is assigned to the correct block and is displayed on the screen

Dependencies
- Raildefs
- Dio192defs
- Unsigned_Types
- IO_Ports

Inputs
- Block (Integer) - the block number
- Cab (Character) - the cab number

Outputs
- IO Write of the cab value to the specified block (port) on DIO192 board

Exceptions
N/A

Set_Polarity

Description
Used to interface with the DIO192 board, the function will set the polarity number for a specific block.

Trigger
- Set_Cab_And_Polarity

Preconditions
- The system is active
- The hardware is active
- The input data is valid

Postconditions
The polarity is assigned to the correct block and is displayed on the screen

Dependencies
- Raildefs
- Dio192defs
- Unsigned_Types
- IO_Ports

Inputs
- Block (Integer) - the block number
- Polarity (Character) - polarity of the block; positive or negative

Outputs
- IO Write of the polarity value to the specified block (port) on DIO192 board

Exceptions
N/A

Set_Cab_And_Polarity

Description
Used to interface with the DIO192 board, the function will set the cab number and polarity for a specific block.

Trigger
- b_command

Preconditions
- The system is active
- The input data is valid

Postconditions
- Software functions set_cab and set_polarity are run

Dependencies
N/A

Inputs
- Block (Integer) - the block number
- Polarity (Character) - polarity of the block; positive or negative
- Cab (Character) - the cab number

Outputs
- Set_Cab
- Set_Polarity

Exceptions
N/A

s_command

Description
Used to process straight commands for turnouts entered into the system or by the system; the function will validate the data entered

Trigger
- User enters ‘s’ during a dialogue loop
- System calls s_command

Preconditions
- The system is active

Postconditions
- The software functions for interfacing with the DIO192 board are run

Dependencies
- Ada.Text_IO
- Turnout_Driver

Inputs
Turnout (Integer) -the turnout being straightened

Outputs
- Straight_Turnout

Exceptions
- Turnout value entered is not in range of 1..19 => output “command ignored”

t_command

Description
Used to process turn commands for turnouts entered into the system or by the system; the function will validate the data entered

Trigger
- User enters ‘t’ during a dialogue loop
- System calls t_command

Preconditions
- The system is active

Postconditions
- The software functions for interfacing with the DIO192 board are run

Dependencies
- Ada.Text_IO
- Turnout_Driver

Inputs
Turnout (Integer) -the turnout being straightened

Outputs
- Turn_Turnout

Exceptions
- Turnout value entered is not in range of 1..19 => output “command ignored”

Straight_Turnout

Description
Used to interface with the DIO192 card, the straight_turnout function sets a specified turnout to its straightened position.

Trigger
- s_command

Precondition
- The system is active
- The hardware is active
- The input data is valid

Postconditions
- The turnout is set to straight and is displayed on the software screen

Dependencies
- Railsdefs
- Dio192defs
- Unsigned_Types
- IO_Ports

Inputs
- Turnout (Integer) - The number of the turnout

Outputs
- IO Write of the value to the specified port on DIO192 card

Exceptions
N/A

Turn_Turnout

Description
Used to interface with the DIO192 card, the straight_turnout function sets a specified turnout to its turned position.

Trigger
- t_command

Precondition
- The system is active
- The hardware is active
- The input data is valid

Postconditions
- The turnout is set to turned and is displayed on the software screen

Dependencies
- Railsdefs
- Dio192defs
- Unsigned_Types
- IO_Ports

Inputs
- Turnout (Integer) - The number of the turnout

Outputs
- IO Write of the value to the specified port on DIO192 card

Exceptions
N/A

o_command

Description
Used to process oval commands entered into the system or by the system; the function will validate the data entered

Trigger
- User enters ‘o’ during a dialogue loop
- System calls o_command

Preconditions
- The system is active

Postconditions
- The software functions for interfacing with the DIO192 and DDA06/Jr boards are run

Dependencies
- Ada.Text_IO
- Oval_Driver

Inputs
N/A

Outputs
- Oval

Exceptions
N/A

Oval

Description
Sets up an oval for train 2. Note by default when 3 trains are selected, Simrail2.Reset places train 2 on block 12. Then the d command will allow you to demonstrate train 2 going around the oval (fixed direction).

Trigger
- o_command

Preconditions
- System is active
- Hardware is active

Postconditions
- Train 2 runs around the track in an oval
- Other trains move out of the way

Dependencies
- Dac_Driver
- Block_Driver
- Turnout_Driver

Inputs
N/A

Outputs
- Dac_commands
- B_commands
- S_commands
- T_commands

Exceptions
N/A

Init_Time_Stamp

Description
Function that will initialize the clock timer

Trigger
- System calls ‘init_time_stamp’ command

Preconditions
- System is active

Postconditions
- Clock timer is initiated and will begin counting

Dependencies
- Ada.Real_Time

Inputs
N/A

Outputs
- Clock timer start

Exceptions
N/A

Time_Stamp

Description
Function that will return a string containing the current runtime of the program

Trigger
- System calls ‘time_stamp’ command

Preconditions
- System is active
- T0 is initiated (timer)

Postconditions
- Timestamp is returned and displayed on the screen

Dependencies
- Ada.Text_IO
- Ada.Real_Time

Inputs
N/A

Outputs
- Timestamp

Exceptions
N/A

Sporadic_Op

Description

Implements the handling of a bounded queue of data (sensor events); worker threads

Trigger

- Data entering the sporadic

Preconditions

N/A

Postconditions

- Data is consumed

Dependencies

- Ada.Real_Time

Inputs

- Sensor Data
- User requests

Outputs

- Timestamp at completion
- Worker thread consumption

Exceptions

N/A

B_command

Description
Used to process bell commands entered into the system or by the system; the function will validate the data entered

Trigger
- User enters ‘B’ during a dialogue loop
- System calls B_command

Preconditions
- The system is active

Postconditions
- The software functions for interfacing with the DAS08 boards are run

Dependencies
- Ada.Text_IO
- Sound_Manager

Inputs
Cab (Integer) - The number of the train

Outputs
- Sound_Bell

Exceptions
- Turnout value entered is not in range of 1..4 => output “command ignored”

h_command

Description
Used to process horn commands entered into the system or by the system; the function will validate the data entered

Trigger
- User enters ‘h’ during a dialogue loop
- System calls h_command

Preconditions
- The system is active

Postconditions
- The software functions for interfacing with the DAS08 board are run

Dependencies
- Ada.Text_IO
- Sound_Manager

Inputs
Cab (Integer) - The number of the train

Outputs
- Sound_Horn

Exceptions
- Turnout value entered is not in range of 1..4 => output “command ignored”

Sound_Bell

Description
Used to interface with the DAS08 card, the sound_bell function toggles the bell between on and off for a train.

Trigger
- s_command

Precondition
- The system is active
- The hardware is active
- The input data is valid

Postconditions
- The bell of a train is toggled between on and off

Dependencies
- Railsdefs
- Das08defs
- Unsigned_Types
- IO_Ports

Inputs
- Cab (Integer) - The number of the train

Outputs
- IO Write of the value to the specified port on DAS08 card

Exceptions
N/A

Sound_Horn

Description
Used to interface with the DAS08 card, the sound_horn function toggles the horn between on and off for a train.

Trigger
- h_command

Precondition
- The system is active
- The hardware is active
- The input data is valid

Postconditions
- The horn of a train is toggled between on and off

Dependencies
- Railsdefs
- Das08defs
- Unsigned_Types
- IO_Ports

Inputs
- Cab (Integer) - The number of the train

Outputs
- IO Write of the value to the specified port on DAS08 card

Exceptions
N/A

e_command

Description
Used to process figure eight commands entered into the system or by the system; the function will validate the data entered

Trigger
- User enters ‘e’ during a dialogue loop
- System calls e_command

Preconditions
- The system is active

Postconditions
- The software functions for interfacing with the DIO192 and DDA06/Jr boards are run

Dependencies
- Ada.Text_IO
- Figure_Eightl_Driver

Inputs
N/A

Outputs
- Figure_Eight

Exceptions
N/A

Figure_Eight

Description
Sets up a figure 8 for train 2. Note by default when 3 trains are selected, Simrail2.Reset places train 2 on block 12. Then the d command will allow you to demonstrate train 2 going around the figure eight(fixed direction).

Trigger
- e_command

Preconditions
- System is active
- Hardware is active

Postconditions
- Train 2 runs around the track in a figure eight
- Other trains move out of the way

Dependencies
- Dac_Driver
- Block_Driver
- Turnout_Driver

Inputs
N/A

Outputs
- Dac_commands
- B_commands
- S_commands
- T_commands

Exceptions
N/A

r_command

Description
Used to process the reverse direction commands entered into the system or by the system; the function will validate the data entered

Trigger
- User enters ‘r’ during a dialogue loop
- System calls r_command

Preconditions
- The system is active

Postconditions
- The software functions for interfacing with the DIO192 and DDA06/Jr boards are run

Dependencies
- Ada.Text_IO
- Figure_Eightl_Driver

Inputs
N/A

Outputs
- Reverse_Direction

Exceptions
N/A

Reverse_Direction

Description
Alternates the direction traveled by the trains during the oval or figure_eight functions.

Trigger
- r_command

Preconditions
- System is active
- Hardware is active

Postconditions
- Train travels opposite to its previous direction

Dependencies
- Dac_Driver
- Block_Driver
- Turnout_Driver

Inputs
N/A

Outputs
- Dac_commands
- B_commands
- S_commands
- T_commands

Exceptions
N/A

Sensor_Register

Description
Checks the state of all sensors and passes on changes to a sporadic, then updates the current array of sensor values.

Trigger
- Sensor change

Preconditions
- System is active
- Hardware is active

Postconditions
- Changed bits are passed onto a sporadic

Dependencies
- Int32defs
- Analyze
- Current_Sensor_Array

Inputs
- Sensor Values

Outputs
- New_Sensor_Array

Exceptions
N/A

Analyze

Description
Compares the current and new values of the sensor then returns which ones have changed.

Trigger
- Called by Sensor_Register

Preconditions
- System is active

Postconditions
- Changed bits are returned to Sensor_Register

Dependencies
- Int32defs
- Analyze
- Current_Sensor_Array

Inputs
- New_Sensor_Array

Outputs
- Changed_Sensor_Array

Exceptions
N/A