May 18, 2016

Project 2B Technical Report

EEEE-663 Real-Time and Embedded Systems

Author(s)	**:**	**Vyoma Sharma, Siddharth Ramkrishnan**
Email ID	**:**	**sg5232@rit.edu, sxr4316@rit.edu**

Table of Contents

1. Project Requirements

Project 2B requires the design and implementation a program exhibiting multitasking characteristics in simultaneously controlling a pair of servo motors using a custom interpreted control language. The system will be responsive to simultaneous independent, externally provided commands. The servo positions are controlled with pulse-width modulation (PWM). Each servo will be controlled by its own recipe of commands in the custom interpreted control language. The servos can be positioned to any of six positions (0, 1, 2, 3, 4, 5) approximately evenly spaced along the servos' potential 160-degree arc. Position 0 is at the extreme clockwise position, and Position 5 is at the extreme counter clockwise position.

1.1. Graduate Student extensions

There are three unused recipes "opcodes" in the command descriptions above. Design one additional command and use one of the unused opcodes. Your report must document its operation, the limiting factors and parameters, etc. Add a new user command if appropriate. The new opcode must be included in the demo recipes to clearly demonstrate this new command's capabilities and limitations.

Table 1. Opcodes required to be implemented

Mnemonic	Opcode	Parameter	Range	Comment
MOV	001XXXXX	Target Position	0 - 5	An Out of range parameter produces an error
WAIT	010XXXXX	Number of 0.1s delay before attempting to execute the next recipe command	0 - 31	Actual delay would be 0.1 s more than provided value
LOOP_START	100XXXX	The number of additional times to execute the following recipe block	0-31	A parameter value of "n" will execute the block once but will repeat it "n" more times
END_LOOP	101XXXXX	Not applicable	-	Marks the end of a recipe loop block
RECIPE_END	000XXXXX	Not applicable	-	-

Table 2. User Input Commands

Command	**Mnemonic**	**Comment**
Pause Recipe Execution	P or p	Not operative after recipe end or error
Continue Recipe execution	C or c	Not operative after recipe end or error
Move 1 position to the right if possible	R or r	Not operative if recipe isn't paused or at extreme right position
Move 1 position to the left if possible	L or l	Not operative if recipe isn't paused or at extreme left position
No-op no new override entered for selected servo	N or n	-
Restart the recipe execution	B or b	Starts the recipe's execution immediately

2. Abstract

The design and implementation, of a pair of simultaneous executable PWM controlled servo motors, requires the knowledge of programming guidelines, basic programming and logic concepts, knowledge of Athena QNX platform hardware, DIO configuration, timers and Interrupt Configuration. The system consists of three primary functions running concurrently, namely the routine evaluation for *motor A*, routine evaluation for *motor B* and function to gather user input data and threads to periodically toggle the DIO output port according to determined duty cycle and period. Based on the status information provided by these functions, the required PWM duty cycles pulses and DIO port strobes are generated to actuate the Servo Motors and status indicator LEDs respectively.

3. Introduction

The program to be implemented requires an algorithm and system design that is capable of accepting predetermined recipes to be evaluated for each servo motor independent of the other. The system should be able to interface with the memory maps to evaluate the corresponding routine when the interrupt is triggered, and DIO ports to control the servo motors and to indicate the status of the system to the user. The system should also be capable of interacting with the Serial Data Communication ports to communicate with the user display monitor and accept keyboard inputs from the user.

The proposed design uses a three function approach, with the two functions operating in a round robin fashion for evaluating the routines and two threads controlling the PWM generation and the status LED displays; and an auxiliary main program, which runs at a lower priority and collects user input and performs the necessary control actions to influence the routine evaluation.

3.1. Graduate Extension

The graduate program, requires the implementation of some additional custom commands be implemented to the interpretation language. For the purpose of this project, the instructions, *SKIP* and FLIP opcodes have been implemented

Table 3. Graduate Extension for Opcode

Mnemonic	Opcode	Parameter	Range	Comment
FLIP	011XXXXX	No Parameter required	-	-
SKIP	110XXXXX	Number of instructions to be skipped	0 - 31	Skips ‘n’ instruction or RECIPE_END whichever occurs earlier

- ***FLIP Opcode***: does not require any parameter, it shifts the servo motor position to the mirror position on the opposite side of the working region of the motor (clockwise ↔ counter clockwise).

- ***SKIP Opcode***: instructs the program to jump a certain number of instructions as indicated by the parameter. The control skips the parameter number of steps, or end of recipe, whichever occurs earlier. The instruction would also terminate any loop if the END_LOOP instruction is skipped.

4. Area of Focus

The system design was primarily partitioned into two components:

- Hardware (Primary Responsibility: Vyoma Sharma)
  - Interrupt configuration,
  - PWM configuration and Pulse generation,
  - GPIO configuration and usage.
  - Grad Extension logic design

- Software (Primary Responsibility : Siddharth Ramkrishnan)
  - Algorithm and design flow
  - Implementation of routines
  - Evaluation of Opcodes
  - Synchronizing the various function
  - Handshaking between function and routine recipe evaluation control

The project implementation was a joint work between the team members, with the responsible person performing any additional study and looking for solutions to specific problems identified during the team discussion and coding sessions.

5. System Block Diagram

The proposed design and implementation is intended to trigger subroutines delay functions supported by the QNX target environment. The evaluation routines, are executed cyclically with a 100ms delay, intended to concurrently evaluate the recipes marked for the individual servos. The main () program, would operate on an infinite loop accepting data from the user, while simultaneously being interrupted, for the recipe to be evaluated. This allows the system to continually operate, without having to wait for user input, or timing out the user input functions.

5.1.Hardware Architecture

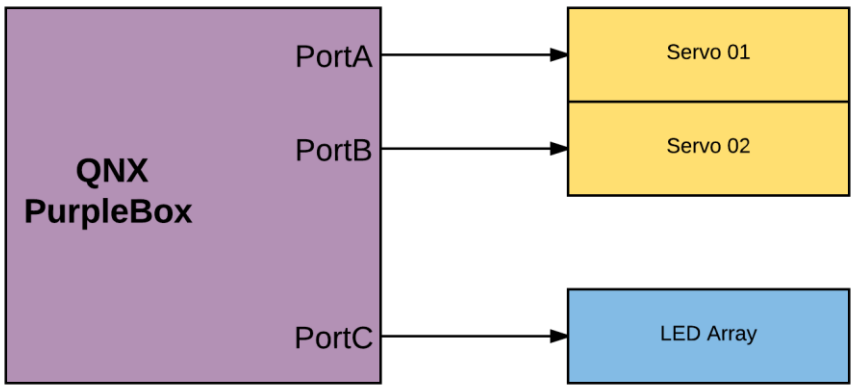

Figure 1. Hardware Architecture of implementation

The implementation utilizes PortA and PortB DIO ports to generate the signal simulating a PWM pulse to control the servo motors. General Purpose Input Output Port PORTC is used to control the LEDs to indicate the stats of the system to the user.

5.2.Software Architecture

The system implementation consists of three main functions and multiple sub-functions.

The main () function has the lowest priority, it initializes the system and loads the recipes for evaluation. It initializes the hardware configuration registers according to the required functionality. The control the loops infinitely between data acquisition from the user and triggering the corresponding control actions on the servo recipe evaluation functions.

The ServoRoutineEval () function checks the control and error status and then evaluates every step of the recipe. If recipe evaluation is permitted, depending on the identified Opcode, specific sub-routines are called. Suitable enable conditions, error diagnosis conditions are added to all functions. Appropriate flags are added in sub-routines that take more than one instruction cycle to complete evaluation (e.g. Move, Wait…). The ServoRoutineEval () function provides the error / system status to be displayed in the LED and the required servo motor position to be actuated.

Separate functions are present, which read the System/Error Status and Servo Position provided by the ServoRoutineEval() function and update the corresponding registers to generate the required signals to be communicated.

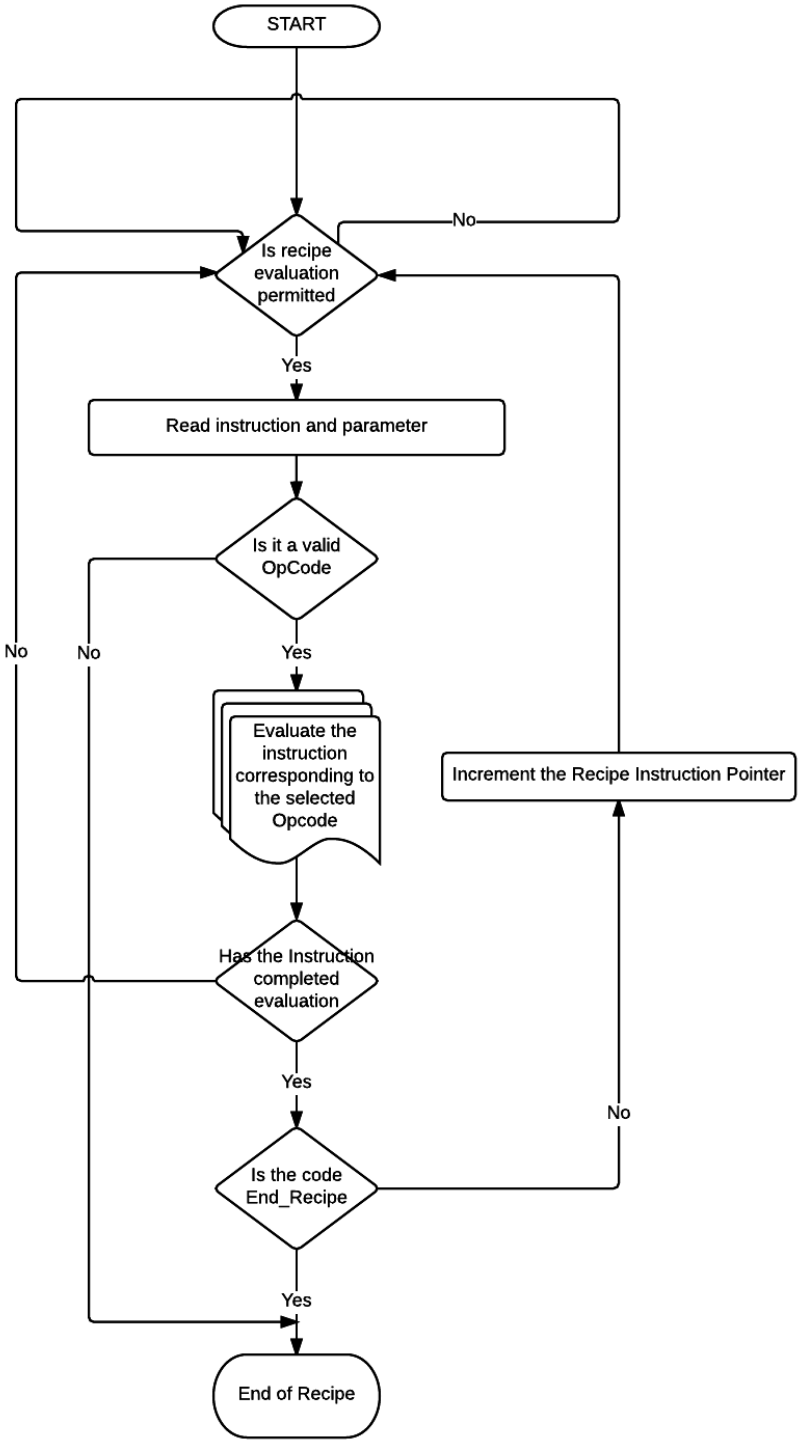

Figure 2. Control diagram for Recipe Evaluation

6. System Operation and Behaviour

Table 4. Complete list of Opcodes implemented

Mnemonic	Opcode	Parameter	Range	Comment
RECIPE_END	000XXXXX	Not applicable	-	-
MOV	001XXXXX	Target Position	0 - 5	An Out of range parameter produces an error
WAIT	010XXXXX	Number of 0.1s delay before attempting to execute the next recipe command	0 - 31	Actual delay would be 0.1 s more than provided value
FLIP	**011XXXXX**	**No Parameter required**	**-**	**-**
LOOP_START	100XXXX	The number of additional times to execute the following recipe block	0-31	A parameter value of "n" will execute the block once but will repeat it "n" more times
END_LOOP	101XXXXX	Not applicable	-	Marks the end of a recipe loop block
SKIP	**110XXXXX**	**Number of instructions to be skipped**	**0 - 31**	**Skips 'n' instruction or RECIPE_END whichever occurs earlier**
UNUSED	111XXXXX	-	-	-

7. Simulation Results and System Output

The system was evaluated for the provided routine and error routine. The graduate extensions were implemented and verified. The clock delay for the instructions is evaluated to be 100 ms, (three instances of Wait+31 was evaluated to have 9.3 seconds). A stuttering of the PWM arm was noted due to irregularity in the PWM signal generated.

8. Conclusion

The project was a good and useful introduction to implementation of complete system model in an embedded environment. Challenging parts of the project was the configuration of the hardware peripherals and the algorithm design. Some of the major challenges involved, include the exception handling when errors are detected and the recipe handling for various different error information and control instructions from the user.