Rochester Institute of Technology

Real Time and Embedded Systems: Project 2a

Overview:

Design and implement a STM32 Discovery board 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). This first portion of the project utilizes the STM32 Discovery board. Each servo will be controlled by its own recipe of commands in the custom interpreted control language.

Servo and PWM Overview:

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 counterclockwise position.

Typical Futaba position control servos require a PWM control signal with a 20 millisecond period. The duty cycle of this PWM signal controls the servo through its range of positions. The typical duty cycle ranges from 2% to over 10% for full range of positions.

Mandatory Demo of PWM Signal:

Before you will be allowed to connect to a servo motor you MUST demonstrate that your code produces a valid PWM signal. You will demonstrate this using an oscilloscope.

Command Language:

As memory space on microcontrollers is often at a premium, each of the recipe commands are encoded into one byte. These recipes are typically implemented as an array of byte values.

A master timer generates interrupts every 100 milliseconds, serving as the time basis the wait commands below.

Command Format:

command opcode	command parameter
3 bits wide	5 bits wide

Recipe Commands:

Mnemonic	Opcode	Parameter	Range	Comments
MOV	001	The target position number	0..5	An out of range parameter value produces an error. *
WAIT	010	The number of 1/10 seconds to delay before attempting to execute next recipe command	0..31	The actual delay will be 1/10 second more than the parameter value.. **
LOOP_START	100	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	101	Not applicable		Marks the end of a recipe loop block.
RECIPE_END	000	Not applicable		

 (Other opcodes such as 011, 110 and 111 are illegal, and if encountered at run time, transition the task into an error state.)

*An out-of-range parameter causes the task to enter an error state. The error state inhibits the operation of the "Continue" override command (see below), but all other overrides remain functional.

** The parsing of a "wait" command (as all other commands) takes a full system clock cycle (e.g. 1/10 second) to execute. Thus a wait + 0 skips 1/10 of a second before execution of the next command, a wait+1 would takes 2/10 seconds, etc.

*** Nested loops are not permissible and places the task in an error state
.
Status:

Display the status of each task on a set of LEDs – four bits allocated to each task:
>   Task 1 status LEDs   Task 2 status
>   LEDs y y y y   y y y y

Where each set of   y y y y values can be interpreted as (and "x" indicates "don't
>   care") 1 x x x recipe command error
>   x 1 x x nested loop error
>   x x 1 x end of recipe
>   encountered x x x 1 recipe
>   paused

During execution of the task, each LED should be toggled per command. Thus the

proper operation of the commands will be indicated by flashing LEDs.

User Commands:

The User commands are entered via the system console keyboard. Each set of overrides are entered as a pair of characters followed by a <CR>. The first character represents the command for the first servo, the second for the second servo. Each character entered will be echoed to the console screen as entered. The <CR> will initiate the "simultaneous" execution of the overrides by each motor and generate a <LF> followed by the override prompt character ">" on the beginning of the next line.

Note that if either "X" or "x" is entered before the <CR>, the entire override command is cancelled and the ">" override prompt character is written on the beginning of a new line on the system console. See below for an example.

User Command Details:

Command	Mnemonic	Comments
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	B or b	Starts the recipe's execution immediately

Example override commands:
- Pc <CR> Pause the recipe on Servo 1 and continue (resume) the recipe on Servo 2.
- Rn <CR> Move Servo 1 to the right one position (if possible), no op of Server 2.
- Bb <CR> Restart Servo 1's recipe and Servo 2's recipes.

As servo motors take time to reach a new position, the programs will need to introduce a delay in fetching and executing the next recipe command after a "mov" command. The delay will need to be adjusted depending upon how far the new position is from the current position. A starting estimate of the necessary added delays to reach a new position is approximately 200 milliseconds per position.

For safety, your program's execution should begin with tasks in the paused state. The startup sequence should include appropriate initialization of the task values.

Include a robust pair of demo recipes (a different one for each servo to demonstrate multitasking.)

The following test recipe snippets must be included in your demo:

```
MOV+0       ; There must NOT be intervening instructions in this group to allow
MOV+5       ; verification of default time delay.
MOV+0
MOV+3
LOOP+0      ;Test the default loop behavior.
MOV+1
MOV+4
END_LOOP
MOV+0
MOV+2
WAIT+0
MOV+3       ; Move to an adjacent position to verify
WAIT+0.
MOV+2
MOV+3       ; Measure the timing precision of the 9.3 second delay with an external
WAIT+31     ; timer.
WAIT+31
WAIT+31
MOV+4
```

Additional test recipes:
- Include a recipe to verify the moves to all possible positions.
- Include a recipe to end normally (i.e. with an "RECIPE_END" command, followed and by a MOV command to a position different from the previous MOV destination. This allows verification of the CONTINUE override.
- Include a recipe with a deliberate error near the end of a task's demo recipe, then follow the erroneous command with a MOV command to a new position.

Graduate Student extensions:

There are three unused recipe "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.

Due Date:

Please refer to the course schedule for the due date. An in-class demo will be required. Please submit your source code and your report (in either PDF or Word compatible format) to the corresponding myCourses drop box.

Grading Criteria:

- Program Operation - 50%
- Program Design - 15%
- Source Code Structure and Readability - 10%
- Report Content - 25%