Multitasking using LPC2148

e-Yantra Team

June 2016

List of Figures

Contents

1 Objective

In this tutorial we will learn how to implement Multitasking using FreeRTOS in Firebird V.

2 Prerequisites

- C Programming
- Basics of LPC2148
- FreeRTOS

3 Hardware Requirements

1. Firebird V (LPC2148)
2. LCD Interfacing for LPC 2148
3. Motions control for Firebird V (LPC2148)
4. Buzzer Operations

4 Software Requirements

- Keil uvision4
- FlashMagic

5 Theory and Description

5.1 Motion Control

In Firebird V ther are two 75RPM geared motors. The robot can achieve zero turning radius by rotating the two wheels in the opposite direction. The two motors are driven using the L293D motor driver IC that provides upto 600mA of current.

Following are the ports that are used to control the motion of the robot and the settings that are required to determine the direction of its motion.

DIRECTION	LEFT BWD (LB) P0.22	LEFT FWD(LF) P1.21	RIGHT FWD(RF) P0.10	RIGHT BWD(RB) P0.11
FORWARD	0	1	1	0
REVERSE	1	0	0	1
RIGHT *(Left wheel forward, Right wheel backward)*	0	1	0	1
LEFT*(Left wheel backward, Right wheel forward,)*	1	0	1	0

Figure 1: Motion Control Settings

5.2 Buzzer

In Firebird V 3Khz piezo buzzer is used. It is connected to port P0.25. In order to start and stop the buzzer we can simly use the commands like IO0SET and IO0CLR.

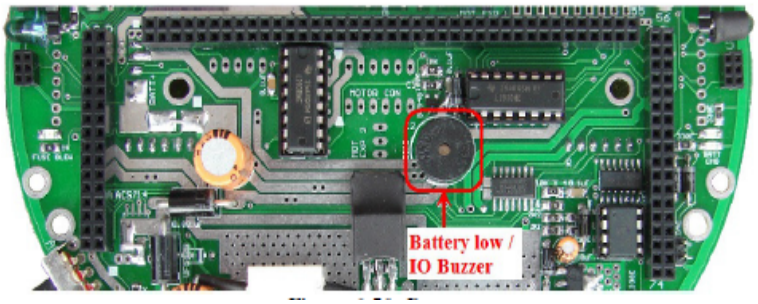

Figure 2: BUZZER

5.3 LCD Interfacing

In Firebird V The LCD can be used in 4-bit mode or in 8-bit mode. When used in 8-bit mode it requires 3 control lines and 8 data lines. In order to reduce the number of I/O used we use it 4-bit mode. Here, we only need four data lines.The three control lines used are EN(Enable),RS(Register Select) and R/W(Read/Write)

EN pin : It is used to establish communication between the micro-controller and the LCD. It is connected to port P1.17. In order to transmit data to LCD, we should make sure that firstly it is low and set other control lines as required and then make it high and wait for some time and then make it low again.

RS pin : The RS pin is connected to the port P1.19. When this is set low then the data that is transmitted is treated command that is used for initialisation of LCD and other functions. If it is set high the then the data transmitted is treated as a text data that is to be displayed.

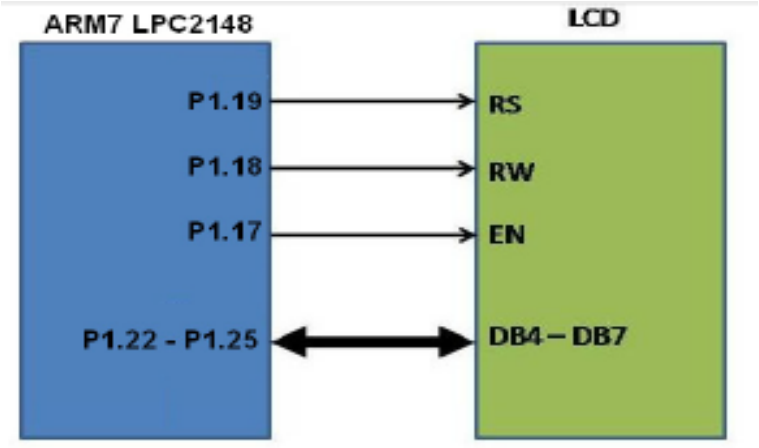

Figure 3: LCD Connections

R/W : This pin is connected to port P1.18. If it is set low then it means the data is to be written on LCD. If it is set high then the data is read from LCD.

Ports P1.22-P1.25 are used for data transmission between the LCD and the micro-controller. These line are bi-directional.

6 Experiment

Use the following code to do Multi-Tasking.

```
#include <stdlib.h>
#include "FreeRTOS.h"
#include "task.h"
#include "lcd.h"


#define DATA_PORT() IO1SET=(1<<19)
#define READ_DATA() IO1SET=(1<<18)
#define EN_HI() IO1SET=(1<<17)

#define COMMAND_PORT() IO1CLR=(1<<19)
#define WRITE_DATA() IO1CLR=(1<<18)
#define EN_LOW() IO1CLR=(1<<17)


TaskHandle_t  xTask1Handle,xTask2Handle,xTask3Handle;

void Init_Motion_Pin(void)
{
 PINSEL0&=0xFF0F3FFF;
 PINSEL0|=0x00000000;              //Set Port pins P0.7, P0.10, P0.11 as GPIO
 PINSEL1&=0xFFFFF0FF;
 PINSEL1|=0x00000000;              //Set Port pins P0.21 and 0.22 as GPIO
 IO0DIR&=0xFF9FF37F;
 IO0DIR|= (1<<10) | (1<<11) | (1<<21) | (1<<22) | (1<<7) | (1<<25);      //Set Port pins
 IO1DIR&=0xFFDFFFFF;
 IO1DIR|= (1<<21);                 // Set P1.21 as output pin
 IO0SET = 0x00200080;
  IO1DIR|=(1<<25) | (1<<24) | (1<<23) | (1<<22) | (1<<19) | (1<<18) | (1<<17);  // Set P
}

//Stop left motor
void L_Stop(void)
{
 IO1CLR = 0x00200000;              //Set P1.21 to logic '0'
 IO0CLR = 0x00400000;              //Set P0.22 to logic '0'
}

//Stop Right motor
void R_Stop(void)
{
 IO0CLR = 0x00000400;              //Set P0.10 to logic '0'
```

```
 IO0CLR = 0x00000800;              //Set P0.11 to logic '0'
}
void Stop(void)
{
 L_Stop();
 R_Stop();
}
//Move Left motor forward
void L_Forward(void)
{
 IO1SET = 0x00200000;              //Set P1.21 to logic '1'
}

//Function to move Left motor backward
void L_Back(void)
{
 IO0SET = 0x00400000;              //Set P0.22 to logic '1'
}

//Move Right motor forward
void R_Forward(void)
{
 IO0SET = 0x00000400;              //Set P0.10 to logic '1'
}

//Move Right motor backward
void R_Back(void)
{
 IO0SET = 0x00000800;              //Set P0.11 to logic '1'
}

//Function to move robot in forward direction
void Forward(void)
{
 Stop();
 L_Forward();
 R_Forward();
}

void BUZZER_ON(void)
{
IO0SET |= (1<<25);
}
void BUZZER_OFF(void)
{
IO0CLR |= (1<<25);
}



//Pin Initialisations
```

```
void Init_Ports(void)
{
 Init_LCD_Pin();
 Init_Motion_Pin();
}

void Init_Peripherals(void)
{
 Init_Ports();
}

//Task Functions
void vbuzzer(void *);
void vmotion(void *);
void lcdprint(void *);


// Buzzer Task
void vbuzzer(void *p)
{
while(1)
{
            BUZZER_ON();
            vTaskDelay(200);
            BUZZER_OFF();
            vTaskDelay(200);
}
}

//Motion Task
void vmotion(void *p)
{
while(1)
{
 Forward();
 vTaskDelay(250);


}
}

 //LCD Display Task
void lcdprint(void *p)
{
unsigned char count = 0;//Initialised a variable
while(1)
{

  if (count == 100)
  {
        count = 0;
```

```
  }
 LCD_Print(1,2,count++,3);
 vTaskDelay(200);


 }
}


int main ()

{
Init_Peripherals();
 while(1)
{
   LCD_Init();//LCD is initialised

/*If the priorities are same then make the #define  configUSE_TIME_SLICING
0*/
/*3 Tasks are created in the following function*/
xTaskCreate(vbuzzer, "noise", configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY+1,&xTask1
xTaskCreate(vmotion, "forward", configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY+2, &xTa
xTaskCreate(lcdprint, "display", configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY+3, &xT
                                        /*stack_depth, priority=1 , Null handle. */


vTaskStartScheduler();

}
}
```

Code Explanation: In the above program, we have created three tasks viz. a buzzer task, a forward motion task and a counter on LCD. These tasks are created using the function "xTaskCreate". Six parameters are defined inside this function.First one is the name of the task that is created as shown in the program. Second is the name of the task in simple terms for the user to understand. Third is the stack size. Fourth parameter is any parameter that we want to pass to that particular task. Fifth is the task priority. We use variable priority in the given example. Last parameter is the name of the taskhandle. Its similar to a pointer. This handle will point to that particular task.
The vTaskStartScheduler is the function that schedules the task according to the priority.
VTaskDelay is delay generated in terms of ms. It will temporarily suspend that particular task for the defined duration.