Kernel Masters

GPIO Lab Experiments

Embedded C & RTOS

Kernel Masters
2/27/2022

Contents

GPIO Lab Assignments: 2
1. GPIO Controller 2
1.1. GPIO Ports Configure to Output: 2
1.2. GPIO Ports Configure to Input with Polling: 2
1.3. GPIO Ports Configure to Input with Interrupt: 3
2. Mini Project: 16x2 Monochrome LCD Interface: 3
2.1. LCD Registers 4
2.2. Important command codes for LCD 4
2.3. LCD Initialization Flow: 5

GPIO Lab Assignments:

1. GPIO Controller

GPIO Controller Pin mapping details show the below table.
The below LED and switches configured to as -ve Level Logic.

GPIO Pin	Pin Function	Device
PB12	GPIO	BUZZER
PB13	GPIO	Red LED
PB14	GPIO	Green LED
PC8	GPIO	SW_UP
PC9	GPIO	SW_DN
PC10	GPIO	SW_ENT

1.1. GPIO Ports Configure to Output:

Lab Experiment 1:
Toggle RED LED ON time is 50ms and RED LED OFF time is 1300ms

Lab Experiment 2:
Airplane wing Lights blinking Delays:
USER LED1 (GREEN): ON (50msec), OFF (50msec), ON (50msec), OFF (150msec) periodically.
USER LED2 (RED): ON (50msec), OFF (250msec), ON (50msec), OFF (500msec) periodically.

1.2. GPIO Ports Configure to Input with Polling:

Lab Experiment 3: Whenever SW_UP is pressed RED LED is ON, whenever SW_DN is pressed GREEN LED is ON.
Assign breakpoints after if condition and click run button in debugger window. Whenever SW1/SW2 presses program stops at any one of the breakpoint.

Lab Experiment 4: Use **"SW_ENT"** and declare "**counter**" global variable. Your program should increment **counter** by one, every time switch is pressed. Note how the value of **counter** changes on each "ENTER SWITCH" press. Use debugger and add **Counter** to "Watch Expression" window. Does the value of **counter** increment by one always?

Note: Define **counter** as a global variable and in debug perspective use continuous refresh option (You will find Continuous Refresh button on top of the Expression Window). You can use step debugging or breakpoints to check the variable value.

Hint: To add variable to Expression Window, select and right click the variable name and select "Add Watch Expression". To view Expression Window, click on View button from Keil menu bar and select Expressions.

Lab Experiment 5: Whenever ENTER SWITCH is Press toggle RED LED.

Lab Experiment 6: Read SW_ENT, if switch is press RED LED is ON otherwise RED LED is OFF. Note that RED LED should remain ON for the duration switch is kept pressed i.e. RED LED should turn OFF when switch is released.

Lab Experiment 7: Whenever SW_ENT Press turn ON BUZZER up to 250msec and Turn OFF BUZZER.

1.3. GPIO Ports Configure to Input with Interrupt:

Lab Experiment 8: Write a program Request an interrupt on the **Falling edge of ENTER Switch,** whenever the user button is pressed and increment a counter in the interrupt and RED LED is ON**.**

Lab Experiment 9: Write an Embedded C Program Whenever SW_UP press RED LED is ON, whenever **SW_DN** press GREEN LED is ON using Falling edge GPIO interrupt.

Lab Experiment 10: Write an Embedded C Program to measure interrupt latency. By default **RED LED** (-ve Logic Level) is ON. Whenever **ENT SWITCH (-ve Logic Level) is** press RED LED is OFF.

Interview Questions

What is Interrupt Latency?

Interrupt latency is the time that elapses from when an interrupt is generated to when the source of the interrupt is serviced. For many operating systems, devices are serviced as soon as the device's interrupt handler is executed. Interrupt latency may be affected by microprocessor design, interrupt controllers, interrupt masking, and the operating system's (OS) interrupt handling methods

How to measure interrupt latency?

To measure a time interval, like interrupt latency, with any accuracy, requires a suitable instrument. The best tool to use is an oscilloscope. One approach is to use one pin on a GPIO interface to generate the interrupt. This pin can be monitored on the 'scope. At the start of the interrupt service routine, another pin, which is also being monitored, is toggled. The interval between the two signals may be easily read from the instrument.

How to reduce interrupt latency?

To reduce interrupt latency ARM Cortex M4 introduce new feature **"tail chain"**

ARM Cortex M4 processor reduces interrupt latency using tail-chain.

2. Mini Project: 16x2 Monochrome LCD Interface:

GPIO Pin	Pin function	Device
PB0	GPIO	LCD_DATA_4
PB1	GPIO	LCD_DATA_5
PB2	GPIO	LCD_DATA_6
PB3	GPIO	LCD_DATA_7
PB4	GPIO	LCD_RS
PB5	GPIO	LCD_RW
PB8	GPIO	LCD_EN

2.1. LCD Registers

RS(Register select): A 16X2 LCD has two registers, namely, command and data. The register select is used to switch from one register to other. RS=0 for command register, whereas RS=1 for data register.

Command Register: The command register stores the command instructions given to the LCD. A command is an instruction given to LCD to do a predefined task like initializing it, clearing its screen, setting the cursor position, controlling display etc. Processing for commands happen in the command register.

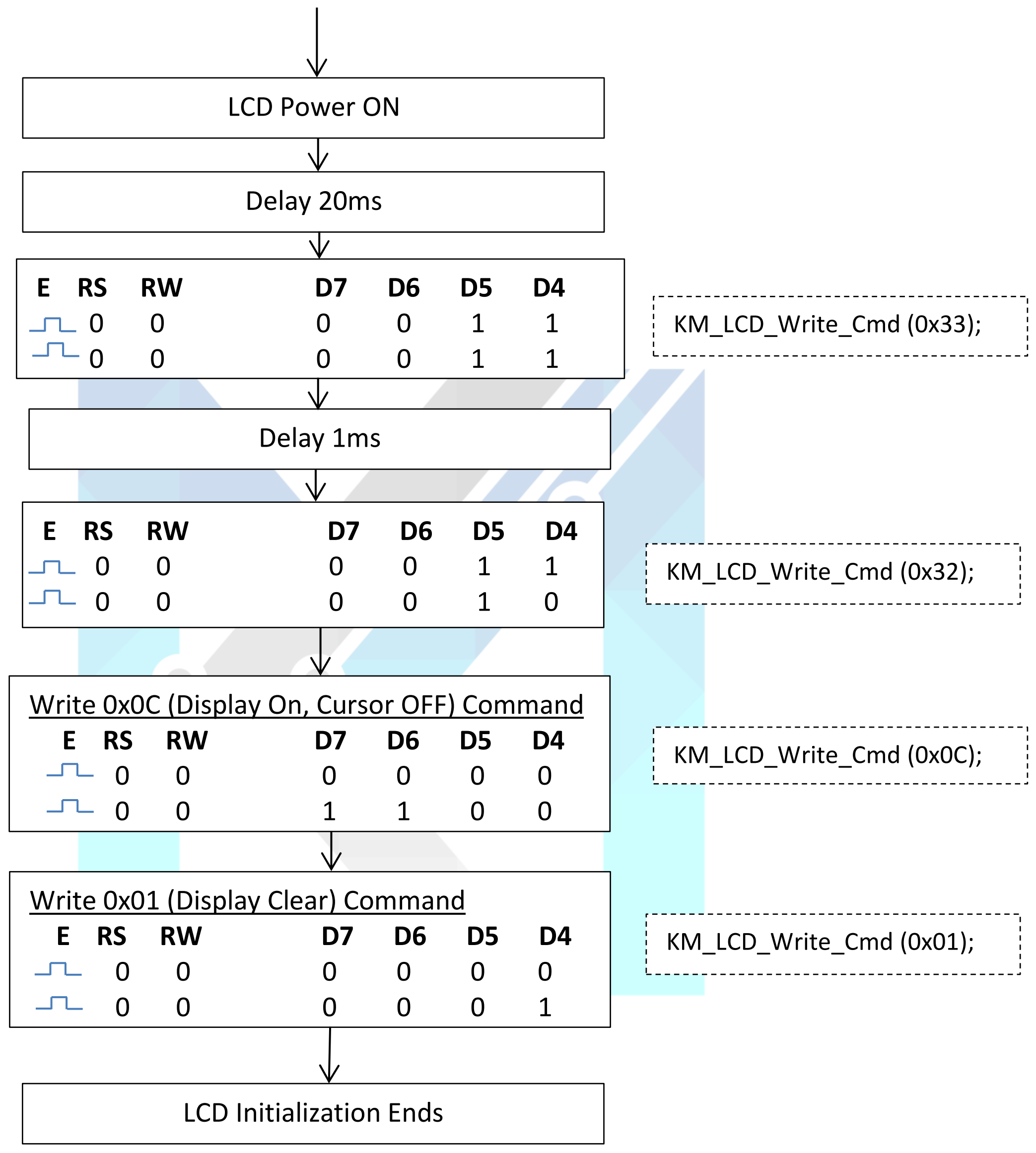
Data Register: The data register stores the data to be displayed on the LCD. The data is the ASCII value of the character to be displayed on the LCD. When we send data to LCD it goes to the data register and is processed there. When RS=1, data register is selected.

2.2. Important command codes for LCD

Hex Code	Command to LCD instruction Register	Hex Code	Command to LCD instruction Register
01	Clear display screen	0E	Display on, cursor blinking
02	Return home	0F	Display on, cursor blinking
04	Decrement cursor (shift cursor to left)	10	Shift cursor position to left
06	Increment cursor (shift cursor to right)	14	Shift cursor position to right
05	Shift display right	18	Shift the entire display to the left
07	Shift display left	1c	Shift the entire display to the right
08	Display OFF, cursor OFF	80	Force cursor to beginning (1st line)
0A	Display OFF, cursor ON	C0	Force cursor to beginning (2nd line)

*/*Write Higher Nibble data in to LCD.*
*Return Value: No return Value; Argument: unsigned char data */*
void write_high_nibble(unsigned char data);
*/*Write Lower Nibble data in to LCD.*
*Return Value: No return Value; Argument: unsigned char data */*
void write_low_nibble(unsigned char data);
*/*LCD Initialization.*
*Return Value: No return Value; Argument: No Argument */*
void KM_LCD_Init(void);
*/*Write a LCD command.*
*Return Value: No return Value; Argument: unsigned char */*
void KM_LCD_Write_Cmd(unsigned char);
*/*Write a LCD Data (single character).*
*Return Value: No return Value; Argument: unsigned char */*
void KM_LCD_Write_Data(unsigned char);
*/*Write a LCD Data (Multiple characters).*
*Return Value: No return Value; Argument: character pointer */*
void KM_LCD_Write_Str(char *);

2.3. LCD Initialization Flow:


LCD Power ON
Delay 20ms
E RS RW D7 D6 D5 D4
0 0 0 0 1 1
0 0 0 0 1 1
KM_LCD_Write_Cmd (0x33);
Delay 1ms
E RS RW D7 D6 D5 D4
0 0 0 0 1 1
0 0 0 0 1 0
KM_LCD_Write_Cmd (0x32);
Write 0x0C (Display On, Cursor OFF) Command
E RS RW D7 D6 D5 D4
0 0 0 0 0 0
0 0 1 1 0 0
KM_LCD_Write_Cmd (0x0C);
Write 0x01 (Display Clear) Command
E RS RW D7 D6 D5 D4
0 0 0 0 0 0
0 0 0 0 0 1
KM_LCD_Write_Cmd (0x01);
LCD Initialization Ends