UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 6 Examination in Engineering: January 2022

Module Number: EE6304 **Module Name: Embedded System Design**

[Three Hours]

[Answer **all questions**, each question carries 10 marks]

Q1 a) i) Design steps of an embedded system can be listed as
1. Requirements and specifications of hardware and software
2. Define architectures of hardware and software
3. Coding and implementation as per architecture,
4. Testing, validation and verification of system.

Draw a flow chart to show how these steps are being used.

[1 Mark]

ii) In embedded systems, the design should support hassle free/efficient methods for testing and validation. Briefly explain the above statement.

[2 Marks]

b) i) State the meaning of a real-time requirement with respect to an embedded system.

[1 Mark]

ii) State one requirement each for functional requirements and quality requirements of an embedded system.

[1 Mark]

c) i) List three key components of a microcontroller.

[1 Mark]

ii) State the main difference between the platform and architecture.

[1 Mark]

iii) Using a figure, illustrate the relationship between the main program, an interrupt service routine, and the interrupt vector.

[1 Mark]

d) Briefly explain how you would assist resource sharing using a semaphore.

[2 Marks]

Q2 a) Consider the assembly code written to control a set of LEDs using a push button shown in Figure Q2.1. It is targeted for a PIC16F877A microcontroller and is written using MPLAB-X IDE V5.15.

Note: Required register summary and Assembly instruction summary are given in Figure Q2.2 and Figure Q2.3, respectively.

i) Identify the addresses of the registers accessed in line 20 and line 23

[1 Mark]

ii) Identify the possible functionalities of line 20 and line 21.

[1 Mark]

iii) Line 23 sets the interrupt to trigger at the rising edge of the input to the external interrupt pin. Identify the functionality of the code written inside the ISR.

[1 Mark]

iv) Identify the functionality of line 2.

[1 Mark]

v) When you run this program in the debug mode, you may not see if the program is running even if it is running properly. State a technique you can adopt to identify if the program is running.

[1 Mark]

vi) State in point form what you would do if you are to test the functionality of the interrupt before uploading the compiled code to the microcontroller.

[2 Marks]

b) i) Identify the property of Atmega328P microcontroller which allows you to program it using Atmel Studio and a USB cable.

[1 Mark]

ii) Identify the important functionality/feature that should be turned off when sending a microcontroller to the debug mode.

[1 Mark]

iii) State an advantage of using an IDE than using a simple text editor to program microcontrollers.

[1 Mark]

Q3 a) You are required to write a C program to drive a stepper motor. Figure Q3 gives the code for stepperMotor.c file written for this purpose. The `Motor` structure consist of four `integer` variables.

1. `phaseA` and `phaseB` corresponding to two phases of the stepper motor.
2. `currentStepIndex` is to store the index of the step that you currently execute.
3. `nextStepTime` is to set the time at which next step should occur.

Following two functions are written in a different C file which you can access by prototyping as given in line 10 and line 12. You can call these functions by their name in you code.

1. `getTime()` is to get the current time as an integer.
2. `driveMotor()` takes an input parameter of type `struct Motor` and set values to the corresponding pins of the motor.

The `initMotor()` function receives a pointer (by reference) to a parameter of type `struct Motor` and it initializes all motor attributes inside its body.

Answer part (i) to (vii) in Figure Q3 itself and attach it to the answer script.

i) Suppose that a parameter of type `struct Motor` is passed by reference (as a pointer) into the `generateNextPattern()` function. Complete the function prototype in line 16.

[1 Mark]

ii) Suppose that a the pointer used inside `generateNextPattern()` function is named as `this_motor`. Complete the function definition given in line 39.

[1 Mark]

iii) Complete line 24 to implement reading the current time into the variable `curentTime`.

[1 Mark]

iv) Suppose that we can execute what is inside the 'if' condition in line 25 only when current time is greater than the time to set the next step (nextStepTime). Complete line 25 to implement this condition.

[1 Mark]

v) Complete line 27 to call the function to set required pins to drive the motor.

[1 Mark]

vi) After each step, two phases of the stepper motor has to be inverted (i.e. one to zero and zero to one). Complete line 40 and line 41 to update the two phases of the motor.

[1 Mark]

vii) Update the time for next step and step index using line 42 and line 43.

[1 Mark]

b) Consider a microcontroller running at 16 Mhz consisting of an 8-bit and 16-bit timers each having the prescalar values 1, 8, 64, 256, and 1024.

i) Suppose that you need to set a time interval between 1 second and 4 seconds. Identify the timer and the prescalar to be used in order to implement this time interval.

Note: Show the steps and calculation used for your selection.

[2 Marks]

ii) Out of the overflow and compare modes, select the mode would you select to generate this timer interrupt.

[1 Mark]

Q4 Suppose that you are taking part in a project to implement a vehicle fleet monitoring system. The device you are supposed to design should satisfy following requirements.

1. Obtain the GPS location every few seconds.
2. Send number of GPS locations at once to a central server at constant intervals.
3. Manually adjust the interval between successive GPS locations and successive transmissions to the server.
4. Display the time interval between successive GPS locations and successive transmissions to the server.

Further, you are requested to use an Atmel 328P microcontroller to implement the device.

a) i) Draw a block diagram to implement the given device.

[1 Mark]

ii) Draw a flow chart to implement the given functionalities.

[2 Marks]

iii) Identify the components/modules to be used for each block.

[1 Mark]

iv) If the GPS module is to be connected via UART, identify the interfaces to be used for each of the other components/modules.

[1 Mark]

b) Suppose that you are asked to implement the 3rd requirement using a different technique than what you have given in (a). State in point form, how you would implement it.

[2 Marks]

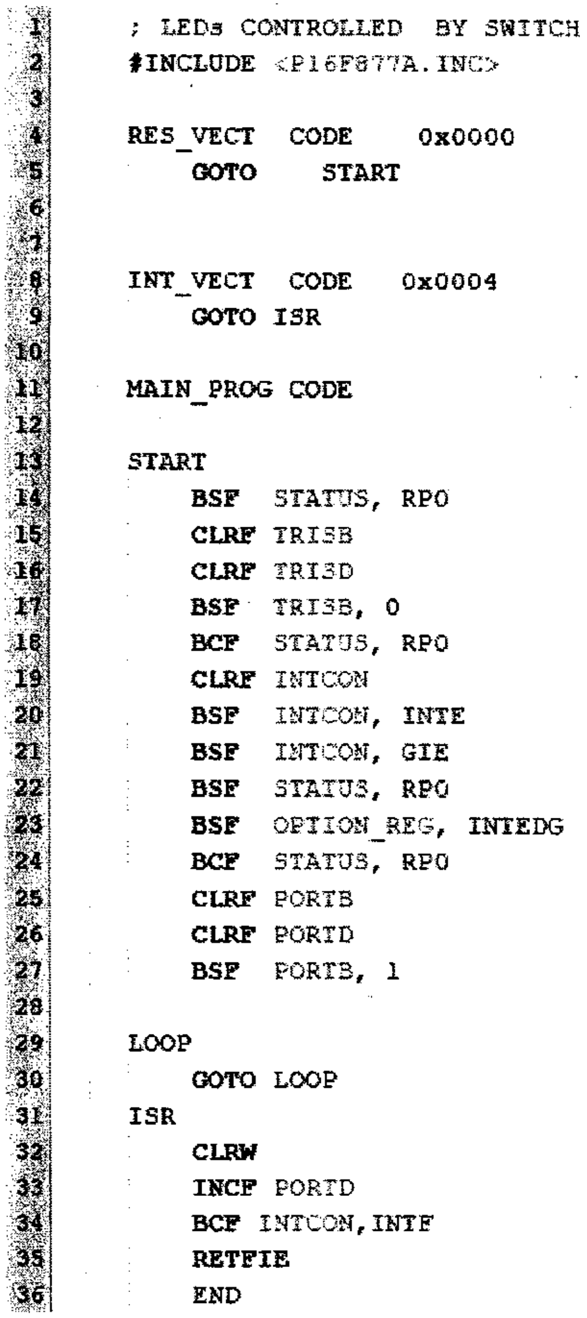
c) i) Suppose that you are to take a digital input from a particular port. Using a figure, show how you would connect an input which is low when it is active and floating when it is inactive.

[1 Mark]

ii) Suppose that you are required to use a push button to implement a 'sleep button', which you need to push for a certain duration to activate. Explain in point form how you would implement this functionality.

Note: you may have to focus on the input type, the implementation logic, etc.

[2 Mark]



```
; LEDs CONTROLLED  BY SWITCH
#INCLUDE <P16F877A.INC>

RES_VECT  CODE    0x0000
    GOTO    START


INT_VECT  CODE   0x0004
    GOTO ISR

MAIN_PROG CODE

START
    BSF  STATUS, RP0
    CLRF TRISB
    CLRF TRISD
    BSF  TRISB, 0
    BCF  STATUS, RP0
    CLRF INTCON
    BSF  INTCON, INTE
    BSF  INTCON, GIE
    BSF  STATUS, RP0
    BSF  OPTION_REG, INTEDG
    BCF  STATUS, RP0
    CLRF PORTB
    CLRF PORTD
    BSF  PORTB, 1

LOOP
    GOTO LOOP
ISR
    CLRW
    INCF PORTD
    BCF INTCON,INTF
    RETFIE
    END
```

Figure Q2.1

	Address		Address		Address		Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	▨	105h	▨	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h	▨	107h	▨	187h
PORTD(1)	08h	TRISD(1)	88h	▨	108h	▨	188h
PORTE(1)	09h	TRISE(1)	89h	▨	109h	▨	189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh	▨	8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h	▨	90h	General Purpose Register 16 Bytes	110h	General Purpose Register 16 Bytes	190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h	▨	95h		115h		195h
CCPR1H	16h	▨	96h		116h		196h
CCP1CON	17h	▨	97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah	▨	9Ah		11Ah		19Ah
CCPR2L	1Bh	▨	9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
			EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h - 7Fh	1F0h
	7Fh		FFh		17Fh		1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

Figure Q2.2: Register Banks for PIC16F877A

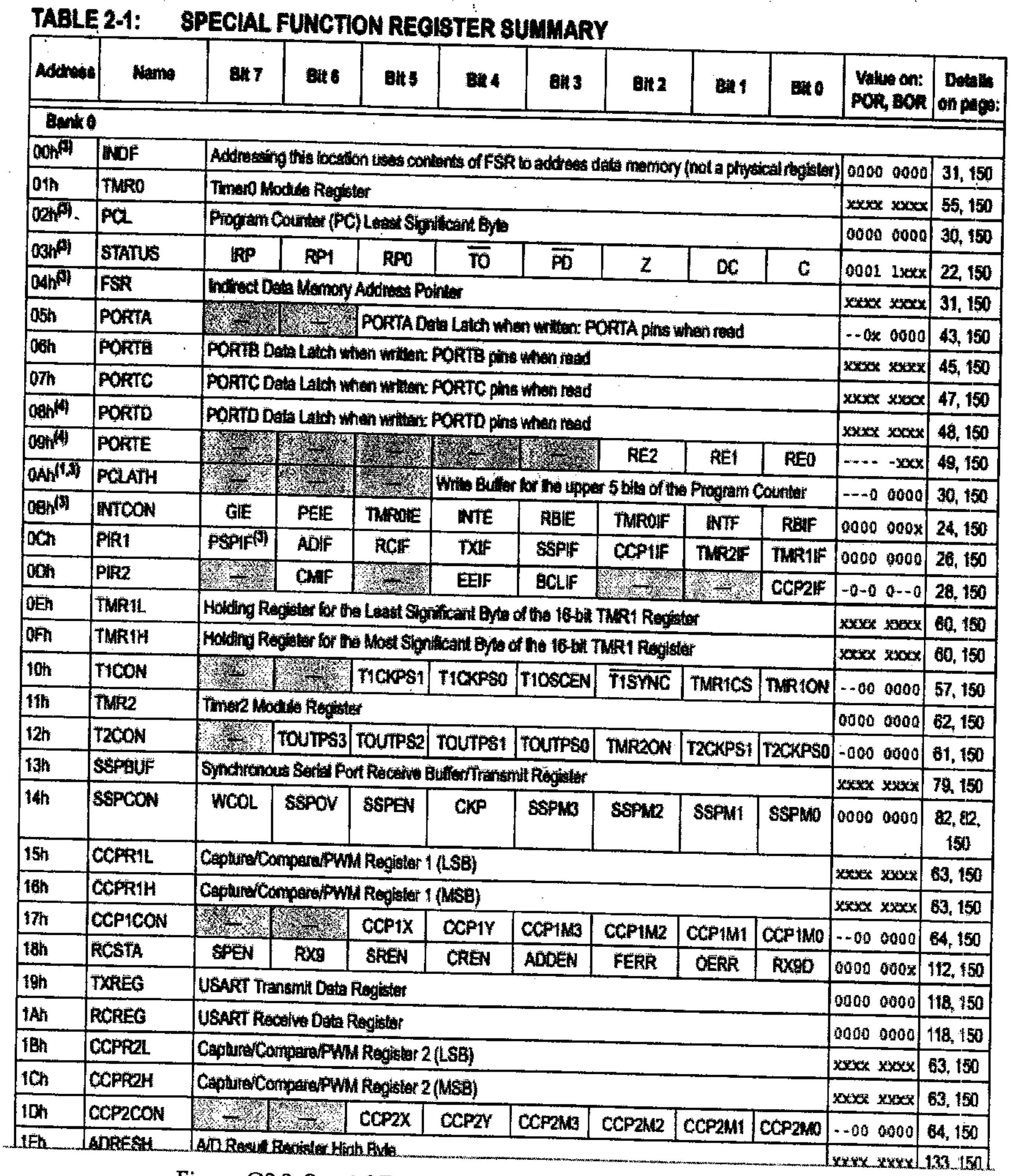


TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h[3]	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
01h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
02h[3]	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
03h[3]	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
04h[3]	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	43, 150
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	45, 150
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	47, 150
08h[4]	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	48, 150
09h[4]	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	49, 150
0Ah[1,3]	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
0Bh[3]	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
0Ch	PIR1	PSPIF[3]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	26, 150
0Dh	PIR2	—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF	-0-0 0--0	28, 150
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	--00 0000	57, 150
11h	TMR2	Timer2 Module Register								0000 0000	62, 150
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	61, 150
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	79, 150
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	82, 82, 150
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	63, 150
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	63, 150
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	64, 150
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	112, 150
19h	TXREG	USART Transmit Data Register								0000 0000	118, 150
1Ah	RCREG	USART Receive Data Register								0000 0000	118, 150
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxx xxxx	63, 150
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxx xxxx	63, 150
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	64, 150
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	133, 150

Figure Q2.3: Special Function Register Summary for PIC16F877A