Seat No.: _____ Enrolment No._____

GUJARAT TECHNOLOGICAL UNIVERSITY

M. E. - SEMESTER – I • EXAMINATION – WINTER 2012

Subject code: 710302N **Date: 09-01-2013**

Subject Name: Advance Microcontroller

Time: 02.30 pm – 05.00 pm **Total Marks: 70**

Instructions:

1. **Attempt all questions.**
2. **Make suitable assumptions wherever necessary and mention it clearly.**
3. **Appropriate comment lines are must in your programs.**
4. **Figures to the right indicate full marks.**
5. **Some of the control/status registers and instruction set of PIC family is provided for your reference in annexure sheet.**

Q.1 **(a)** A parking space having two gates, one for entry and other for exit, has a maximum capacity of 50 cars. Design an embedded system using PIC18F452 microcontroller which counts the total number of cars present in parking space at a particular time and display it on 7 segment LEDs. Remember that a car can enter and leave from the parking space. Sensors are mounted in both gates. When parking space becomes full then it should give indication on LED by toggling it continuously. Write an assembly language program for the same. Make use of at least one interrupt in your program. Write your assumptions explicitly. **10**

(b) Answer the following in brief **04**

(1) Complement the data stored in data register 0x05 without using COMF instruction. Store the result in 0x0A.

(2) What is the maximum delay obtained when timer 0 is running in 16-bit mode and the MCU clock frequency is 20MHz?

Q.2 **(a)** Draw interfacing of three common cathode seven-segment LEDs to PIC18F452 microcontroller using PORTA and PORTB. Write an assembly language program to display a three-digit BCD number “942” on these LEDs continuously. **07**

(b) Add 10 BCD data bytes stored in data registers beginning from 0x100. Store the result in data registers. **07**

OR

(b) A string of 10 characters is stored in internal RAM beginning from 0x200. If the string is palindrome store 0xAA else store 0xFF in data memory 0x300. **07**

Q.3 **(a)** Given a packed BCD number in data register 0x50. Convert this number in binary and store the result in data register 0x51. **07**

(b) A data block of signed numbers is stored in memory location beginning from 0x500. Length of this block is stored in memory location 0x05. Separate out positive and negative numbers from this block and store them in memory locations beginning from 0x600 and 0x700 respectively. **07**

OR

Q.3 **(a)** The 16-bit number 0xA237 is stored in memory location 0x10 (LSB) and 0x11 (MSB). Multiply this number by 0x3A that is stored in memory location 0x12. Store the product in data registers 0x100 (LSB) onwards. **07**

(b) Explain interfacing and timing diagram of LCD with PIC microcontroller. **07**

Q.4	**(a)**	Draw interfacing of 8 push-button keys (K0 to K7) to PORTB and 8 common cathode LEDs (LED0 to LED7) to PORTA of PIC18F452 microcontroller. Write an assembly language program that checks for a key closure, debounces multiple key contacts and displays the binary number corresponding to pressed key on LED port (e.g. if key K5 is pressed then display 0000 0101 on LED port).	**10**
	(b)	Write a macro named ADDUNSGN that adds two numbers passed as arguments and stores the result in data memory location 0x09.	**04**

OR

Q.4	**(a)**	Interface DAC with PIC18F452 microcontroller. Generate a saw tooth waveform using this DAC. How can we change the frequency of this waveform?	**10**
	(b)	Write a program to copy the value 0xA3 into RAM locations 0x230 and 0x530 using direct addressing mode only.	**04**
Q.5	**(a)**	Write a subroutine to set up timer 0 in the 16-bit mode with the internal clock at 20 MHz to generate a 100 ms delay and provides multiples of 100 ms delay based on the count in data register 0x10. Select an appropriate prescaler and show your calculations of delay. Using this subroutine generate a square wave of 1 Hz.	**10**
	(b)	Explain block diagram of PIC18F452 ADC module in brief.	**04**

OR

Q.5	**(a)**	Write a program to generate a square wave of 2 KHz at pin RC2/CCP1 of the PORTC using the interrupt method and Timer1, assuming a crystal frequency of 20 MHz.	**10**
	(b)	Explain about priority of interrupts in PIC18F452 microcontroller.	**04**

****

ANNEXURE SHEET

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC(1)	C(2)

STATUS REGISTER

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
SPPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF

PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIF	CMIF	USBIF	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF

PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE

PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIE	CMIE	USBIE	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE

PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
SPPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP

IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
OSCFIP	CMIP	USBIP	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP

IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

R/W-0	R/W-1(1)	U-0	R/W-1	R-1	R-1	R/W-0(2)	R/W-0
IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$

RCON: RESET CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF(1)

INTCON: INTERRUPT CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SWDTEN(1)

WDTCON: WATCHDOG TIMER CONTROL REGISTER

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON

T1CON: TIMER1 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0

T2CON: TIMER2 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON

T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0

CCP1CON: ECCP CONTROL REGISTER

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PRSEN	PDC6[1]	PDC5[1]	PDC4[1]	PDC3[1]	PDC2[1]	PDC1[1]	PDC0[1]

ECCP1DEL: PWM DEAD-BAND DELAY REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1[1]	PSSBD0[1]

ECCP1AS: ENHANCED CAPTURE/COMPARE/PWM AUTO-SHUTDOWN CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON

ADCON0: A/D CONTROL REGISTER 0

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0

ADCON2: A/D CONTROL REGISTER 2

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0

CMCON: COMPARATOR CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP

INTCON2: INTERRUPT CONTROL REGISTER 2

R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF

INTCON3: INTERRUPT CONTROL REGISTER 3

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SWDTEN[1]

WDTCON: WATCHDOG TIMER CONTROL REGISTER

BYTE-ORIENTED OPERATIONS		
ADDWF	f, d, a	Add WREG and f
ADDWFC	f, d, a	Add WREG and Carry bit to f
ANDWF	f, d, a	AND WREG with f
CLRF	f, a	Clear f
COMF	f, d, a	Complement f
CPFSEQ	f, a	Compare f with WREG, Skip =
CPFSGT	f, a	Compare f with WREG, Skip >
CPFSLT	f, a	Compare f with WREG, Skip <
DECF	f, d, a	Decrement f
DECFSZ	f, d, a	Decrement f, Skip if 0
DCFSNZ	f, d, a	Decrement f, Skip if Not 0
INCF	f, d, a	Increment f
INCFSZ	f, d, a	Increment f, Skip if 0
INFSNZ	f, d, a	Increment f, Skip if Not 0
IORWF	f, d, a	Inclusive OR WREG with f
MOVF	f, d, a	Move f
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word
MOVWF	f, a	Move WREG to f
MULWF	f, a	Multiply WREG with f
NEGF	f, a	Negate f
RLCF	f, d, a	Rotate Left f through Carry
RLNCF	f, d, a	Rotate Left f (No Carry)
RRCF	f, d, a	Rotate Right f through Carry
RRNCF	f, d, a	Rotate Right f (No Carry)
SETF	f, a	Set f
SUBFWB	f, d, a	Subtract f from WREG with Borrow
SUBWF	f, d, a	Subtract WREG from f
SUBWFB	f, d, a	Subtract WREG from f with Borrow
SWAPF	f, d, a	Swap Nibbles in f
TSTFSZ	f, a	Test f, Skip if 0
XORWF	f, d, a	Exclusive OR WREG with f

BIT-ORIENTED OPERATIONS		
BCF	f, b, a	Bit Clear f
BSF	f, b, a	Bit Set f
BTFSC	f, b, a	Bit Test f, Skip if Clear
BTFSS	f, b, a	Bit Test f, Skip if Set
BTG	f, d, a	Bit Toggle f
CONTROL OPERATIONS		
BC	n	Branch if Carry
BN	n	Branch if Negative
BNC	n	Branch if Not Carry
BNN	n	Branch if Not Negative
BNOV	n	Branch if Not Overflow
BNZ	n	Branch if Not Zero
BOV	n	Branch if Overflow
BRA	n	Branch Unconditionally
BZ	n	Branch if Zero
CALL	n, s	Call Subroutine 1st word 2nd word
CLRWDT	—	Clear Watchdog Timer
DAW	—	Decimal Adjust WREG
GOTO	n	Go to Address 1st word 2nd word
NOP	—	No Operation
NOP	—	No Operation
POP	—	Pop Top of Return Stack (TOS)
PUSH	—	Push Top of Return Stack (TOS)
RCALL	n	Relative Call
RESET		Software Device Reset
RETFIE	s	Return from Interrupt Enable
RETLW	k	Return with Literal in WREG
RETURN	s	Return from Subroutine
SLEEP	—	Go into Standby mode

LITERAL OPERATIONS		
ADDLW	k	Add Literal and WREG
ANDLW	k	AND Literal with WREG
IORLW	k	Inclusive OR Literal with WREG
LFSR	f, k	Move Literal (12-bit) 2nd word to FSR(f) 1st word
MOVLB	k	Move Literal to BSR<3:0>
MOVLW	k	Move Literal to WREG
MULLW	k	Multiply Literal with WREG
RETLW	k	Return with Literal in WREG
SUBLW	k	Subtract WREG from Literal
XORLW	k	Exclusive OR Literal with WREG
DATA MEMORY ↔ PROGRAM MEMORY		
TBLRD*		Table Read
TBLRD*+		Table Read with Post-Increment
TBLRD*-		Table Read with Post-Decrement
TBLRD+*		Table Read with Pre-Increment
TBLWT*		Table Write
TBLWT*+		Table Write with Post-Increment
TBLWT*-		Table Write with Post-Decrement
TBLWT+*		Table Write with Pre-Increment