**Name:**………………………………………………………. **Signature**………………… **ID-No**:…………………….

**CMPE423 Embedded System Design Fall 2017-18, HW3 (return the solution before the Quiz Hour. Late HWs cannot be graded.)**

INFOSHEET for ADC Configuration, Reading

**Digital/Analog/Reference settings of the i/o pins.**
**A: Analog, D:Digital.**

| Microcontroller Pins | | | | | | | | | | ADCON1[3..0] |
|---|---|---|---|---|---|---|---|---|---|---|
| RA0 | RA1 | RA2 | RA3 | RA5 | RE0 | RE1 | RE2 | Vdd | Vss | PCFG |
| D | D | D | D | D | D | D | D | | | 011x |
| A | D | D | D | D | D | D | D | $V_{R+}$ | $V_{R-}$ | 1110 |
| A | A | D | A | D | D | D | D | $V_{R+}$ | $V_{R-}$ | 0100 |
| A | A | A | A | A | D | D | D | $V_{R+}$ | $V_{R-}$ | 0010 |
| A | A | A | A | A | A | D | D | $V_{R+}$ | $V_{R-}$ | 1001 |
| A | A | A | A | A | A | A | A | $V_{R+}$ | $V_{R-}$ | 0000 |
| A | A | D | $V_{R+}$ | D | D | D | D | | $V_{R-}$ | 0101 |
| A | A | A | $V_{R+}$ | A | D | D | D | | $V_{R-}$ | 0011 |
| A | A | A | $V_{R+}$ | A | A | D | D | | $V_{R-}$ | 1010 |
| A | A | A | $V_{R+}$ | A | A | A | A | | $V_{R-}$ | 0001 |
| A | D | $V_{R-}$ | $V_{R+}$ | D | D | D | D | | | 1111 |
| A | A | $V_{R-}$ | $V_{R+}$ | D | D | D | D | | | 1101 |
| A | A | $V_{R-}$ | $V_{R+}$ | A | D | D | D | | | 1100 |
| A | A | $V_{R-}$ | $V_{R+}$ | A | A | D | D | | | 1011 |
| A | A | $V_{R-}$ | $V_{R+}$ | A | A | A | A | | | 1000 |

**ADC related configuration registers.**

| SFR | bit | |
|---|---|---|
| ADCON0 | 7,6 | ADCS1, ADCS0 (11 for ADC-RC clock) |
| | 5,4,3 | 0,..,7 selects RA0,RA1,RA2,RA3,RA5,RE0,RE1,RE2 |
| | 2 | GO_DONE : Set to begin conversion. |
| | 1 | don't care |
| | 0 | ADON : 1=ADC is powered up |
| ADCON1 | 7 | ADFM: ADC result format<br>0: MSB justified, ADRESH:ADRESL = xxxxxxxx xx000000<br>1: LSB justified, ADRESH:ADRESL = 000000xx xxxxxxxx |
| | 6 | ADCS2 , (0 for ADC-RC clock). |
| | 5,4 | don't care |
| | 3,2,1,0 | PCFG: Port configuration bits |
| ADRESH | 7,...,0 | 8-bit ADC result register. |
| ADRES | 15,...,0 | 16-bit ADC result register. |

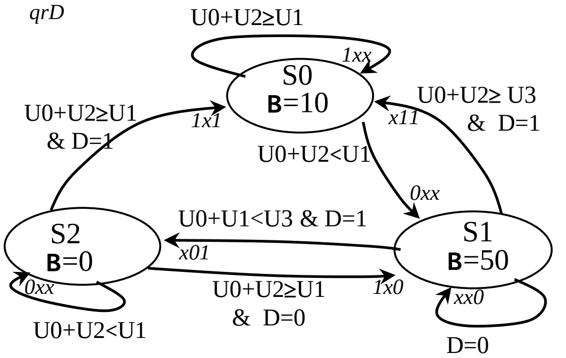
**ADC Frequency Setting**

Use always internal ADC-RC-clock by setting **(ADCS1, ADCS0)=(1,1); ADCS2=0**.

**Q1.** Consider a PIC18F452 system with four analog input voltages **U0, U1, U2** and **U3,** and a digital input signal **D** from **RA4**. Voltages **U0** and **U1** are in the range of (**0… 2V),** and **U2, U3** are in the range **(0…5V)**.

The system shall provide an 8-bit output **B** from **PORTB** according to the described FSM chart. The hardware of the system is already connected to satisfy the followings:

**RA0** is input for (Analog) **U0**, **RA1** is input for (Analog) **U1**,
**RA2** is input for (Analog) **U2**, **RA3** is input for (Vref) **2V**,
**RA4** outputs Blinkalive LED, **RA5** is input for (Analog) **U3**,
**PORTB** outputs **B.**




**Part-I**

-Find the ADC port config.
code to read u0 and u1 **[ PCFG01 =** ……………….]

-Find the ADC port config.
code to read u2 and u3 [ **PCFG23 =** ……………….]

-Find **ADCON1** register for 8-bit
conversion of u0 and u1. **[ADCON1u01=** ……………………………]

-Find **ADCON0** register for
conversion of u0. **[ADCON0u0=** …………………………..…]

-Find **ADCON1** register for 10-bit
conversion of u2 and u3. **[ADCON1u01=** ……………………………]

-Find **ADCON0** register for
conversion of u3. **[ADCON0u3=** ……………………………]

- Find the digital readings **Nu0** for **U0=1.2V** and **Nu3** for **U=1,2V**.
[**Nu0**= …………] [**Nu3**= …………]

-You can code the voltage conditions to integers {**q**, **r**}
(i.e., **q=0** if **NU0+NU2<NU1** ; **q=1** if **NU0+NU2>=NU1**;
and **r=0** if **NU0+NU2<NU3**; **r=1** if **NU0+NU2>=NU3**) ;
and code the states by **S**={**0, 1, 2**}.
Complete coefficients to construct the
index for state address table **ix= …*S+…*q +…*r+D**.
Complete the missing part of the state transition table
according to the FSM diagram. (Easiest method is first
note down the **q,r,D** values on each FSM arrow.
Then transfer them to the table.)

| Output Table | |
|---|---|
| S | B |
| 0 | …… |
| 1 | …… |
| 2 | …… |

| Next State Table | | | | |
|---|---|---|---|---|
| S | q | r | D | NS |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | …... |
| 1 | 0 | 0 | 1 | …… |
| 1 | 0 | 1 | 0 | …… |
| 1 | 0 | 1 | 1 | …… |
| 1 | 1 | 0 | 0 | …… |
| 1 | 1 | 0 | 1 | …… |
| 1 | 1 | 1 | 0 | …… |
| 1 | 1 | 1 | 1 | …… |
| 2 | 0 | 0 | 0 | 2 |
| 2 | 0 | 0 | 1 | 2 |
| 2 | 0 | 1 | 0 | 2 |
| 2 | 0 | 1 | 1 | 2 |
| 2 | 1 | 0 | 0 | 1 |
| 2 | 1 | 0 | 1 | 0 |
| 2 | 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 1 | 0 |

- Complete the following next-state and output arrays for index **ix**.

```
const char NST[]= (… , …, …, …,   …, …, …, …,   …, …, …, … ,
                   … , …, …, …,   …, …, …, …,   …, …, …, …
);

const char BOT[]= ( …, …, … );
```

**Name:**……………………………………………………………. **Signature**………………… **ID-No**:…………………….

**CMPE423 Embedded System Design Fall 2017-18, HW3 (return the solution before the Quiz Hour. Late HWs cannot be graded.)**

## Part-II

- Write a CC8E program applying the following steps

(a) declare global variables char `NU0, NU1, D, q, r, s, ix, B`. and uns16 `NU2,NU3.`

(b) declare global constant char arrays for state transition table `NST[]` and output table `BOT[];`

In the main procedure

(c) initialize ports for input-output as required by the problem statement.

(d) initialize `ADCON1` and `ADCON0` for reading 8-bit analog voltage from `RA0`;

(e) initialize the state `s`=0; and `ix=0`;

(f) start the main-loop;

(g) Convert input voltage (`U0` at `RA0`) to digital and store it into char `NU0`;

(h) Convert input voltage (`U1` at `RA1`) to digital and store it into char `NU1`;

(i) Convert input voltage (`U2` at `RA2`) to digital and store it into uns16 `NU2`;

(j) Convert input voltage (`U3` at `RA5`) to digital and store it into uns16 `NU3`;

(k) Calculate `q` and `r` using `NU0`, `NU1`, `NU2`, `NU3`,

(l) Calculate `ix` consistent to `NST[]` and `BOT[]`

(m) read state `s` from the next state table;

(n) read FSM output `B` from the output table;

(o) send `B` to `PORTB`;

(p) repeat main-loop forever;

(q) close the main procedure.

## Part-III

- Write ADC related codes using macro definitions that packs the ADC conversion using settings of ADCON1 and ADCON0 and the result as parameters, i.e., `ADCstart(A0, A1)`.
- Write the FSM related coding into procedure void `fsm(void){…}`.