HW06-1(Exceptions-General)

Exceptions:

- come in one of two types: **interrupts** and **resets**
- can be caused by illegal or erroneous instructions **software**
- is a break in normal program flow.
- can be divided into two types: **Hardware** interrupts, e.g an external interrupt when a pin changes, and **Software** interrupts, e.g. a divide by zero occurs"

Interrupts can be divided into two types: **asynchronous** which can happen at any time, and **synchronous** which occur at predetermined intervals.

Trying to execute an illegal instruction produces a **software** interrupt.

Another name for a software interrupt is a **trap** because it can catch a problem with a software instruction

Why are interrupts so important when dealing with I/O? That is, what do interrupts allow that polling does not
interrupt will stop. polling will continue even though an interrupt was detected.

A **interrupt** is an exception which saves the current processor state before executing the service routine

Which of the following are potential sources for maskable interrupts? **port H,J,P,PWM,SPI,SCI**

There are two types of interrupts which are determined by their timing.
An example if an **Asynchronous** interrupt is when a process detects an error condition.
An example of a **Synchronous** interrupt is a timer used to write to a display at specified periodic intervals.

There are two types of interrupts which are determined by their timing. **Synchronous** interrupts can be used to switch between processes at predetermined intervals.

A **Maskable** interrupt can be disabled and enabled through software.

Interrupts can be divided into two types: **Maskable** which can be turned on or off through the clearing or setting of certain register bits, and **Non-maskable** which can be turned on in software, but cannot be turned off with software.

An **Interrupt** is an exception which saves the current processor state before executing the service routine.

What is a clock monitor reset?
When a clock monitor senses a clock error and resets the clock?

What is the difference between a power-on reset and an external reset?
While they are both triggered by low-active reset pins, the Power-On Reset is activated at startup to guarantee a known initial state while the external reset is triggered by an external switch.

What is a Power-On Reset (POR)?
A low-active reset pin; it is activated at startup and guarantees a known initial state.

Hardware exceptions can be caused by resets initiated by the user or processor or caused by external interrupts and internal timers.

What does the programmer need to do every time an interrupt is handled by an interrupt service routine?

A[n] **interrupt** is an exception which saves the current processor state before executing the service routine.

A[n] **reset** is an exception which does not save the state of processor before executing its service routine.

A **clock monitor** reset occurs when the processor detects the system clock is not within a valid frequency range.

A[n] example of an **External** Reset is when a low-active pin is activated by a user pressing a switch to force the processor to reset.

A[n] **reset** is an exception described as a "one-way ticket" because the processor doesn't return to what it was doing when the exception occurred.

There are two types of interrupts which are determined by their timing. An example an **asynchronous** interrupt is when a processor detects an error condition.

What is a Watchdog Reset? A reset that happens if a signal has not been sent to the watchdog to reset the timer in a given amount of time.

What is a COP Reset? Same as above

Hardware exceptions can be caused by resets initiated by the user or processor or caused by external interrupts and internal timers.

Software exceptions can be caused by illegal or erroneous instructions

HW06-2(PIC32MX-Resets)

- Give a Single C instruction which will set variable **n** to a **1** if a PIC32MX was *not* in Sleep Mode when it was reset.
  if (RCONbits.SLEEP==0)
  n = 1;
  if (!RCONbits.SLEEP)
  n = 1;
- Give a Single C instruction which will set variable **n** to a **1** if a PIC32MX was in Sleep Mode when it was reset.
  All of these are valid answers:
  if (RCONbits.SLEEP==1) n = 1;
  if (RCONbits.SLEEP==1) n=1;
  if (RCONbits.SLEEP==1) n = 1
  if (RCONbits.SLEEP==1) n=1
  if (RCON & 0x0008) n = 1;
  if (RCONbits.SLEEP) n = 1;
  if (RCONbits.SLEEP) n=1;
  if (RCONbits.SLEEP) n = 1
  if (RCONbits.SLEEP) n=1
- Give a single C instruction which will set variable **n** to a **1** if a software reset has occurred on a PIC32MX.
  if (RCONbits.SWR==1) n = 1;
  if (RCONbits.SWR) n = 1;
  if (RCON & 0x0040) n = 1;
- Give a single C instruction which will enable a Software Reset on a PIC32MX.
  RSWRSTbits.SWRST = 1;
- Give a single C instructions which will disable a Software Reset on a PIC32MX.
  RSWRSTbits.SWRST = 0;
- Give a single C instructions which will set variable **n** to a **1** if a Power-On Reset has *not* occurred on a PIC32MX.
  if (RCONbits.POR==0) n = 1;
  if (RCONbits.POR==0) n=1;
  if (RCONbits.POR==0) n = 1
  if (RCONbits.POR==0) n=1
  if (!(RCON & 0x0001)) n = 1;
  if (!RCONbits.POR) n = 1;
  if (!RCONbits.POR) n=1;
  if (!RCONbits.POR) n = 1
  if (!RCONbits.POR) n=1
- Give a single C instructions which will set variable n to a 1 if a Brown-Out Reset has *not* occurred on a PIC32MX.
  if (RCONbits.BOR==0) n = 1;
- Give a single C instructions which will set variable n to a 1 if a Watchdog Timeout Reset has not occurred on a PIC32MX.
  if (RCONbits.WDTO==0) n = 1;
  if (RCONbits.WDTO==0) n=1;
  if (RCONbits.WDTO==0) n = 1
  if (RCONbits.WDTO==0) n=1
  if (!(RCON & 0x0010)) n = 1;
  if (!RCONbits.WDTO) n = 1;
  if (!RCONbits.WDTO) n=1;
  if (!RCONbits.WDTO) n = 1
  if (!RCONbits.WDTO) n=1
- Give a single C instructions which will set variable n to a 1 if a PIC32MX was in Idle Mode when it was reset.
  if (RCONbits.IDLE==1) n = 1;
  if (RCONbits.IDLE==1) n=1;

HW06-3(PIC32-InterruptInit)

External Interrupt	.INTx
Input Capture	.ICx
Comparator	CMPx

Write the C instructions necessary to initialize the External Interrupt 2 interrupt on a PIC32MX by:

1. Clearing the flag
2. Setting the priority to level 5
3. Setting the sub-priority to level 0
4. Enabling the interrupt.

1. IFS0bits.INT2IF = 0;
2. IPC2bits.INT2IP = 5;
3. IPC2bits.INT2IS = 0;
4. IEC0bits.INT2IE = 1;

Write the C instructions necessary to initialize the Output Compare 3 interrupt on a PIC32MX by:

1. Clearing the interrupt flag.
2. Setting the priority to level 5.
3. Setting the sub-priority to level 0.
4. Enabling the interrupt.
   1. **IFS0bits.OC3IF = 0;**
   2. **IPC3bits.OC3IP = 5;**
   3. **IPC3bits.OC3IS = 0;**
   4. **IEC0bits.OC3IE = 1;**

Write the C instructions necessary to initialize the I2C2 Slave Event interrupt on a PIC32MX by:

1. Clearing the interrupt flag.
2. Setting the priority to level 6.
3. Setting the sub-priority to level 1.
4. Enabling the interrupt.

   1. **IFS1bits.I2C2SIF = 0;**
   2. **IPC9bits.I2C2IP = 6;**
   3. **IPC9bits.I2C2IS = 1;**
   4. **IEC1bits.I2C2SIE = 1;**

Write the C instructions necessary to initIFialize the I2C1 Bus Collision Event interrupt on a PIC32MX by:

1. Clearing the interrupt flag. **IFS1bits.I2C1BIF = 0;**
2. Setting the priority to level 4. **IPC8bits.I2C1IP = 4;**
3. Setting the sub-priority to level 0. **IPC8bits.I2C1IS = 0;**
4. Enabling the interrupt. **IEC1bits.I2C1BIE = 1;**

Write the C instructions necessary to initialize the SPI1 Transfer Done interrupt on a PIC32MX by:

1. Clearing the interrrupt flag. **IFS1bits.SPI1TXIF = 0;**
2. Setting the priority to level 6. **IPC7bits.SPI1IP = 6;**
3. Setting the sub-priority to level 1. **IPC7bits.SPI1IS = 1;**
4. Enabling the interrupt. **IEC1bits.SPI1TXIE = 1;**

Write the C instructions necessary to initialize the I2C1 Master Event interrupt on a PIC32MX by:

1. Clearing the interrrupt flag. **IFS1bits.I2C1MIF = 0;**
2. Setting the priority to level 3. **IPC8bits.I2C1IP = 3**;
3. Setting the sub-priority to level 0. **IPC8bits.I2C1IS = 0;**
4. Enabling the interrupt. **IEC1bits.I2C1MIE = 1;**

Write the C instructions necessary to initialize the SPI2 Fault interrupt on a PIC32MX by:

1. Clearing the interrrupt flag. **IFS1bits.SPI2EIF = 0;**
2. Setting the priority to level 1. **IPC9bits.SPI2IP = 1;**
3. Setting the sub-priority to level 3. **IPC9bits.SPI2IS = 3;**

Write the C instructions necessary to initialize the Input Capture 1 interrupt on a PIC32MX by:

1. Clearing the flag
2. Setting the priority to level 1
3. Setting the sub-priority to level 3
4. Enabling the interrupt.

1. IFS0bits.IC1IF = 0;
2. IPC1bits.IC1IP = 1;
3. IPC1bits.IC1IS = 3;
4. IEC0bits.IC1E = 1;

Write the C instructions necessary to initialize the Comparator Interrupt 2 on a PIC32MX by:

1. Clearing the flag
2. Setting the priority to level 1
3. Setting the sub-priority to level 3
4. Enabling the interrupt.

1. IFS1bits.CMP2IF = 0;
2. IPC7bits.CMP2IP = 0;
3. IPC7bits.CMP2IS = 2;
4. IEC1bits.CMP2IE = 1;

Write the C instructions necessary to initialize the External Interrupt 4 interrupt on a PIC32MX by:

1. Clearing the interrupt flag.
2. Setting the priority to level 5.
3. Setting the sub-priority to level 3.
4. Enabling the interrupt.
   1. **IFS0bits.INT4IF = 0;**
   2. **IPC4bits.INT4IP = 5;**
   3. **IPC4bits.INT4IS = 3;**
   4. **IEC0bits.INT4IE = 1;**

Write the C instructions necessary to initialize the Timer3 interrupt on a PIC32MX by:

1. Clearing the interrupt flag.
2. Setting the priority to level 6.
3. Setting the sub-priority to level 3.
4. Enabling the interrupt.
   1.IFS0bits.T3IF = 0;
   2.IPC3bits.T3IP = 6;
   3.IPC3bits.T3IS = 3;
   4.IEC0bits.T3IE = 1;

Write the C instructions necessary to initialize the Timer2 interrupt on a PIC32MX by:

1. Clearing the interrupt flag.
2. Setting the priority to level 3.
3. Setting the sub-priority to level 0.
4. Enabling the interrupt.
   1.IFS0bits.T2IF = 0;
   2.IPC2bits.T2IP = 3;
   3.IPC2bits.T2IS = 0;
   4.IEC0bits.T2IE = 1;

HW06-3(PIC32-ExternalInterruptInit)
Write the C instruction necessary to have the external interrupt INT1 on a PIC32MX look for a Falling edge.
INTCONbits.INT1EP = 0;

..For INT4:
INTCONbits.INT4EP = 0;

HW06-3(PIC32-InterruptInit2)
See Table 7-1 for reference.
Write the C instructions to initialize the Input Capture 4 Error interrupt on a PIC32MX by:

1. Clearing the flag
2. Setting the priority to level 1
3. Setting the sub-priority to level 3
4. Enabling the interrupt.

1. IFS0bits.IC4EIF = 0;
2. IPC4bits.IC4IP = 7;
3. IPC4bits.IC4IS = 3;
4. IEC0bits.IC4IE = 1;

Write the C instructions necessary to initialize the PORTC Input Change Interrupt interrupt on a PIC32MX by:

1. Clearing the interrupt flag.
2. Setting the priority to level 1.
3. Setting the sub-priority to level 2.
4. Enabling the interrupt.
   1. **IFS1bits.CNCIF = 0;**
   2. **IPC8bits.CNIP = 1;**
   3. **IPC8bits.CNIS = 2;**
   4. **IEC1bits.CNCIE = 1;**

Write the C instructions necessary to initialize the PORTA Input Change Interrupt interrupt on a PIC32MX by:

1. Clearing the interrupt flag. **IFS1bits.CNAIF = 0;**
2. Setting the priority to level 0. **IPC8bits.CNIP = 0;**
3. Setting the sub-priority to level 3. **IPC8bits.CNIS = 3;**
4. Enabling the interrupt. **IEC1bits.CNAIE = 1;**

Write the C instructions necessary to initialize the UART1 Fault interrupt on a PIC32MX by:**WHY IS THIS "C8" AND NOT "C6"?**

1. Clearing the flag. **IFS1bits.U2RXIF = 0;**
2. Setting the priority to level 5. **IPC9bits.U2IP = 1;**
3. Setting the sub-priority to level 0. **IPC9bits.U2IS = 2;**
4. Enabling the interrupt. **IEC1bits.U2RXIE = 1;**

Write the C instructions necessary to initialize the UART2 Receiver interrupt on a PIC32MX by:

1. Clearing the flag. **IFS1bits.U1EIF = 0;**
2. Setting the priority to level 5. **IPC8bits.U1IP = 1;**
3. Setting the sub-priority to level 0. **IPC8bits.U1IS = 0;**
4. Enabling the interrupt. **IEC1bits.U1EIE = 1;**

Write the C instructions necessary to initialize the UART2 Transmitter interrupt on a PIC32MX by:

1. Clearing the interrupt flag. **IFS1bits.U2TXIF = 0;**
2. Setting the priority to level 5. **IPC9bits.U2IP = 5;**
3. Setting the sub-priority to level 1. **IPC9bits.U2IS = 1;**
4. Enabling the interrupt. **IEC1bits.U2TXIE = 1;**

Write the C instructions necessary to initialize the I2C1 Slave Event interrupt on a PIC32MX by:

1. Clearing the flag
2. Setting the priority to level 6
3. Setting the sub-priority to level 0
4. Enabling the interrupt
   1. IFS1bits.I2C1SIF = 0;
   2. IPC8bits.I2C1IP = 6;
   3. IPC8bits.I2C1IS = 0;
   4. IEC1bits.I2C1SIE = 1;

4. Enabling the interrupt. **IEC1bits.SPI2EIE = 1;**

Write the C instructions necessary to initialize the Core Software Interrupt 1 on a PIC32MX by:

1. Clearing the interrupt flag. **IFS0bits.CS1IF = 0;**
2. Setting the priority to level 1. **IPC0bits.CS1IP = 1;**
3. Setting the sub-priority to level 3. **IPC0bits.CS1IS = 0;**
4. Enabling the interrupt. **IEC0bits.CS1IE = 1;**

Write the C instructions necessary to initialize the Flash Control Event on a PIC32MX by:

1. Clearing the interrupt flag. **IFS0bits.FCEIF = 0;**
2. Setting the priority to level 1. **IPC6bits.FCEIP = 1;**
3. Setting the sub-priority to level 3. **IPC6bits.FCEIS = 0;**
4. Enabling the interrupt. **IEC0bits.FCEIE = 1;**

Write the C instructions necessary to initialize the External Interrupt 1 interrupt on a PIC32MX by:

1. Clearing the interrupt flag. **IFS0bits.INT1IF = 0;**
2. Setting the priority to level 3. **IPC1bits.INT1IP = 3;**
3. Setting the sub-priority to level 2. **IPC1bits.INT1IS = 2;**
4. Enabling the interrupt. **IEC0bits.INT1IE = 1;**

Write the C instructions necessary to initialize the Output Compare 1 interrupt on a PIC32MX by:

1. Clearing the interrupt flag. **IFS0bits.OC1IF = 0;**
2. Setting the priority to level 1. **IPC1bits.OC1IP = 1;**
3. Setting the sub-priority to level 3. **IPC1bits.OC1IS = 3;**
4. Enabling the interrupt **IEC0bits.OC1IE = 1;**

HW06-3(PIC32MX-Interrupts)
How many different Priority values are there in the sub-priority level of a PIC32MX? **4**
What is the *lowest* priority *value* in the *sub*-priority level of a PIC32MX? **0**

How is a persistent interrupt different form a non-persistent interrupt?
Persistent interrupts will remain active and the associated interrupt flag set until the issue causing the interrupt serviced. In non persistent interrupts, the interrupt is recorded once to the interrupt controller which presents it to the CPU.

What is the highest priority value in the main priority level? **7**

How many different priority *levels* are there in the PIC32MX? **2**

What header file provides functions and definitions to handle PIC32MX interrupts? **int.h**

what is the lowest priority value in the main priority level? **0**

what external events can cause an external interrupt? **a reset**

How many possible interrupt sources does a PIC32 have? **64**

What does the INTSTAT register in the PIC32 tell the programmer?
The priority number and vector number of the latest interrupt presented to the CPU. - Pg 91 Data sheet
what does the IP17 mean in the code below?
#pragma interrupt Int0_IRQ ip17 vector 3

What does the **ipl2** mean in the code below?

```
void __ISR(_EXTERNAL_0_VECTOR,
ipl2) Int0_IRQ(void);
```

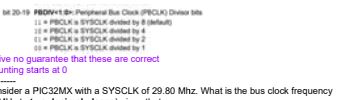
What external event(s) can cause an external interrupt on a PIC32MX?
The low active reset pin being activated (set HIGH) *This isn't a confirmed answer*

What does a PIC32MX do if two interrupts with the *same* priority value occur at the same time?
it compares the sub priorities
Chapter 7

HW07-2(PIC32-OSCCON-Frequency)
Consider a PIC32MX with a SYSCLK of 34.20 Mhz. What is the bus clock frequency (in MHz to two decimal places) given that
OSCCON = 0x1950724F
PBCLK = 8.55 MHz

Instructions:

1. Take the bits 19 and 20 from the binary value of OSCCON:
   1100101010**00**00111001001001111 (start count from right to left, starting from 0)
2. Refer to the chart/image below (found on pg. 97 of datasheet) & take the frequency given and divide by whatever is says in the image to get answer. In this case: 34.20 MHz ÷ 4 = 8.55 MHz



*I give no guarantee that these are correct
*counting starts at 0

Consider a PIC32MX with a SYSCLK of 29.80 Mhz. What is the bus clock frequency (in MHz to **two decimal places**) given that
OSCCON = 0x394172DD
PBCLK = 29.80 MHz

Consider a PIC32MX with a SYSCLK of 15.60 Mhz. What is the bus clock frequency (in MHz to two decimal places) given that
OSCCON = 0x2A7E12A3
PBCLK = 1.95 MHz

Consider a PIC32MX with a SYSCLK of 10.10 Mhz. What is the bus clock frequency (in MHz to two decimal places) given that
OSCCON = 0x027842B4
PBCLK = 1.26 MHz

Consider a PIC32MX with a SYSCLK of 14.30 Mhz. What is the bus clock frequency (in MHz to two decimal places) given that
OSCCON = 0x07023278
PBCLK = 14.3 MHz

HW07-2(PIC32-OSCCON-PBDIV)
Consider a PIC32MX with a SYSCLK of 20.90 MHz. Give a single C instruction to produce a bus clock as close to 14.72 MHz as possible.
OSCCONbits.PBDIV = 1;

I can't find the exact steps on how to do this, so here is my best guess:

1. Take the SYSCLK frequency and divide it by 1,2,4, and 8. You can do this all at once in your calculator by typing: `20.90/{1,2,4,8}`
   a. This should give you 4 results in brackets: `{20.9 10.45 5.225 2.6125}`
   b. All this command does is divide one number by multiple numbers all at once.
2. Take the result that is closest to the desired bus clock frequency, in this case 10.45 is closest to 14.72MHz, and note which number you divided it by. In this case, 10.45 was the result of 20.90/2, so that divisor was 2.
3. Using the chart used in the previous question (7-2 OSCCON-Frequency), take the divisor bits that match up with your value determined in step 2 and convert to decimal. In this example, the line that says "...divided by 2", has divisor bits 01, which is just decimal value 1. This means you need to set this command equal to

---

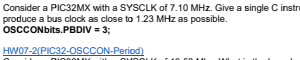
Consider a PIC32MX with a SYSCLK of 26.60 MHz. Give a single C instruction to produce a bus clock as close to 5.20 MHz as possible.
OSCCONbits.PBDIV = 2;
`26.60/{1,2,4,8} = {26.6 13.3 6.65 3.325}`

Consider a PIC32MX with a SYSCLK of 11.40 MHz. Give a single C instruction to produce a bus clock as close to 3.96 MHz as possible.
OSCCONbits.PBDIV = 2;

Consider a PIC32MX with a SYSCLK of 11.70 MHz. Give a single C instruction to produce a bus clock as close to 4.53 MHz as possible.
OSCCONbits.PBDIV = 1;

Consider a PIC32MX with a SYSCLK of 7.10 MHz. Give a single C instruction to produce a bus clock as close to 1.23 MHz as possible.
OSCCONbits.PBDIV = 3;

HW07-2(PIC32-OSCCON-Period)
Consider a PIC32MX with a SYSCLK of 19.50 Mhz. What is the bus clock period (in ns to 2 decimal places) given that OSCCON = 0x2B04326B
Period of PBCLK = **51.28** ns



Use the same chart from the last two 7-2 problems.
EQUATION:
(n=number in "divided by n", F=SYSCLK frequency (**in Hz**)
Period = 10^9/(F*10^6 * (1/n))

1. Convert hex number to binary and get bits 20-19 to figure out what to divide SYSCLK by, this is your value for *n*.
2. Answer = 10^9/(19.5*10^6 * (1/1)) = 51.28205 ns = 51.28 ns

Consider a PIC32MX with a SYSCLK of 37.40 Mhz. What is the bus clock period (in ns to 2 decimal places) given that OSCCON = 0x200F02D1
Period of PBCLK = **53.48** ns

Consider a PIC32MX with a SYSCLK of 24.00 Mhz. What is the bus clock period (in ns to 2 decimal places) given that OSCCON = 0x166E026B
Period of PBCLK = **83.33** ns

Consider a PIC32MX with a SYSCLK of 6.20 Mhz. What is the bus clock period (in ns to 2 decimal places) given that OSCCON = 0x135622F8
Period of PBCLK = **645.16** ns

Consider a PIC32MX with a SYSCLK of 15.60 Mhz. What is the bus clock period (*in ns to 2 decimal places*) given that OSCCON = 0x3A25124A
Period of PBCLK = **64.10** ns

HW07-3(PIC32-CoreTimer-CompareForFrequency)

Given the following C instructions on a PIC32MX running at 10.5 MHz,

```
_CP0_SET_COUNT(0);
_CP0_SET_COMPARE(compare);
IFS0bits.CTIF = 0;
IFS0bits.CTIE = 1;
```

what does the value of compare need to be to give a Core Timer interrupt frequency as close to 0.656 Hz as possible?
Answer = **8003049**

Formula: $f_{CT}=f_{SYSCLK}/(2 * COMPARE)$
Using the above problem, solve for COMPARE: 0.656 = 10.5*10^6 / (2 * COMPARE), COMPARE = 8003049.
I use the solve() function on the TI-89 to avoid confusion, and less work. On the TI-89 you can just enter: `solve(0.656=10.5e6/(2*x),x)`

Given the following C instructions on a PIC32MX running at 6.8 MHz,

```
_CP0_SET_COUNT(0);
_CP0_SET_COMPARE(compare);
IFS0bits.CTIF = 0;
IFS0bits.CTIE = 1;
```

what does the value of compare need to be to give a Core Timer interrupt frequency as close to 701 Hz as possible?
Answer = **4850**

Given the following C instructions on a PIC32MX running at 19.4 MHz,

```
_CP0_SET_COUNT(0);
_CP0_SET_COMPARE(compare);
IFS0bits.CTIF = 0;
IFS0bits.CTIE = 1;
```

what does the value of compare need to be to give a Core Timer interrupt frequency as close to 2.4 MHz as possible?
Answer = **4.0417**

HW07-3(PIC32-CoreTimer-CompareForPeriod)
Formula: $f_{CT}=f_{SYSCLK}/(2 * COMPARE)$
Given the following C instructions on a PIC32MX running at 19.6 MHz,

```
_CP0_SET_COUNT(0);
_CP0_SET_COMPARE(compare);
IFS0bits.CTIF = 0;
IFS0bits.CTIE = 1;
```

what does the value of compare need to be to give a Core Timer interrupt frequency as close to 183 kHz as possible?
Answer = **53.55**
Same process as CompareForFrequency above.
If given time, f=1/T

Given the following C instructions on a PIC32MX running at 2.1 MHz,

```
_CP0_SET_COUNT(0);
_CP0_SET_COMPARE(compare);
IFS0bits.CTIF = 0;
IFS0bits.CTIE = 1;
```

what does the value of compare need to be to give a Core Timer interrupt frequency as close to 363 kHz as possible?
Answer = **2892.56**

HW07-3(PIC32-CoreTimer-WhatIsFrequency)
Given the following C instructions on a PIC32MX running at 13 MHz,

```
_CP0_SET_COUNT(0);
_CP0_SET_COMPARE(104202);
IFS0bits.CTIF = 0;
IFS0bits.CTIE = 1;
```

what is the frequency of the Core Timer interrupt in Hz (to one decimal place)?
Answer = **62.4**
Same equation as CompareForFrequency and CompareForPeriod above, but you're solving for f_{CT} this time. These problems seem to be picky about the decimal places *sometimes*.Formula: $f_{CT}=f_{SYSCLK}/(2 * COMPARE)$

HW07-3(PIC32-CoreTimer-WhatIsPeriod)
Given the following C instructions on a PIC32MX running at 3.9 MHz,

```
_CP0_SET_COUNT(0);
_CP0_SET_COMPARE(220);
IFS0bits.CTIF = 0;
IFS0bits.CTIE = 1;
```

what is the period of the Core Timer interrupt in **micro** seconds (to one decimal place)?
Answer = **112.8**

Instructions
1. Use the same formula as above: $f_{CT}=f_{SYSCLK}/(2 * COMPARE)$
   a. To get T_{CT} (period of core timer) in micro seconds: $T_{CT} = 10^6/f_{CT}$
   b. To get T_{CT} in **milli**seconds: $T_{CT} = 10^3/f_{CT}$

Given the following C instructions on a PIC32MX running at 3 MHz,

```
_CP0_SET_COUNT(0);
_CP0_SET_COMPARE(859590);
IFS0bits.CTIF = 0;
IFS0bits.CTIE = 1;
0
```

what is the period of the Core Timer interrupt in **milliseconds** (to one decimal place)?
Answer = **573.1**

HW07-4(PIC32-T1CON)
Give a single C instruction to configure a PIC32MX's Timer 1 with the following properties.

Timer is enabled
Continue module operation when the device enters Idle Mode.
Writes to Timer1 are ignored until pending write operation completes.
1:64 prescale value

External clock input IS synchronized.
External clock.

Answer: **T1CON = 0x9026;**

Instructions:
See page 157 of the datasheet for Timers 2-5 and page 152 for Timer 1 for reference/further info
1. Build the binary number: 0000000000000000**A**0**CDE**000**G**0**II**0**KL**0
   - A = Timer on bit (1=on)
   - C = SIDL (1=discontinue operation in idle mode)
   - D = TWDIS (1=writes to timer are ignored until pending write is complete)
   - E = TWIP (1=asynchronous write to timer1 is in progress)
   - G = TGATE *This bit depends on the TCS bit (L). If ignored, =0.*
   - II = TCKPS (2 bits) (00=1,01=8,10=64,11=256)
   - K = TSYNC (1=external clock input is synchronized
   - L = TCS (1=external clock)
2. For this problem, this would leave you with this (with leading zeros excluded): 1001000000100110 (Note: I underlined the middle zeros to make this easier)
3. Convert that binary number to hex and make it into the command.

Give a single C instruction to configure a PIC32MX's Timer 1 with the following properties.

Timer is disabled.
Discontinue module operation when the device enters Idle Mode.
Back-to-back writes are enabled.
1:64 prescale value
External clock input IS synchronized.
External clock.

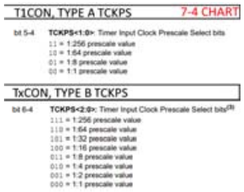
Answer = **T1CON = 0x2026;**
Steps I took to "build" this binary number:
1. 0-100---0-10-11-
2. 0010000000100110
3. In hex: 0x2026

HW07-4(PIC32-TimerClockFrequency)
A PIC32MX has a peripheral bus clock frequency of 21.4 MHz.
What is the Timer 4 clock frequency (in MHz to three decimal places) given the following register setting?

T4CON = 0x00008010
Answer = **10.700;**

Note: Timer 1 is a type A and timers 2-5 are type B timers, but I don't think that is important for this. -It matters for which bits you look at, Type A is only 2 bits, B is 3
1. Convert hex to binary
2. Get bits 4-6 and look at TCKPS value and find the prescale (PS) value
3. Plug that into this equation: $C_{BUS} = C_{CORE} / PS$ and solve for C_{CORE} (Core clock frequency)

That will look like this: `21.4E6 * (1/2) * 10-6 = 10.700`


T1CON, TYPE A TCKPS — 7-4 CHART
bit 5-4 TCKPS<1:0>: Timer Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

TxCON, TYPE B TCKPS
bit 6-4 TCKPS<2:0>: Timer Input Clock Prescale Select bits[3]
111 = 1:256 prescale value
110 = 1:64 prescale value
101 = 1:32 prescale value
100 = 1:16 prescale value
011 = 1:8 prescale value
010 = 1:4 prescale value
001 = 1:2 prescale value
000 = 1:1 prescale value

A PIC32MX has a peripheral bus clock frequency of 6.3 MHz.
What is the Timer 4 clock frequency (in MHz to three decimal places) given the following register setting?

T4CON = 0x00008010;
Answer = **3.150**

HW07-4(PIC32-TimerClockPeriod)
A PIC32MX has a peripheral bus clock frequency of 30.1 MHz
What is the Timer 1 clock period (in μs to three decimal places) given the following register setting?
T1CON = 0x000088B0;
Answer = **8.505**

Similar to the previous 7-4 problem, get the binary value of the hex number and read the prescale (PS) value from the *7-4 Chart*. Use the equation below to get the result in microseconds. Remember PS is a fraction (e.g. ⅛ or 1/256)

$T_{CLK} = 10^6/(C_{BUS} \times PS)$
Using above problem: `10⁶/(30.1E6*1/256)`

A PIC32MX has a peripheral bus clock frequency of 6.3 MHz.
What is the Timer 3 clock period (in μs to three decimal places) given the following register setting?
T3CON = 0x00008008;
Answer = **0.159**

HW07-4(PIC32-TimerPeriod-MinMax)
If a PIC32MX has a bus clock of 16.60 MHz, what does TCKPS need to be to be able to produce timer periods from 0.07 μs to 3.8 ms with Timer 4? (Type 'X' if not possible.)
Answer = **0**

Use the equation in TimerClockPeriod, but solve for PS. *WORK IN BASE UNITS*
$T_{CLK} = 10^6/(C_{BUS} \times PS)$
1. A = PS using minimum timer period (0.07 us)
2. B = PS using max timer period (3.8 ms)
3. Choose the prescale value from *Chart 7-4* that is just smaller than the min calculated between A and B.
4. The binary value (bits 6-4) associated with the prescale value is converted to decimal and is the final answer
5. If B is greater than 65536 (2^16), The answer is **X**
6. If B is less than 65536, then you match it up with the closest possible prescale value in the *7-4 Chart* and that should be your answer. I haven't gotten anything other than 0 or X though.

You can also just use this equation in the calculator to get A & B all at once:
`csolve(1/(CBUS*1/n)={TMIN, TMAX},n)`. Example: `csolve(1/(16.6E6*1/n)={.07E-6,3.8E-3}` Result: `n={1.162, 63080}`

```
Check if Possible
     TMin = 1/bus              TMax = (1/bus)*(2^16-1)
if   Tmin < min    &           Tmax > max ?        ans is not X
```

If a PIC32MX has a bus clock of 5.10 MHz, what does TCKPS need to be to be able to produce timer periods from 0.20 μs to 10 ms with Timer 1? (Type 'X' if not possible.)
Answer = **0** {1.02, 51000}

If a PIC32MX has a bus clock of 15.10 MHz, what does TCKPS need to be to be able to produce timer periods from 0.07 μs to 4.4 ms with Timer 3? (Type 'X' if not possible.)
Answer = **X** {1.057, 66440}   Tmax = .0043 is less than .0044 => X

If a PIC32MX has a bus clock of 18.90 MHz, what does TCKPS need to be to be able to produce timer periods from 0.06 μs to 3.4 ms with Timer 1? (Type 'X' if not possible.)

Answer = **0** {1.134, 64260}   Tmax = 0.00346 is greater than 0.0034 => not X

If a PIC32MX has a bus clock of 10.20 MHz, what does TCKPS need to be to be able to produce timer periods from 26 μs to 1.7 seconds with Timer 2? (Type 'X' if not possible.)
Answer = **X** {265.2, 17340000}

If a PIC32MX has a bus clock of 18.90 MHz, what does TCKPS need to be to be able to produce timer periods from 15 μs to .85 seconds with Timer 4? (Type 'X' if not possible.)
Answer = **7** {283.5, 16065000}   doesn't work?
Can someone explain why this is 7 and not X? Is it because the PS is 256 and 256*65536 is the new value MAX should be compared too? *It's a mystery*.

If a PIC32MX has a bus clock of **19.80 MHz**, what does **TCKPS** need to be to be able to produce timer periods from **13 μs** to **0.80 seconds** with **Timer 4**? (Type 'X' if not possible.)
Answer = **7**

HW07-4(PIC32-TimerPeriod)
What is the timer timeout length (in ms to two decimal places) of a PIC32MX's Timer 1 after running the following instructions given a bus clock frequency of 33.34 MHz?

T1CON = 0x00000020; → TCKPS bits are 10 (b/c Timer 1) → Prescale is 1:64
TMR1 = 0;
PR1 = 3649;
T1CONSET = 0x00008000;
Answer = **7.00**

1. Get the PS value from the T1CON value using the *7-4 Chart*. This one (above) is 1/64. **Make sure to use the appropriate chart like the other ones above. This is Timer1 (16-bit) but yours could be Timers2-5 (32-bit)**.
2. Plug your PS value into this equation: $(PS/C_{BUS})*PR1$
3. This will give you the answer in seconds
   a. I find using the second PS number works
   b. (64)/(33.34M)*3649*10^3 = 7.0046

What is the timer timeout length (in micro seconds to one decimal place) of a PIC32MX's Timer 4 after running the following instructions given a bus clock frequency of 29.94 MHz? **Note that this is timer 4!**

T4CON = 0x00000050; -->TCKPS bits are 101, which is prescale 1:32
TMR4 = 0;
PR4 = 32;

T4CONSET = 0x00008000;
Answer = **34.2**

What is the timer timeout length (in ms to two decimal place) of a PIC32MX's Timer 1 after running the following instructions given a bus clock frequency of 7.33 MHz?

T1CON = 0x00000020;
TMR1 = 0;
PR1 = 3183;
T1CONSET = 0x00008000;
Answer = **27.79**

What is the timer timeout length (in micro seconds) of a PIC32MX's Timer 4 after running the following instructions given a bus clock frequency of 7.65 MHz?

T4CON = 0x00000020;
TMR4 = 0;
PR4 = 2104;
T4CONSET = 0x00008000;
Answer = **1100.0**

HW07-4(PIC32-Timers-TCKPSForFreqeuncy)
Using a PIC32MX with a bus clock frequency of 16.90 MHz, produce a Timer 2 clock frequency as close to 0.085 MHz as possible with a single C instruction assuming PBCLK is used.
T2CONbits.TCKPS = 7;

1. Solve for prescale (PS) using equation from TimerClockFrequency question:
   a. $C_{BUS} = C_{CORE} / (PS)$
   b. Or use this equation: $PS = C_{CORE}/(C_{BUS})$
   c. Result for above question: PS = 1:198.82
2. In the *7-4 Chart*, find the prescale value that is the ***next highest*** to the PS value you determined and get the bits needed to set that as the prescale value
   Result, using above question: 1:256 → 111
3. In the command, make sure to use the appropriate timer number and make the command equal to the decimal result of that binary number you got in #2.
   T*2*CONbits.TCKPS = *7*;

Using a PIC32MX with a bus clock frequency of 18.00 MHz, produce a Timer 5 clock frequency as close to 3.516 MHz as possible with a single C instruction assuming PBCLK is used.
T5CONbits.TCKPS = 2;

Using a PIC32MX with a bus clock frequency of 7.50 MHz, produce a Timer 2 clock frequency as close to 0.148 MHz as possible with a single C instruction assuming PBCLK is used.
T2CONbits.TCKPS = 6;

Using a PIC32MX with a bus clock frequency of 8.60 MHz, produce a Timer 3 clock frequency as close to 0.026 MHz as possible with a single C instruction assuming PBCLKis used.
T3CONbits.TCKPS = 7;

HW07-4(PIC32-Timers-TCKPSForPeriod)
Using a PIC32MX with a bus clock frequency of 14.90 MHz, produce a Timer 3 clock period as close to 0.403 micro seconds as possible with a single C instruction assuming PBCLK is used.
T3CONbits.TCKPS = 3;

Same thing as *TCKPSForFrequency* questions above, except you need to take the clock period and invert it, after which it is the same process as before.
$PS = C_{CORE}/(1/t_{CLK})$. This can also be done with $PS = C_{CORE} * t_{CLK}$. Choose the closest PS value that is closest to what is calculated.
A. (1/P) = target frequency
B. $C_{BUS} / x = F_{target}$
C. Choose closet prescale value to x (round up OR down)

Using a PIC32MX with a bus clock frequency of 10.80 MHz, produce a Timer 3 clock period as close to 9.156 micro seconds as possible with a single C instruction assuming PBCLK is used.
T3CONbits.TCKPS = 7;

Using a PIC32MX with a bus clock frequency of 5.10 MHz, produce a Timer 5 clock period as close to 26.102 micro seconds as possible with a single C instruction assuming PBCLK is used.
T5CONbits.TCKPS = 7;

Using a PIC32MX with a bus clock frequency of 8.60 MHz, produce a Timer 4 clock period as close to 0.148 micro seconds as possible with a single C instruction assuming PBCLK is used.
T4CONbits.TCKPS = 0;

Using a PIC32MX with a bus clock frequency of 7.50 MHz, produce a Timer 3 clock period as close to 1.323 micro seconds as possible with a single C instruction assuming PBCLK is used.
T3CONbits.TCKPS = 3;

Using a PIC32MX with a bus clock frequency of 13.60 MHz, produce a Timer 2 clock period as close to 1.459 micro seconds as possible with a single C instruction assuming PBCLK is used.
T2CONbits.TCKPS = 4;

HW07-4(PIC32-TxCON)
Give a single C instruction to configure a PIC32MX's Timer 4 with the following properties.

Timer is ENabled
Discontinue module operation when the device enters Idle Mode.
Gated time accumulation is DISabled.
1:1 prescale value.
32-bit Mode.
INternal peripheral Clock.
1. 1010000000001000
2. Hex: 0xA008

T4CON = 0xA008;

Instructions:
This is basically the PIC32-T1CON question, but for the other timers. Use the datasheet, page 157, for reference.
Build the binary number in this format:
A0**B**0,0000,**CDDD**,**E**0**F**0
A = ON
B = SIDL (1=discontinue in idle)
C = TGATE (gate time is enabled. Irrelevant if TC is on)
D = TCKPS (3bits) (0=1 scale, 1=2,2=4,3=8,4=16,5=32,6=64,7=256)
E = T32 (1=32 bit mode)
F = TCS (1=external clock)
Have fun!

Give a single C instruction to configure a PIC32MX's Timer 4 with the following properties.

Timer is ENabled
Continue module operation when the device enters Idle Mode.
Gated time accumulation is ENabled.
1:2 prescale value.
32-bit Mode.
INternal peripheral Clock.
T4CON = 0x8098;

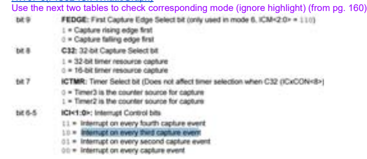
Give a single C instruction to configure a PIC32MX's Timer 2 with the following properties.

Timer is ENabled
Discontinue module operation when the device enters Idle Mode.
Gated time accumulation is ENabled.
1:64 prescale value
32-bit Mode.
INternal peripheral Clock.
T2CON = 0xA0E8;

Give a single C instruction to configure a PIC32MX's Timer 4 with the following properties.

Timer is ENabled
Continue module operation when the device enters Idle Mode.
Gated time accumulation is ENabled.
1:32 prescale value.
32-bit Mode.
INternal peripheral Clock.
T4CON = 0x80D8

HW07-5(PIC32-ICx-WhatIsGraph)
Use the next two tables to check corresponding mode (ignore highlight) (from pg. 160)

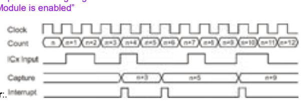

bit 9 FEDGE: First Capture Edge Select bit (only used in mode 6, ICM<2:0> = 110)
1 = Capture rising edge first
0 = Capture falling edge first
bit 8 C32: 32-bit Capture Select bit
1 = 32-bit timer resource capture
0 = 16-bit timer resource capture
bit 7 ICTMR: Timer Select bit (Does not affect timer selection when C32 (ICxCON<8>)
0 = Timer3 is the counter source for capture
1 = Timer2 is the counter source for capture
bit 6-5 ICI<1:0>: Interrupt Control bits
11 = Interrupt on every fourth capture event
10 = Interrupt on every third capture event
01 = Interrupt on every second capture event
00 = Interrupt on every capture event


ICM<2:0>: Input Capture Mode Select bits
111 = Interrupt-Only mode (only supported while in Sleep mode or Idle mode)
110 = Simple Capture Event mode – every edge, specified edge first and every edge thereafter
101 = Prescaled Capture Event mode – every sixteenth rising edge
100 = Prescaled Capture Event mode – every fourth rising edge
011 = Simple Capture Event mode – every rising edge
010 = Simple Capture Event mode – every falling edge
001 = Edge Detect mode – every edge (rising and falling)
000 = Input Capture module is disabled

Example 1
Consider the code below for a PIC32MX's Input Capture module:

IC1CONbits.ICM = 2;
IC1CONbits.ICI = 0;
IC1CONbits.FEDGE = 0;
IC1CONbits.ON = 1;

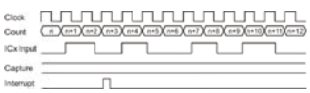
Which graph below correlates best to these settings?
1. "Simple capture event mode, every falling edge"
2. "Interrupt on every capture event"
3. "Capture on falling edge first"
4. "Module is enabled"

Answer:


Example 2
Consider the code below for a PIC32MX's Input Capture module:

IC1CONbits.ICM = 7;
IC1CONbits.ICI = 1;
IC1CONbits.FEDGE = 1;
IC1CONbits.ON = 1;

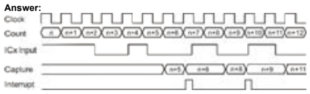
Which graph below correlates best to these settings?
1. "Interrupt-Only Mode"
2. "Interrupt on every second capture event"
3. "Capture on rising edge first"
4. "Module is enabled"



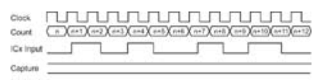
Example 3
Consider the code below for a PIC32MX's Input Capture module:

IC1CONbits.ICM = 6;
IC1CONbits.ICI = 1;
IC1CONbits.FEDGE = 1;
IC1CONbits.ON = 1;

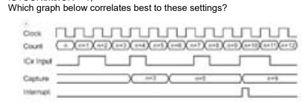
Which graph below correlates best to these settings?
1. "Simple Capture Event mode – every edge, specified edge first and every edge thereafter"
2. "Interrupt on every second capture event"
3. "Capture on rising edge first"
4. "Module is enabled"

Answer:


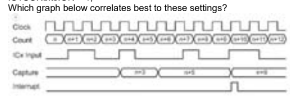
Example 4
Consider the code below for a PIC32MX's Input Capture module:
IC1CONbits.ICM = 0; "Input Capture module is disabled"
IC1CONbits.ICI = 2; "Interrupt on every capture event" → This doesn't matter b/c of ICM
IC1CONbits.FEDGE = 1; "Capture rising edge first" → Also doesn't matter b/c of ICM
IC1CONbits.ON = 1;
Which graph below correlates best to these settings?
Answer:



Example 5
Consider the code below for a PIC32MX's Input Capture module:
IC1CONbits.ICM = 2; "Simple Capture Event Mode-Every Falling Edge"
IC1CONbits.ICI = 2; "Interrupt on every third capture event"
IC1CONbits.FEDGE = 1; "Capture rising edge first"
IC1CONbits.ON = 1;
Which graph below correlates best to these settings?

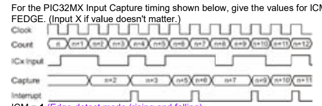


Example 6
Consider the code below for a PIC32MX's Input Capture module:
IC1CONbits.ICM = 2; "Simple Capture Event Mode-Every Falling Edge"
IC1CONbits.ICI = 2; "Interrupt on every third capture event"
IC1CONbits.FEDGE = 0; "Capture falling edge first"
IC1CONbits.ON = 1;
Which graph below correlates best to these settings?



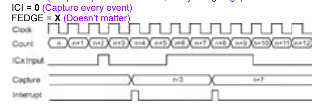
HW07-5(PIC32-ICx-WhatIsMode)
***NO SEMI-COLONS**

For the PIC32MX Input Capture timing shown below, give the values for ICM, ICI, and FEDGE. (Input X if value doesn't matter.)

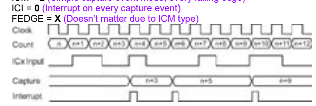


ICM = **1** (Edge-detect mode (rising and falling)
ICI = **1** (Interrupt on every 2nd capture event)
FEDGE = **X** (Doesn't matter due to ICM value)

For the PIC32MX Input Capture timing shown below, give the values for ICM, ICI, and FEDGE. (Input X if value doesn't matter.)
ICM = **3** (Simple capture event mode, every rising edge)
ICI = **0** (Capture every event)
FEDGE = **X** (Doesn't matter)

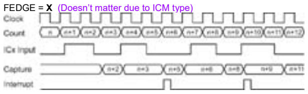


For the PIC32MX Input Capture timing shown below, give the values for ICM, ICI, and FEDGE. (Input X if value doesn't matter.)
ICM = **2** (Simple capture event mode, every falling edge)
ICI = **0** (Interrupt on every capture event)
FEDGE = **X** (Doesn't matter due to ICM type)

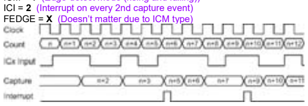


For the PIC32MX Input Capture timing shown below, give the values for ICM, ICI, and FEDGE. (Input X if value doesn't matter.)
ICM = **1** (Edge-detect mode (rising and falling))
ICI = **2** (Interrupt on every 2nd capture event)

FEDGE = **X** (Doesn't matter due to ICM type)



For the PIC32MX Input Capture timing shown below, give the values for ICM, ICI, and FEDGE. (Input X if value doesn't matter.)
ICM = **1** (Edge-detect mode (rising and falling))
ICI = **2** (Interrupt on every 2nd capture event)
FEDGE = **X** (Doesn't matter due to ICM type)



HW07-5(PIC32-ICxCON)

here for formatting. Delete when stuff is ready to be .
Instructions:
This is basically the PIC32-T1CON question, but for the Input Capture. Use the datasheet, page 160, for reference.
Build the binary number in this format (hyphens are placeholders for 0's): **A-B---CDEFFGHIII**
A = ON
B = SIDL
C = FEDGE
D = C32
E = ICTMR
F = ICI
G = ICOV (if not mentioned, set equal to 0)
H = ICBNE (if not mentioned, set equal to 0)
I = ICM
Have fun!

Give a single C instruction to configure a PIC32MX's Input Capture Module 2 with the following properties:

Module is DISabled.
Discontinue module operation when in Idle Mode.
Capture Rising Edge First.
32-bit mode.
Timer 3 is counter source.
Interrupt on every fourth capture event.
Simple Capture Event Mode - every falling edge.
1. 0-1---1101100010
2. 0010001101100010
3. 0x2362

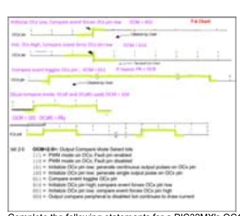
Answer: **IC2CON = 0x2362;**

Give a single C instruction to configure a PIC32MX's Input Capture Module 4 with the following properties:

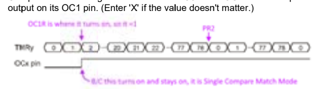
Module is ENabled
Continue module operation when in Idle Mode.
Capture Rising Edge First.
32-bit mode.
Timer 3 is counter source.
Interrupt every capture event.
Edge Detect Mode.

Answer: **IC4CON = 0x8301;**

HW07-6(PIC32-OCx-WhatAreSettings)
.OCM command depends on graph below (from pg. 164)
PR2 is the highest value the TMRy value gets to before resetting to 0
OC1R is the value where the rising edge starts at (see graphic below)
OC1RS is the value of the *falling* edge of the first pulse. This only applies to diagrams that have pulses in them. If they don't, this is just X.

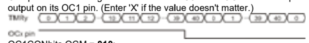


Complete the following statements for a PIC32MX's OC1 module to produce the given output on its OC1 pin. (Enter 'X' if the value doesn't matter.)



OC1CONbits.OCM = **001**;
PR2 = **78**;
OC1R = **1**;
OC1RS = **X**;

Complete the following statements for a PIC32MX's OC1 module to produce the given output on its OC1 pin. (Enter 'X' if the value doesn't matter.)

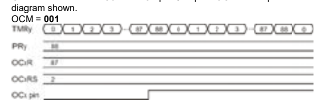


OC1CONbits.OCM = **010**;
PR2 = **40**;
OC1R = **39**;
OC1RS = **X**;

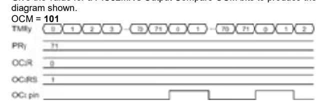
Complete the following statements for a PIC32MX's OC1 module to produce the given output on its OC1 pin. (Enter 'X' if the value doesn't matter.)

OC1CONbits.OCM = **101**;
PR2 = **62**;
OC1R = **52**;
OC1RS = **62**;

HW07-6(PIC32-OCx-WhatIsMode)
Give the value for a PIC32MX's Output Compare OCM bits to produce the timing diagram shown.
OCM = **001**



Give the value for a PIC32MX's Output Compare OCM bits to produce the timing diagram shown.
OCM = **101**



Give the value for a PIC32MX's Output Compare OCM bits to produce the timing diagram shown.
OCM = **011**

Give the value for a PIC32MX's Output Compare **OCM** bits to produce the timing diagram shown.
OCM = 100

HW07-6(PIC32-OCxCON)

This is the same process as the 7-5 ICxCON. Use pg. 164 for reference.
Give a single C instruction to configure a PIC32MX's Output Capture Module 4 with the following properties:

Module is ENabled
Continue module operation when in Idle Mode.
32-bit mode.
Timer 2 is clock source.
PWM mode on OCx. Fault pin DISabled.
OC4CON = 0x8026;

Give a single C instruction to configure a **PIC32MX's** Output Capture Module 2 with the following properties:

Module is DISabled.
Discontinue module operation when in Idle Mode.
32-bit mode.
Timer 2 is clock source.
Initialize OCx pin LOW. Event forces pin HIGH.
OC2CON = 0x2021;

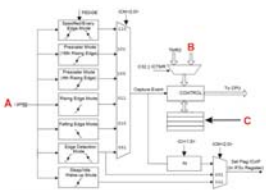
}
Answer = **222**
Example 5: (For incase you really need a lot of help like me.)
A PIC32MX with a bus clock frequency of 23.6 MHz has been set up with the following code for Timer 5:

```
void __ISR(_TIMER_5_VECTOR, ipl7) Timer_ISR;

T5CON = 0x00000060;
TMR5 = 0;
PR5 = 584;
T5CONSET = 0x000080000;
What does the value of C need to be in the ISR code below to get a square wave frequency as close to 14 Hz as possible?
void __ISR(_TIMER_5_VECTOR, ipl7) Timer_ISR;
{ static n=0;

if (++n >= C) {
LATAbits.LATA0 = ~LATAbits.LATA0;
n = 0;
}
IFS0bits.T5IF = 0;
}
```

Answer: 23.0

HW07-7(PIC32-Timer-CountForSquareWave-Period)
This is the same thing as CountForSquareWave-Frequency above.
Use the same formula, but note that they give you t_{SQR} directly this time.
$t_{SQR}=(C \times PR \times 2 \times n)/f_{BUS}$
A PIC32MX with a bus clock frequency of 23.9 MHz has been set up with the following code for Timer 4:

```
void __ISR(_TIMER_4_VECTOR, ipl7) Timer_ISR;


T4CON = 0x00000010; → (binary) 10000 → Prescale 1:2→ n=2 (TxCON table)
TMR4 = 0;
PR4 = 5;
T4CONSET = 0x000080000;
What does the value of C need to be in the ISR code below to get a square wave period as close to 1.5 ms as possible?
void __ISR(_TIMER_4_VECTOR, ipl7) Timer_ISR;
{ static n=0;
```



Which PIC32MX timers can have an asynchronous external clock? **Timer 1**
Which PIC32MX timers cannot be used for 32 bit timers? **Timer 1**
What is the lowest value possible in a PIC32MX's TMR1 register ? 0
What is the highest value possible in a PIC32MX's TMR5 register? **0xFFFF, FFFF, 65535**
What PIC32MX integrated peripheral counts clock pulses until an input signal changes state? **Input Capture**
What PIC32MX integrated peripheral works like a stopwatch? **Input Capture**

Chapter 8

HW08-1(WatchdoqTimers-General)
When is a watchdog a necessity in an embedded system?
A watchdog timer is a piece hardware that can be used to automatically detect software anomalies and reset the processor if any occur. Basically, A watchdog Timer is a counter that counts down to zero. However it's supposed to be reset every time the Program resets, if it isn't reset then it's assumed to be malfunctioning and the watchdog resets the program.

What happens if a watchdog chips timer reaches zero? **It restarts the Processor.**

What peripheral chip can be used to reset a microcontroller if it cannot get out of a loop structure?
Watchdog

Complete the following C instructions to produce a duty factor as close to 83.5 % as possible on a PIC32MX. 52 ÷ .835 - 1= 61.275

OC1R = 52;
OC1RS = 52;
PR2 = **61**;

Complete the following C instructions to produce a duty factor as close to 75.8 % as possible on a PIC32MX. 66 ÷ .758 - 1= 86.071

OC1R = 66;
OC1RS = 66;
PR2 = **86**;

HW09-2(PIC32-MaxPWMResolution-PR)
Equation: Max PWM Resolution Bits = $\log_2$(PRx)
This and the stuff below all (mostly) use this equation from the slides. I find it hard to understand/read quickly so I made the equation better.

Give the the maximum PWM resolution (in bits to two decimal places) of a PIC32MX PWM signal given a PBLCK of 19.5 MHz, and the following instructions.
T2CONbits.TCKPS = 0;
PR1 = 12;
Answer = **3.58** $\log_2((19.5e6 \times 12)/19.5e6)$

Give the the maximum PWM resolution (in bits to two decimal places) of a PIC32MX PWM signal given a PBLCK of 33.2 MHz, and the following instructions.
T2CONbits.TCKPS = 2;
PR1 = 386038;
Answer = **18.56**

Give the the maximum PWM resolution (in bits to two decimal places) of a PIC32MX PWM signal given a PBLCK of 37.7 MHz, and the following instructions.

T2CONbits.TCKPS = 1;
PR1 = 173775;
Answer = **17.41**

HW09-2(PIC32-MaxPWMResolution)
Equation: $\log_2((f_{PBCLK}/(f_{PWM} \times \text{prescale}))$? it works. Prescale comes from the binary number (Type B: TxCON chart)
On TI-89 you can do log base 2 by entering: $\log(f_{PBCLK}/(f_{PWM} \times \text{prescale}),2)$

HW07-7(Application-Shaft-MaxRPMs)
Consider a shaft which has 5 equally-spaced magnets placed around its circumference to trigger a hall-effect sensor connected to IC2 of a PIC32MX.

Assuming IC2 has been set up to capture every rising edge with a Timer 2 frequency of 26.4 kHz, what is the speed of the shaft (in RPMs) if the last value of TMR2 was 5991 and the new value of TMR2 is 20868, assuming no overflow has taken place?
dAnswer = **21.3**

Equation: $(t_{ticks} \times f_{TMR})/(n \times (\text{New-Old}))$
(60 * 26.4k) / (5*(20868 - 5991)
n = Amount of "ticks" per rotation. In this case it would be 5 b/c of the 5 magnets
t_{TMR} = Time you're measuring for in seconds. The question asks for the answer in RPM, so this 1 will be 1 minute, or 60 seconds.
New/Old = The values of the timer at the new and old measurement points

---

Consider a shaft which has 4 equally-spaced magnets placed around its circumference to trigger a hall-effect sensor connected to IC2 of a PIC32MX.

Assuming IC2 has been set up to capture every rising edge with a Timer 2 frequency of 4.5 MHz, what is the speed of the shaft (in RPMs) if the last value of TMR2 was 8396 and the new value of TMR2 is 49615, assuming no overflow has taken place?
Answer = **1638**

HW07-7(PIC32-Timer-CountForSquareWave-Frequency)
Get the prescale value using the 7-5 chart.
n = prescale value denominator: (e.g. 0x00000010 → (binary) 10000, which is prescale value 1:2, so n = 2)
t_{SQR}= 1 / f_{SQR} (Hz → s)
Equation: $f_{SQR}=(C \times PR \times 2 \times n)/f_{BUS}$
(f_{SQR} * f_{BUS}) / (PR x 2 x n) = C
Use that equation and solve for *C*, which is what question asks for.
*round up OR down

Example 1:
A PIC32MX with a bus clock frequency of 24.2 MHz has been set up with the following code for Timer 2:

```
void __ISR(_TIMER_2_VECTOR, ipl7) Timer_ISR;
T2CON = 0x00000030;
TMR2 = 0;
PR2 = 100;
T2CONSET = 0x000080000;
```

```
if (++n >= C)
{
LATAbits.LATA0 = ~LATAbits.LATA0;
n = 0;
}

IFS0bits.T4IF = 0;
}
```

$1.5 \times 10^{-3} = (C \times 5 \times 2 \times 2)/23.9 \times 10^{6}$. Solve for C. Why is n = 2? ***See addition in pink***
Answer = **1793**

HW07-7(PIC32-Timer-SquareWave-Frequency)
This time you're solving for f_{SQR}, which is from $f_{SQR}=1/t_{SQR}$, but you're still using the same equation: $f_{SQR}=(C \times PR \times 2 \times n)/f_{BUS}$
Example 1:
A PIC32MX with a bus clock frequency of 11.1 MHz has been set up with the following code for Timer 3:

```
void __ISR(_TIMER_3_VECTOR, ipl7) Timer_ISR;

T3CON = 0x00000040;
TMR3 = 0;
PR3 = 385;
T3CONSET = 0x000080000;
What is the frequency (in Hz to two decimal places) of the square wave generated on pin 0 of Port A generated with the ISR code?
void __ISR(_TIMER_3_VECTOR, ipl7) Timer_ISR;
{ static n=0;
if (++n >= 76)
{
LATAbits.LATA0 = ~LATAbits.LATA0;
n = 0;
}

IFS0bits.T3IF = 0;
}
```

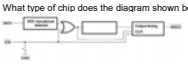
Answer = **11.85**

Example 2:
A PIC32MX with a bus clock frequency of 32.8 MHz has been set up with the following code for Timer 3:

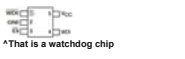
```
void __ISR(_TIMER_3_VECTOR, ipl7) Timer_ISR;
```

What peripheral chip can sometimes be used to reset a microcontroller if its supply voltage falls to an invalid value? **Watchdog**
What does a programmer have to do to prevent a Watchdog from timing out and resetting a processor?
It will have to be 'kicked' regularly, i.e. communicated with, once every given period.

What type of chip does the diagram shown below? **Watchdog Timer**



What is the purpose of the WDI pin? **watchdog input, likewise WDO is watchdog output**



^That is a watchdog chip

HW08-2(PIC32-InstructionsPerTimeout)
Equation: Instructions = $C_{SYSCLK} \times T_{TIMEOUT}$
Since all/most of these equations ask for the answer in the thousands, just use this:
Instructions = $C_{SYSCLK} \times T_{TIMEOUT} \times 10^{-3}$

How many typical instructions (in millions) could a PIC32MX with a SYSCLK of 2.8 MHz run before its watchdog times out given the following settings:
$(2.8*10^{6}Hz * 262.144s) = 734003200$ → Convert to millions (10^{6})→ 734.0032 (see slide 9 for table ~~edit: posted above)
WDTCONbits.WDTPS = 18;
734

How many typical instructions (in thousands) could a PIC32MX with a SYSCLK of 36.1 MHz run before its Watchdog times out given the following settings:
WDTCONbits.WDTPS = 2;
Answer = **144.0**

How many typical instructions (in thousands) could a PIC32MX with a SYSCLK of 33.9 MHz run before its Watchdog times out given the following settings:
WDTCONbits.WDTPS = 2;
Answer = **136.0**

Give the the maximum PWM resolution (in bits to two decimal places) of a PIC32MX PWM signal given a PWM frequency of 5.2 kHz, a PBLCK of 18.6 MHz, and the following instruction.
T2CONbits.TCKPS = 2;
Answer = **9.8**

Give the the maximum PWM resolution (in bits to two decimal places) of a PIC32MX PWM signal given a PWM frequency of 784 kHz, a PBLCK of 35.5 MHz, and the following instruction.
T2CONbits.TCKPS = 1;
Answer = **4.50**

Give the the maximum PWM resolution (in bits to two decimal places) of a PIC32MX PWM signal given a PWM frequency of 2.3 Hz, a PBLCK of 23.4 MHz, and the following instruction.
T2CONbits.TCKPS = 5;
Answer = **18.28**

HW09-2(PIC32-PWM-RMS)
Formula: $\sqrt{(OC1R)(PR1 + 1)} \times V_{PWM}$
Give the RMS voltage (to three decimal places) of a 4.5-Volt PWM signal on a PIC32MX with a PBLCLK of 22.9 MHz given the following settings:
PR1 = 66;
OC1R = 26;
OC1RS = 26;
T2CONbits.TCKPS = 7;
Answer = **2.803** V $\sqrt{(26/(66 + 1))} \times 4.5$

Give the RMS voltage (to three decimal places) of a 4.6-Volt PWM signal on a PIC32MX with a PBLCLK of 32.2 MHz given the following settings:
PR1 = 16;
OC1R = 7;
OC1RS = 7;
T2CONbits.TCKPS = 6;
Answer = **2.952** V

HW09-2(PIC32-PWMFrequency)
This requires 2 different pages of the datasheet.
A = PBDIV divider
B = TCKPS determined prescale value
Equation: $T_{PWM} = (PR + 1) \times (A / f_{SYSCLK}) \times (B)$
These questions are asking for the *frequency*, so you need to convert the output that is in time to that by doing $1/T_{PWM}$

What does the value of C need to be in the ISR code below to get a square wave frequency as close to 775 Hz as possible?

```
void __ISR(_TIMER_2_VECTOR, ipl7) Timer_ISR;
{ static n=0;

if (++n >= C) //the ++n means the value of n after it has been incremented
{
LATAbits.LATA0 = ~LATAbits.LATA0;
n = 0;
}

IFS0bits.T2IF = 0;
}
```

Answer = **20**
1/775 = (C*100*2*8) / (24.2M) , C = 19.512

Example 2:
A PIC32MX with a bus clock frequency of 30.5 MHz has been set up with the following code for Timer 4:

```
void __ISR(_TIMER_4_VECTOR, ipl7) Timer_ISR;


T4CON = 0x00000060;
TMR4 = 0;
PR4 = 39;
T4CONSET = 0x000080000;
What does the value of C need to be in the ISR code below to get a square wave frequency as close to 6 Hz as possible?
void __ISR(_TIMER_4_VECTOR, ipl7) Timer_ISR;
{ static n=0;

if (++n >= C)
{
LATAbits.LATA0 = ~LATAbits.LATA0;
n = 0;
}

IFS0bits.T4IF = 0;
}
```

Answer = **1018.0**

Example 3:

```
T3CON = 0x00000040;
TMR3 = 0;
PR3 = 7566;
T3CONSET = 0x000080000;
What is the frequency (in Hz to two decimal places) of the square wave generated on pin 0 of Port A generated with the ISR code?
void __ISR(_TIMER_3_VECTOR, ipl7) Timer_ISR;
{ static n=0;
if (++n >= 46)
{
LATAbits.LATA0 = ~LATAbits.LATA0;
n = 0;
}

IFS0bits.T3IF = 0;
}
```

Answer = **2.95**

HW07-7(PIC32-Timer-SquareWave-Period)
Same equation as the previous problems, but solving for period instead. This means you just need to find t_{SQR} without modifying (inverting) it.
$t_{SQR}=(C \times PR \times 2 \times n)/f_{BUS}$
A PIC32MX with a bus clock frequency of **25.3 MHz** has been set up with the following code for Timer 5:

```
void __ISR(_TIMER_5_VECTOR, ipl7) Timer_ISR;


T5CON = 0x00000050;
TMR5 = 0;
PR5 = 70;
T5CONSET = 0x000080000;
What is the period (in **micro seconds**) of the square wave generated on pin 0 of Port A generated with the ISR code?
void __ISR(_TIMER_5_VECTOR, ipl7) Timer_ISR;
{ static n=0;

if (++n >= 49)
{
LATAbits.LATA0 = ~LATAbits.LATA0;
n = 0;
```

HW08-2(PIC32-WatchdoqTimer-TimeoutPeriod)
LOOK AT THE CHART
Give the Time-out Period (in milliseconds) of a PIC32MX Watchdog Timer given the following instruction:

WDTCONbits.WDTPS = 3;
Answer = **8 ms**

Give the Time-out Period (in milliseconds) of a PIC32MX Watchdog Timer given the following instruction:
WDTCONbits.WDTPS = 21;
Answer = **1048576 ms**
The *Note 1* comment at the bottom of the chart is applied to this question

Give the Time-out Period (in milliseconds) of a PIC32MX Watchdog Timer given the following instruction:
WDTCONbits.WDTPS = 15;
32768 (slide 9 of 08-2…. Pasted above)
Give the Time-out Period (in milliseconds) of a PIC32MX Watchdog Timer given the following instruction:
WDTCONbits.WDTPS = 30;
1045876

HW08-2(PIC32-WatchdoqTimer-WDTPS)

Give a single C instruction to set the Time-out Period of a PIC32MX Watchdog Timer to as close to 1 min as possible.
WDTConbits.WDTPS = 16;

Give a single C instruction to set the Time-out Period of a PIC32MX Watchdog Timer to as close to 10 min as possible.
WDTConbits.WDTPS = 19;

Give a single C instruction to set the Time-out Period of a PIC32MX Watchdog Timer to as close to 1.2 ms as possible.
WDTConbits.WDTPS = 0;

Chapter 9

HW09-2(PIC32-DutyFactor-OCR)
Equation: OC1R = OC1RS = (PR2+1) x %
Complete the following C instructions to produce a duty factor as close to 59.9 % as possible on a PIC32MX. (84+1)*0.599=50.915

Give the frequency (in kHz to two decimal places) of a PIC32MX PWM waveform created by Timer 2 given the following C instructions, given that the SYSCLK is 3.7 MHz.
OSCCONbits.PBDIV = 1; → This would line up with "PBCLK is SYSCLK divided by 2" where 2 is A
T2CONbits.TCKPS = 0; → This means the prescale value is 1:1, so B = 1
PR2 = 955;
Answer = **1.94** (955+1) x (2/(3.7x10^{6})) x 1 → 5.167x10^{-4} → 1/5.167x10^{-4} → 1935.1 Hz → 1.94 kHz

Give the frequency (in Hz) of a PIC32MX PWM waveform created by Timer 2 given the following C instructions, given that the SYSCLK is 33.4 MHz.
OSCCONbits.PBDIV = 1;
T2CONbits.TCKPS = 2;
PR2 = 864;
Answer = **4827.0**

Give the freuency (in Hz to one decimal place) of a PIC32MX PWM waveform created by Timer 2 given the following C instructions, given that the SYSCLK is 17.3 MHz.
OSCCONbits.PBDIV = 0;
T2CONbits.TCKPS = 1;
PR2 = 27370;
Answer = **316.0**

HW09-2(PIC32-PWMPeriod)
Use the same equation used in the problem above, but don't convert it back to frequency. The equation, by default, gives the answer as a period (time), T_{PWM}.
Give the period (in milliseconds to three decimal places) of a PIC32MX PWM waveform created by Timer 2 given the following C instructions, given that the SYSCLK is 12.7 MHz.
OSCCONbits.PBDIV = 0;
T2CONbits.TCKPS = 1;
PR2 = 3523;
Answer = **0.555** ms

Give the period (in milliseconds to three decimal places) of a PIC32MX PWM waveform created by Timer 2 given the following C instructions, given that the SYSCLK is 27.2 MHz.
OSCCONbits.PBDIV = 2;
T2CONbits.TCKPS = 4;
PR2 = 566;
Answer = **1.334** ms

A PIC32MX with a bus clock frequency of 28.7 MHz has been set up with the following code for Timer 4:

```
void __ISR(_TIMER_4_VECTOR, ipl7) Timer_ISR;
T4CON = 0x00000050;
TMR4 = 0;
PR4 = 2;
T4CONSET = 0x000080000;
What does the value of C need to be in the ISR code below to get a square wave frequency as close to 11 Hz as possible?
void __ISR(_TIMER_4_VECTOR, ipl7) Timer_ISR;
{ static n=0;

if (++n >= C)
{
LATAbits.LATA0 = ~LATAbits.LATA0;
n = 0;
}

IFS0bits.T4IF = 0;
}
```

Answer = **20384.0**

Example 4:
A PIC32MX with a bus clock frequency of 35.1 MHz has been set up with the following code for Timer 5:

```
void __ISR(_TIMER_5_VECTOR, ipl7) Timer_ISR;
T5CON = 0x00000010;
TMR5 = 0;
PR5 = 4;
T5CONSET = 0x000080000;
What does the value of C need to be in the ISR code below to get a square wave frequency as close to 9.9 kHz as possible?
void __ISR(_TIMER_5_VECTOR, ipl7) Timer_ISR;
{ static n=0;

if (++n >= C)
{
LATAbits.LATA0 = ~LATAbits.LATA0;
n = 0;
}

IFS0bits.T5IF = 0;

}

IFS0bits.T5IF = 0;
}
```

Answer = **8677**

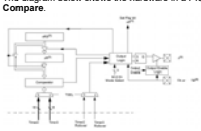
HW07(PIC32-Timers-General)
What signal values/transitions can a PIC32MX's output compare module produce at a given pin?
Toggle, Low Value, High Value, Low-High-Low Pulse, and Continuous Pulses

What PIC32MX integrated peripheral works like a "Alarm Clock?"
Output Compare

What do you call a timer which allows an external signal to control when a timer counts and when it doesn't?
Gated Timer

What is register in a PIC32MX contains the timer count values? **TMR**
Which register in a PIC32MX contains the value a timer counts up to? **PR or PRx**

The diagram below shows the hardware in a PIC32MX used to perform a\[n] **Output Compare**.



For the PIC32MX module diagram below, to what does Label B refer?
Answer = **TMR3**
C is the FIFO Buffer
A is the ICx
Found on pg.1 of 7-5 notes

OC1R = **51**;
OC1RS = **51**;
PR2 = 84;

Complete the following C instructions to produce a duty factor as close to 26.0% as possible on a PIC32MX. (86+1)*0.260=22.62
OC1R = **23**;
OC1RS = **23**;
PR2 = 86;

Complete the following C instructions to produce a duty factor as close to 78.4% as possible on a PIC32MX. (50+1)*0.784 = 39.984
OC1R = **40**;
OC1RS = **40**;
PR2 = 50;

HW09-2(PIC32-DutyFactor-PR)
(OC1R+DF)-1 = PR ~~***Round the result***~~
Complete the following C instructions to produce a duty factor as close to 16.4 % as possible on a PIC32MX. 63+0.164=384.146 → 383

OC1R = 63;
OC1RS = 63;
PR2 = **383**;

Complete the following C instructions to produce a duty factor as close to 1.0 % as possible on a PIC32MX. 43+0.01-1 = 4299
OC1R = 43;
OC1RS = 43;
PR2 = **4299**;

Complete the following C instructions to produce a duty factor as close to 70.3 % as possible on a PIC32MX. 44+0.703-1=61.589
OC1R = 44;
OC1RS = 44;
PR2 = **62**;

Complete the following C instructions to produce a duty factor as close to 13.1 % as possible on a PIC32MX. 47 + .131 - 1= 357.78

OC1R = 47;
OC1RS = 47;
PR2 = **358**;

HW09-2(PIC32-PWMs-General)

What does PWM stand for?
Pulse Width Modulation
What property/characteristic of a microcontroller allows a PWM to be useful?
PWM allows a microcontroller to interface with many analog components. The microcontroller, being a digital device, only outputs a binary result, so PWM allows some components to be "tricked" by sending pulses and a specific frequency.
Give an example of a practical use of a PWM waveform.
Controlling a motor's speed
How do you get a 100% duty factor using the PWM of a PIC32MX?
By having OCxR greater than PRy
What does "Modulation" mean in PWM?
A technique used in communication systems for coding the height of the signal. So it controls the amplitude of the signals.
What is the maximum number of duty factors that a 16-bit Output Compare module on a PIC32MX can produce?
65536
What is the maximum number of PWM that can be running at any given time on a PIC32MX?
5
What is the maximum number of different PWM periods that a 32-bit Output Compare module can produce on a PIC32MX?
4294967296
How do you get a 100% duty factor using the PWM of a PIC32MX?
When OCxR is loaded with a 0x0000, the OCx pin will remain low giving a 0% duty cycle.
What is the maxium number of duty factors that a 32-bit Output Compare module on a PIC32MX can produce?
2^32
What is the purpose/use of PWM "fault protection?"
To prevent a steady high from being left on the PWM control line in case of a reset/power up, etc.
What is the maximum number of PWM that can be running at any given time on a PIC32MX?
5
What is the maximum number of duty factors that a 16-bit Output Compare module on a PIC32MX can produce?
2^16 = 65536

ECE 3710 TEST 3

Does anyone know the answer to these questions?
What is the minimum sampling time in the sample-convert sequence when using a PIC32MX's ADC?

Because acquisition has a minimum time of 132ns and the clock period of the ADC is at least 65ns while conversion takes 12 TAD, the minimum sample time = (65ns * 12 TAD) + 132ns = 912ns.
Thank you. That's what I thought, but the notes were not very clear.

Are we sure this is right?

Is there a minimum conversion time in the sample-convert sequence when using a PIC32MX's ADC? If so, what is it?

Found this in the ADC datasheet:

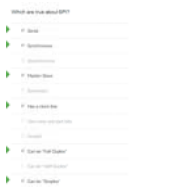
Param No.	Symbol	Characteristics	Min.	Typical(1)	Max.	Units	Conditions
Clock Parameters							
AD50	TAD	ADC Clock Period(3)	65	—	—	ns	See Table 29-33
Conversion Rate							
AD55	TCONV	Conversion Time	—	12 TAD	—	—	—
AD56	FCNV	Throughput Rate (Sampling Speed)	—	—	1000	ksps	AVDD = 3.3V to 3.6V
			—	—	400	ksps	AVDD = 2.5V to 3.6V
AD57	TSAMP	Sample Time	1 TAD	—	—	—	TSAMP must be ≥ 132 ns

So technically there isn't a minimum conversion time, but the typical conversion time is 12 TAD.

Chapters 10-20 (no 14 - 18)
Homework Equations and Explanations
Explanations and equations are in purple.
*when typing your answers please don't type word for word
***All free response questions are unconfirmed, take them with a bag of rice
If you have a question with a picture, you can save the picture and the default file name contains the answer usually (For HW only)

Chapter 10

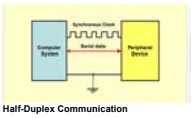
HW10-1(SynchronousSerialComm-General)
What type of Serial Communication is shown below?



Give the name of all SPI signal lines / SPI has the following signals.

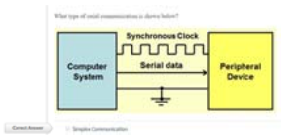
MOSI, MISO, SS#, SCK

HW10-3(SPI-PIC32-Clock)
Helpful Chart



Half-Duplex Communication

What type of serial Communication has two serial data lines each allowing communication in one direction?
Full Duplex Communication



What makes Synchronous Serial Communication "Synchronous"?
There is a protocol where each device communicating uses a common clock to synchronize data transmission.

What is Half-Duplex communication?
A system through which each party can communicate with each other, but not at the same time.
Example: Walkie-Talkie

What is Duplex communication?
Duplex, same as full duplex, is a form of communication through which both parties can send and receive data at the same time.

NOTE:
- Full-Duplex allows for communication in either direction at any time (2 lines). An example is HTTP Pipelining.
- Half-Duplex allows for communication in one direction at one time then can switch to the opposite direction (one line two arrows. An example is the POP3 Protocol (as well as FTP, NNTP, SMTP).
- Simplex allows for communication in one direction only and doesn't allow switching directions. This is sometimes specified programmatically as Simplex Server or Simplex Client.

Which of the following things must a sender and receiver agree upon before they can communicate in a synchronous serial fashion?

The speed of the clock, the order of the bits (MSb or LSb first), Number of bits per symbol, Simplex or Duplex communication.

Give an example of synchronous serial communication protocols mentioned in the notes.

What type of serial communication has a single serial data lines allowing communication in both directions? **Half-Duplex Communication**

HW10-2(SPI-General)
What does MISO stand for?
Master In-Slave Out

What function does the 74HC595 chip perform?
Serial-to-Parallel Conversion

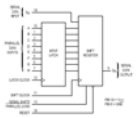
What does SPI Stand for?
Serial Peripheral Interface

SPI has the following signals.
MOSI, MISO, SS#, SCK

Identify the 7400-series chip shown.

74HC595

Identify the 7400-series chip shown.



74HC597

What does the SPI **SS#** line do?
The slave select line is normally low active and once activated lets the slave know it is about to be sent data. Normally each slave has a different slave select line.

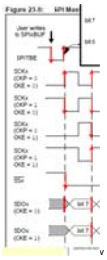
SPI is a _____ protocol.
Master-slave synchronous serial

What's the purpose of a **74HC597** chip when using a **SPI** Port?
This chip is a shift register, so it can be configured to take a serial input from the SPI port and convert it to a parallel output.
Maybe I'm dumb but on his scripts it has the following (which I think is different):
2 The 74H595 IC is a serial-to-parallel chip which takes in a byte of serial data and then drives its 8 outputs with the data read in.
3 The 74H597 does the opposite, taking in a parallel data byte, and then transmitting that byte serially at its output.

Give the name of all SPI signal lines / SPI has the following signals.

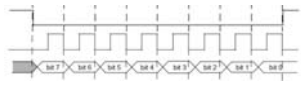
MOSI, MISO, SS#, SCK

HW10-3(SPI-PIC32-Clock)
Helpful Chart



Which graph represents a PIC32MX SPI timing with where **CKP = 1** and **CKE = 0**?

Which graph represents PIC32MX SPI timing with where **CKP=0** and **CKE=1?**



HW10-3(SPI-PIC32-SPIxCON) - **the table from the manual works too (pg 166--168)**

A00B 0CCC D000 00EF G0HI JJ0K LM0N OOPP
A. FRMEN Frame SPI support (1=enabled)
B. MSSEN: Master Mode Slave Select Enable (1=slave select SPI enabled)
C. FRMCNT: Frame Sync Pulse Counter (0=1 chars, 5 [max]=32 chars, x=2^n chars)
D. MCLKSEL: Master Clock Enable (1=BRG uses REFCLK,0=BRG uses PBCLK)
E. SPIFE: Frame Sync Pulse Edge Select (1=FSP coincides with first bit clock, 0=precedes)
F. ENHBUF: Enhanced Buffer Enable (1=enabled)
G. ON: SPI Peripheral on (1=enabled)
H. SIDL: Stop In Idle Mode (1=discontinue in idle mode)
I. DISSDO: Disable SDOx pin (1=SDOx pin not used by module-controlled by PORT)
J. MODE: if auden=0->0=8 bit mode, 1=16 bit mode, 2/3=32 bit mode
K. CKE: SPI Clock Edge Select (1=serial output data changes when from active to idle)
L. SSEN: Slave Select Enable (1=SSx pin used for slave mode)
M. CKP: Clock Polarity Select (1=idle state for clock is high & active is low)
N. DISSDI: Disable SDI (1=SDI not used by SPI-used by PORT)
O. STXISEL: SPI Transmit Buffer Empty INT(3=not full,2=½ empty,1=empty,0=when done)
P. SRXISEL: SPI Receive Buffer Full INT (3=full,2=½ full,1=not empty,0=last word read

Give a Single C instruction to configure a **PIC32MX's** SPI Module 2 with the following properties:
-Slave Select Support Enabled
-Enhanced buffer mode enabled
-16-bit Mode
-Receive is generated when the buffer is NOT empty.
Answer: SPI2CON = 0x10010401;

Give a Single C instruction to configure a **PIC32MX's** SPI Module 2 with the following properties:
-Frame Support Enabled
-REFCLK is used for Baud Rate
-Frame Synchronization pulse coincides with first by clock
-Discontinue Module operation when the device enters Idle Mode
-Receive is generated when the buffer is NOT EMPTY
Answer: SPI2CON = 0x80822001;

Give a Single C instruction to configure a **PIC32MX's** SPI Module 2 with the following properties:
-Slave Select Support Enabled
-Frame Synchronization pulse Coincides with first Bit clock.
-16-bit Mode
-Serial output data changes on transition from active clock state to idle clock state
-Receive is generated when the buffer is NOT EMPTY
Answer: SPI2CON = 0x10020501;

Give a single C instruction to configure a **PIC32MX's** SPI Module 2 with the following properties:
- Slave select support ENabled.
- Frame synchronization pulse coincides with first bit clock.
- Discontinue module operation when the device enters Idle Mode.
- Idle state for clock is a HIGH level.
- Receive is generated when the buffer is NOT EMPTY.
(Assume all parameters not mentioned are 0's).
SPI2CON = 0x10022041;

Give a single C instruction to configure a PIC32MX's SPI Module 1 with the following properties:
- Frame support ENabled.
- Generate a frame sync pulse on every 16 data characters.
- REFCLK is used for Baud Rate.
- SDOx pin is NOT controlled by module.
- Receive is generated when the buffer is at least HALF FULL.
Answer: SPI1CON = 0x84801002;

Give a single C instruction to configure a PIC32MX's SPI Module 1 with the following properties:

- Slave select support ENabled.
- Frame synchronization pulse coincides with first bit clock.
- Discontinue module operation when the device enters Idle Mode.
- Serial output data changes on transition from active clock state to idle clock state.
- Receive is generated when the buffer is NOT EMPTY.
Answer: SPI1CON = 0x10022101;

HW10-3(SPI-PIC32-SPIxCON2)

NOTE: The data sheet may have a typo b/c it doesn't show bit 4, just enter 0 for bit 4.
Give a single C instruction to configure a PIC32MX's SPI Module 2 with the following properties:

- Data from RX FIFO is sign extended.
- Audio Protocol ENabled.
- Audio Data is Mono.
- PCM/DSP Mode
SPI2CON2 = 0x808B;

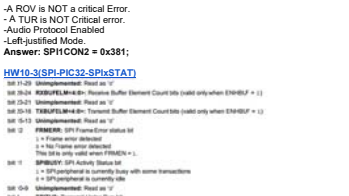
Give a single C instruction to configure a PIC32MX's SPI Module 2 with the following properties:

- A ROV is NOT a critical error.
- Audio Protocol ENabled.
- Audio Data is Mono.
- PCM/DSP Mode
SPI2CON2 = 0x28B;

Give a single C instruction to configure a **PIC32MX's** SPI Module 1 with the following properties:

-A ROV is NOT a critical Error.
- A TUR is NOT Critical error.
-Audio Protocol Enabled
-Left-justified Mode.
Answer: SPI1CON2 = 0x381;

HW10-3(SPI-PIC32-SPIxSTAT)



Given the value of a PIC32MX's **SPIxSTAT** register shown below,
SPI1STAT = 0x4000923;
is the following statement true or false?
SPI peripheral is busy?
TRUE
look at bit 11

Given the value of a PIC32MX's **SPIxSTAT** register shown below,
SPI2STAT = 0x31B11A9;
is the following statement true or false?

Transmit Buffer Element Count Bits are NOT equal to 27.
FALSE

```
SPI2STAT = 0x81318C1;
```

is the following statement true or false?
Transmit Buffer IS empty. **FALSE**

HW10-3(SPI-PIC32-WhatIsBaudRate)
Baud Rate = (f_{PBCLK})/(2*(SPIxBRG+1)) (note the units)

A PIC32MX has a PBCLK frequency of 32.8 Mhz. What is the SPI1 Baud Rate (in kb/s to one decimal place) given the following register settings?
SPIxBRG = 473;
Answer = 34.6

--
A PIC32MX has a PBCLK frequency of 30.3 MHz. What is the SPI1 Baud Rate (in kb/s to one decimal place) given the following register settings?

SPI1BRG = 465;
Answer = 32.5

A PIC32MX has a PBCLK frequency of 33.5 MHz. What is the SPI1 Baud Rate (in kb/s to one decimal place) given the following register settings?

SPI1BRG = 0x071;
Answer = 146.9

HW10-3(SPI-PIC32-WhatIsSPIxBR)
Given a PIC32MX PBCLK value of **29.5 MHz**, give the value of **SPIxBRG** to produce a Baud Rate as close to **216.0 kb/s** as possible.
baud rate = ($f_{frequencyPBCLK}$)/(2* (SPIxBRG +1))
SPIxBRG = ($f_{frequencyPBCLK}$)/(2*baud rate) - 1
216.0*10^3 b/s = (29.5*10^6 Hz)/(2*(SPIxBRG +1))
SPIxBRG = 67

Given a PIC32MX PBCLK value of 14.2 MHz, give the value of SPIxBRG to produce a Baud Rate as close to 31.6 kb/s as possible.
SPIxBRG = **224**

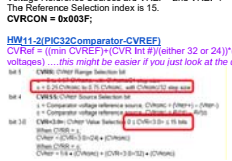
Chapter 11

HW11-2(PIC32-CVRCON)

Give a single C instruction to configure the PIC32MX comparator Modules with the following Properties:
Module is Enabled
Voltage level is output to pin.
Voltage Range is 0.25 CVRSRC to 0.75 CVRSRC.
Voltage Reference Sources are VREF+ and VREF-.
The Reference Selection index is 12. This is the last four bits of the binary value
CVRCON = 0x805C;
Give a single C instruction to configure the PIC32MX comparator Modules with the following Properties:
Module is Enabled
Voltage level is disconnected from output pin
Voltage Range is 0.25 CVRSRC to 0.75 CVRSRC.

Voltage Reference Sources are VREF+ and VREF-.
The reference selection index is 11.
CVRCON = 0x801B;
Give a single C instruction to configure the PIC32MX comparator Modules with the following Properties:
Module is Disabled.
Voltage level is disconnected from output pin.
Voltage Range is 0 to .67 CVRSRC.
Voltage Reference Sources are VREF+ and VREF-.
The Reference Selection index is 15.
CVRCON = 0x003F;

HW11-2(PIC32Comparator-CVREF)
CVRef = {(min CVREF)+(CVR Int #)/(either 32 or 24)*(Difference between CVRR voltages) ….*this might be easier if you just look at the datasheet?*



Given the following C instructions, what is the value of CVRef for the PIC32 comparator (to three decimal places) given that VREF+ = 3.13, VREF- = 0.58, and AVDD = 2.85, AVSS = 1.35?
CVRCONbits.CVRR = 1;
CVRCONbits.CVRSS = 1;
CVRCONbits.CVR = 2;
Answer = 0.212

Given the following C instructions, what is the value of CVRef for the PIC32 comparator (to three decimal places) given that VREF+ = 2.75, VREF- = 0.31, and AVDD = 2.43, AVSS = 0.99?

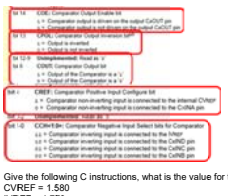
CVRCONbits.CVRR = 1;
CVRCONbits.CVRSS = 0;
CVRCONbits.CVR = 14;
Answer = 0.84
14/24*(2.43-.99) = 0.84

Given the following C instructions, what is the value of CVRef for the PIC32 comparator (to three decimal places) given that VREF+ = 2.06, VREF- = 1.16, and AVDD = 2.19, AVSS = 0.35?

```
CVRCONbits.CVRR = 0;
CVRCONbits.CVRSS = 1;
CVRCONbits.CVR = 4;
```
Answer 0.338
(2.06-1.16)/4*(2.06-1.16)*4/32

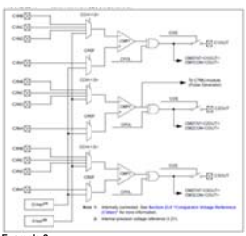
HW11-2(PIC32Comparator-Output)
Just look at the picture below
and reason through it.
if INV>NONInv, output = 0
The output is zero by default
and if there are not ones being
outputted it is a tri state (when COE = 0).

CREF voltage < CCH voltage => 0 if CPOL = 0, 1 if CPOL = 1
CREF voltage > CCH voltage => 1 if CPOL = 0, 0 if CPOL = 1
COE = 0 => HiZ



Give the following C instructions, what is the value for the PIC32 comparator given that
CVREF = 1.580
IVREF = 1.770
C1INA = 2.000
C1INB = 2.481
C1INC = 0.114
C1IND = 0.841
CM1CONbits.COE = **1**;
CM1CONbits.CPOL = 0;
CM1CONbits.CREF = 0;

CM1CONbits.CCH = **1**;



Example 2
Given the following C instructions, what is the value for the PIC32 comparator given that

CVREF = 2.690
IVREF = 1.792
C2INA = 1.770
C2INB = 2.543
C2INC = 0.961
C2IND = 0.182

CM2CONbits.COE = **1**;
CM2CONbits.CPOL = 1;
CM2CONbits.CREF = 0;
CM2CONbits.CCH = 2;
COUT = 0

Example 3
Given the following C instructions, what is the value for the PIC32 comparator given that

CVREF = 2.920
IVREF = 0.164
C2INA = 1.280
C2INB = 0.905
C2INC = 1.720
C2IND = 2.549

CM2CONbits.COE = 0;
CM2CONbits.CPOL = 1;
CM2CONbits.CREF = 1;
CM2CONbits.CCH = 2;
COUT = HiZ

Example 3
Given the following C instructions, what is the value for the PIC32 comparator given that

CVREF = 2.720
IVREF = 1.696
C1INA = 1.540
C1INB = 0.963F
C1INC = 0.126
C1IND = 2.641

CM1CONbits.COE = 1;
CM1CONbits.CPOL = 0;
CM1CONbits.CREF = 0;
CM1CONbits.CCH = 3;
COUT = 1 **Shouldn't this be 0?**

Chapter 12
HW12-1(DACs-Bits)
DACbits = log_2((High Voltage - Low Voltage) / (Increment Voltage)) (Always round up to the next highest integer. Never Round Down.

If we need to produce voltages from 0 to 10 Volts, in increments of 2 mV, what's the fewest number of bits that our DAC input must have?
DACbits = log((High Voltage - Low Voltage)/Increment Voltage)/log(2) (Round up)
log((10-0)/(2*10^-3))/log(2) = **13** Bits

HW12-1(DACs-General)

Every answer above can be found here:

https://www.maximintegrated.com/en/app-notes/index.mvp/id/641

What is a “relative accuracy error” as it pertains to a DAC?
It’s the Output error between the measured response and a line running from the output given an input of -0, and the output given an input of $2^N – 1$.
Give an Example of how a **DAC** can be used in an embedded system.
They could be used to generate an audio signal from 0’s and 1’s to an analog signal that could be sent to a handphone amp.
what is a “zero order hold”?
A zero order hold is used in the reconstruction of an analog signal for a DAC. Basically, Its used to describe the effect of converting a discrete time signal into a continuous time signal.
What does "unipolar" mean as it pertains to a DAC?
The binary input can only be a positive number.

What is a "full scale error" as it pertains to a DAC?
The difference between the actual value that triggers the transition to full-scale and the ideal analog full-scale transition value.

What is a the "zero error" of a DAC?
It indicates how well the actual transfer function matches the ideal transfer function at a single point.

What does “offset binary” mean as it pertains to a DAC?

Why are you not generally going to find a **DAC** on a microcontroller?"

HW12-1(DACs-Levels)
*** For the following, there are two equations based on whether the DAC is unipolar or bipolar
*** And pay attention to whether they want the answer in V or mV
If a 11-bit unipolar DAC has a reference voltage of 6.7 Volts, what is the smallest non-zero voltage (in mV) that can be produced?
3.271
Unipolar Equation: SmallestVoltage = RefVolt / (2^Bits)

If a 10-bit bipolar DAC has a reference voltage of 10.8 V, what is the smallest positive voltage (in mV to three decimal places) that can be produced?
21.094
Bipolar Equation: SmallestVoltage = RefVolt / (2^(Bits - 1))

If a 12-bit bipolar DAC has a reference voltage of 13.3 Volts, what is the **maximum** voltage (in Volts to four decimal places) that can be produced?
13.2935
Bipolar Equation: MaxVoltage = RefVolt * [2^(Bits-1) - 1] / 2^(Bits-1)

If a 10-bit unipolar DAC has a reference voltage of 12.8 Volts, what is the maximum voltage (in Volts to four decimal places) that can be produced?
12.7875
Unipolar Equation: MaxVoltage = RefVolt * [2^(Bits) - 1] / 2^(Bits)

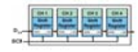
HW12-1(DACs-Values-Offset)
Given a 10-bit Offset-Binary bipolar DAC with a Vref of 10.0 V, what voltage would be output (to 3 decimal places) for an input code of 01 0010 0001?
SmallestVoltage = [(Vref / ($2^{(Bits - 1)}$)) * Binary]-Vref
((10/(2^(10-1)))*289)-10 = **-4.355** V

HW12-1(DACs-Values)
***Again, watch for whether the DAC is unipolar (use bits) or bipolar (use bits-1)
Given a 10-bit unipolar DAC with a Vref of 10.0V, what voltage would be output (to 3 decimal places) for an input code 0f 01 1000 0001?
3.760
Unipolar Equation: Voltage = [Vref / (2^bits)] * (binary code converted to decimal)
= (10 / 2^10) * (385)
= 3.760

Bipolar Equation: Voltage = (Vref / 2^(bits - 1)) * (binary code converted to decimal)

HW12-1(MultichannelDAC)
EXAMPLE SET 1
What voltage (to three decimal places) would come out of Channel 3 of the 6-bit DAC below given Vref = 3 V if the following bits are sent and the right-most bits are transmitted first.
0111 0001 0100 0101 0001 1010
Vout = (Vref*D_{in}) / 2^(#ofBits)
D_{in} = Decimal value of (# of Bits) in specified channel
Always start counting from the right. Pay attention to MSB (Most significant bit) and LSB (Least significant bit),
(3*20)/(2^6) = **0.938** V

What voltage (to three decimal places) would come out of **Channel 1** of the **7-bit** DAC below given Vref = **3 V** if the following bits are sent and the right-most bits are transmitted first.
1110 0101 0000 1000 0110 1001 0000



2.672
What voltage (to three decimal places) would come out of Channel 1 of the 5-bit DAC below given Vref = 3 V if the following bits are sent and the right-most bits are transmitted first.
0101 0011 1010 0100 0110 1010 0110

0.563

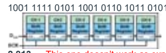
EXAMPLE SET 2
What voltage (to three decimal places) would come out of Channel 4 of the 5-bit DAC below given Vref = 3 V if the following bits are sent and the right-most bits are transmitted first.
0110 1110 1011 0000 1101

1.219

What voltage (to three decimal places) would come out of Channel 4 of the 7-bit DAC below given Vref = 5 V if the following bits are sent and the right-most bits are transmitted first.
1011 1100 0111 0110 0111 0010 1111 0001 1101

3.672

What voltage (to three decimal places) would come out of Channel 3 of the 5-bit DAC below given Vref = 5 V if the following bits are sent and the right-most bits are transmitted first.
1001 1111 0101 1001 0110 1011 0101 1001



2.813 → This one doesn’t work as expected

HW12-2(ADCBits)
ADCbits = log_2((High Voltage - Low Voltage) / Resolution Voltage) (Always round up to the next highest integer. ***Never NEVER NEVER*** Round Down.

If we need to sample voltages from 0 to 11.4 Volts, with a resolution of 62 mV, what's the fewest number of bits that our ADC output must have?
ADCbits = log((High Voltage -Low Voltage)/Resolution Voltage)/log(2) (Round up)
log((11.4-0)/(62*10^-3))/log(2) = **8**

HW12-2(ADCLevels)
*** For the following, there are two equations based on whether the DAC is unipolar or bipolar
*** And pay attention to whether they want the answer in V or mV
If a 10-bit unipolar ADC has a reference voltage of 12.5 Volts, what voltage (in mV) does an output value of 1 represent?
12.207
Unipolar Equation: Voltage = Vref / 2^bits

If a 13-bit bipolar ADC has a reference voltage of 5.3 Volts, what voltage (in V to three decimal places) does an output of 1 represent?
-5.299
Bipolar Equation: Voltage = (Vref / 2^(bits-1)) - Vref

If a 12-bit bipolar ADC has a reference voltage of 13.9 Volts, what voltage (in V to four decimal places) does an output code value of all 1's represent?
Bipolar Equation: Voltage = (Vref / 2^(bits-1)) - Vref
All 1’s -> make positive
13.8932

If a 10-bit unipolar ADC has a reference voltage of 4.4 Volts, what voltage (in V to four decimal places) does an output code value of all 1's represent?
Bipolar Equation worked here for some reason
4.3957

HW12-3(PIC32-ADC-WhatIsCode-Decimal-16bit)

*Leading 1: -512 (signed)
*Leading 0: +512 (signed)

If form=0, find decimal of last 10 bits

If form=1, 10th bit is sign (every bit before it should be the same) and last 9 bits are the decimal (signed int), remember 0x200=-512 and then you add the next 9 bits to -512 if 10th bit is 1

If form=2, first 10 bits are decimal and answer = decimal / (2^n) (n is always 10, so 1024)

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 1;
what decimal value does the following binary code sequence represent?
1111 1110 1101 1110
-290.0 -512+0b11011110

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 0;
what decimal value does the following binary code sequence represent?
0000 0001 0110 0001
353.0

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 2;
what decimal value does the following binary code sequence represent?
0110 0011 1000 0000
0.389

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 3;
what decimal value does the following binary code sequence represent?
1001 1000 1100 0000
-0.403
FORM = 3 means it’s a Signed Fractional 16-bit, so you disregard the last six 0’s
The first bit is the sign, so you start with -2^9 = -512
You then convert the next 9 binary numbers to decimal
001100011 = 99
-512 + 99 = -413
Answer = decimal / 2^n (n is always 10 so 1024)
-413 / 1024 = -0.403

HW12-3(PIC32-ADC-WhatIsCode-Decimal-32bit)
If form=5, bit 9 is sign and every bit before it should be the same, 0xFFFFFF000=-512
If form=6, first 10 bits are decimal, answer=number/(2^n), n=10
If form=7, 1st bit is sign, next 9 are decimal, answer=signed int/(2^n), n=10

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 5;
what decimal value does the following binary code sequence represent?
1111 1111 1111 1111 1111 1111 1111 0100
-12.0

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 7;
what decimal value does the following binary code sequence represent?
0111 1100 0000 0000 0000 0000 0000 0000
0.484

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 7;
what decimal value does the following binary code sequence represent?
1111 0110 1000 0000 0000 0000 0000 0000
-0.037 (-512+0b111011010)/1024

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 5;
what decimal value does the following binary code sequence represent?
0000 0000 0000 0000 0000 0001 0100 1001
329.0 0b111011010 → dec (because sign bit is 0)

HW12-3(PIC32-ADCVoltage-Fractional-16bit)

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 2;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 4.2 Volts.
AVSS = 0 Volts.
VREF+ = 2.6 Volts.
VREF- = 0 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
1001 1110 1000 0000
1.61 0b1001111010/1024*2.6

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 6;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 2.4 Volts.
AVSS = 0 Volts.
VREF+ = 2.1 Volts.
VREF- = 0 Volts.
What voltage (to three decimal places) value does the following binary sequence represent?
1011 1010 0000 0000 0000 0000 0000 0000
1.744 0b1011101000/1024*2.4 ...but this doesn’t make sense?...ya idk wtf

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 6;
AD1CON2bits.VCFG = 1;
and has the following Reference voltages:
AVDD = 3.2 Volts.
AVSS = 0 Volts.
VREF+ = 2.8 Volts.
VREF- = 0 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
1111 1010 1000 0000 0000 0000 0000 0000
2.74 0b1111101010/1024*2.8

HW12-3(PIC32-ADCVoltage-Fractional-32bit)
AD1CON1bits.FORM = 6;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 2.3 Volts.
AVSS = 0 Volts.
VREF+ = 3.6 Volts.
VREF- = 0 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
0100 0010 1000 0000 0000 0000 0000 0000
0.935 0b100001010/1024*3.6=0.935

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 6;
AD1CON2bits.VCFG = 0;
and has the following Reference voltages:
AVDD = 2.6 Volts.
AVSS = 0 Volts.
VREF+ = 2.5 Volts.
VREF- = 0 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
1010 1000 0000 0000 0000 0000 0000 0000
Answer = 1.706

HW12-3(PIC32-ADCVoltage-Fractional-Difference-16bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 2;
AD1CON2bits.VCFG = 0;
and has the following Reference voltages:
AVDD = 4.6 Volts.
AVSS = 0 Volts.
VREF+ = 4.8 Volts.
VREF- = 0.3 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
0111 0000 1000 0000
2.021

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 2;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 4 Volts.
AVSS = 0.9 Volts.
VREF+ = 4.6 Volts.
VREF- = 0.3 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
0101 0100 1100 0000
1.724 (0b101010011)/1024*(4.6-0.3)+0.3

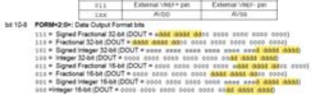
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 2;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 4 Volts.
AVSS = 1.3 Volts.
VREF+ = 3.9 Volts.
VREF- = 1.4 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
0001 0101 1000 0000
1.610

Note: Most of 12-3 makes use of this chart.

DIFFERENCE EQUATION SUMMARY	
Signed	Unsigned
((B±512)/(2^10))(V_{REFH}-V_{REFL})+(V_{REFL})	(B/(2^10))(V_{REFH}-V_{REFL})+(V_{REFL})

B = All *d* bits converted to binary. These are the bits referenced in the *FORM* command.
*Leading 1: -512 (signed)
*Leading 0: +512 (signed)
****With signed, include the sign bit in B**

	VREFH	VREFL
000	AVDD	AVSS
001	External VREF+ pin	AVSS
010	AVDD	External VREF- pin
011	External VREF+ pin	External VREF- pin
1xx	AVDD	AVSS



Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 2;
AD1CON2bits.VCFG = 1;
and has the following Reference voltages:
AVDD = 4.4 Volts.
AVSS = 0.7 Volts.
VREF+ = 4.4 Volts.
VREF- = 0.8 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
1111 0110 0100 0000
4.259 0b1111011001/1024*(4.4-.7)+.7

HW12-3(PIC32-ADCVoltage-Fractional-Difference-32bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 1;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 4.8 Volts.
AVSS = 0 Volts.
VREF+ = 3.1 Volts.
VREF- = 0 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1110 0001 0010
0.0545 0b10010*3.1/1024

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 6;
AD1CON2bits.VCFG = 2;
and has the following Reference voltages:
AVDD = 4.8 Volts.
AVSS = 0.1 Volts.
VREF+ = 3.6 Volts.
VREF- = 0.8 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
1100 1101 0100 0000 0000 0000 0000 0000
4.007
(0b1100110101 / 1024) * (4.8 - 0.8) - (-0.8)

HW12-3(PIC32-ADCVoltage-Signed-16bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 5;
AD1CON2bits.VCFG = 2;
and has the following Reference voltages:
AVDD = 3.5 Volts.
AVSS = 0 Volts.
VREF+ = 3.1 Volts.
VREF- = 0 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1111 1111 1111 1111 1111 1111 1011
1.7329

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 1;
AD1CON2bits.VCFG = 1;
and has the following Reference voltages:
AVDD = 2.1 Volts.
AVSS = 0 Volts.
VREF+ = 2.2 Volts.
VREF- = 0 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1111 0010 1011
0.6424
((0b1100101011 - 512) / 1024) * 2.2

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 1;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 3.3 Volts.
AVSS = 0 Volts.
VREF+ = 2.6 Volts.
VREF- = 0 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1110 1110 0101
0.5814

HW12-3(PIC32-ADCVoltage-SignedFractional-16bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 3;
AD1CON2bits.VCFG = 1;
and has the following Reference voltages:
AVDD = 4.8 Volts.
AVSS = 0 Volts.
VREF+ = 4.8 Volts.
VREF- = 0 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1001 0001 1100 0000
0.3328
((0b1001000111 - 512) / 1024) * 4.8

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 3;
AD1CON2bits.VCFG = 1;
and has the following Reference voltages:
AVDD = 3.8 Volts.
AVSS = 0 Volts.
VREF+ = 4.7 Volts.
VREF- = 0 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1011 1000 0000
2.2674

HW12-3(PIC32-ADCVoltage-SignedFractional-32bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 7;
AD1CON2bits.VCFG = 2;
and has the following Reference voltages:
AVDD = 5 Volts.
AVSS = 0 Volts.
VREF+ = 4.1 Volts.
VREF- = 0 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1011 1111 0000 0000 0000 0000 0000 0000
1.2305
((0b1011111100 - 512) / 1024) * 5

HW12-3(PIC32-ADCVoltage-Signed-32bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 5;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 4.6 Volts.
AVSS = 0 Volts.
VREF+ = 2.8 Volts.
VREF- = 0 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1111 1111 1111 1111 1110 1001 0101
0.4074
((0b1010010101 - 512) / 1024) * 2.8

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 5;
AD1CON2bits.VCFG = 2;
and has the following Reference voltages:
AVDD = 2.9 Volts.
AVSS = 0 Volts.
VREF+ = 3.7 Volts.
VREF- = 0 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1111 1111 1111 1111 1110 1101 1001
0.6146

HW12-3(PIC32-ADCVoltage-SignedDifference-16bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 1;
AD1CON2bits.VCFG = 1;
and has the following Reference voltages:
AVDD = 4.1 Volts.
AVSS = 1.2 Volts.
VREF+ = 3.7 Volts.
VREF- = 1.6 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1111 1101 1110
2.367
[((0b1111011110 - 512) / 1024) * (3.7 - 1.2)] + 1.2

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 1;
AD1CON2bits.VCFG = 0;
and has the following Reference voltages:
AVDD = 4.6 Volts.
AVSS = 1.2 Volts.
VREF+ = 2.8 Volts.
VREF- = 1.4 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1111 1111 0110
2.8668

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 1;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 4.3 Volts.
AVSS = 0 Volts.
VREF+ = 4.4 Volts.
VREF- = 0.3 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1110 0011 0000
0.4922

HW12-3(PIC32-ADCVoltage-SignedDifference-32bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 5;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 4.6 Volts.
AVSS = 1.5 Volts.
VREF+ = 5 Volts.
VREF- = 1.4 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1111 1111 1111 1111 1111 0100 1010
2.5602
[((0b1101001010 - 512) / 1024) * (5 - 1.4)] + 1.4

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 5;
AD1CON2bits.VCFG = 2;
and has the following Reference voltages:
AVDD = 3.7 Volts.
AVSS = 0.5 Volts.
VREF+ = 3.6 Volts.
VREF- = 0.7 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1111 1111 1111 1111 1111 0001 0000
1.4969

HW12-3(PIC32-ADCVoltage-SignedFractional-Difference-16bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 3;
AD1CON2bits.VCFG = 2;
and has the following Reference voltages:
AVDD = 3.8 Volts.
AVSS = 0.5 Volts.
VREF+ = 4.6 Volts.
VREF- = 1 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1111 1010 1100 0000
2.3426 0b111101011/1024*(3.8-1)+1

HW12-3(PIC32-ADCVoltage-SignedFractionalDifference-32bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 7;
AD1CON2bits.VCFG = 2;
and has the following Reference voltages:
AVDD = 4.5 Volts.
AVSS = 1 Volts.
VREF+ = 4.6 Volts.
VREF- = 1.4 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
1110 1111 1000 0000 0000 0000 0000 0000
2.7502
((0b110111110 / 1024) * (4.5 - 1.4)) + 1.4

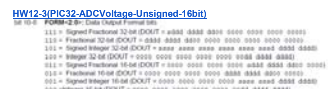
HW12-3(PIC32-ADCVoltage-SignedFractionalPositiveDifference-16bit)
(Binary value + 512)/1024*(V_{REFH}-V_{REFL})+V_{REFL}

HW12-3(PIC32-ADCVoltage-SignedFractionalPositiveDifference-32bit)
(Binary value + 512)/1024*(V_{REFH}-V_{REFL})+V_{REFL}

HW12-3(PIC32-ADCVoltage-SignedFractionalPositive-16bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 3;
AD1CON2bits.VCFG = 2;
and has the following Reference voltages:
AVDD = 3.2 Volts.
AVSS = 0 Volts.
VREF+ = 3.2 Volts.
VREF- = 0 Volts.
What voltage value does the following binary sequence represent?
0110 1001 1000 0000
2.9188
(Binary value + 512)/1024*(V_{REFH})

HW12-3(PIC32-ADCVoltage-SignedPositive-16bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 1;
AD1CON2bits.VCFG = 0;
and has the following Reference voltages:
AVDD = 3 Volts.
AVSS = 0 Volts.
VREF+ = 3.1 Volts.
VREF- = 0 Volts.
What voltage value does the following binary sequence represent (to four decimal places)?
0000 0001 1010 1011
2.751

HW12-3(PIC32-ADCVoltage-Unsigned-16bit)



Voltage = (binary converted to decimal / 1024) * Positive Voltage Reference

Consider a PIC32MX with the following ADC1 settings,

	VREFH	VREFL
000	AVDD	AVSS
001	External VREF+ pin	AVSS
010	AVDD	External VREF- pin
011	External VREF+ pin	External VREF- pin
1xx	AVDD	AVSS

AD1CON1bits.FORM = 0;
AD1CON2bits.VCFG = 1;
and has the following Reference voltages:
AVDD = 3.8 Volts.
AVSS = 0 Volts.
VREF+ = 3.5 Volts.
VREF- = 0 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
0000 0011 0111 1110
3.056
Since FORM = 0, convert the rightmost 10 bits to decimal.
1101111110 = 894
Since VCFG = 1, the positive reference voltage is VREF+.
(894/2^10) * 3.5 = 3.056

HW12-3(PIC32-ADCVoltage-Unsigned-32bit)
Same as HW 12-3(PIC32-ADCVoltage-Unsigned-16bit) above

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 4;
AD1CON2bits.VCFG = 0;
and has the following Reference voltages:
AVDD = 4.6 Volts.
AVSS = 0 Volts.
VREF+ = 3.9 Volts.
VREF- = 0 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
0000 0000 0000 0000 0000 0000 0010 1001
0.184

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 4;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 2.1 Volts.
AVSS = 0 Volts.
VREF+ = 2.5 Volts.
VREF- = 0 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
0000 0000 0000 0000 0000 0001 1011 0111 439/2^10*2.5
1.072

HW12-3(PIC32-ADCVoltage-UnsignedDifference-16bit)
Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 0;
AD1CON2bits.VCFG = 3;
and has the following Reference voltages:
AVDD = 4.3 Volts.
AVSS = 2.5 Volts.
VREF+ = 4.9 Volts.
VREF- = 2.6 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
0000 00nn11 0000 1001
4.345

Consider a PIC32MX with the following ADC1 settings,
AD1CON1bits.FORM = 0;
AD1CON2bits.VCFG = 1;
and has the following Reference voltages:

AVDD = 4.5 Volts.
AVSS = 2.6 Volts.
VREF+ = 4.3 Volts.
VREF- = 1.5 Volts.
What voltage value (to three decimal places) does the following binary sequence represent?
0000 0000 0100 0110
2.716

HW12-3(PIC32-ADCVoltage-UnsignedDifference-32bit)
One more...

HW12-3(PIC32-ADCxCHS)
Give a single C instruction to configure a **PIC32MX's** ADC Module with the following properties:
- Channel 0 negative input is VREFL for MUX B.
- Channel 0 positive input is AN10 for MUX B.
- Channel 0 negative input is AN1 for MUX A.
- Channel 0 positive input is AN1 for MUX A.
AD1CHS = 0x0A810000;

HW12-3(PIC32-ADCxCON1)
Give a single C instruction for a PIC32MX which will tell the Analog to Digital peripheral module to output the sample as a fractional 32-bit value without modifying any other parameters. (Leave a single space between all variables and/or operators and use no parentheses.
AD1CON1bits.FORM = 6;

Use the info from the data sheet found under HW 12-3 (PIC32-ADCxCON1Multi)

Give a single C instruction which will turn off the Analog to Digital peripheral module in a PIC32MX without modifying any other parameters. (Leave a single space between all variables and/or operators and use no parentheses.)
AD1CON1bits.ON = 0;

HW12-3(PIC32-ADCxCON1Multi)
Give a single C instruction to configure a PIC32MX's ADC Module with the following properties:
-ADC module IS operating.
-Continue Module operation when the device enters Idle Mode.
-16-bit Signed Integer Output.
-Internal counter ends sampling and starts conversion.
-Sampling begins when SAMP bit is set.
-When SSRC = 000, writing a '0' to SAMP will end sampling and start conversion.

AD1CON1 = 0x81E0;

HW12-3(PIC32-ADCxCON2)
Give a single C instruction for a **PIC32MX** which will tell the Analog to Digital peripheral module to use **AVDD**and **AVSS** as its voltage references without modifying any other parameters. (Leave a single space between all variables and/or operators and use no parentheses.)
AD1CON2bits.VCFG = 0;

Use the info from the data sheet found under HW 12-3 (PIC32-ADCxCON2Multi)

Give a single C instruction for a PIC32MX which will tell the Analog to Digital peripheral module to interrupt after every five sample sequences without modifying any other parameters. (Leave a single space between all variables and/or operators and use no parentheses.)
AD1CON2bits.SMPI = 4;

HW12-3(PIC32-ADCxCON2Multi)
Give a single C instruction to configure a **PIC32MX's** ADC Module with the following properties:

- VREFH uses AVDD, VREFL uses external VREF- pin.
- Disable Offset Calibration Mode.
- Scan the Inputs.
- Interrupt after completion of each 3rd conversion.
- Buffer configured as one 16-word buffer.
- Always use Sample A MUX settings for input.

AD1CON2 = 0x4408;

Use the following

HW12-3(PIC32-ADCxCON3)
Give a single C instruction for a **PIC32MX** which will have the Analog to Digital peripheral module use a sample period of **four TAD**s without modifying any other parameters. (Leave a single space between all variables and/or operators and use no parentheses.)
AD1CON3bits.SAMC = 4;

Use the info from the data sheet found under HW 12-3(PIC32-ADCxCON3Multi)

HW12-3(PIC32-ADCxCON3Multi)

Give a single C instruction to configure a PIC32MX's ADC Module with the following properties:

- Clock derived from FRC.
- Sample time = 0 TAD.
- TAD = 2 TPB.
AD1CON3 = 0x8000;

Give a single C instruction to configure a **PIC32MX's** ADC Module with the following properties:

- Clock derived from FRC.
- Sample time = 5 TAD.
- TAD = 10 TPB.

AD1CON3 = 0x8504;

Give a single C instruction to configure a PIC32MX's ADC Module with the following properties:

- Clock derived from FRC.
- Sample time = 1 TAD.
- TAD = 6 TPB.
AD1CON3 = 0x8102

HW12-3(PIC32-ADCxCSSL)
Give a single C instruction to select the following pins as input for MUX A of a **PIC32MX's** ADC Module:

AN2
AN3
AN12
VSS

AD1CSSL = 0x900C;

If VSS is listed, bit 15 = 1.
If IVREF is listed, bit 14 = 1.
If CTMU is listed, bit 13 = 1.
Set each corresponding bit to 1 that's listed.

For the above problem, bit 15 = 1, bit 12 = 1, bit 3 = 1, and bit 2 = 1. The rest are 0.
1001 0000 0000 1100 = 0x900C

Give a single C instruction to select the following pins as input for MUX A of a PIC32MX's ADC Module:

AN8
AN11
AN12
CTMU

AD1CSSL = 0x3900;

HW12-4(PIC32-ADC-ProduceASequence)

Example 1
Complete the C instructions below to produce the following sampling sequence using ADC1 of a PIC32MX.
AD1BUF0 = MUXA AN3
AD1BUF1 = MUXB AN4
AD1BUF2 = MUXA AN3
AD1BUF3 = MUXB AN4
AD1BUF4 = MUXA AN3
AD1BUF5 = MUXB AN4
< Interrupt Generated >
AD1BUF8 = MUXA AN3
AD1BUF9 = MUXB AN4
AD1BUF10 = MUXA AN3
AD1BUF11 = MUXB AN4
AD1BUF12 = MUXA AN3
AD1BUF13 = MUXB AN4
< Interrupt Generated >
etc...
AD1CON2bits.CSCNA = **0**; ***1 if the ANx values change within any MUXy, 0 otherwise.
AD1CON2bits.SMPI = **5**; Amount of buffers between interrupts minus 1.
AD1CON2bits.BUFM = **1**;
Explanation 1: BUFM = 0 when conversion results are written sequentially starting at ADC1BUF0 and goes until the number of samples defined by SMPI and starts over after the interrupt
AD1CON2bits.ALTS = **1**; If they alternate between MUXA and MUXB, this is =1, otherwise 0.
AD1CHSbits.CH0SB = **4**; If the MUXB values correspond to one AN value (like this example), this is just equal to the value of that ANx. If it doesn't correspond to just one, this is =X.
AD1CHSbits.CH0SA = **3**; Same thing as CH0SB, but for MUXA commands.
AD1CSSL = **X**; ***Use the CSSL reference and put a 1 for every ANx value used. 0s when you don't. Unless it's only 1 value, then it's an X

Example 2
Complete the C instructions below to produce the following sampling sequence using ADC1 of a PIC32MX .
AD1BUF0 = MUXA AN0
AD1BUF1 = MUXA AN5

AD1BUF2 = MUXA AN6
AD1BUF3 = MUXA AN7
AD1BUF4 = MUXA AN0
< Interrupt Generated >
AD1BUF0 = MUXA AN5
AD1BUF1 = MUXA AN6
AD1BUF2 = MUXA AN7
AD1BUF3 = MUXA AN0
AD1BUF4 = MUXA AN5
< Interrupt Generated >
etc...
AD1CON2bits.CSCNA = **1**;
AD1CON2bits.SMPI = **4**;
AD1CON2bits.BUFM = **0**;
AD1CON2bits.ALTS = **0**;
AD1CHSbits.CH0SB = **X**;
AD1CHSbits.CH0SA = **X**;
AD1CSSL = **0xE1**;

Example 3
Complete the C instructions below to produce the following sampling sequence using ADC1 of a PIC32MX .
AD1BUF0 = MUXA AN10
AD1BUF1 = MUXB AN2
AD1BUF2 = MUXA AN10
AD1BUF3 = MUXB AN2
AD1BUF4 = MUXA AN10
AD1BUF5 = MUXB AN2
AD1BUF6 = MUXA AN10
< Interrupt Generated >
AD1BUF8 = MUXB AN2
AD1BUF9 = MUXA AN10
AD1BUF10 = MUXB AN2
AD1BUF11 = MUXA AN10
AD1BUF12 = MUXB AN2
AD1BUF13 = MUXA AN10
AD1BUF14 = MUXB AN2
< Interrupt Generated >
Etc…
AD1CON2bits.CSCNA = **0**;
AD1CON2bits.SMPI = **6**;
AD1CON2bits.BUFM = **1**;
AD1CON2bits.ALTS = **1**;
AD1CHSbits.CH0SB = **2**;

AD1CHSbits.CH0SA = **10**;
AD1CSSL = **X**;

Example 4
Complete the C instructions below to produce the following sampling sequence using ADC1 of a PIC32MX .
AD1BUF0 = MUXA AN0
AD1BUF1 = MUXB IVREF
AD1BUF2 = MUXA AN2
< Interrupt Generated >
AD1BUF8 = MUXB IVREF
AD1BUF9 = MUXA AN4
AD1BUF10 = MUXB IVREF
< Interrupt Generated >
AD1BUF0 = MUXA AN5
AD1BUF1 = MUXB IVREF
AD1BUF2 = MUXA AN7
< Interrupt Generated >
AD1BUF8 = MUXB IVREF
AD1BUF9 = MUXA AN0
AD1BUF10 = MUXB IVREF
< Interrupt Generated >
etc...
AD1CON2bits.CSCNA = **1**;
AD1CON2bits.SMPI = **2**;
AD1CON2bits.BUFM = **1**;
AD1CON2bits.ALTS = **1**;
AD1CHSbits.CH0SB = **14**;
AD1CHSbits.CH0SA = **X**;
AD1CSSL = **0x00B5**;

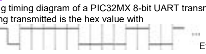
HW12-4(PIC32-ADC-WhatIsSequence)
Given the following register settings in a PIC32MX, give the sampling sequence that takes place showing what buffer(s) is/are filled by what input (including which MUX) and show when an interrupt would be generated.
AD1CON2 = 0x00000007
AD1CHS = 0x83880000
AD1CSSL = 0x000000CD

Chapter 13
HW13-2(RS-232-General)
What does Modem stand for? **Modulate-Demodulate**
What does CTS in RS-232 stand for? **Clear to Send**
What does DTR in RS-232 stand for? **Data Terminal Ready**

What does RTS in RS-232 stand for? **Request to Send**
What does DSR in RS-232 stand for? **Data Set Ready**
What was RS-232 and modems originally designed to communicate over? **Telephone Lines**
What does RI in RS-232 stant for? **Ring Indicator**
What does RxD in RS-232 stand for? **Receive Data**
What does DTE in RS-232 stand for? **Data Terminal Equipment**
What does DCD in RS-232 stand for? **Data Carrier Detect**
What's the minimum number of wires necessary to communicate with the **RS-232** protocol? **3**

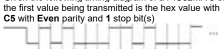
HW13-3(UART-General)
What does Baud Rate mean? **Baud rate is the "symbols per second"**
How many Start Bits does a UART have? **One**
How many Stop Bits does a UART have? **Variable/programmable**
What is the purpose of a UART? **It transmits and receives serial data using a shift register**
What is a 9th UART bit usually used for? **The 9th bit determines whether or not the remaining**
8 bits transmitted contains a device address, or data for the selected device.
What does UART stand for? **Universal Asynchronous Receiver Transmitter**
What type of communication does UART perform? **Asynchronous Serial**
How many start bits does a UART have? **One**
What causes a UART Overrun Error? **If a new byte arrives before the byte in the buffer is moved into the CPU, an Overrun Error occurs.**

HW13-3(UART-WhatIsTransmission-Parity)
Explanation below
Given the following timing diagram of a PIC32MX 8-bit UART transmission with parity, the first value being transmitted is the hex value with



1F with **Odd** (Even/Odd) parity and **2** stop bit(s)

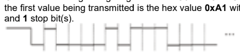
Given the following timing diagram of a PIC32MX 8-bit UART transmission with parity, the first value being transmitted is the hex value with
C5 with **Even** parity and **1** stop bit(s)



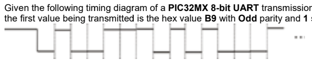
Given the following timing diagram of a PIC32MX 8-bit UART transmission with parity, the first value being transmitted is the hex value withs
29 with **Odd** parity and **2** stop bit(s)

Take the binary value (in red) and flip it: 10010100 → 00101001 and convert that to a hex value. This the data being transmitted.
The parity bit is **ODD** if there is an odd number of bits that are 1 in the transmission plus the parity bit if it is 1, and it's **EVEN** if there is an even number of bits that are 1 in the transmission plus the parity bit if it is 1. For the above example, there are three 1's and you don't add the parity bit because it is 0, which gives us 3, an odd number, therefore it is an odd parity.
The number of stop bits is just the number of high values in the blue section before the next low, which is 2 in this case.
Given the following timing diagram of a PIC32MX 8-bit UART transmission with parity, the first value being transmitted is the hex value **0x34** with **Odd** (Even/Odd) parity and **1** stop bit(s).

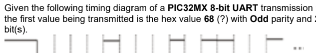
Given the following timing diagram of a PIC32MX 8-bit UART transmission with parity, the first value being transmitted is the hex value **0xA1** with **Even** (Even/Odd) parity and **1** stop bit(s).



Given the following timing diagram of a **PIC32MX 8-bit UART** transmission ***with*** parity, the first value being transmitted is the hex value **B9** with **Odd** parity and **1** stop bit(s).

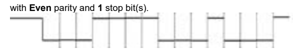


Given the following timing diagram of a **PIC32MX 8-bit UART** transmission ***with*** parity, the first value being transmitted is the hex value **68** (?) with **Odd** parity and **2** stop bit(s).

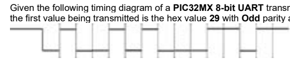


Given the following timing diagram of a **PIC32MX 8-bit UART** transmission ***with*** parity, the first value being transmitted is the hex value **3C**

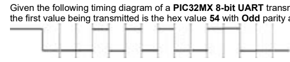
with **Even** parity and **1** stop bit(s).



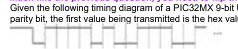
Given the following timing diagram of a **PIC32MX 8-bit UART** transmission ***with*** parity, the first value being transmitted is the hex value **29** with **Odd** parity and **2** stop bit(s).



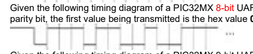
Given the following timing diagram of a **PIC32MX 8-bit UART** transmission ***with*** parity, the first value being transmitted is the hex value **54** with **Odd** parity and **2** stop bit(s).



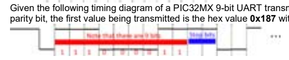
HW13-3(UART-WhatIsTransmission)
Much like the previous question, you need to flip the binary value (shown in red).
Given the following timing diagram of a PIC32MX 9-bit UART transmission with no parity bit, the first value being transmitted is the hex value **0x2F** with **2** stop bits.



Given the following timing diagram of a PIC32MX 8-bit UART transmission with no parity bit, the first value being transmitted is the hex value **0x37** with **1** stop bits.



Given the following timing diagram of a PIC32MX 9-bit UART transmission with no parity bit, the first value being transmitted is the hex value **0x187** with **2** stop bits.



HW13-4(PIC32-UART-UxMODE)
Give a single C instruction to configure UART 1 of a PIC32MX with the following parameters:(Assume all parameters not mentioned are 0's.)
Discontinue UART operation when the device enters Idle Mode. (bit13 = 1)
UxTX, UxRX, and UxBCLK pins are enabled. (bits9-8 = 11)
Loopback Mode Enabled. (bit6 = 1)
8-bit Data, No Parity. (bits2-1 = 00)
2 Stop Bits. (bit0 = 1)

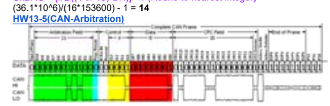
U1MODE = 0x2341;

HW13-4(PIC32-UART-UxSTA)
Give a single C instruction to configure UART 2of a PIC32MX with the following parameters:(Assume all parameters not mentioned are 0's.)
Automatic Address Detection is ENabled. (bit24 = 1)
Automatic Address Are: (bits23-16 = 1000 0001)
bit 0 = 1
bit 1 = 0
bit 2 = 0
bit 3 = 0
bit 4 = 0
bit 5 = 0
bit 6 = 0
bit 7 = 1
Interrupt is generated when transmit buffer is empty. (bits15-14 = 10)
Send Break on next Transmission. (bit11 = 1)
Address Detect Mode is Enabled. (bit5 = 1)

U2STA = 0x1818820;

HW13-4(PIC32-UART-WhatIsBaudRate)
1 => 4, 0 => 16
What is the baud rate (in bits/s to two decimal places) of a PIC32MX's UART 2 given the following C statements if fPBCLK is 45.90 MHz?
U2MODEbits.BRGH = 1; (Rate = 4) **There are only 2 modes, rate=4 or rate=16
U2BRG = 27653;
BR = f_{PBCLK} / [(4 or 16)*(UxBRG + 1)]
(45.9*10^6)/(4*(27653 + 1)) = **414.95** bits/s

HW13-4(PIC32-UART-WhatIsUxBRG)
What does a PIC32MX's UxBRG register need to be to give a baud rate of 153600 bps assuming a fPBCLK of 36.10 MHz and BRGH = 0? (Rate = 16)
UxBRG = [f_{PB}/((4 or 16)*BR)] - 1 (Round to nearest integer)
(36.1*10^6)/(16*153600) - 1 = **14**
HW13-5(CAN-Arbitration)



The bits below show the data of a CAN transmission for various nodes wanting to communicate on the bus all at the same time. Which node will get to use the bus first?
Node A: 001001000000000000001000000.... (576)
Node B: 001001010001000000011100000.... (593)
Node C: 001001000111000001011110000.... (583)
Node D: 001001111101000010011100000.... (637)
Node E: 001001100000000100010010000.... (608)
Node **A** will get control of the bus. (Lowest Value = Highest Priority)

HW13-5(CAN-Format-ACK-Delimeter)
Given the data of the CAN transmission shown..
0100101011010000001111010010011111010111010111111111110
...the value of the ACK Delimeter is **1** (answer in gray)

HW13-5(CAN-Format-ACK)
Given the data of the CAN transmission shown..
0111000101100000001001011011000100011101000001111111110
...the value of the ACK field is **0**

HW13-5(CAN-Format-CRC-Delimeter)
Given the data of the CAN transmission shown..
0101011100000000001001111000111000110101100101111111110
...the value of the CRC Delimeter is **0**

HW13-5(CAN-Format-CRC)
Given the data of the CAN transmission shown..
0111001110010000001110011010110000111101100011111111110
...the value of the CRC bits is **0x30F6** (in hex).

HW13-5(CAN-Format-Data)
Given the data of the CAN transmission shown..
00100000011100000001010001000011011101101010001111111110
...the value of the Data bits is **0x44** (in hex).

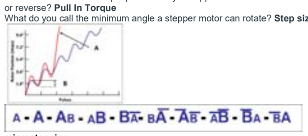
HW13-5(CAN-Format-DataLength)
Given the data of the CAN transmission shown
0011110010100001000111001111110001110001100101111111110
...the value of the Data Length is **8** (if binary value is greater than 8, the answer is 8)

HW13-5(CAN-Format-ID)
Given the data of the CAN transmission shown..
0011101110010000001011100100010011100111011001111111110
...the value of the Identifier is **0x3B9** (in hex).

HW13-5(CAN-Format-RR)
Given the data of the CAN transmission shown..
0111000001111000001101000010110001101111101001111111110
...the value of the Request Remote field is **1**

Test 3 only covers parts 10 - 13

Chapter 19
HW19(StepperMotors-General)
What do you call the part of a stepper motor that is generally stationary and is held in place by the outer case? **Stator**
What type of stepper motor generally has the smallest step angles? **Hybrid Stepper Motor**
What type of stepper motor has a commuator to produce a magnetic field on the rotor? **None**
What do you call the part of a stepper motor that are radial projections on the mtor's poles used to concentrate the magnetic flux to certain positions on the stator? **Stator Teeth**
What is the following stepper motor graph exhibiting by label 'A'? **Loss of Synchronization**
What is the maximum torque produced by a stepper motor where it can still stop, start, or reverse? **Pull In Torque**
What do you call the minimum angle a stepper motor can rotate? **Step size**



microstepping

HW19(StepperMotors-Hybrid-RotorTeeth)
From the notes: "The number of Rotor Teeth is somewhat larger than the Stator Teeth."
So take the stator teeth number and go to the next highest amount.
Given a hybrid stepper motor with 4 poles and 40 stator teeth, which number would be the best
choice for the number of rotor teeth the motor has? **50**
Given a hybrid stepper motor with **3** poles and **36** stator teeth, which number would be the best choice for the number of rotor teeth the motor has? **46**

Given a hybrid stepper motor with 2 poles and 40 stator teeth, which number would be the best choice for the number of rotor teeth the motor has? **40**
Given a hybrid stepper motor with 5 poles and 40 stator teeth, which number would be the best choice for the number of rotor teeth the motor has? **50**
Given a hybrid stepper motor with 3 poles and 24 stator teeth, which number would be the best choice for the number of rotor teeth the motor has? **48**
Given a hybrid stepper motor with 8 poles and 24 stator teeth, which number would be the best choice for the number of rotor teeth the motor has? **32**
Given a hybrid stepper motor with 3 poles and 60 stator teeth, which number would be the best choice for the number of rotor teeth the motor has? **70**
HW19(StepperMotors-Hybrid-StatorTeeth)
From note: "The Stator Teeth are a multiple of the poles and typically are 16 to 32."
Pick closest multiple of # of poles that is less than # of rotor teeth
Given a hybrid stepper motor with 4 poles and 28 rotor teeth, which number would be the best choice for the number of stator teeth the motor has? **20**
Given a hybrid stepper motor with 5 poles and 20 rotor teeth, which number would be the best choice for the number of stator teeth the motor has? **15**
Given a hybrid stepper motor with 2 poles and 30 rotor teeth, which number would be the best choice for the number of stator teeth the motor has? **21**
Given a hybrid stepper motor with 6 poles and 20 rotor teeth, which number would be the best choice for the number of stator teeth the motor has? **12**
Given a hybrid stepper motor with 4 poles and 44 rotor teeth, which number would be the best choice for the number of stator teeth the motor has? **36**
Given a hybrid stepper motor with 5 poles and 50 rotor teeth, which number would be the best choice for the number of stator teeth the motor has? **35**

HW19(StepperMotors-Hybrid-StepSize)
Step size = 360*(RT-ST)/(RT*ST)
Given a 4-phase Hybrid stepper motor with 72 stator teeth and 90 rotor teeth, what is the step size of the motor (in degrees to two decimal places)? **1.00**

HW19(StepperMotors-Hybrid-ToothPitch)
Tooth pitch = 360/RT
Given a 5-phase Hybrid stepper motor with 39 stator teeth and 49 rotor teeth, what is the tooth pitch of the motor (in degrees to two decimal places)? **7.35**
Given a 3-phase Hybrid stepper motor with 21 stator teeth and 35 rotor teeth, what is the tooth pitch of the motor (in degrees to two decimal places)? **10.29**

HW19(StepperMotors-VR-MS-PitchStepSize)
Step size = 360/(phase*RT)
Given a 5-phase Multi-stack Variable Reluctance stepper motor with 20 rotor teeth, what is the step size of the motor (in degrees to two decimal places)?**3.6**

HW19(StepperMotors-VR-MS-StatorTeeth)
Number of Stator Teeth equals Number of Rotor Teeth because the offset of each stack from the others are what produce the next step angle.

Given a multi-stack variable-reluctance stepper motor with 4 phases and 32 stator teeth, how many rotor teeth must the motor have? **32**

HW19(StepperMotors-VR-SS-PitchStepSize)
Tooth pitch = 360/RT
Given a 5-phase Single-stack Variable Reluctance stepper motor with 8 rotor teeth, what is the tooth pitch of the motor (in degrees to two decimal places)? **45**

HW19(StepperMotors-VR-SS-RotorTeeth)
Not the same number but the next closest (greater than or less than)
Given a single-stack variable-reluctance stepper motor with 6 phases and 18 stator teeth, which number would be the best choice for the number of rotor teeth the motor has? **12**

HW19(StepperMotors-VR-SS-StatorTeeth)
Next closest number that is a multiple of the number of phases (greater than or less than)
Given a single-stack variable-reluctance stepper motor with 2 phases and 8 rotor teeth, which number would be the best choice for the number of stator teeth the motor has? **8**

Given a single-stack variable-reluctance stepper motor with 5 phases and 20 rotor teeth, which number would be the best choice for the number of stator teeth the motor has? **30**

Hardware exceptions: user or processor, external interrupts or internal timers.
Software exceptions: Illegal instructions, divide by 0, traps i.e. *(int*) (0x0000000)
Exception Handling: Resets – power on reset, external resets, watchdog, and clock monitor reset.
Watchdog: counts down to 0x0 and then resets processor, can't turn off after turning on (except reset).
Clock Monitor: resets when system clock frequency drops below prescribed value.
Brown-out: supply voltage goes below threshold.

- Power-on Reset (POR)
- Master Clear Reset pin (MCLR)
- Software Reset (SWR)
- Watchdog Timer Reset (WDTR)
- Brown-out Reset (BOR)
- Configuration Mismatch Reset (CMR)

Asynchronous Interrupt: I/O - communication, Errors
Synchronous Interrupts: I/O – update periodically, Processes – switch between tasks, Timing- measure elapsed time.
Maskable Interrupt: enabled and disabled in software
Non-Maskable: enabled by software, only disabled by hardware.

RCON:

bit 31-10 **Unimplemented:** Read as '0'
bit 9 **CMR:** Configuration Mismatch Reset Flag bit
1 = Configuration mismatch Reset has occurred
0 = Configuration mismatch Reset has not occurred
bit 8 **VREGS:** Voltage Regulator Standby Enable bit
1 = Regulator is enabled and is on during Sleep mode
0 = Regulator is disabled and is off during Sleep mode
bit 7 **EXTR:** External Reset (MCLR) Pin Flag bit
1 = Master Clear (pin) Reset has occurred
0 = Master Clear (pin) Reset has not occurred
bit 6 **SWR:** Software Reset Flag bit
1 = Software Reset was executed
0 = Software Reset as not executed
bit 5 **Unimplemented:** Read as '0'
bit 4 **WDTO:** Watchdog Timer Time-out Flag bit
1 = WDT Time-out has occurred
0 = WDT Time-out has not occurred
bit 3 **SLEEP:** Wake From Sleep Flag bit
1 = Device was in Sleep mode
0 = Device was not in Sleep mode
bit 2 **IDLE:** Wake From Idle Flag bit
1 = Device was in Idle mode
0 = Device was not in Idle mode
bit 1 **BOR:** Brown-out Reset Flag bit[1]
1 = Brown-out Reset has occurred
0 = Brown-out Reset has not occurred
bit 0 **POR:** Power-on Reset Flag bit[1]
1 = Power-on Reset has occurred
0 = Power-on Reset has not occurred

IP, IS: bits 7:0 – IP00, IS00. bits 15:8 – IP01, IS01. bits 23:16 – IP02, IS02. Bits 31:24 – IP03, IS03.
External interrupts: INTCONbits.INTxEP – sets edge polarity (0 falling, 1 rising).

- void __ISR(_EXTERNAL_0_VECTOR, ipl2) Int0_IRQ(void);
- void __attribute__((interrupt(ipl2), vector(_EXTERNAL_0_VECTOR))) Int0_IRQ (void);
-

TABLE 7-1: INTERRUPT IRQ, VECTOR AND BIT LOCATION

Interrupt Source[1]	IRQ #	Vector #	Flag	Enable	Priority	Sub-priority	Persistent Interrupt
Highest Natural Order Priority							
CT – Core Timer Interrupt	0	0	IFS0<0>	IEC0<0>	IPC0<4:2>	IPC0<1:0>	No
CS0 – Core Software Interrupt 0	1	1	IFS0<1>	IEC0<1>	IPC0<12:10>	IPC0<9:8>	No
CS1 – Core Software Interrupt 1	2	2	IFS0<2>	IEC0<2>	IPC0<20:18>	IPC0<17:16>	No
INT0 – External Interrupt	3	3	IFS0<3>	IEC0<3>	IPC0<28:26>	IPC0<25:24>	No
T1 – Timer1	4	4	IFS0<4>	IEC0<4>	IPC1<4:2>	IPC1<1:0>	No
IC1E – Input Capture 1 Error	5	5	IFS0<5>	IEC0<5>	IPC1<12:10>	IPC1<9:8>	Yes
IC1 – Input Capture 1	6	5	IFS0<6>	IEC0<6>	IPC1<12:10>	IPC1<9:8>	Yes
OC1 – Output Compare 1	7	6	IFS0<7>	IEC0<7>	IPC1<20:18>	IPC1<17:16>	No
INT1 – External Interrupt 1	8	7	IFS0<8>	IEC0<8>	IPC1<28:26>	IPC1<25:24>	No
T2 – Timer2	9	8	IFS0<9>	IEC0<9>	IPC2<4:2>	IPC2<1:0>	No
IC2E – Input Capture 2	10	9	IFS0<10>	IEC0<10>	IPC2<12:10>	IPC2<9:8>	Yes
IC2 – Input Capture 2	11	9	IFS0<11>	IEC0<11>	IPC2<12:10>	IPC2<9:8>	Yes
OC2 – Output Compare 2	12	10	IFS0<12>	IEC0<12>	IPC2<20:18>	IPC2<17:16>	No
INT2 – External Interrupt 2	13	11	IFS0<13>	IEC0<13>	IPC2<28:26>	IPC2<25:24>	No
T3 – Timer3	14	12	IFS0<14>	IEC0<14>	IPC3<4:2>	IPC3<1:0>	No
IC3E – Input Capture 3	15	13	IFS0<15>	IEC0<15>	IPC3<12:10>	IPC3<9:8>	Yes
IC3 – Input Capture 3	16	13	IFS0<16>	IEC0<16>	IPC3<12:10>	IPC3<9:8>	Yes
OC3 – Output Compare 3	17	14	IFS0<17>	IEC0<17>	IPC3<20:18>	IPC3<17:16>	No
INT3 – External Interrupt 3	18	15	IFS0<18>	IEC0<18>	IPC3<28:26>	IPC3<25:24>	No
T4 – Timer4	19	16	IFS0<19>	IEC0<19>	IPC4<4:2>	IPC4<1:0>	No
IC4E – Input Capture 4 Error	20	17	IFS0<20>	IEC0<20>	IPC4<12:10>	IPC4<9:8>	Yes
IC4 – Input Capture 4	21	17	IFS0<21>	IEC0<21>	IPC4<12:10>	IPC4<9:8>	Yes
OC4 – Output Compare 4	22	18	IFS0<22>	IEC0<22>	IPC4<20:18>	IPC4<17:16>	No
INT4 – External Interrupt 4	23	19	IFS0<23>	IEC0<23>	IPC4<28:26>	IPC4<25:24>	No
T5 – Timer5	24	20	IFS0<24>	IEC0<24>	IPC5<4:2>	IPC5<1:0>	No
IC5E – Input Capture 5 Error	25	21	IFS0<25>	IEC0<25>	IPC5<12:10>	IPC5<9:8>	Yes
IC5 – Input Capture 5	26	21	IFS0<26>	IEC0<26>	IPC5<12:10>	IPC5<9:8>	Yes
OC5 – Output Compare 5	27	22	IFS0<27>	IEC0<27>	IPC5<20:18>	IPC5<17:16>	No
AD1 – ADC1 Convert done	28	23	IFS0<28>	IEC0<28>	IPC5<28:26>	IPC5<25:24>	Yes
FSCM – Fail-Safe Clock Monitor	29	24	IFS0<29>	IEC0<29>	IPC6<4:2>	IPC6<1:0>	No
RTCC – Real-Time Clock and Calendar	30	25	IFS0<30>	IEC0<30>	IPC6<12:10>	IPC6<9:8>	No
FCE – Flash Control Event	31	26	IFS0<31>	IEC0<31>	IPC6<20:18>	IPC6<17:16>	No
CMP1 – Comparator Interrupt	32	27	IFS1<0>	IEC1<0>	IPC6<28:26>	IPC6<25:24>	No
CMP2 – Comparator Interrupt	33	28	IFS1<1>	IEC1<1>	IPC7<4:2>	IPC7<1:0>	No
CMP3 – Comparator Interrupt	34	29	IFS1<2>	IEC1<2>	IPC7<12:10>	IPC7<9:8>	No
USB – USB Interrupts	35	30	IFS1<3>	IEC1<3>	IPC7<20:18>	IPC7<17:16>	Yes
SPI1E – SPI1 Fault	36	31	IFS1<4>	IEC1<4>	IPC7<28:26>	IPC7<25:24>	Yes
SPI1RX – SPI1 Receive Done	37	31	IFS1<5>	IEC1<5>	IPC7<28:26>	IPC7<25:24>	Yes
SPI1TX – SPI1 Transfer Done	38	31	IFS1<6>	IEC1<6>	IPC7<28:26>	IPC7<25:24>	Yes

TABLE 7-1: INTERRUPT IRQ, VECTOR AND BIT LOCATION (CONTINUED)

Interrupt Source[1]	IRQ #	Vector #	Flag	Enable	Priority	Sub-priority	Persistent Interrupt
U1E – UART1 Fault	39	32	IFS1<7>	IEC1<7>	IPC8<4:2>	IPC8<1:0>	Yes
U1RX – UART1 Receive Done	40	32	IFS1<8>	IEC1<8>	IPC8<4:2>	IPC8<1:0>	Yes
U1TX – UART1 Transfer Done	41	32	IFS1<9>	IEC1<9>	IPC8<4:2>	IPC8<1:0>	Yes
I2C1B – I2C1 Bus Collision Event	42	33	IFS1<10>	IEC1<10>	IPC8<12:10>	IPC8<9:8>	Yes
I2C1S – I2C1 Slave Event	43	33	IFS1<11>	IEC1<11>	IPC8<12:10>	IPC8<9:8>	Yes
I2C1M – I2C1 Master Event	44	33	IFS1<12>	IEC1<12>	IPC8<12:10>	IPC8<9:8>	Yes
CNA – PORTA Input Change Interrupt	45	34	IFS1<13>	IEC1<13>	IPC8<20:18>	IPC8<17:16>	Yes
CNB – PORTB Input Change Interrupt	46	34	IFS1<14>	IEC1<14>	IPC8<20:18>	IPC8<17:16>	Yes
CNC – PORTC Input Change Interrupt	47	34	IFS1<15>	IEC1<15>	IPC8<20:18>	IPC8<17:16>	Yes
PMP – Parallel Master Port	48	35	IFS1<16>	IEC1<16>	IPC8<28:26>	IPC8<25:24>	Yes
PMPE – Parallel Master Port Error	49	35	IFS1<17>	IEC1<17>	IPC8<28:26>	IPC8<25:24>	Yes
SPI2E – SPI2 Fault	50	36	IFS1<18>	IEC1<18>	IPC9<4:2>	IPC9<1:0>	Yes
SPI2RX – SPI2 Receive Done	51	36	IFS1<19>	IEC1<19>	IPC9<4:2>	IPC9<1:0>	Yes
SPI2TX – SPI2 Transfer Done	52	36	IFS1<20>	IEC1<20>	IPC9<4:2>	IPC9<1:0>	Yes
U2E – UART2 Error	53	37	IFS1<21>	IEC1<21>	IPC9<12:10>	IPC9<9:8>	Yes
U2RX – UART2 Receiver	54	37	IFS1<22>	IEC1<22>	IPC9<12:10>	IPC9<9:8>	Yes
U2TX – UART2 Transmitter	55	37	IFS1<23>	IEC1<23>	IPC9<12:10>	IPC9<9:8>	Yes
I2C2B – I2C2 Bus Collision Event	56	38	IFS1<24>	IEC1<24>	IPC9<20:18>	IPC9<17:16>	Yes
I2C2S – I2C2 Slave Event	57	38	IFS1<25>	IEC1<25>	IPC9<20:18>	IPC9<17:16>	Yes
I2C2M – I2C2 Master Event	58	38	IFS1<26>	IEC1<26>	IPC9<20:18>	IPC9<17:16>	Yes
CTMU – CTMU Event	59	39	IFS1<27>	IEC1<27>	IPC9<28:26>	IPC9<25:24>	Yes
DMA0 – DMA Channel 0	60	40	IFS1<28>	IEC1<28>	IPC10<4:2>	IPC10<1:0>	No
DMA1 – DMA Channel 1	61	41	IFS1<29>	IEC1<29>	IPC10<12:10>	IPC10<9:8>	No
DMA2 – DMA Channel 2	62	42	IFS1<30>	IEC1<30>	IPC10<20:18>	IPC10<17:16>	No
DMA3 – DMA Channel 3	63	43	IFS1<31>	IEC1<31>	IPC10<28:26>	IPC10<25:24>	No
Lowest Natural Order Priority							

Note 1: Not all interrupt sources are available on all devices. See TABLE 1: "PIC32MX1XX 28/36/44-Pin General Purpose Family Features" and TABLE 2: "PIC32MX2XX 28/36/44-pin USB Family Features" for the lists of available peripherals.

- **Input capture:** counts pulses until rising/falling edge to determine duty factors, measure Pulse width etc.
- **Output compare:** like alarm clock, create signal after predefined time has elapsed. Make PWM
- **OSCCON -** oscillator function
- bit 21 – PBDIVRDY and 20/19 PBDIV
- PBDIV divides SYSCLK by 1, 2, 4, 8 (default) for 00, 01, 10, 11
-

- **Core timer –** uses COUNT and COMPARE registers can be used to generate interrupt when COMPARE = COUNT. Interrupt 0.
- COUNT has ½ frequency of SYSCLK.

$$f_{CT} = \frac{f_{SYSCLK}}{2X(COMPARE)} \qquad T_{CT} = \frac{2X(COMPARE)}{f_{SYSCLK}}$$

- CP0_GET_COUNT(), CPO_SET_COUNT(value), CP0_GET_COMPARE(), CP0_SET_COMPARE(val).
- **Timers:** PIC32 has TMR1-TMR5.
- **Type A Timer:** TMR1 sync/async 16-bit
- Type B Timers: TMR2- TMR5 syn 16-bit but can be Combined, 2/3 & 4/5 to use as 32-bit

Type A: T1CON

bit 31-16 **Unimplemented:** Read as '0'
bit 15 **ON:** Timer On bit[1]
1 = Timer is enabled
0 = Timer is disabled
bit 14 **Unimplemented:** Read as '0'
bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when the device enters Idle mode
0 = Continue module operation when the device enters Idle mode
bit 12 **TWDIS:** Asynchronous Timer Write Disable bit
1 = Writes to Timer1 are ignored until pending write operation completes
0 = Back-to-back writes are enabled (Legacy Asynchronous Timer functionality)
bit 11 **TWIP:** Asynchronous Timer Write in Progress bit
In Asynchronous Timer mode:
1 = Asynchronous write to the Timer1 register in progress
0 = Asynchronous write to Timer1 register is complete
In Synchronous Timer mode:
This bit is read as '0'.
bit 10-8 **Unimplemented:** Read as '0'
bit 7 **TGATE:** Timer Gated Time Accumulation Enable bit
When TCS = 1:
This bit is ignored.
When TCS = 0:
1 = Gated time accumulation is enabled
0 = Gated time accumulation is disabled
bit 6 **Unimplemented:** Read as '0'
bit 5-4 **TCKPS<1:0>:** Timer Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

REGISTER 12-1: T1CON: TYPE A TIMER CONTROL REGISTER (CONTINUED)
bit 3 **Unimplemented:** Read as '0'
bit 2 **TSYNC:** Timer External Clock Input Synchronization Selection bit
When TCS = 1:
1 = External clock input is synchronized
0 = External clock input is not synchronized
When TCS = 0:
This bit is ignored.
bit 1 **TCS:** Timer Clock Source Select bit
1 = External clock from TxCKI pin
0 = Internal peripheral clock
bit 0 **Unimplemented:** Read as '0'

- Input Capture: PIC32 has 5

ADD 1 TO PRy

Type B: TxCON, x = [2,5]

bit 31-16 **Unimplemented:** Read as '0'
bit 15 **ON:** Timer On bit[1,3]
1 = Module is enabled
0 = Module is disabled
bit 14 **Unimplemented:** Read as '0'
bit 13 **SIDL:** Stop in Idle Mode bit[4]
1 = Discontinue module operation when the device enters Idle mode
0 = Continue module operation when the device enters Idle mode
bit 12-8 **Unimplemented:** Read as '0'
bit 7 **TGATE:** Timer Gated Time Accumulation Enable bit[3]
When TCS = 1:
This bit is ignored and is read as '0'.
When TCS = 0:
1 = Gated time accumulation is enabled
0 = Gated time accumulation is disabled
bit 6-4 **TCKPS<2:0>:** Timer Input Clock Prescale Select bits[3]
111 = 1:256 prescale value
110 = 1:64 prescale value
101 = 1:32 prescale value
100 = 1:16 prescale value
011 = 1:8 prescale value
010 = 1:4 prescale value
001 = 1:2 prescale value
000 = 1:1 prescale value

REGISTER 13-1: TXCON: TYPE B TIMER CONTROL REGISTER (CONTINUED)
bit 3 **T32:** 32-Bit Timer Mode Select bit[2]
1 = Odd numbered and even numbered timers form a 32-bit timer
0 = Odd numbered and even numbered timers form a separate 16-bit timer
bit 2 **Unimplemented:** Read as '0'
bit 1 **TCS:** Timer Clock Source Select bit[3]
1 = External clock from TxCK pin
0 = Internal peripheral clock
bit 0 **Unimplemented:** Read as '0'

Note 1: When using 1:1 PBCLK divisor, the user's software should not read/write the peripheral SFRs in the SYSCLK cycle immediately following the instruction that clears the module's ON bit.
2: This bit is available only on even numbered timers (Timer2 and Timer4).
3: While operating in 32-bit mode, this bit has no effect for odd numbered timers (Timer3, and Timer5). All timer functions are set through the even numbered timers.
4: While operating in 32-bit mode, this bit must be cleared on odd numbered timers to enable the 32-bit timer in Idle mode.

Input Capture: ICxCON, x = [1,5]

bit 31-16 **Unimplemented:** Read as '0'
bit 15 **ON:** Input Capture Module Enable bit[1]
1 = Module is enabled
0 = Disable and reset module, disable clocks, disable interrupt generation and allow SFR modifications
bit 14 **Unimplemented:** Read as '0'
bit 13 **SIDL:** Stop in Idle Control bit
1 = Halt in Idle mode
0 = Continue to operate in Idle mode
bit 12-10 **Unimplemented:** Read as '0'
bit 9 **FEDGE:** First Capture Edge Select bit (only used in mode 6, ICM<2:0> = 110)
1 = Capture rising edge first
0 = Capture falling edge first
bit 8 **C32:** 32-bit Capture Select bit
1 = 32-bit timer resource capture
0 = 16-bit timer resource capture
bit 7 **ICTMR:** Timer Select bit (Does not affect timer selection when C32 (ICxCON<8>) is '1')
0 = Timer3 is the counter source for capture
1 = Timer2 is the counter source for capture
bit 6-5 **ICI<1:0>:** Interrupt Control bits
11 = Interrupt on every fourth capture event
10 = Interrupt on every third capture event
01 = Interrupt on every second capture event
00 = Interrupt on every capture event
bit 4 **ICOV:** Input Capture Overflow Status Flag bit (read-only)
1 = Input capture overflow has occurred
0 = No input capture overflow has occurred
bit 3 **ICBNE:** Input Capture Buffer Not Empty Status bit (read-only)
1 = Input capture buffer is not empty; at least one more capture value can be read
0 = Input capture buffer is empty

Note 1: When using 1:1 PBCLK divisor, the user's software should not read/write the peripheral's SFRs in the SYSCLK cycle immediately following the instruction that clears the module's ON bit.

REGISTER 15-1: ICxCON: INPUT CAPTURE 'x' CONTROL REGISTER (CONTINUED)
bit 2-0 **ICM<2:0>:** Input Capture Mode Select bits
111 = Interrupt-Only mode (only supported while in Sleep mode or Idle mode)
110 = Simple Capture Event mode – every edge, specified edge first and every edge thereafter
101 = Prescaled Capture Event mode – every sixteenth rising edge
100 = Prescaled Capture Event mode – every fourth rising edge
011 = Simple Capture Event mode – every rising edge
010 = Simple Capture Event mode – every falling edge
001 = Edge Detect mode – every edge (rising and falling)
000 = Input Capture module is disabled

Note 1: When using 1:1 PBCLK divisor, the user's software should not read/write the peripheral's SFRs in the SYSCLK cycle immediately following the instruction that clears the module's ON bit.

Output compare – OCxCON, x = [1,5]

bit 31-16 **Unimplemented:** Read as '0'
bit 15 **ON:** Output Compare Peripheral On bit[1]
1 = Output Compare peripheral is enabled
0 = Output Compare peripheral is disabled
bit 14 **Unimplemented:** Read as '0'
bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when the device enters Idle mode
0 = Continue module operation when the device enters Idle mode
bit 12-6 **Unimplemented:** Read as '0'
bit 5 **OC32:** 32-bit Compare Mode bit
1 = OCxR<31:0> and/or OCxRS<31:0> are used for comparisons to the 32-bit timer source
0 = OCxR<15:0> and OCxRS<15:0> are used for comparisons to the 16-bit timer source
bit 4 **OCFLT:** PWM Fault Condition Status bit[2]
1 = PWM Fault condition has occurred (cleared in hardware only)
0 = No PWM Fault condition has occurred
bit 3 **OCTSEL:** Output Compare Timer Select bit
1 = Timer3 is the clock source for this Output Compare module
0 = Timer2 is the clock source for this Output Compare module
bit 2-0 **OCM<2:0>:** Output Compare Mode Select bits
111 = PWM mode on OCx; Fault pin enabled
110 = PWM mode on OCx; Fault pin disabled
101 = Initialize OCx pin low; generate continuous output pulses on OCx pin
100 = Initialize OCx pin low; generate single output pulse on OCx pin
011 = Compare event toggles OCx pin
010 = Initialize OCx pin high; compare event forces OCx pin low
001 = Initialize OCx pin low; compare event forces OCx pin high
000 = Output compare peripheral is disabled but continues to draw current

Note 1: When using 1:1 PBCLK divisor, the user's software should not read/write the peripheral's SFRs in the SYSCLK cycle immediately following the instruction that clears the module's ON bit.
2: This bit is only used when OCM<2:0> = '111'. It is read as '0' in all other modes.

- **Square Wave Calcs:** final frequency / 2 - half period on.
- **RPM:** Let N = number of "equally spaced events"
f_tmr = timer frequency
ticks = TMRx(new) – TMRx(old)
RPM = 60 / (N * (ticks/f_tmr))
- Watchdog timer – STWD100

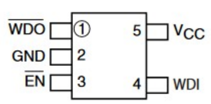


Table 9-3: WDT Time-out Period vs. Postscaler Settings[1, 2]

FWDTPS<4:0>	Postscaler Ratio	Time-out Period
00000	1:1	1 ms
00001	1:2	2 ms
00010	1:4	4 ms
00011	1:8	8 ms
00100	1:16	16 ms
00101	1:32	32 ms
00110	1:64	64 ms
00111	1:128	128 ms
01000	1:256	256 ms
01001	1:512	512 ms
01010	1:1024	1.024 s
01011	1:2048	2.048 s
01100	1:4096	4.096 s
01101	1:8192	8.192 s
01110	1:16384	16.384 s
01111	1:32768	32.768 s
10000	1:65536	65.536 s
10001	1:131072	131.072 s
10010	1:262144	262.144 s
10011	1:524288	524.288 s
10100	1:1045876	1048.576 s

Note 1: All other combinations will result in operation as if the prescaler was set to 10100.
2: The periods listed are based on a 32 kHz (nominal) input clock.

DEVCGF1 – FWDTEN 1 = on

bit 31-10 **Unimplemented:** Read as '0'
bit 9 **CMR:** Configuration Mismatch Reset Flag bit
1 = Configuration mismatch Reset has occurred
0 = Configuration mismatch Reset has not occurred
bit 8 **VREGS:** Voltage Regulator Standby Enable bit
1 = Regulator is enabled and is on during Sleep mode
0 = Regulator is disabled and is off during Sleep mode
bit 7 **EXTR:** External Reset (MCLR) Pin Flag bit
1 = Master Clear (pin) Reset has occurred
0 = Master Clear (pin) Reset has not occurred
bit 6 **SWR:** Software Reset Flag bit
1 = Software Reset was executed
0 = Software Reset as not executed
bit 5 **Unimplemented:** Read as '0'
bit 4 **WDTO:** Watchdog Timer Time-out Flag bit
1 = WDT Time-out has occurred
0 = WDT Time-out has not occurred
bit 3 **SLEEP:** Wake From Sleep Flag bit
1 = Device was in Sleep mode
0 = Device was not in Sleep mode
bit 2 **IDLE:** Wake From Idle Flag bit
1 = Device was in Idle mode
0 = Device was not in Idle mode
bit 1 **BOR:** Brown-out Reset Flag bit[1]
1 = Brown-out Reset has occurred
0 = Brown-out Reset has not occurred
bit 0 **POR:** Power-on Reset Flag bit[1]
1 = Power-on Reset has occurred
0 = Power-on Reset has not occurred

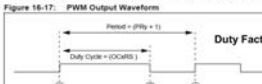
Note 1: User software must clear this bit to view next detection.

RCON --^ ------ PWM ------v

bit 31-16 **Unimplemented:** Read as '0'
bit 15 **ON:** Output Compare Peripheral On bit[1]
1 = Output Compare peripheral is enabled
0 = Output Compare peripheral is disabled
bit 14 **Unimplemented:** Read as '0'
bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when the device enters Idle mode
0 = Continue module operation when the device enters Idle mode
bit 12-6 **Unimplemented:** Read as '0'
bit 5 **OC32:** 32-bit Compare Mode bit
1 = OCxR<31:0> and/or OCxRS<31:0> are used for comparisons to the 32-bit timer source
0 = OCxR<15:0> and OCxRS<15:0> are used for comparisons to the 16-bit timer source
bit 4 **OCFLT:** PWM Fault Condition Status bit[2]
1 = PWM Fault condition has occurred (cleared in hardware only)
0 = No PWM Fault condition has occurred
bit 3 **OCTSEL:** Output Compare Timer Select bit
1 = Timer3 is the clock source for this Output Compare module
0 = Timer2 is the clock source for this Output Compare module
bit 2-0 **OCM<2:0>:** Output Compare Mode Select bits
111 = PWM mode on OCx; Fault pin enabled
110 = PWM mode on OCx; Fault pin disabled
101 = Initialize OCx pin low; generate continuous output pulses on OCx pin
100 = Initialize OCx pin low; generate single output pulse on OCx pin
011 = Compare event toggles OCx pin
010 = Initialize OCx pin high; compare event forces OCx pin low
001 = Initialize OCx pin low; compare event forces OCx pin high
000 = Output compare peripheral is disabled but continues to draw current

Note 1: When using 1:1 PBCLK divisor, the user's software should not read/write the peripheral's SFRs in the SYSCLK cycle immediately following the instruction that clears the module's ON bit.
2: This bit is only used when OCM<2:0> = '111'. It is read as '0' in all other modes.

Figure 16-17: PWM Output Waveform



$$\text{Duty Factor} = \frac{OCxRS}{PRy + 1}$$

① Timery is cleared and the new duty cycle value is loaded from OCxRS into OCxR.
② Timer value equals the value in the OCxR register, OCx Pin is driven low.
③ Timer overflow, value from OCxRS is loaded into OCxR, OCx pin is driven high. TyIF interrupt flag is asserted.

Equation 16-1: Calculating the PWM Period

$$\text{PWM Period} = [(PR + 1) \cdot T_{PB} \cdot (\text{TMR Prescale Value})]$$

$$\text{PWM Frequency} = 1/[\text{PWM Period}]$$

Equation 16-2: Calculation for Maximum PWM Resolution

$$\text{Maximum PWM Resolution (bits)} = \frac{\log_{10}\left(\frac{F_{PB}}{F_{PWM} \cdot TMRy \cdot \text{Prescaler bits}}\right)}{\log_{10}(2)}$$

Max RES^^ DON'T ROUND/TRUNCATE

Serial Peripheral Interface (SPI) - Full Duplex
Inter Integrated Circuits (I^2C) - Half Duplex
MOSI – Master Out Slave In (Data out)
MISO – Master In Slave Out (Data in)
SCK – Serial clock
SS# - Slave Select
PIC32 has **2 SPI** peripherals available

REGISTER 17-1: SPIxCON: SPI CONTROL REGISTER

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0 FRMEN	R/W-0 FRMSYNC	R/W-0 FRMPOL	R/W-0 MSSEN	R/W-0 FRMSYPW	R/W-0 FRMCNT<2:0>	R/W-0	R/W-0
23:16	R/W-0 MCLKSEL[2]	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	R/W-0 SPIFE	R/W-0 ENHBUF[2]
15:8	R/W-0 ON[1]	U-0 —	R/W-0 SIDL	R/W-0 DISSDO	R/W-0 MODE32	R/W-0 MODE16	R/W-0 SMP	R/W-0 CKE[3]
7:0	R/W-0 SSEN	R/W-0 CKP[4]	R/W-0 MSTEN	R/W-0 DISSDI	R/W-0 STXISEL<1:0>	R/W-0	R/W-0 SRXISEL<1:0>	R/W-0

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

bit 31 **FRMEN:** Framed SPI Support bit
1 = Framed SPI support is enabled (SSx pin used as FSYNC input/output)
0 = Framed SPI support is disabled
bit 30 **FRMSYNC:** Frame Sync Pulse Direction Control on SSx pin bit (Framed SPI mode only)
1 = Frame sync pulse input (Slave mode)
0 = Frame sync pulse output (Master mode)
bit 29 **FRMPOL:** Frame Sync Polarity bit (Framed SPI mode only)
1 = Frame pulse is active-high
0 = Frame pulse is active-low
bit 28 **MSSEN:** Master Mode Slave Select Enable bit
1 = Slave select SPI support enabled. The SS pin is automatically driven during transmission in Master mode. Polarity is determined by the FRMPOL bit.
0 = Slave select SPI support is disabled.
bit 27 **FRMSYPW:** Frame Sync Pulse Width bit
1 = Frame sync pulse is one character wide
0 = Frame sync pulse is one clock wide
bit 26-24 **FRMCNT<2:0>:** Frame Sync Pulse Counter bits. Controls the number of data characters transmitted per pulse. This bit is only valid in FRAMED_SYNC mode.
111 = Reserved; do not use
110 = Reserved; do not use
101 = Generate a frame sync pulse on every 32 data characters
100 = Generate a frame sync pulse on every 16 data characters
011 = Generate a frame sync pulse on every 8 data characters
010 = Generate a frame sync pulse on every 4 data characters
001 = Generate a frame sync pulse on every 2 data characters
000 = Generate a frame sync pulse on every data character
bit 23 **MCLKSEL:** Master Clock Enable bit[2]
1 = REFCLK is used by the Baud Rate Generator
0 = PBCLK is used by the Baud Rate Generator
bit 22-18 **Unimplemented:** Read as '0'

Note 1: When using the 1:1 PBCLK divisor, the user's software should not read or write the peripheral's SFRs in the SYSCLK cycle immediately following the instruction that clears the module's ON bit.
2: This bit can only be written when the ON bit = 0.
3: This bit is not used in the Framed SPI mode. The user should program this bit to '0' for the Framed SPI mode (FRMEN = 1).
4: When AUDEN = 1, the SPI module functions as if the CKP bit is equal to '1', regardless of the actual value of CKP.

bit 17 **SPIFE:** Frame Sync Pulse Edge Select bit (Framed SPI mode only)
1 = Frame synchronization pulse coincides with the first bit clock
0 = Frame synchronization pulse precedes the first bit clock
bit 16 **ENHBUF:** Enhanced Buffer Enable bit[2]
1 = Enhanced Buffer mode is enabled
0 = Enhanced Buffer mode is disabled
bit 15 **ON:** SPI Peripheral On bit[1]
1 = SPI Peripheral is enabled
0 = SPI Peripheral is disabled
bit 14 **Unimplemented:** Read as '0'
bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when the device enters Idle mode
0 = Continue module operation when the device enters Idle mode
bit 12 **DISSDO:** Disable SDOx pin bit
1 = SDOx pin is not used by the module. Pin is controlled by associated PORT register
0 = SDOx pin is controlled by the module
bit 11-10 **MODE<32,16>:** 32/16-Bit Communication Select bits
When AUDEN = 1:

MODE32	MODE16	Communication
1	1	24-bit Data, 32-bit FIFO, 32-bit Channel/64-bit Frame
1	0	32-bit Data, 32-bit FIFO, 32-bit Channel/64-bit Frame
0	1	16-bit Data, 16-bit FIFO, 32-bit Channel/64-bit Frame
0	0	16-bit Data, 16-bit FIFO, 16-bit Channel/32-bit Frame

When AUDEN = 0:

MODE32	MODE16	Communication
1	x	32-bit
0	1	16-bit
0	0	8-bit

bit 9 **SMP:** SPI Data Input Sample Phase bit
Master mode (MSTEN = 1):
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
Slave mode (MSTEN = 0):
SMP value is ignored when SPI is used in Slave mode. The module always uses SMP = 0.
To write a '1' to this bit, the MSTEN value = 1 must first be written.

bit 8 **CKE:** SPI Clock Edge Select bit[3]
1 = Serial output data changes on transition from active clock state to Idle clock state (see the CKP bit)
0 = Serial output data changes on transition from Idle clock state to active clock state (see the CKP bit)
bit 7 **SSEN:** Slave Select Enable (Slave mode) bit
1 = SSx pin used for Slave mode
0 = SSx pin not used for Slave mode, pin controlled by port function.
bit 6 **CKP:** Clock Polarity Select bit[4]
1 = Idle state for clock is a high level; active state is a low level
0 = Idle state for clock is a low level; active state is a high level

Note 1: When using the 1:1 PBCLK divisor, the user's software should not read or write the peripheral's SFRs in the SYSCLK cycle immediately following the instruction that clears the module's ON bit.
2: This bit can only be written when the ON bit = 0.
3: This bit is not used in the Framed SPI mode. The user should program this bit to '0' for the Framed SPI mode (FRMEN = 1).
4: When AUDEN = 1, the SPI module functions as if the CKP bit is equal to '1', regardless of the actual value of CKP.

bit 5 **MSTEN:** Master Mode Enable bit
1 = Master mode
0 = Slave mode
bit 4 **DISSDI:** Disable SDI bit
1 = SDI pin is not used by the SPI module (pin is controlled by PORT function)
0 = SDI pin is controlled by the SPI module
bit 3-2 **STXISEL<1:0>:** SPI Transmit Buffer Empty Interrupt Mode bits
11 = Interrupt is generated when the buffer is not full (has one or more empty elements)
10 = Interrupt is generated when the buffer is empty by one-half or more
01 = Interrupt is generated when the buffer is completely empty
00 = Interrupt is generated when the last transfer is shifted out of SPISR and transmit operations are complete
bit 1-0 **SRXISEL<1:0>:** SPI Receive Buffer Full Interrupt Mode bits
11 = Interrupt is generated when the buffer is full
10 = Interrupt is generated when the buffer is full by one-half or more
01 = Interrupt is generated when the buffer is not empty
00 = Interrupt is generated when the last word in the receive buffer is read (i.e., buffer is empty)

Note 1: When using the 1:1 PBCLK divisor, the user's software should not read or write the peripheral's SFRs in the SYSCLK cycle immediately following the instruction that clears the module's ON bit.
2: This bit can only be written when the ON bit = 0.
3: This bit is not used in the Framed SPI mode. The user should program this bit to '0' for the Framed SPI mode (FRMEN = 1).
4: When AUDEN = 1, the SPI module functions as if the CKP bit is equal to '1', regardless of the actual value of CKP.

REGISTER 17-2: SPIxCON2: SPI CONTROL REGISTER 2

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —
23:16	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —	U-0 —
15:8	R/W-0 SPISGNEXT	U-0 —	U-0 —	R/W-0 FRMERREN	R/W-0 SPIROVEN	R/W-0 SPITUREN	R/W-0 IGNROV	R/W-0 IGNTUR
7:0	R/W-0 AUDEN[1]	U-0 —	U-0 —	U-0 —	R/W-0 AUDMONO[1,2]	U-0 —	R/W-0 AUDMOD<1:0>[1,2]	R/W-0

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

bit 31-16 **Unimplemented:** Read as '0'
bit 15 **SPISGNEXT:** Sign Extend Read Data from the RX FIFO bit
1 = Data from RX FIFO is sign extended
0 = Data from RX FIFO is not sign extended
bit 14-13 **Unimplemented:** Read as '0'
bit 12 **FRMERREN:** Enable Interrupt Events via FRMERR bit
1 = Frame Error overflow generates error events
0 = Frame Error does not generate error events
bit 11 **SPIROVEN:** Enable Interrupt Events via SPIROV bit
1 = Receive overflow generates error events
0 = Receive overflow does not generate error events
bit 10 **SPITUREN:** Enable Interrupt Events via SPITUR bit
1 = Transmit underrun generates error events
0 = Transmit underrun does not generate error events
bit 9 **IGNROV:** Ignore Receive Overflow bit (for Audio Data Transmissions)
1 = A ROV is not a critical error; during ROV data in the FIFO is not overwritten by receive data
0 = A ROV is a critical error that stops SPI operation
bit 8 **IGNTUR:** Ignore Transmit Underrun bit (for Audio Data Transmissions)
1 = A TUR is not a critical error and zeros are transmitted until the SPIxTXB is not empty
0 = A TUR is a critical error that stops SPI operation
bit 7 **AUDEN:** Enable Audio CODEC Support bit[1]
1 = Audio protocol enabled
0 = Audio protocol disabled
bit 6-5 **Unimplemented:** Read as '0'
bit 3 **AUDMONO:** Transmit Audio Data Format bit[1,2]
1 = Audio data is mono (Each data word is transmitted on both left and right channels)
0 = Audio data is stereo
bit 2 **Unimplemented:** Read as '0'
bit 1-0 **AUDMOD<1:0>:** Audio Protocol Mode bit[1,2]
11 = PCM/DSP mode
10 = Right-Justified mode
01 = Left-Justified mode
00 = I²S mode

Note 1: This bit can only be written when the ON bit = 0.
2: This bit is only valid for AUDEN = 1.

Equation 23-1:

$$F_{SCK} = \frac{F_{PB}}{2 \cdot (SPIxBRG + 1)}$$

- **SPIxBRG<0:8>** - Baud rate register

REGISTER 17-3: SPIxSTAT: SPI STATUS REGISTER

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	R-0	R-0	R-0	R-0	R-0
	—	—	—	RXBUFELM<4:0>				
23:16	U-0	U-0	U-0	R-0	R-0	R-0	R-0	R-0
	—	—	—	TXBUFELM<4:0>				
15:8	U-0	U-0	U-0	R/C-0, HS	R-0	U-0	U-0	R-0
	—	—	—	FRMERR	SPIBUSY	—	—	SPITUR
7:0	R-0	R/W-0	R-0	U-0	R-1	U-0	R-0	R-0
	SRMT	SPIROV	SPIRBE	—	SPITBE	—	SPITBF	SPIRBF

Legend: C = Clearable bit; HS = Set in hardware; 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

bit 31-29 **Unimplemented:** Read as '0'
bit 28-24 **RXBUFELM<4:0>:** Receive Buffer Element Count bits (valid only when ENHBUF = 1)
bit 23-21 **Unimplemented:** Read as '0'
bit 20-16 **TXBUFELM<4:0>:** Transmit Buffer Element Count bits (valid only when ENHBUF = 1)
bit 15-13 **Unimplemented:** Read as '0'
bit 12 **FRMERR:** SPI Frame Error status bit
1 = Frame error detected
0 = No Frame error detected
This bit is only valid when FRMEN = 1.
bit 11 **SPIBUSY:** SPI Activity Status bit
1 = SPI peripheral is currently busy with some transactions
0 = SPI peripheral is currently idle
bit 10-9 **Unimplemented:** Read as '0'
bit 8 **SPITUR:** Transmit Under Run bit
1 = Transmit buffer has encountered an underrun condition
0 = Transmit buffer has no underrun condition
This bit is only valid in Framed Sync mode; the underrun condition must be cleared by disabling (ON bit = 0) and re-enabling (ON bit = 1) the module, or writing a '0' to SPITUR.
bit 7 **SRMT:** Shift Register Empty bit (valid only when ENHBUF = 1)
1 = When SPI module shift register is empty
0 = When SPI module shift register is not empty
bit 6 **SPIROV:** Receive Overflow Flag bit
1 = A new data is completely received and discarded. The user software has not read the previous data in the SPIxBUF register.
0 = No overflow has occurred
This bit is set in hardware; can bit only be cleared by disabling (ON bit = 0) and re-enabling (ON bit = 1) the module, or by writing a '0' to SPIROV.
bit 5 **SPIRBE:** RX FIFO Empty bit (valid only when ENHBUF = 1)
1 = RX FIFO is empty (CRPTR = SWPTR)
0 = RX FIFO is not empty (CRPTR ≠ SWPTR)
bit 4 **Unimplemented:** Read as '0'
bit 3 **SPITBE:** SPI Transmit Buffer Empty Status bit
1 = Transmit buffer, SPIxTXB is empty
0 = Transmit buffer, SPIxTXB is not empty
Automatically set in hardware when SPI transfers data from SPIxTXB to SPIxSR.
Automatically cleared in hardware when SPIxBUF is written to, loading SPIxTXB.
bit 2 **Unimplemented:** Read as '0'
bit 1 **SPITBF:** SPI Transmit Buffer Full Status bit
1 = Transmit not yet started, SPITXB is full
0 = Transmit buffer is not full
Standard Buffer Mode:
Automatically set in hardware when the core writes to the SPIBUF location, loading SPITXB.
Automatically cleared in hardware when the SPI module transfers data from SPITXB to SPISR.
Enhanced Buffer Mode:
Set when CWPTR + 1 = SRPTR; cleared otherwise
bit 0 **SPIRBF:** SPI Receive Buffer Full Status bit
1 = Receive buffer, SPIxRXB is full
0 = Receive buffer, SPIxRXB is not full
Standard Buffer Mode:
Automatically set in hardware when the SPI module transfers data from SPIxSR to SPIxRXB.
Automatically cleared in hardware when SPIxBUF is read from, reading SPIxRXB.
Enhanced Buffer Mode:
Set when SWPTR + 1 = CRPTR; cleared otherwise

PIC32 has 3 comparators
CVRCON

bit 31-16 **Unimplemented:** Read as '0'
bit 15 **ON:** Comparator Voltage Reference On bit[1]
1 = Module is enabled
Setting this bit does not affect other bits in the register.
0 = Module is disabled and does not consume current.
Clearing this bit does not affect the other bits in the register.
bit 14-7 **Unimplemented:** Read as '0'
bit 6 **CVROE:** CVREFOUT Enable bit
1 = Voltage level is output on CVREFOUT pin
0 = Voltage level is disconnected from CVREFOUT pin
bit 5 **CVRR:** CVREF Range Selection bit
1 = 0 to 0.67 CVRSRC, with CVRSRC/24 step size
0 = 0.25 CVRSRC to 0.75 CVRSRC, with CVRSRC/32 step size
bit 4 **CVRSS:** CVREF Source Selection bit
1 = Comparator voltage reference source, CVRSRC = (VREF+) – (VREF-)
0 = Comparator voltage reference source, CVRSRC = AVDD – AVSS
bit 3-0 **CVR<3:0>:** CVREF Value Selection 0 ≤ CVR<3:0> ≤ 15 bits
When CVRR = 1:
CVREF = (CVR<3:0>/24) • (CVRSRC)
When CVRR = 0:
CVREF = 1/4 • (CVRSRC) + (CVR<3:0>/32) • (CVRSRC)

Note 1: When using 1:1 PBCLK divisor, the user's software should not read/write the peripheral's SFRs in th SYSCLK cycle immediately following the instruction that clears the module's ON bit.

REGISTER 23-1: CMXCON: COMPARATOR CONTROL REGISTER

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	R-0
	ON[1]	COE	CPOL[2]	—	—	—	—	COUT
7:0	R/W-1	R/W-1	U-0	R/W-0	U-0	U-0	R/W-1	R/W-1
	EVPOL<1:0>		—	CREF	—	—	CCH<1:0>	

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

bit 31-16 **Unimplemented:** Read as '0'
bit 15 **ON:** Comparator ON bit[1]
1 = Module is enabled. Setting this bit does not affect the other bits in this register
0 = Module is disabled and does not consume current. Clearing this bit does not affect the other bits in this register
bit 14 **COE:** Comparator Output Enable bit
1 = Comparator output is driven on the output CxOUT pin
0 = Comparator output is not driven on the output CxOUT pin
bit 13 **CPOL:** Comparator Output Inversion bit[2]
1 = Output is inverted
0 = Output is not inverted
bit 12-9 **Unimplemented:** Read as '0'
bit 8 **COUT:** Comparator Output bit
1 = Output of the Comparator is a '1'
0 = Output of the Comparator is a '0'
bit 7-6 **EVPOL<1:0>:** Interrupt Event Polarity Select bits
11 = Comparator interrupt is generated on a low-to-high or high-to-low transition of the comparator output
10 = Comparator interrupt is generated on a high-to-low transition of the comparator output
01 = Comparator interrupt is generated on a low-to-high transition of the comparator output
00 = Comparator interrupt generation is disabled
bit 5 **Unimplemented:** Read as '0'
bit 4 **CREF:** Comparator Positive Input Configure bit
1 = Comparator non-inverting input is connected to the internal CVREF
0 = Comparator non-inverting input is connected to the CxINA pin
bit 3-2 **Unimplemented:** Read as '0'
bit 1-0 **CCH<1:0>:** Comparator Negative Input Select bits for Comparator
11 = Comparator inverting input is connected to the IVREF
10 = Comparator inverting input is connected to the CxIND pin
01 = Comparator inverting input is connected to the CxINC pin
00 = Comparator inverting input is connected to the CxINB pin

Note 1: When using the 1:1 PBCLK divisor, the user's software should not read/write the peripheral's SFRs in the SYSCLK cycle immediately following the instruction that clears the module's ON bit.
2: Setting this bit will invert the signal to the comparator interrupt generator as well. This will result in an interrupt being generated on the opposite edge from the one selected by EVPOL<1:0>.

D to A:

$$\textbf{Output} = K\, V_{REF} \frac{\text{Code}}{2^N}\ \textbf{Volts}$$

$$\textbf{Resolution} = \Delta = K\, V_{REF} \frac{1}{2^N}$$

$$\textbf{Maximum Output Voltage} = K\, V_{REF} \frac{2^{N-1}}{2^N} = K\, V_{REF} - \Delta$$

Zero Error – error when input = 0
Full-Scale Error - error when input is the maximum
Relative Accuracy Error – error between actual and the line from output_max and output_min.
Monotonicity - output always increasing as input increases
Settling Time - The required for the output to reach with a D/2 of the stable output when the input changes from 0 to 2^N-1, or 2^N-1 to 0.
A-to-D:
Offset Error - Error of analog transition point nearest 0 Volts.
Full Scale (Gain) Error - Error of analog transition point farthest from 0 Volts.
Differential Nonlinearity Error - The maximum difference between a step D and the constant ideal D.
DNLE = Maximum $|D_k - D|$

A *Successive Approximation Register* (**SAR**) is an ADC where each output bit is determined sequentially, starting with the most significant bit (MSB) working down to the least significant bit (LSB) in a binary search algorithm.
Dual-Ramp ADCs use an RC Operational Amplifier Circuit coupled with a timer to measure the value of an input.

- Slower but more precise because it needs to charge and discharge a capacitor.

PIC32 A-to-D:

- 10 bit, up to 16 analog inputs, internal VREF, multiple channel scan, 16-word buffer, 8 conversion formats and can run during idle/sleep mode.

REGISTER 22-1: AD1CON1: ADC CONTROL REGISTER 1

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	U-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
	ON[1]	—	SIDL	—	—	FORM<2:0>		
7:0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0, HSC	R/C-0, HSC
	SSRC<2:0>			CLRASAM	—	ASAM	SAMP[2]	DONE[3]

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

bit 31-16 **Unimplemented:** Read as '0'
bit 15 **ON:** ADC Operating Mode bit[1]
1 = ADC module is operating
0 = ADC module is not operating
bit 14 **Unimplemented:** Read as '0'
bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation when the device enters Idle mode
bit 12-11 **Unimplemented:** Read as '0'
bit 10-8 **FORM<2:0>:** Data Output Format bits
111 = Signed Fractional 32-bit (DOUT = sddd dddd dd00 0000 0000 0000 0000 0000)
110 = Fractional 32-bit (DOUT = dddd dddd dd00 0000 0000 0000 0000 0000)
101 = Signed Integer 32-bit (DOUT = ssss ssss ssss ssss ssss sssd dddd dddd)
100 = Integer 32-bit (DOUT = 0000 0000 0000 0000 0000 00dd dddd dddd)
011 = Signed Fractional 16-bit (DOUT = 0000 0000 0000 0000 sddd dddd dd00 0000)
010 = Fractional 16-bit (DOUT = 0000 0000 0000 0000 dddd dddd dd00 0000)
001 = Signed Integer 16-bit (DOUT = 0000 0000 0000 0000 ssss sssd dddd dddd)
000 = Integer 16-bit (DOUT = 0000 0000 0000 0000 0000 00dd dddd dddd)
bit 7-5 **SSRC<2:0>:** Conversion Trigger Source Select bits
111 = Internal counter ends sampling and starts conversion (auto convert)
110 = Reserved
101 = Reserved
100 = Reserved
011 = CTMU ends sampling and starts conversion
010 = Timer 3 period match ends sampling and starts conversion
001 = Active transition on INT0 pin ends sampling and starts conversion
000 = Clearing SAMP bit ends sampling and starts conversion

REGISTER 22-1: AD1CON1: ADC CONTROL REGISTER 1 (CONTINUED)

bit 4 **CLRASAM:** Stop Conversion Sequence bit (when the first ADC interrupt is generated)
1 = Stop conversions when the first ADC interrupt is generated. Hardware clears the ASAM bit when the ADC interrupt is generated.
0 = Normal operation, buffer contents will be overwritten by the next conversion sequence
bit 3 **Unimplemented:** Read as '0'
bit 2 **ASAM:** ADC Sample Auto-Start bit
1 = Sampling begins immediately after last conversion completes; SAMP bit is automatically set.
0 = Sampling begins when SAMP bit is set
bit 1 **SAMP:** ADC Sample Enable bit[2]
1 = The ADC sample and hold amplifier is sampling
0 = The ADC sample/hold amplifier is holding
When ASAM = 0, writing '1' to this bit starts sampling.
When SSRC = 000, writing '0' to this bit will end sampling and start conversion.
bit 0 **DONE:** Analog-to-Digital Conversion Status bit[3]
1 = Analog-to-digital conversion is done
0 = Analog-to-digital conversion is not done or has not started
Clearing this bit will not affect any operation in progress.

Note 1: When using 1:1 PBCLK divisor, the user's software should not read/write the peripheral's SFRs in the SYSCLK cycle immediately following the instruction that clears the module's ON bit.
2: If ASAM = 0, software can write a '1' to start sampling. This bit is automatically set by hardware if ASAM = 1. If SSRC = 0, software can write a '0' to end sampling and start conversion. If SSRC ≠ '0', this bit is automatically cleared by hardware to end sampling and start conversion.
3: This bit is automatically set by hardware when analog-to-digital conversion is complete. Software can write a '0' to clear this bit (a write of '1' is not allowed). Clearing this bit does not affect any operation already in progress. This bit is automatically cleared by hardware at the start of a new conversion.

- Signed voltages: Vad = (# / 1024) * VREF+
  - (VREF+ - VREF-) / 2 + Vad
  - For fractions add 1 if POSITIVE

REGISTER 22-2: AD1CON2: ADC CONTROL REGISTER 2

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0	U-0
	VCFG<2:0>			OFFCAL	—	CSCNA	—	—
7:0	R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	BUFS	—	SMPI<3:0>				BUFM	ALTS

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

bit 31-16 **Unimplemented:** Read as '0'
bit 15-13 **VCFG<2:0>:** Voltage Reference Configuration bits

	VREFH	VREFL
000	AVDD	AVSS
001	External VREF+ pin	AVSS
010	AVDD	External VREF- pin
011	External VREF+ pin	External VREF- pin
1xx	AVDD	AVSS

bit 12 **OFFCAL:** Input Offset Calibration Mode Select bit
1 = Enable Offset Calibration mode
Positive and negative inputs of the sample and hold amplifier are connected to VREFL
0 = Disable Offset Calibration mode
The inputs to the sample and hold amplifier are controlled by AD1CHS or AD1CSSL
bit 11 **Unimplemented:** Read as '0'
bit 10 **CSCNA:** Input Scan Select bit
1 = Scan inputs
0 = Do not scan inputs
bit 9-8 **Unimplemented:** Read as '0'
bit 7 **BUFS:** Buffer Fill Status bit
Only valid when BUFM = 1.
1 = ADC is currently filling buffer 0x8-0xF, user should access data in 0x0-0x7
0 = ADC is currently filling buffer 0x0-0x7, user should access data in 0x8-0xF
bit 6 **Unimplemented:** Read as '0'
bit 5-2 **SMPI<3:0>:** Sample/Convert Sequences Per Interrupt Selection bits
1111 = Interrupts at the completion of conversion for each 16th sample/convert sequence
1110 = Interrupts at the completion of conversion for each 15th sample/convert sequence
•
•
•
0001 = Interrupts at the completion of conversion for each 2nd sample/convert sequence
0000 = Interrupts at the completion of conversion for each sample/convert sequence
bit 1 **BUFM:** ADC Result Buffer Mode Select bit
1 = Buffer configured as two 8-word buffers, ADC1BUF7-ADC1BUF0, ADC1BUFF-ADCBUF8
0 = Buffer configured as one 16-word buffer ADC1BUFF-ADC1BUF0
bit 0 **ALTS:** Alternate Input Sample Mode Select bit
1 = Uses Sample A input multiplexer settings for first sample, then alternates between Sample B and Sample A input multiplexer settings for all subsequent samples
0 = Always use Sample A input multiplexer settings

REGISTER 22-3: AD1CON3: ADC CONTROL REGISTER 3

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADRC	—	—	SAMC<4:0>[1]				
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W	R/W-0
	ADCS<7:0>[2]							

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

bit 31-16 **Unimplemented:** Read as '0'
bit 15 **ADRC:** ADC Conversion Clock Source bit
1 = Clock derived from FRC
0 = Clock derived from Peripheral Bus Clock (PBCLK)
bit 14-13 **Unimplemented:** Read as '0'
bit 12-8 **SAMC<4:0>:** Auto-Sample Time bits[1]
11111 = 31 TAD
•
•
•
00001 = 1 TAD
00000 = 0 TAD (Not allowed)
bit 7-0 **ADCS<7:0>:** ADC Conversion Clock Select bits[2]
11111111 = TPB • 2 • (ADCS<7:0> + 1) = 512 • TPB = TAD
•
•
•
00000001 = TPB • 2 • (ADCS<7:0> + 1) = 4 • TPB = TAD
00000000 = TPB • 2 • (ADCS<7:0> + 1) = 2 • TPB = TAD

Note 1: This bit is only used if the SSRC<2:0> bits (AD1CON1<7:5>) = 111.
2: This bit is not used if the ADRC (AD1CON3<15>) bit = 1.

REGISTER 22-4: AD1CHS: ADC INPUT SELECT REGISTER

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	CH0NB	—	—	—	CH0SB<3:0>			
23:16	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	CH0NA	—	—	—	CH0SA<3:0>			
15:8	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
7:0	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—

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

bit 31 **CH0NB:** Negative Input Select bit for Sample B
1 = Channel 0 negative input is AN1
0 = Channel 0 negative input is VREFL
bit 30-28 **Unimplemented:** Read as '0'
bit 27-24 **CH0SB<3:0>:** Positive Input Select bits for Sample B
1111 = Channel 0 positive input is Open[1]
1110 = Channel 0 positive input is IVREF[2]
1101 = Channel 0 positive input is CTMU temperature sensor (CTMUT)[3]
1100 = Channel 0 positive input is AN12[4]
•
•
•
0001 = Channel 0 positive input is AN1
0000 = Channel 0 positive input is AN0
bit 23 **CH0NA:** Negative Input Select bit for Sample A Multiplexer Setting[2]
1 = Channel 0 negative input is AN1
0 = Channel 0 negative input is VREFL
bit 22-20 **Unimplemented:** Read as '0'
bit 19-16 **CH0SA<3:0>:** Positive Input Select bits for Sample A Multiplexer Setting
1111 = Channel 0 positive input is Open[1]
1110 = Channel 0 positive input is IVREF[2]
1101 = Channel 0 positive input is CTMU temperature (CTMUT)[3]
1100 = Channel 0 positive input is AN12[4]
•
•
•
0001 = Channel 0 positive input is AN1
0000 = Channel 0 positive input is AN0
bit 15-0 **Unimplemented:** Read as '0'

Note 1: This selection is only used with CTMU capacitive and time measurement.
2: See **Section 24.0 "Comparator Voltage Reference (CVREF)"** for more information.
3: See **Section 25.0 "Charge Time Measurement Unit (CTMU)"** for more information.
4: AN12 is only available on 44-pin devices. AN6-AN8 are not available on 28-pin devices.

REGISTER 22-5: AD1CSSL: ADC INPUT SCAN SELECT REGISTER

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0

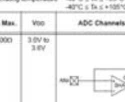
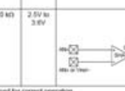
Legend: R = Readable bit; W = Writable bit; U = Unimplemented bit, read as '0'; -n = Value at POR; '1' = Bit is set; '0' = Bit is cleared; x = Bit is unknown

bit 31-16 **Unimplemented:** Read as '0'
bit 15-0 **CSSL<15:0>:** ADC Input Pin Scan Selection bits[1,2]
1 = Select ANx for input scan
0 = Skip ANx for input scan

Note 1: CSSL = ANx, where 'x' = 0-12; CSSL13 selects CTMU input for scan; CSSL14 selects IVREF for scan; CSSL15 selects VSS for scan.
2: On devices with less than 13 analog inputs, all CSSLx bits can be selected; however, inputs selected for scan without a corresponding input on the device will convert to VREFL.

TABLE 30-35: 10-BIT CONVERSION RATE PARAMETERS

AC CHARACTERISTICS[2] — Standard Operating Conditions (see Note 3): 2.5V to 3.6V (unless otherwise stated). Operating temperature -40°C ≤ TA ≤ +85°C for Industrial; -40°C ≤ TA ≤ +105°C for V-temp

ADC Speed	TAD Min.	Sampling Time Min.	Rs Max.	VDD	ADC Channels Configuration
1 Msps to 400 ksps[1]	65 ns	132 ns	500Ω	3.0V to 3.6V	
Up to 400 ksps	200 ns	200 ns	5.0 kΩ	2.5V to 3.6V	

Note 1: External VREF- and VREF+ pins must be used for correct operation.
2: These parameters are characterized, but not tested in manufacturing.
3: The ADC module is functional at VBORMIN < VDD < 2.5V, but with degraded performance. Unless otherwise stated, module functionality is tested, but not characterized.

TABLE 30-36: ANALOG-TO-DIGITAL CONVERSION TIMING REQUIREMENTS

AC CHARACTERISTICS — Standard Operating Conditions (see Note 4): 2.5V to 3.6V (unless otherwise stated). Operating temperature -40°C ≤ TA ≤ +85°C for Industrial; -40°C ≤ TA ≤ +105°C for V-temp

Param. No.	Symbol	Characteristics	Min.	Typical[1]	Max.	Units	Conditions
Clock Parameters							
AD50	TAD	ADC Clock Period[2]	65	—	—	ns	See Table 30-35
Conversion Rate							
AD55	TCONV	Conversion Time	—	12 TAD	—	—	—
AD56	FCNV	Throughput Rate (Sampling Speed)	—	—	1000	ksps	AVDD = 3.0V to 3.6V
			—	—	400	ksps	AVDD = 2.5V to 3.6V
AD57	TSAMP	Sample Time	1 TAD	—	—	—	TSAMP must be ≥ 132 ns
Timing Parameters							
AD60	TPCS	Conversion Start from Sample Trigger[3]	—	1.0 TAD	—	—	Auto-Convert Trigger (SSRC<2:0> = 111) not selected
AD61	TPSS	Sample Start from Setting Sample (SAMP) bit	0.5 TAD	—	1.5 TAD	—	—
AD62	TCSS	Conversion Completion to Sample Start (ASAM = 1)[3]	—	0.5 TAD	—	—	—
AD63	TDPU	Time to Stabilize Analog Stage from ADC Off to ADC On[3]	—	—	2	μs	—

Note 1: These parameters are characterized, but not tested in manufacturing.
2: Because the sample caps will eventually lose charge, clock rates below 10 kHz can affect linearity performance, especially at elevated temperatures.
3: Characterized by design but not tested.
4: The ADC module is functional at VBORMIN < VDD < 2.5V, but with degraded performance. Unless otherwise stated, module functionality is tested, but not characterized.

Equation 17-1: ADC Conversion Clock Period

$$T_{AD} = 2 \bullet (T_{PB} \bullet (ADCS + 1))$$

$$ADCS = \left(\frac{T_{AD}}{2 \bullet T_{PB}}\right) - 1$$

Equation 17-2: Available Sampling Time, Sequential Sampling

$$T_{SMP} = TriggerPulseInterval(T_{SEQ}) - ConversionTime(T_{CONV})$$

$$T_{SMP} = T_{SEQ} - T_{CONV}$$

Note: TSEQ is the trigger pulse interval time.

UARTS:
Overrun Error – A new character has arrived before previous character has been read.
Underrun Condition – The UART's transmit buffer is empty while communication has been taking place.
Framing Error – Start and/or stop bits are in error. That is, the line is not low where a start bit should be, or it is not high where the stop bit(s) should be located.
Parity Error – The parity bit (if present) is not the value should be for the transmitted data bits.
Break Condition – A certain number of consecutive 0's (generally more than a complete character) have been received. Can be sent intentionally to tell the receiver communication is finished.
Baud Rate: Symbols per second

REGISTER 19-1: UxMODE: UARTx MODE REGISTER

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
	ON[1]	—	SIDL	IREN	RTSMD	—	UEN<1:0>	
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	WAKE	LPBACK	ABAUD	RXINV	BRGH	PDSEL<1:0>		STSEL

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

bit 31-16 **Unimplemented:** Read as '0'
bit 15 **ON:** UARTx Enable bit[1]
1 = UARTx is enabled. UARTx pins are controlled by UARTx as defined by UEN<1:0> and UTXEN control bits
0 = UARTx is disabled. All UARTx pins are controlled by corresponding bits in the PORTx, TRISx and LATx registers; UARTx power consumption is minimal
bit 14 **Unimplemented:** Read as '0'
bit 13 **SIDL:** Stop in Idle Mode bit
1 = Discontinue operation when device enters Idle mode
0 = Continue operation in Idle mode
bit 12 **IREN:** IrDA® Encoder and Decoder Enable bit
1 = IrDA is enabled
0 = IrDA is disabled
bit 11 **RTSMD:** Mode Selection for UxRTS Pin bit[2]
1 = UxRTS pin is in Simplex mode
0 = UxRTS pin is in Flow Control mode
bit 10 **Unimplemented:** Read as '0'
bit 9-8 **UEN<1:0>:** UARTx Enable bits[2]
11 = UxTX, UxRX and UxBCLK pins are enabled and used; UxCTS pin is controlled by corresponding bits in the PORTx register
10 = UxTX, UxRX, UxCTS and UxRTS pins are enabled and used
01 = UxTX, UxRX and UxRTS pins are enabled and used; UxCTS pin is controlled by corresponding bits in the PORTx register
00 = UxTX and UxRX pins are enabled and used; UxCTS and UxRTS/UxBCLK pins are controlled by corresponding bits in the PORTx register
bit 7 **WAKE:** Enable Wake-up on Start bit Detect During Sleep Mode bit
1 = Wake-up enabled
0 = Wake-up disabled
bit 6 **LPBACK:** UARTx Loopback Mode Select bit
1 = Loopback mode is enabled
0 = Loopback mode is disabled

Note 1: When using the 1:1 PBCLK divisor, the user software should not read/write the peripheral SFRs in the SYSCLK cycle immediately following the instruction that clears the module's ON bit.
2: These bits are not available in some UART modules. Refer to the **"UART"** chapter in the specific device data sheet for more information on availability of these bits in different UART modules.

Register 21-1: UxMODE: UARTx Mode Register (Continued)

bit 5 **ABAUD:** Auto-Baud Enable bit
1 = Enable baud rate measurement on the next character – requires reception of Sync character (0x55); cleared by hardware upon completion
0 = Baud rate measurement disabled or completed
bit 4 **RXINV:** Receive Polarity Inversion bit
1 = UxRX Idle state is '0'
0 = UxRX Idle state is '1'
bit 3 **BRGH:** High Baud Rate Enable bit
1 = High-Speed mode – 4x baud clock enabled
0 = Standard Speed mode – 16x baud clock enabled
bit 2-1 **PDSEL<1:0>:** Parity and Data Selection bits
11 = 9-bit data, no parity
10 = 8-bit data, odd parity
01 = 8-bit data, even parity
00 = 8-bit data, no parity
bit 0 **STSEL:** Stop Selection bit
1 = 2 Stop bits
0 = 1 Stop bit

REGISTER 19-2: UxSTA: UARTx STATUS AND CONTROL REGISTER

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
	—	—	—	—	—	—	—	ADM_EN
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	ADDR<7:0>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-1
	UTXISEL<1:0>		UTXINV	URXEN	UTXBRK	UTXEN	UTXBF	TRMT
7:0	R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/W-0	R-0
	URXISEL<1:0>		ADDEN	RIDLE	PERR	FERR	OERR	URXDA

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

bit 31-25 **Unimplemented:** Read as '0'
bit 24 **ADM_EN:** Automatic Address Detect Mode Enable bit
1 = Automatic Address Detect mode is enabled
0 = Automatic Address Detect mode is disabled
bit 23-16 **ADDR<7:0>:** Automatic Address Mask bits
When the ADM_EN bit is '1', this value defines the address character to use for automatic address detection.
bit 15-14 **UTXISEL<1:0>:** TX Interrupt Mode Selection bits
11 = Reserved, do not use
10 = Interrupt is generated and asserted while the transmit buffer is empty
01 = Interrupt is generated and asserted when all characters have been transmitted
00 = Interrupt is generated and asserted while the transmit buffer contains at least one empty space
bit 13 **UTXINV:** Transmit Polarity Inversion bit
If IrDA mode is disabled (i.e., IREN (UxMODE<12>) is '0'):
1 = UxTX Idle state is '0'
0 = UxTX Idle state is '1'
If IrDA mode is enabled (i.e., IREN (UxMODE<12>) is '1'):
1 = IrDA encoded UxTX Idle state is '1'
0 = IrDA encoded UxTX Idle state is '0'
bit 12 **URXEN:** Receiver Enable bit
1 = UARTx receiver is enabled. UxRX pin is controlled by UARTx (if ON = 1)
0 = UARTx receiver is disabled. UxRX pin is ignored by the UARTx module. UxRX pin is controlled by port.
bit 11 **UTXBRK:** Transmit Break bit
1 = Send Break on next transmission. Start bit followed by twelve '0' bits, followed by Stop bit; cleared by hardware upon completion
0 = Break transmission is disabled or completed
bit 10 **UTXEN:** Transmit Enable bit
1 = UARTx transmitter is enabled. UxTX pin is controlled by UARTx (if ON = 1).
0 = UARTx transmitter is disabled. Any pending transmission is aborted and buffer is reset. UxTX pin is controlled by port.
bit 9 **UTXBF:** Transmit Buffer Full Status bit (read-only)
1 = Transmit buffer is full
0 = Transmit buffer is not full, at least one more character can be written
bit 8 **TRMT:** Transmit Shift Register is Empty bit (read-only)
1 = Transmit shift register is empty and transmit buffer is empty (the last transmission has completed)
0 = Transmit shift register is not empty, a transmission is in progress or queued in the transmit buffer

REGISTER 19-2: UxSTA: UARTx STATUS AND CONTROL REGISTER (CONTINUED)

bit 7-6 **URXISEL<1:0>:** Receive Interrupt Mode Selection bit
11 = Reserved; do not use
10 = Interrupt flag bit is asserted while receive buffer is 3/4 or more full (i.e., has 6 or more data characters)
01 = Interrupt flag bit is asserted while receive buffer is 1/2 or more full (i.e., has 4 or more data characters)
00 = Interrupt flag bit is asserted while receive buffer is not empty (i.e., has at least 1 data character)
bit 5 **ADDEN:** Address Character Detect bit (bit 8 of received data = 1)
1 = Address Detect mode is enabled. If 9-bit mode is not selected, this control bit has no effect.
0 = Address Detect mode is disabled
bit 4 **RIDLE:** Receiver Idle bit (read-only)
1 = Receiver is Idle
0 = Data is being received
bit 3 **PERR:** Parity Error Status bit (read-only)
1 = Parity error has been detected for the current character
0 = Parity error has not been detected
bit 2 **FERR:** Framing Error Status bit (read-only)
1 = Framing error has been detected for the current character
0 = Framing error has not been detected
bit 1 **OERR:** Receive Buffer Overrun Error Status bit.
This bit is set in hardware and can only be cleared (= 0) in software. Clearing a previously set OERR bit resets the receiver buffer and the RSR to an empty state.
1 = Receive buffer has overflowed
0 = Receive buffer has not overflowed
bit 0 **URXDA:** Receive Buffer Data Available bit (read-only)
1 = Receive buffer has data, at least one more character can be read
0 = Receive buffer is empty

Register 21-5: UxBRG: UARTx Baud Rate Register

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	BRG<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	BRG<7:0>							

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

bit 31-16 **Unimplemented:** Read as '0'
bit 15-0 **BRG<15:0>:** Baud Rate Divisor bits

Equation 21-1: UART Baud Rate with BRGH = 0

$$Baud\ Rate = \frac{F_{PB}}{16 \cdot (UxBRG + 1)}$$

$$UxBRG = \frac{F_{PB}}{16 \cdot Baud\ Rate} - 1$$

Note: F_{PB} denotes the PBCLK frequency.

If BRGH = 1 then replace 16 with 4 ^.

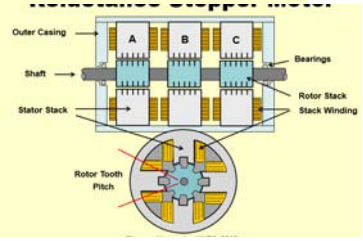
Stepper Motors:
- Unlike many motors, they have full torque at stand-still and have excellent response to starting, stopping, and reversing.
- They don't need brushes for commutation, so they have a longer motor life depending primarily on bearing and coil life.
- They can achieve very low speeds and only require open-loop control
- Furthermore, their speed is determined by the period of these pulses, not by their voltage.
- **Disadvantages:**
Resonance can be a problem if not controlled.
Do not perform as well for high-speed applications.
Stepping Modes:
Wave Drive Mode: one by one A – B – A' – B'
Full Step Drive: 2 at a time AB – BA' – A'B' – B'A
Half Step Drive: combo of above two AB-A-BA'-A'-A'B'-B'-B'A-A
Micro Stepping: smaller step sizes
Stator – Stationary (generally) part of the motor held in place by the outer casing of the motor.
Rotor – Inner part that rotates in response to changes in the activation pattern. It is supported at each end by bearings and includes a projecting shaft for the connection of the load.
Stator Teeth – Radial projections on the stator which provide a more precise path for the magnetic flux to flow between the stator and the rotor and produce the motor's movement.
Rotor Teeth – Radial projections on the rotor which are also used to provide a path for the magnetic field that produces the motor's movement.
Multi-Stack Variable Reluctance: rotor teeth lose to (not equal) stator is multiple of phases.

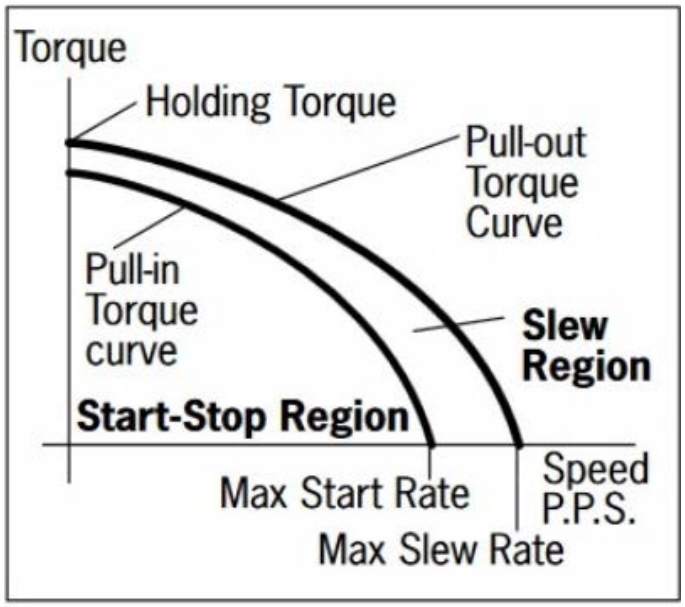
$$\text{Step Size} = \frac{360^\circ}{\Phi t} \quad \text{where} \quad \Phi = \text{Number of Phases}, \; t = \text{Number of Rotor Teeth}$$

$$\text{Tooth Pitch} = {}^{360^\circ}/_{12} = 30^\circ$$

12 = rotor teeth
Hybrid: Rotor teeth > Stator Teeth (still multiple of poles)

$$\text{Step size} = \frac{360^\circ\,(RT-ST)}{(RT \times ST)}$$

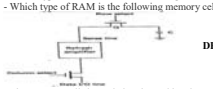




ON SCRATCH PAPER WRITE BINARY 0 to F!
- An **embedded system** is an electronic system controlled by one or more computers/microprocessors that are internal to the system. That is, the system is not thought of as a computer, but something else.
- An example of a system that needs to be a "realtime" system and what the functionality of it is in real time: **Air traffic control system, the current location.**
- A **microcontroller** is a "computer on a chip"
- Who makes the processor discussed in this class? There are two answers: **Microchip Founded 1987.**
- A **real-time** system must insure that a task is completed within a certain time span.
- Name five things discussed in class, which must be considered when designing an embedded System. **Real-Time Execution; Physical Size and Environment; Power Consumption; User Interface; Multirate Operation; Cost; Memory Needs; Hardware versus Software**
-Old chip was: **Mips32 M4K Core**
-In a time span: **real-time system**
-Diagnostic PIC32: **JTAG**
-Functions: **CTMU, Timer, PWM, ADC, UART, SPI, RTCC, Compare**
-CTMU External Input Pins : **11**
-UARTs: **2**

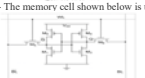
- The HC12 has two 8bit accumulators, **A** and **B**, which can be ganged together
- The HC12 has two **8**-bit accumulators, which can be ganged together.
- The HC12 has at most **128**–kB of flash on board.
- *What is the size of the registers in (in bits) of the HC12 processor? (What size are the standard addresses (in bits) of the HC12 processor?) i.e. "The HC12 is a ___ -bit machine." **16**
- The HC12 has a maximum **8** MHz bus clock.
- The HC12 has a **16**-bit stack pointer.
- The program counter in the HC12 is **16** bits.
- The HC9S12 was an improvement on the HC12 and has a maximum **25**-MHz Clock.
- The HC9S12 has at most **512**-kB of flash on board.
- The HC12 has two [a]-bit registers, X and Y. **16**

- How is SRAM generally used in a system? **Lvl1 cache, Lvl 2 cache, NVRAM**
- What type of Memory is used to store variables in a(n) (embedded) system? **RAM, Volatile, SRAM, DRAM**
- What is the advantage of using DRAM over SRAM? **Lower power at use, Higher Density, Less expensive.**
- *What type of memory is made out of transistors? (What type of RAM is made out of transistors?) **SRAM**
- Which of the following is volatile? **SRAM, DRAM, SDRAM**
- What type of memory is used to store the Stack in an embedded system? **RAM, Volatile, SRAM, DRAM**
- Which type of memory is generally used to store system states in an embedded system? **ROM, NonVolatile, PROM, EPROM, EEPROM**
- What type of memory is generally used to store Program Instructions in an embedded system? **ROM, NonVolatile, PROM, EPROM, EEPROM**
- What type of memory is generally used to store Tables or Images in an embedded system? **ROM, NonVolatile, PROM, EPROM, EEPROM**
- What is the advantage of using SRAM over DRAM? **Lower power consumption (at rest), Simpler to implement, Faster, Can be placed on same die as processor logic**
- What type of memory is generally used to store User Settings in an embedded system? **ROM or NonVolatile or PROM or EPROM or EEPROM**
- Which of the following are nonvolatile? **EEPROM, EPROM, NVRAM, OTP ROM, Flash, Masked ROM, PROM.**
- Which type of RAM stores its data using small capacitors? **DRAM**
- Which type of RAM is the following memory cell found in?



DRAM

- The memory cell shown below is used in what type of memory?

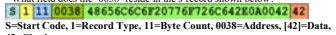


SRAM

- Flowchart, how many times does the programmer try to program a difficult cell before it quits? **10**
- Timing diagram, how long must programmer wait to insure the DATA is valid after the EPROM is told to output the cell data written to it? **tDV**
- Flow chart, how does the programmer program the EPROM? **Programs all the cells, then checks each one for validity**
- Timing diagram, at what point in time does the EPROM programmer read the DATA to be verified from the bus? **The second rising edge of CE#**
- Timing diagram, how long must the DATA be stable before the EPROM latches the data into memory? **tDS**
- Timing diagram, how long must the ADDRESS be stable before the EPROM latches data into memory? **tAS**
- Timing diagram, what signal tells the EPROM to output the cell data which was written to it? **CE#**
- Timing diagram, at what point in time does the EPROM read the DATA it is to program into its cell? **The first rising edge of CE#**
- Timing diagram, how long must the programmer leave the ADDRESS on the bus after they have read the data to be verified? **tAH**

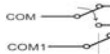
- Timing diagram, what signal determines when the EPROM reads the data to be programmed into the cell? **CE#**

New Section Name four different things that you generally find on a microcontroller project board. **Reset Switch; Com Ports; LCD Display; 7Segment Display; LEDs; Keypad/Switches; Amplifiers/Op Amps; OptoIsolators; Relays; SCRs/Triacs; Buzzers; Potentiometers; DACs; Dip Swiches**
- What's the name of an electromechanical device that is used to switch higher voltage and/or higher current devices? **Relay/SCR/Triac**
- What field does the '0038' reside in the s-record shown below?

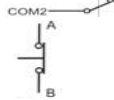


S=Start Code, 1=Record Type, 11=Byte Count, 0038=Address, [42]=Data, 42=Checksum
- What is circuitry that is used to interface two different chips together called? **Glue logic**
- What kind of IC is used to isolate a low voltage microprocessor circuit from a higher voltage motor circuit? **Opto-isolator**
- What is the name for a program which takes C code written on a computer and creates code for a microprocontroller? That is, it creates for a processor other than the one it runs on. **Cross Compiler**
- What is the name for software and/or hardware which allows you to start and stop execution and examine the state of the processor while it is executing your code? **Debugger**
- What is the name for the file generated by a compiler to run on a Motorola/Freescale processor? **S19/S-record**
- What might a potentiometer be used for in an embedded system? **Used to as a volume control in audio devices. Act as a variable resistor**
- What is a relay used for in a circuit? **Relays can control current flow without having hardware/physical device that can degrade of time**
- What are dip switches used for in an embedded system? **For setting up various configuration options**

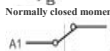
- How is a "momentary switch" different than a regular "on/off" switch? **Momentary is like a push button, on/off switch is like light switch**
- What do you call a switch that is normally off, and must be held down (depressed) to be on? That is, once it is released it is off again. **Push Button Switch/ Momentary Switch**
- What does SPST on a relay stand for? **Single Pole Single Throw**
- What does DPDT on a relay stand for? **Double Pole Double Throw**
- What does SPDT on a relay stand for? **Single Pole Double Throw**



Single Pole Double Throw

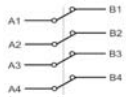


Double Pole Single Throw

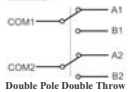


Normally closed momentary switch **Normally open mom. switch**

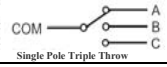
Triple Pole Single Throw



Quad Pole Single Throw

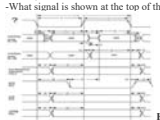


Double Pole Double Throw



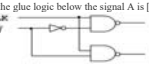
Single Pole Triple Throw

- The "256" in the part number MC9S12DP256B refers to the amount of **memory** contained in the chip.
- The term **glue logic** refers to the added circuitry necessary to interface two or more chips together.
- The **flash** memory in the chip is used to hold code instructions for the chip.
- What signal from the MC9S112 is used in conjunction with the R/W# line to determine whether or not 8 or 16 –bit data is being read? **LSTRB**
- Which of the following is/are not found on the MC9S12? **Piezo Electric Buzzer, Digital to Analog Converter, Temperature Sensor**
- What type of data/information is stored at the upper end of the MC9S12 memory map? **Interrupt Vector Table**
- The [a] memory in the chip is used to hold code instructions for the chip. **Flash**
- What is a compiler called which generates code for a process other than the one it is running on? **Cross Compiler**
- What signal from the MC9S12 is used to determine whether or not a memory chip is being read from or written to? **R/W#**
- What type of memory is located at the very beginning (lowest addresses) of the MC9S12 Memory Map by default? **Registers**
- What type of memory in the 9S12 would you use to store program setting in? **EEPROM**
- What signal from the MC9S12 is used for memory reads when external memory is used, but is not used in single chip mode? **LSTRB**
-What signal is shown at the top of the timing diagram? (As indicated by a '?')



ECLK

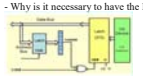
- In the glue logic below the signal A is [a] and the signal B is [b].



A = WE#; B = OE#

Battery*(AmpHour/24*30*months or 24*365*year) = Avg Draw
Need(x) + Don't Need(1-x) = Avg Draw

- What is the purpose/function of a pull-down resistor? **Pulls an input down to a zero by latching it to ground**
- Give an advantage and disadvantage of using isolated(port-based I/O. **Pro – Simultaneous memory and I/O. Cons- Additional pins needed for address, data, controls; additional instructions needed to access the I/O bus**
- What latch chip was used in class to **write** to I/O devices? **74374**
- *(What does the following line of C instruct the compiler to do? What does it allow the programmer to do? (What does unsigned char* do?) (What does volatile mean?)
#define PTH *(volatile unsigned char *)0x260
Variable can be changed from outside the program (external device can drive high/low)
- Give an advantage and disadvantage of using memory-mapped I/O. **Pro – many operations, no addition pins or instructions necessary; Con- I/O error on a memory map cannot be caught and results in program crashing, less memory can be addressed.**
- Give an advantage and disadvantage of using "linear select addressing of I/O. **Pro – Simple selection logic and one port per address-line; Con – Wasteful Addressing**
- Why is a latch needed to store the address from the 9S12 when communicating with an I/O device? **Bus is multiplexed so address must stored while data is put on the bus**
- Why is the 373 Latch's G input connected to 5 volts? **The latche's G input is not-ed so it has to connect to 5 volts to make it a zero.**
-What is the difference between the 74-373 and the 74-374? **74373 uses an enable, 74374 latches values on the rising edge of a clock**
- What latch chip was used in class to **read** from I/O devices? **74373**
-*What is a typical size/value of a pull-up/ pull-down resistor? **4.7KOhm - 10KOhm**
-The ISA does not support serial I/O directly. Why? **Because everything starts out as parallel within the CPU**
- Everything done in the CPU is done in parallel (not serial). Why? **This is done because there are plenty of jobs that computer has to do and most all of them are performed individually. Therefore, parallelism makes everything WAY faster and other programs do not have to wait for another to complete before it executes.**
- Why is it necessary to have the Latch on the left (light blue)?



It uses the clock pulse to latch the values from the address bus to the decoder. Prevents all values from being latched at once.

- An open drain output allows for "wired-Oring" or pins. **True**
- This register can be used to detect overloads or shorts at port pins. **PTIn**
-This register can be used to internally connect a resistor to form a port pin to either Vcc or Vdd if the corresponding pin is set to input. **PERn**
- In order to know which pin an interrupt has occurred on, this register must be read. **PIFn**
- A [a] resistor is generally used to make an input pin's default value 1. **Pull up**
- This register can determine whether the pull resistor is "pull-up" or "pull-down". **PPSn**
- This register can determine whether the interrupt is detected on either the rising edge or falling edge of a signal. **PPSn**
- Which register is written to in order to change the values of a certain port's pins? **PTn**
- What Port only has 4 pins instead of 8? **J**
- Which register needs to be configured before ever outputting a value to a certain port pin? **DDRn**
- This register determines whether or not a transition on a port pin will produce an interrupt. **PIEn**
- The ISA level directly supports serial instructions. **False**
- Which register determines if port pins will be used as input or output? **DDRn**
- Which register determines if port pins will act as an OR gate when its pins are tied together, or not? **WOMn**
- This register reads the state of the register buffer if reading a pin is set to output. **PTn**
- This register tells whether or not an interrupt has occurred a specific pin of a port. **PIFn**
- This register can be used to limit the amount of current supplied to a circuit connected to the corresponding port. **RDRn**
- This Register ignores data written to it if its pins are set up for input. **PTn**

The first address of an **8-K** memory chip is **144 K** in a system with 20-bit addresses.

The chip has **[a]** address lines,

and every valid address for this chip has the form

[b]

where X's represent address bits that can be either 1 or 0.
(Put a space between each group of four bits.)
Specified Answer for: a 13
Specified Answer for: b 0010 010X XXXX XXXX XXXX

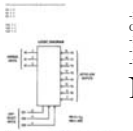
- 2^? = size of memory chip => Address Lines (a)
- Everything beyond address line is an X
- Using values below, add up to get address number, place 1 where number is used (All values are in K(thousands))
- 19=512,18=256,17=128,16=64,15=32,14=16,13=8,12=4,11=2,10=1

Clear PTH: PTH = PTH & 0x52;
Analog: ANSELxSET = 0xA000;
Digital: ANSELxCLR = 0xA000;
Change Notice Interrupt
Disable: CNENxCLR = hex;
Enable: CNENxSET = hex;
Internal Pull Down
Enable: CNPDxSET = hex;
Disable: CNPDxCLR = hex;

Internal Pull Up
Enable: CNPUxSET = hex;
Disable: CNPUxCLR = hex;
TRISx: 1-> in 0-> out
PORTx = read (value of pins)
n = PORTx & 0xA234;
LATx = write
Zeros: LATxCLR = hex;
Ones: LATxSET = hex;

Outputs Open Drain:
ODCxSET = hex;
Outputs Pull Push:
ODCxCLR = hex;

Port[0] is right most port

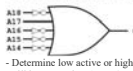


-Match A's with table, CS1=Y0, CS2=Y1, CS3=Y2
-If they all match, find the 0 in the line
-If they don't match, no outputs
-Usually….

NO OUTPUT

A2	A1	A0	/Y0	/Y1	/Y2	/Y3	/Y4	/Y5	/Y6	/Y7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0
X	X	X	1	1	1	1	1	1	1	1

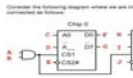
Assume the following gate is connected to the LOW active chip select line of a 8-KB memory chip used in a byte-addressable system with 20-bit addresses. Which of the following system addresses would address the chip?



Watch for Skips

- Determine low active or high active, pay attention to not's
- Fill in appropriate 1's or 0's to get correct output on each line
- If line is excluded, put an x => x can be 1 or 0
- Convert A19-A0 to hex

Consider the following diagram where we are interfacing two memory chips to a MC9S12. Given that the pins are connected as follows:



a. 2^(D-10)
b. FG and KL=>D0 to D7 is low order chip (F = D0, G = D7 -> low) (F = D8, G = D15 -> high)
c. A19-A0=> Put 1 where A and B are, 0's otherwise, convert to HEX
d. A19-A0=>Put 1 at A and B, at D put 1's onward, otherwise 0's
e. Big endian if LSTRB goes into low-order chip (high –> little endian)

Consider the following diagram where we are interfacing two memory chips to a MC9S12. Given that the pins are connected as follows:

High/Low is Given

- C and H are A1
- D and I=> take ending address, covert to binary, where bit's are all 1's, that starting position is D and I
- E and J=>If big endian **LSTRB** goes into Low order chip, Little endian LSTRB goes into high order chip, **A0** goes into opposite chip
- FG and KL=>Low order chip is D0-D7, other is D8-D15
-Chip 0 = 2^(D-10)

The upper address lines of a 20-bit address are to be connected to the inputs of a 10-input PAL to decode memory-mapped I/O addresses in a system.

The PAL is to generate a low signal when an address in the range
0xEA800 – 0xEAFFF
is selected.

What do the input values of the PAL need to be for it to output a low?
(1, 0, or type in **X** if input address not used.) The highest order inputs of the PAL should be used first and the lower ones not connected if not needed. i.e. **I9** is connected to **A19**, **I8** to **A18**, etc…

- Convert both to binary=>Write starting address on top of ending
- Put x where they start to differ and onward

Given the following desired initialization settings for Port H of the HCS12, give the **hexadecimal** value for each register shown below.
(Do not put a **0x** in front of each number. Assume pins that aren't input pins at output, and vice versa. Make each bit that doesn't matter a X.)

```
Input: bit(s) 0 1 2 3 5 6
Reduced Drive: bit(s) 7
Pull Device Enabled: bit(s) 3 5
Interrupt Enabled: bit(s) 1 2
Falling Edge Triggered Interrupts: bit(s)
Rising Edge Triggered Interrupts: bit(s) 1 2
Pull Up: bit(s)
Pull Down: bit(s) 3 5

DDRH = [a]
PERH = [b]
PIEH = [c]
PPSH = [d]
RDRH = [e]
```

- DDRH=Either output or inverted input bits
- PERH=Pull Device Enabled Bits
- PIEH=Interrupt Enabled Bits
- PPSH=Rising Edge Triggered Interrupt bits and pull down bits
-RDRH=Reduced Drive Bits

How would you **set** the following pins of **Port B** without affecting any other pins?
(Leave exactly one space between all variables and/or operators and don't use any parentheses.)

```
Set bit(a) 2 3 4

PTH = [a]
```

- Convert to Binary
- If set, OR with PTH — PTH | 0xSomething
- If clear, AND with PTH — PTH & 0xSomething

How would you **clear** the following pins of **Port B** without affecting any other pins?
(Leave exactly one space between all variables and/or operators and don't use any parentheses.)

```
Clear bit(s) 1 4 6

PTH = [a]
```

- Convert to Binary=>Invert the binary number=>Convert to hex

Given the following code, when must the program drop out of the while loop and execute the instruction **PTJ = 0**!

```
while ((PTH & 0x20) == 0);
PTJ = 0;
```

- Convert to binary
- If while loop = 0=> Every bit that is a 1, put Port[x]=1

- If while loop = 1=> No possible solutions

What is the value of **ChangedSw** (in hex) after this code is run?

```
unsigned char OldSw = 0xD1;
unsigned char NewSw = 0x1E;
unsigned char ChangedSw;

ChangedSw = (OldSw ^ ~NewSw) & ~OldSw;
```

- Convert all to binary
- When raising binary to binary => bits are same, 0; bits are different, 1
- Follow equation

What does the following C code "look for?" (i.e. What kind of change does this C code detect?)

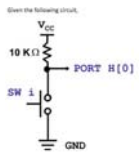
```
/* ... */
unsigned char OldSw, NewSw, ChangedSw;
while (OldSw == NewSw) NewSw = PTH;
ChangedSw = (NewSw ^ OldSw) & ~NewSw;
OldSw = NewSw;
/* ... */
```

Look where turned to 1

- Use 1010 for Old->Use 1001 for new->Follow code, see what changed

KEYPAD
- Rewrite Keypad as
- Circle keys pressed->0's if pressed->1's otherwise (in form of keypad)
- For bitshifting->convert across to binary (Nott if needed)
- Otherwise look at keypad and determine where button was pressed
  - Single button in row, it's that one; Two single buttons in two different rows, it's the top button; Two buttons in same row or none pushed, it's 0

Given the following circuit,



Switch on bottom

Switch on Top

Consider two chip technologies, A and B, which have the following electrical input and output characteristics:

```
VIH(A) = 3.75 V VIH(B) = 3.21 V
VIL(A) = 4.88 V VIL(B) = 3.63 V
VOH(A) = 3.98 V VOH(B) = 4.06 V
VOL(A) = 4.70 V VOL(B) = 4.63 V

IIH(A) = 39.65 mA IIH(B) = 81.34 mA
IIL(A) = 14.88 mA IIL(B) = 4.76 mA
IOH(A) = 6.83 mA IOH(B) = 56.24 mA
IOL(A) = 0.26 mA IOL(B) = 68.06 mA
```

The fanout for chip A outputs driving chip B inputs is **[a]**. (Type '0' if chip A cannot drive chip B.)

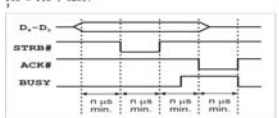
The fanout for chip B outputs driving chip A inputs is **[b]**. (Type '0' if chip B cannot drive chip A.)

Round Down
Only one will be 0

- First look at IF's->A to B make sure Max A>Min B if not A = 0
- Repeat for B to A-> Max B>Min A if not B = 0
- Repeat steps for IO's
-If pass both test; A to B get minimum of IOH(A)/IIH(B) and IOL(A)/IIL(B)
- Whichever is min, that is = A -> Switch A and B's for B

The following program is used to write data to a printer using the SPP protocol showed below.

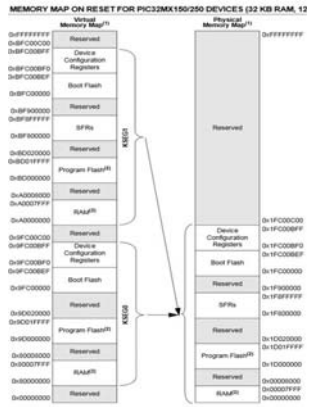
```
void PutChar(unsigned char n)
{
while ((PTJ & 0x02) != 0 );
PTH = n;
PTJ = PTJ & ~0x01;
NOP;
//...
PTJ = PTJ | 0x01;
}
```



- See how man blocks STRB# = # (in this example 1)
- ((#*N*(# of Mhz))/(clock ticks))-1 (don't use x10^-3 or any multiplier)
-round up

ADDRESS DECODING

Remember: address/size = int
Check 0s, 1s always *
Make a table



28 OFN

Pin #	Full Pin Name
1	PGED1/AN2/C1IND/C2INB/C3IND/RPB0/PMD0/RB0
2	PGEC1/AN3/C1INC/C2INA/RPB1/CTED12/PMD1/RB1
3	AN4/C1INB/C2IND/RPB2/SDA2/CTED13/PMD2/RB2
4	AN5/C1INA/C2INC/RTCC/RPB3/SCL2/PMWR/RB3
5	VSS
6	OSC1/CLKI/RPA2/RA2
7	OSC2/CLKO/RPA3/PMA0/RA3
8	SOSCI/RPB4/RB4
9	SOSCO/RPA4/T1CK/CTED9/PMA1/RA4
10	VDD
11	TMS/RPB5/USBID/RB5
12	VBUS
13	TDI/RPB7/CTED3/PMD5/INT0/RB7
14	TCK/RPB8/SCL1/CTED10/PMD4/RB8

Pin #	Full Pin Name
15	TDO/RPB9/SDA1/CTED4/PMD3/RB9
16	VSS
17	VCAP
18	PGED2/RPB10/D+/CTED11/RB10
19	PGEC2/RPB11/D-/RB11
20	VUSB3V3
21	AN11/RPB13/CTPLS/PMRD/RB13
22	CVREFOUT/AN10/C3INB/RPB14/VBUSON/SCK1/CTED5/RB14
23	AN9/C3INA/RPB15/SCK2/CTED6/PMCS1/RB15
24	AVSS
25	AVDD
26	MCLR
27	PGED3/VREF+/CVREF+/AN0/C3INC/RPA0/CTED1/PMD7/RA0
28	PGEC3/VREF-/CVREF-/AN1/RPA1/CTED2/PMD6/RA1

36 VTLA

Pin #	Full Pin Name
1	AN4/C1INB/C2IND/RPB2/SDA2/CTED13/PMD2/RB2
2	AN5/C1INA/C2INC/RTCC/RPB3/SCL2/PMWR/RB3
3	PGED4[4]/AN6/RPC0/RC0
4	PGEC4[4]/AN7/RPC1/RC1
5	VDD
6	VSS
7	OSC1/CLKI/RPA2/RA2
8	OSC2/CLKO/RPA3/PMA0/RA3
9	SOSCI/RPB4/RB4
10	SOSCO/RPA4/T1CK/CTED9/PMA1/RA4
11	AN12/RPC3/RC3
12	VSS
13	VDD
14	TMS/RPB5/USBID/RB5
16	VBUS
17	TDI/RPB7/CTED3/PMD5/INT0/RB7
18	TCK/RPB8/SCL1/CTED10/PMD4/RB8

Pin #	Full Pin Name
19	TDO/RPB9/SDA1/CTED4/PMD3/RB9
20	RPC9/CTED7/RC9
21	VSS
22	VCAP
23	VDD
24	PGED2/RPB10/D+/CTED11/RB10
25	PGEC2/RPB11/D-/RB11
26	VUSB3V3
27	AN11/RPB13/CTPLS/PMRD/RB13
28	CVREFOUT/AN10/C3INB/RPB14/VBUSON/SCK1/CTED5/RB14
29	AN9/C3INA/RPB15/SCK2/CTED6/PMCS1/RB15
30	AVSS
31	AVDD
32	MCLR
33	PGED3/VREF+/CVREF+/AN0/C3INC/RPA0/CTED1/PMD7/RA0
34	PGEC3/VREF-/CVREF-/AN1/RPA1/CTED2/PMD6/RA1
35	PGED1/AN2/C1IND/C2INB/C3IND/RPB0/PMD0/RB0
36	PGEC1/AN3/C1INC/C2INA/RPB1/CTED12/PMD1/RB1

44

Pin #	Full Pin Name
1	RPB9/SDA1/CTED4/PMD3/RB9
2	RPC6/PMA1/RC6
3	RPC7/PMA0/RC7
4	RPC8/PMA5/RC8
5	RPC9/CTED7/PMA6/RC9
6	VSS
7	VCAP
8	PGED2/RPB10/D+/CTED11/RB10
9	PGEC2/RPB11/D-/RB11
10	VUSB3V3
11	AN11/RPB13/CTPLS/PMRD/RB13
12	PGED4/TMS/PMA10/RA10
13	PGEC4/TCK/CTED8/PMA7/RA7
14	CVREFOUT/AN10/C3INB/RPB14/VBUSON/SCK1/CTED5/RB14
15	AN9/C3INA/RPB15/SCK2/CTED6/PMCS1/RB15
16	AVSS
17	AVDD
18	MCLR
19	PGED3/VREF+/CVREF+/AN0/C3INC/RPA0/CTED1/PMD7/RA0
20	PGEC3/VREF-/CVREF-/AN1/RPA1/CTED2/PMD6/RA1
21	PGED1/AN2/C1IND/C2INB/C3IND/RPB0/PMD0/RB0
22	PGEC1/AN3/C1INC/C2INA/RPB1/CTED12/PMD1/RB1

Pin #	Full Pin Name
23	AN4/C1INB/C2IND/RPB2/SDA2/CTED13/PMD2/RB2
24	AN5/C1INA/C2INC/RTCC/RPB3/SCL2/PMWR/RB3
25	AN6/RPC0/RC0
26	AN7/RPC1/RC1
27	AN8/RPC2/PMA2/RC2
28	VDD
29	VSS
30	OSC1/CLKI/RPA2/RA2
31	OSC2/CLKO/RPA3/RA3
32	TDO/RPA8/PMA8/RA8
33	SOSCI/RPB4/RB4
34	SOSCO/RPA4/T1CK/CTED9/RA4
35	TDI/RPA9/PMA9/RA9
36	AN12/RPC3/RC3
37	RPC4/PMA4/RC4
38	RPC5/PMA3/RC5
39	VSS
40	VDD
41	RPB5/USBID/RB5
42	VBUS
43	RPB7/CTED3/PMD5/INT0/RB7
44	RPB8/SCL1/CTED10/PMD4/RB8

28 QFN

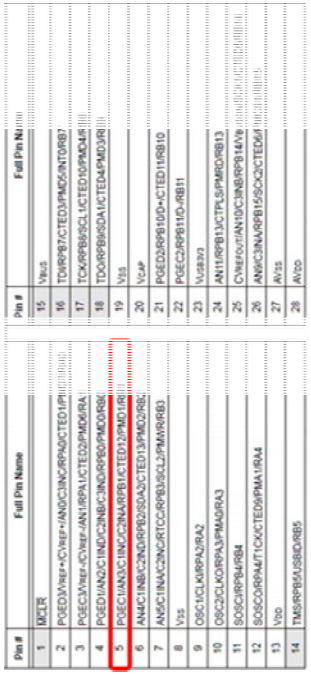


TABLE 4-1: SFR MEMORY MAP

Peripheral	Virtual Address	
	Base	Offset Start
Watchdog Timer	0xBF80	0x0000
RTCC		0x0200
Timer1-5		0x0600
Input Capture 1-5		0x2000
Output Compare 1-5		0x3000
IC1 and IC2		0x5000
SPI1 and SPI2		0x5800
UART1 and UART2		0x6000
PMP		0x7000
ADC		0x9000
CVREF		0x9800
Comparator		0xA000
CTMU		0xA200
Oscillator		0xF000
Device and Revision ID		0xF220
Peripheral Module Disable		0xF240
Flash Controller		0xF400
Reset		0xF600
PPS		0xFA04
Interrupts	0xBF88	0x1000
Bus Matrix		0x2000
DMA		0x3000
USB		0x5050
PORTA-PORTC		0x6000
Configuration	0xBFC0	0x0BF0

TABLE 26-2: PERIPHERAL MODULE DISABLE REGISTER MAP

Virtual Address (BF80_#)	Register(1) Name	Bit Range	Bits															All Resets
			31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0	
F240	PMD1	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	CVRMD	—	—	—	CTMUMD	—	—	—	—	—	—	—	AD1MD	0000
F250	PMD2	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	—	CMP3MD	CMP2MD	CMP1MD	0000
F260	PMD3	31:16	—	—	—	—	—	—	—	—	—	—	—	OC5MD	OC4MD	OC3MD	OC2MD	OC1MD	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	IC5MD	IC4MD	IC3MD	IC2MD	IC1MD	0000
F270	PMD4	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	T5MD	T4MD	T3MD	T2MD	T1MD	0000
F280	PMD5	31:16	—	—	—	—	—	—	—	USB1MD	—	—	—	—	—	—	I2C1MD	I2C1MD	0000
		15:0	—	—	—	—	—	—	SPI2MD	SPI1MD	—	—	—	—	—	—	U2MD	U1MD	0000
F290	PMD6	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	PMPMD	0000
		15:0	—	—	—	—	—	—	—	—	—	—	—	—	—	—	REFOMD	RTCCMD	0000

Legend: x = unknown value on Reset; — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

Note 1: All registers in this table have corresponding CLR, SET and INV registers at their virtual addresses, plus offsets of 0x4, 0x8 and 0xC, respectively. See Section 11.2 "CLR, SET and INV Registers" for more information.