MICROPROCESSORS A (17.383)

Fall 2010

Lecture Outline

Class # 07

October 26, 2010

Dohn Bowden

Today's Lecture

- Syllabus review
- Microcontroller Hardware and/or Interface
  - Finish Analog to Digital Conversion
- Programming/Software
  - Code used to perform Analog to Digital Conversions
- Lab
  - Lab #3 – 7 Segment LED Interface
- Homework

Course Admin

Administrative

- Admin for tonight …
  - Course Project
    - Requirements will be passed out next week
  - Syllabus Highlights
    - Lab report for Lab # 3 is due on November 9th
    - For planning purposes …
      - Exam #2 is November 30th … 5 weeks from today

Syllabus Review

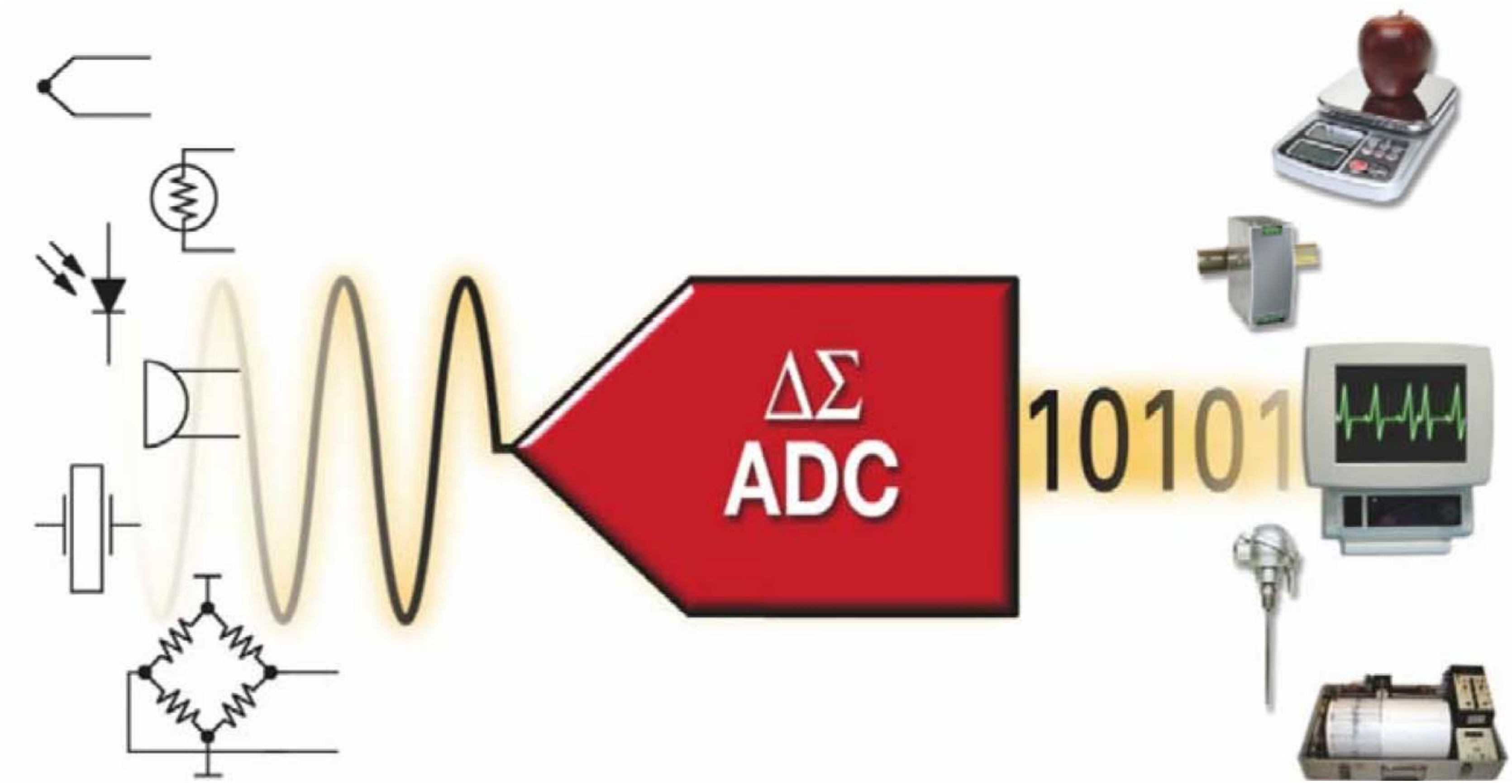
Week	Date	Topics	Lab	Lab Report Due
~~1~~	~~09/07/10~~	~~Intro, Course & Lab Overview, Microcontroller Basics~~	~~1~~	
~~2~~	~~09/14/10~~	~~PIC16F684 Overview and General Input/Output~~	~~1 con't~~	
~~3~~	~~09/21/10~~	~~Switches~~	~~2~~	
~~4~~	~~09/28/10~~	~~Seven Segment LEDs~~	~~2 con't~~	~~1~~
~~5~~	~~10/05/10~~	~~**Examination 1**~~		
~~**X**~~	~~**10/12/10**~~	~~**No Class – Monday Schedule**~~		
~~6~~	~~10/19/10~~	~~Analog to Digital Conversion~~	~~3~~	~~2~~
7	10/26/10	Analog to Digital Conversion con't	3 con't	
8	11/02/10	LCD Interface and Assembly Language	4	
9	11/09/10	Comparators	4 con't	3
10	11/16/10	Timers and Pulse Width Modulation (PWM)	5	
11	11/23/10	Mixed C & Assembly Programming/Course Project	Project	4
12	11/30/10	**Examination 2**		
13	12/07/10	Course Project	Project	5
14	12/14/10	Final Exam/Course Project Brief and Demonstration	Demo	

Microcontroller Hardware and / or Interfaces

ADC Review from the last Lecture …

A Quick Review!

Analog To Digital Conversion Overview



PIC16F684 Hardware

- We turned our attention to the analog features
  - Analog to Digital Conversion (ADC or A/D conversion)

Last Lecture ...

- We went through the basics of Analog To Digital Conversion
    - Any Questions?

Analog Signals

- We stated that … Real world signals are analog
  - For example … sensors
- We need to be able to take these signals and convert them to digital in order to be able to process them using the microcontroller
- The PIC16F684 is capable of performing the required conversion with it's built in analog to digital converter

Analog to Digital Converter

- **What is an *Analog-to-Digital converter*?**

  – An ***Analog-to-Digital converter*** (ADC) is an electronic circuit that changes or converts a continuous analog signal into a digital signal without altering its critical content

Representation of an Analog Signal as a Digital Value

- The ADC represents an analog signal as a digital string of 1's and 0's with finite resolution

  - The ADC outputs a finite number of digital values …

    – equal to 2^N

      (where N is the number of bits of the ADC)

Resolution

- ***Resolution* ...** Is a measure of the smallest change in analog input that can be discriminated by an A/D Converter
  - The more binary digits of output that are available …
  - The more resolution that is possible, and …
  - The more precision we can encode the analog signal
- The ***resolution*** of an 'N' bit A/D converter with a voltage range of "0 – X" volts is …

$$\frac{X - 0}{2^N}$$

The PIC16F684 Analog-to-Digital Converter

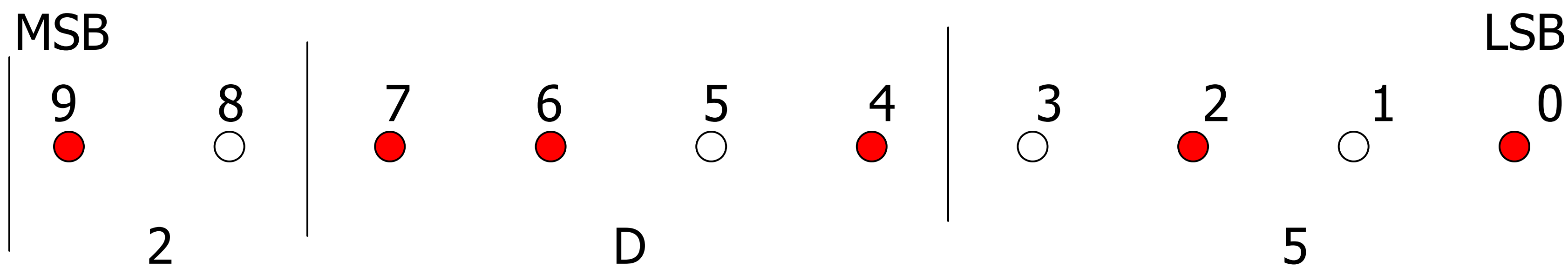
- Contains a Successive-approximation-register (SAR) type Analog to Digital converters

    - 10-bit resolution

    - 8 channels

ADC Output

- LEDs (Digital) representation of Analog voltage …

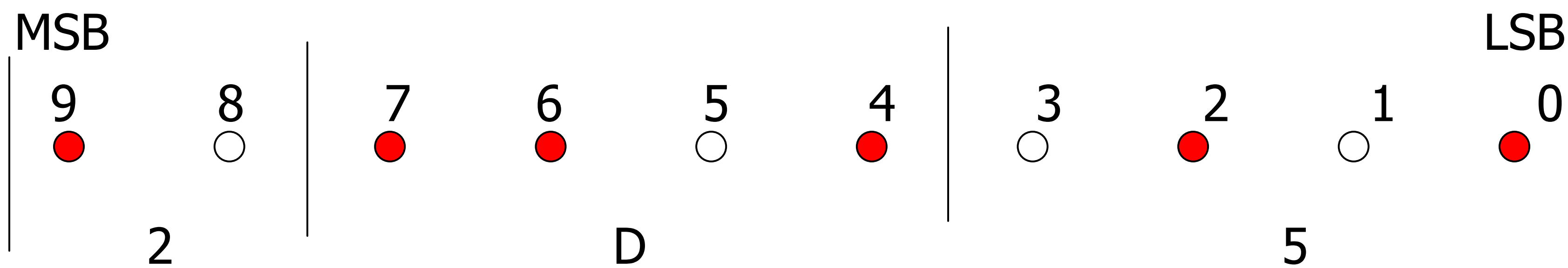


$(\#\#\#h)*(Vmax/2^N) = voltage$ $2D5h = 725_{10}$

$(2D5h)*(5/2^{10}) = 3.54$ volts

ADC Output – another method

- LEDs (Digital) representation of Analog voltage …



$(0bxxx)*(Vmax/2^N)$ = voltage

$(725_{10})*(5/2^{10})$ = 3.54 volts

$(2^0)*1 =$	1	$(2^5)*0 =$	0
$(2^1)*0 =$	0	$(2^6)*1 =$	64
$(2^2)*1 =$	4	$(2^7)*1 =$	128
$(2^3)*0 =$	0	$(2^8)*0 =$	0
$(2^4)*1 =$	16	$(2^9)*1 =$	<u>512</u>
			725

The PIC16F684 Analog-To-Digital Converter Module Specifics

ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

- The PIC16F684 Analog-to-Digital converter (A/D) allows
    - Conversion of an analog input signal to a 10-bit binary representation of that signal
- The PIC16F684 has eight analog inputs,
    - Multiplexed into one sample and hold circuit
- The output of the sample and hold is connected to the input of the converter

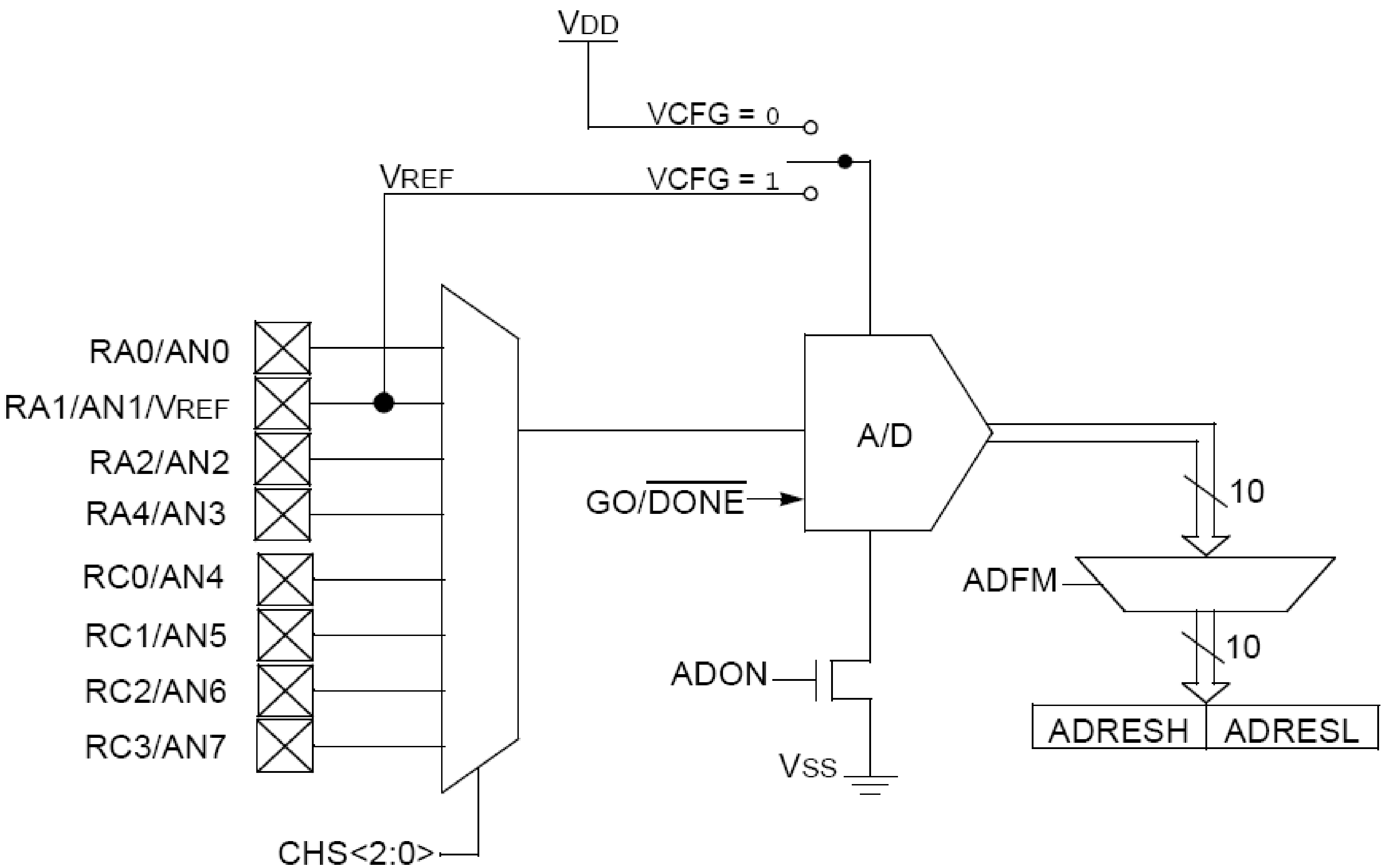
ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE (con't)

- The converter generates a binary result via successive approximation … and …
  - Stores the 10 bit result in 2 eight bit registers
- The voltage reference used in the conversion is software selectable to either …
  - VDD (internal to the PIC) … typically 5.0 volts

  or

  - A voltage applied by the V_{REF} pin (external source)
- The negative voltage reference is always connected to the ground reference

A/D BLOCK DIAGRAM



Configuring The PIC16F684 A/D Module

- To use the ADC feature of the PIC16F684 we will need to configure the device

- To configure the PIC16F684 … three registers need to be setup

  - ANSEL (Analog Select Register)

  - ADCON1 (A/D Control Register 1)

  - ADCON0 (A/D Control Register 0)

ANSEL (Analog Select)

- The ANSEL register is …
  - Used to configure the Input mode of an I/O pin to analog
- Setting the appropriate ANSEL bit high will …
  - Cause all digital reads on the pin to be read as ‘0’ … and …
  - Allow analog functions on the pin to operate correctly

ANSEL (Analog Select)

- The state of the ANSEL bits …
  - Has no affect on digital output functions
- A pin with …
  - TRIS clear … and …
  - ANSEL set … will …
    - Still operate as a digital output … but …
      - The Input mode will be analog
        - » This can cause unexpected behavior when executing read-modify-write instructions on the affected port

ANSEL (Analog Select)

Note:	Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

REGISTER 9-1: ANSEL – ANALOG SELECT REGISTER (ADDRESS: 91h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
bit 7							bit 0

bit 7-0: **ANS<7:0>**: Analog Select bits

Analog select between analog or digital function on pins AN<7:0>, respectively.

`1` = Analog input. Pin is assigned as analog input[(1)].

`0` = Digital I/O. Pin is assigned to port or special function.

Note 1: Setting a pin to an analog input automatically disables the digital input circuitry, weak pull-ups, and interrupt-on-change if available. The corresponding TRIS bit must be set to input mode in order to allow external control of the voltage on the pin.

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

ADCON1 (A/D Control Register 1)

- Bit 6-4 Conversion clock select bits
  - An accurate conversion requires a time of 1.6 μs or greater
    - There is no point making this longer
  - The internal oscillator provides a conversion time of approximately 4 μs, although this can vary between 2 and 6μs
    - We are using the internal oscillator, therefore we will use the A/D RC option (111)
- No other bits are used in this register

REGISTER 9-3: ADCON1 – A/D CONTROL REGISTER 1 (ADDRESS: 9Fh)

U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	ADCS2	ADCS1	ADCS0	—	—	—	—
bit 7							bit 0

bit 7: **Unimplemented:** Read as ‘0’

bit 6-4: **ADCS<2:0>:** A/D Conversion Clock Select bits

- 000 = FOSC/2
- 001 = FOSC/8
- 010 = FOSC/32
- x11 = FRC (clock derived from a dedicated internal oscillator = 500 kHz max)
- 100 = FOSC/4
- 101 = FOSC/16
- 110 = FOSC/64

bit 3-0: **Unimplemented:** Read as ‘0’

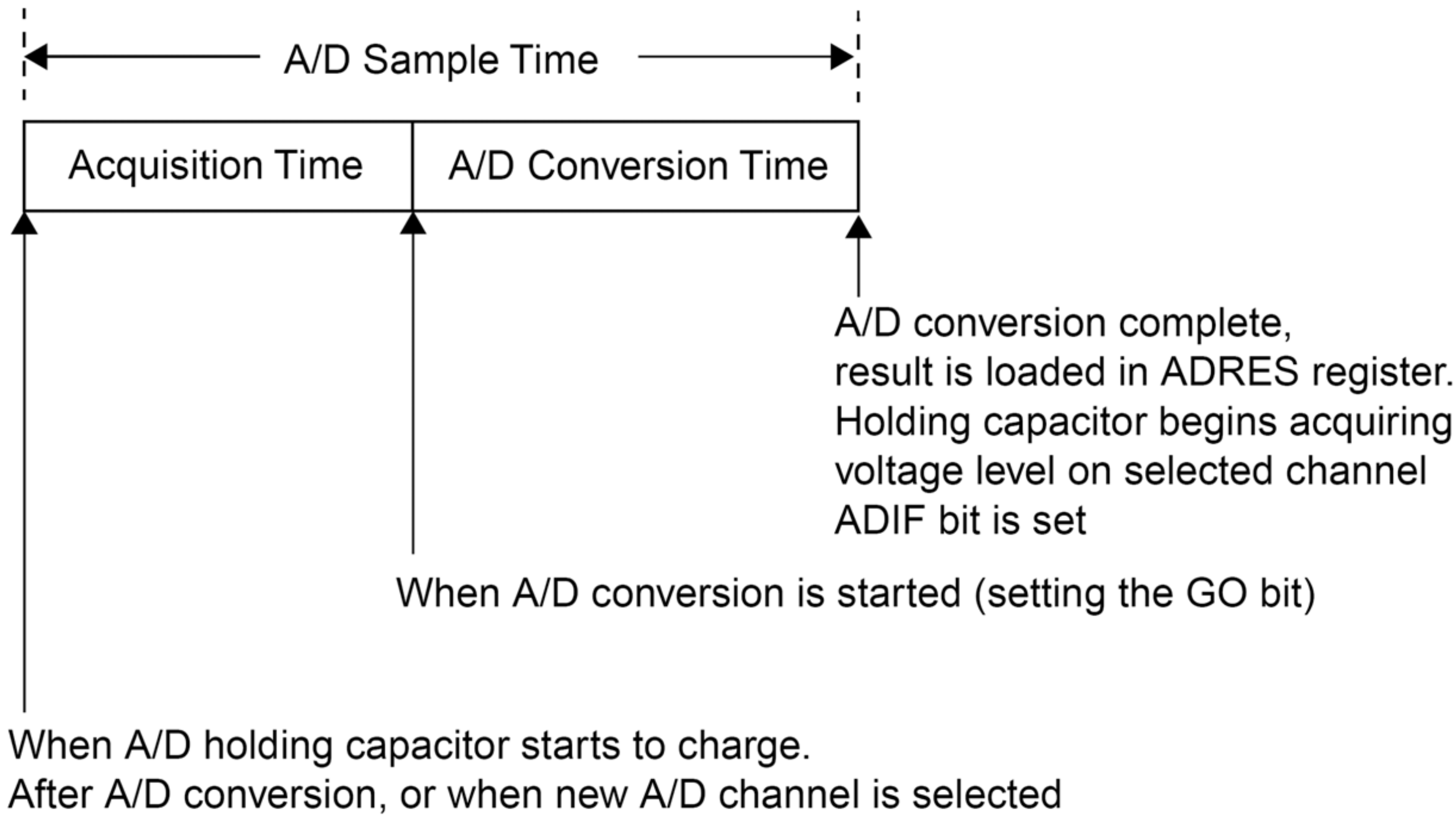
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

ADC Timing

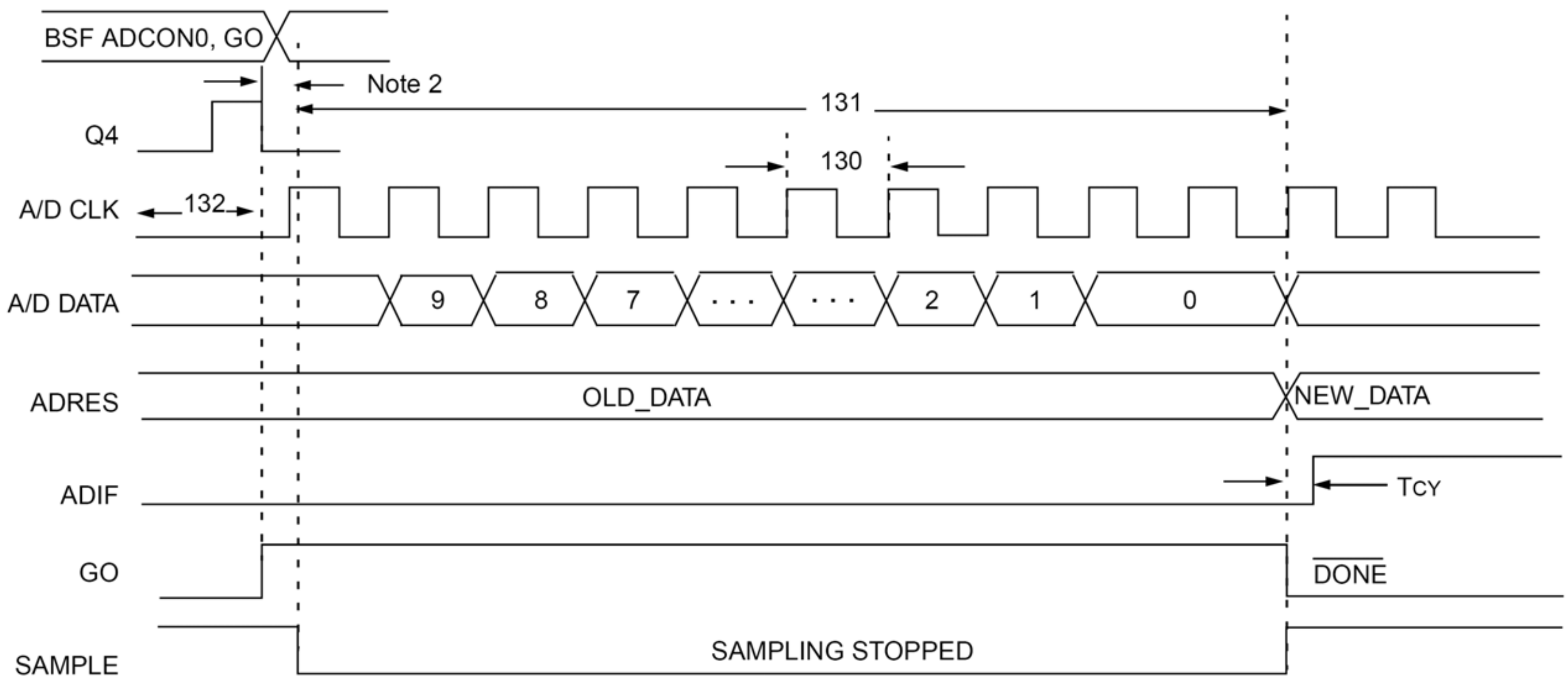
Definitions

- TAD … In the A/D Converter, the time for a single bit of the analog voltage to be converted to a digital value
- Tosc … The time for the device oscillator to do a single period
- Four external clocks (Tosc) make one instruction cycle (TCY)
- INTRC … Internal Resistor-Capacitor (RC) Oscillator. The PIC16F684 has an Internal 4 MHz Resistor/Capacitor oscillator
- Fosc … Frequency of the device oscillator
- TCY … The time for an instruction to complete. This time is equal to Fosc/4 and is divided into four Q-cycles
- Q-cycles … This is the same as a device oscillator cycle. There are 4 Q-cycles for each instruction cycle

A/D Conversion Sequence



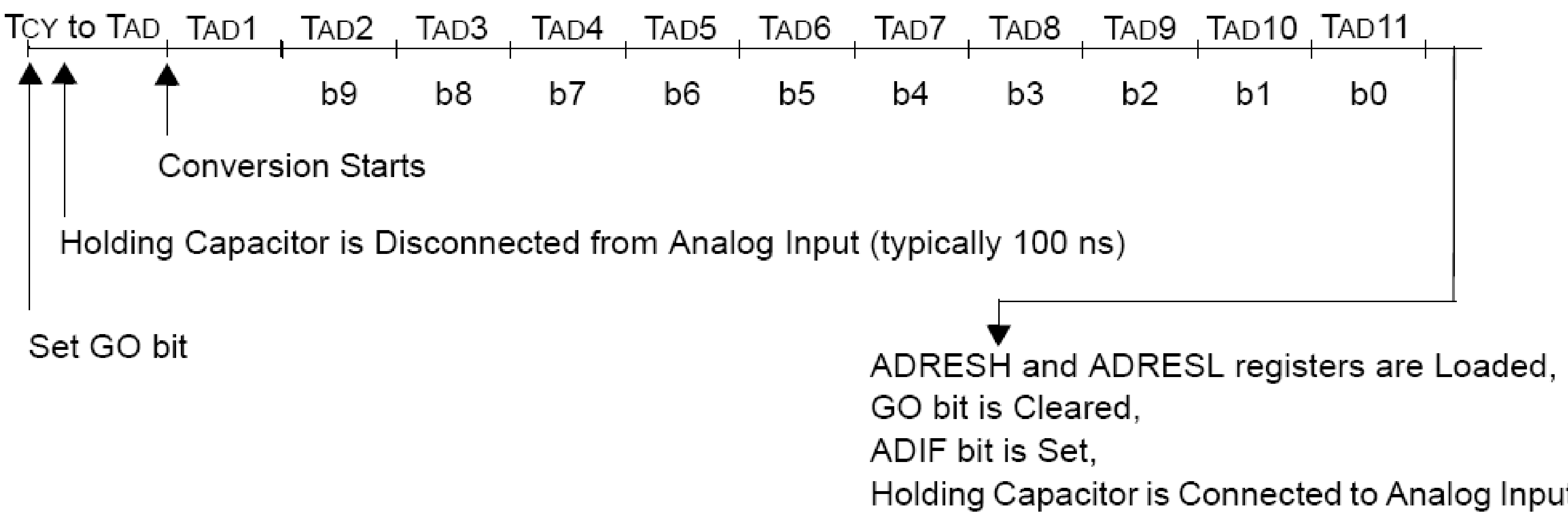
10-Bit A/D Conversion Timing Waveforms



Note 1: If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the `SLEEP` instruction to be executed.

2: This is a minimal RC delay (typically 100 nS), which also disconnects the holding capacitor from the analog input.

A/D CONVERSION TAD CYCLES



A/D conversion time per bit is defined as TAD

Calculating Conversion Times (TAD) - *Example*

- For example … if we use a processor running at 4 MHz
- Clock period = 1/4,000,000 = .000000250 = 250 nSec
- So … if we use 8 TOSC, we would have …

  Clock Period * TAD Operation = 250 nSec * 8 = 2 µSec

- The internal RC clock … has a typical TAD time of 4 µsec
  - Although this can vary between 2 and 6µs

TAD VS. DEVICE OPERATING FREQUENCIES

TABLE 9-1: TAD VS. DEVICE OPERATING FREQUENCIES

A/D Clock Source (TAD)		Device Frequency			
Operation	**ADCS2:ADCS0**	**20 MHz**	**5 MHz**	**4 MHz**	**1.25 MHz**
2 TOSC	`000`	100 ns(2)	400 ns(2)	500 ns(2)	1.6 μs
4 TOSC	`100`	200 ns(2)	800 ns(2)	1.0 μs(2)	3.2 μs
8 TOSC	`001`	400 ns(2)	1.6 μs	2.0 μs	6.4 μs
16 TOSC	`101`	800 ns(2)	3.2 μs	4.0 μs	12.8 μs(3)
32 TOSC	`010`	1.6 μs	6.4 μs	8.0 μs(3)	25.6 μs(3)
64 TOSC	`110`	3.2 μs	12.8 μs(3)	16.0 μs(3)	51.2 μs(3)
A/D RC	`x11`	2-6 μs(1,4)	2-6 μs(1,4)	2-6 μs(1,4)	2-6 μs(1,4)

Legend: Shaded cells are outside of recommended range.

Note 1: The A/D RC source has a typical TAD time of 4 μs for VDD > 3.0V.

2: These values violate the minimum required TAD time.

3: For faster conversion times, the selection of another clock source is recommended.

4: When the device frequency is greater than 1 MHz, the A/D RC clock source is only recommended if the conversion will be performed during Sleep.

VOLTAGE REFERENCE

- There are two options for the voltage reference to the A/D converter … either …
  - VDD is used … or …
  - An analog voltage applied to V_{REF}
- The VCFG bit (ADCON0<6>) controls the voltage reference selection
  - If VCFG is set … then the voltage on the VREF pin is the reference
  - otherwise, VDD is the reference

Reference Voltage

- V_{REF} (Reference voltage)
  - Minimum 3.0V
  - Maximum V_{DD} + 0.3 V
- V_{DD} Supply Voltage
  - 2.0 min
  - 5.5V Max

ADCON0 (A/D Control Register 0)

- Bit 0
  - Turns on or off the A/D converter
    - 1 = On
    - 0 = Off

- Bit 4-2
  - Selects the channel to use (AN0 – AN7)

- Bit 6
  - Selects where the reference voltage is from

- Bit 7
  - Results format (right or left justified)

REGISTER 9-2: ADCON0 – A/D CONTROL REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	VCFG	—	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

bit 7 **ADFM:** A/D Result Formed Select bit
1 = Right justified
0 = Left justified

bit 6 **VCFG:** Voltage Reference bit
1 = V_{REF} pin
0 = V_{DD}

bit 5 **Unimplemented:** Read as '0'

bit 4-2 **CHS<2:0>:** Analog Channel Select bits
000 = Channel 00 (AN0)
001 = Channel 01 (AN1)
010 = Channel 02 (AN2)
011 = Channel 03 (AN3)
100 = Channel 04 (AN4)
101 = Channel 05 (AN5)
110 = Channel 06 (AN6)
111 = Channel 07 (AN7)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle.
This bit is automatically cleared by hardware when the A/D conversion has completed.
0 = A/D conversion completed/not in progress

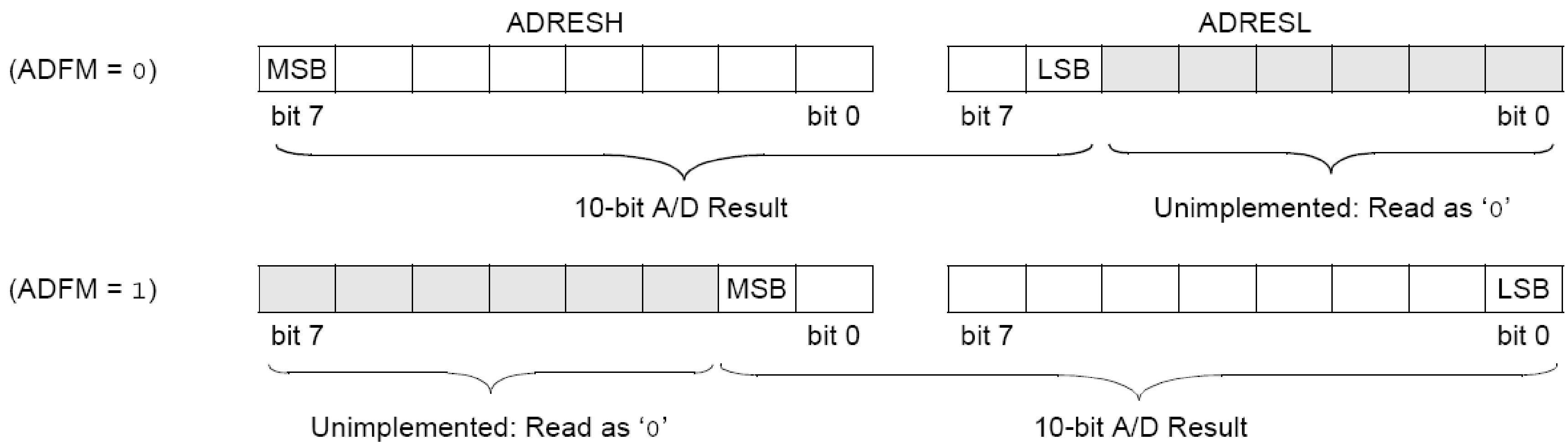
bit 0 **ADON:** A/D Conversion Status bit
1 = A/D converter module is operating
0 = A/D converter is shut-off and consumes no operating current

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

CONVERSION OUTPUT

- The A/D conversion can be supplied in two formats …
  - Left …
  - or right shifted
- The ADFM bit (ADCON0<7>) controls the output format
- The next slide shows the output formats

10-BIT A/D RESULT FORMAT



SUMMARY OF A/D REGISTERS

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOD	Value on all other Resets
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--xx xxxx	--uu uuuu
07h	PORTC	—	—	RC5	RC4	RC3	RC2	RC1	RC0	--xx xxxx	--uu uuuu
0Bh/ 8Bh	INTCON	GIE	PEIE	T0IE	INTE	RAIE	T0IF	INTF	RAIF	0000 0000	0000 0000
0Ch	PIR1	EEIF	ADIF	CCP1IF	C2IF	C1IF	OSFIF	TMR2IF	TMR1IF	0000 0000	0000 0000
1Eh	ADRESH	Most Significant 8 bits of the left shifted A/D result or 2 bits of the right shifted result								xxxx xxxx	uuuu uuuu
1Fh	ADCON0	ADFM	VCFG	—	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	00-0 0000	00-0 0000
85h	TRISA	—	—	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	--11 1111	--11 1111
87h	TRISC	—	—	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	--11 1111	--11 1111
8Ch	PIE1	EEIE	ADIE	CCP1IE	C2IE	C1IE	OSFIE	TMR2IE	TMR1IE	0000 0000	0000 0000
91h	ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	1111 1111	1111 1111
9Eh	ADRESL	Least Significant 2 bits of the left shifted A/D result or 8 bits of the right shifted result								xxxx xxxx	uuuu uuuu
9Fh	ADCON1	—	ADCS2	ADCS1	ADCS0	—	—	—	—	-000 ----	-000 ----

Legend: x = unknown, u = unchanged, — = unimplemented read as '0'. Shaded cells are not used for A/D module.

STARTING A CONVERSION

- The A/D conversion is initiated by setting the GO/DONE bit (ADCON0<1>)

- When the conversion is complete, the A/D module …

  - Clears the GO/DONE bit
  - Sets the ADIF flag (PIR1<6>)
  - Generates an interrupt (if enabled)

Steps to Follow for A/D Conversion

1. Configure the A/D module
2. Configure A/D interrupt (if desired)
3. Wait the required *acquisition* time
4. Start the conversion
5. Wait for A/D *conversion* to complete
6. Read the A/D Result register pair
   - Use/store the result of the conversion
7. Perform the next conversion by going back to step 1 or step 2 as required

Detailed Steps …

Step 1
Configure the A/D module

- `ANSEL`
  - Configures the analog/digital Input/Output (I/O) pins
- `ADCON0`
  - Configures the voltage reference
  - Select the A/D input channel
  - Turns on the A/D module
- `ADCON1`
  - Select the A/D conversion clock

Step 1 (continued)
Configure the A/D module (ANSEL)

- **ANSEL** … select between analog or digital function on pins AN<7:0> (The Register is shown on the next slide)
  - **ANSEL** Bits are **ANS0** through **ANS7**

    Setting the bits …   "1" --- Analog

    Clearing the bits …   "0" --- Digital

- In our example below, we only want pin AN0 as analog. The others can be digital, therefore set ANS0 to 1, all others are 0

- The analog input channels must have their corresponding TRIS bits selected as inputs

```
ANSEL = 1;       // Only AN0 is an Analog Input, all others are digital
TRISA0 = 1;      // Corresponding TRIS bit is set as input
```

REGISTER 9-1: ANSEL – ANALOG SELECT REGISTER (ADDRESS: 91h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
bit 7							bit 0

bit 7-0: **ANS<7:0>**: Analog Select bits

Analog select between analog or digital function on pins AN<7:0>, respectively.

`1` = Analog input. Pin is assigned as analog input[1].

`0` = Digital I/O. Pin is assigned to port or special function.

Note 1: Setting a pin to an analog input automatically disables the digital input circuitry, weak pull-ups, and interrupt-on-change if available. The corresponding TRIS bit must be set to input mode in order to allow external control of the voltage on the pin.

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

Step 1 (continued)
Configure the A/D module (`ADCON0`)

- `ADCON0` ... (AD Control Register)
  - The Register is shown on the next slide

```
ADCON0 = 0b00000001;   //   Bit 7     - Left Justified Sample
                       //   Bit 6     - Use VDD
                       //   Bit 5     - Not Used
                       //   Bit 4:2 - Channel 0 (AN0)
                       //   Bit 1     - Do not Start the conversion
                       //   Bit 0     - Turn on ADC
```

REGISTER 9-2: ADCON0 – A/D CONTROL REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	VCFG	—	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

bit 7 **ADFM:** A/D Result Formed Select bit
1 = Right justified
0 = Left justified

bit 6 **VCFG:** Voltage Reference bit
1 = VREF pin
0 = VDD

bit 5 **Unimplemented:** Read as '0'

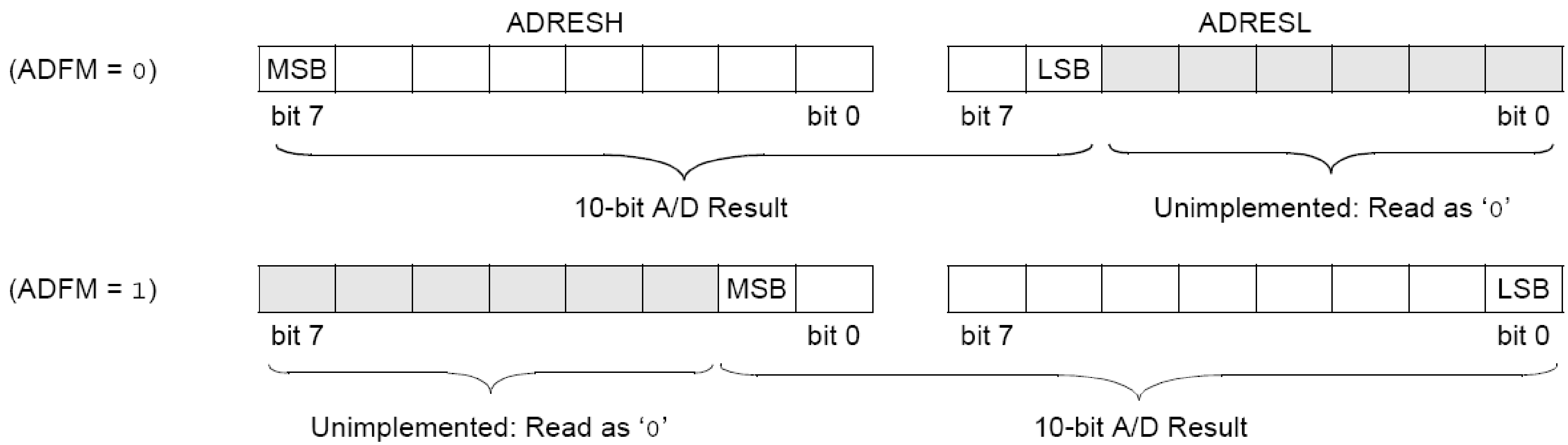
bit 4-2 **CHS<2:0>:** Analog Channel Select bits
000 = Channel 00 (AN0)
001 = Channel 01 (AN1)
010 = Channel 02 (AN2)
011 = Channel 03 (AN3)
100 = Channel 04 (AN4)
101 = Channel 05 (AN5)
110 = Channel 06 (AN6)
111 = Channel 07 (AN7)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle.
This bit is automatically cleared by hardware when the A/D conversion has completed.
0 = A/D conversion completed/not in progress

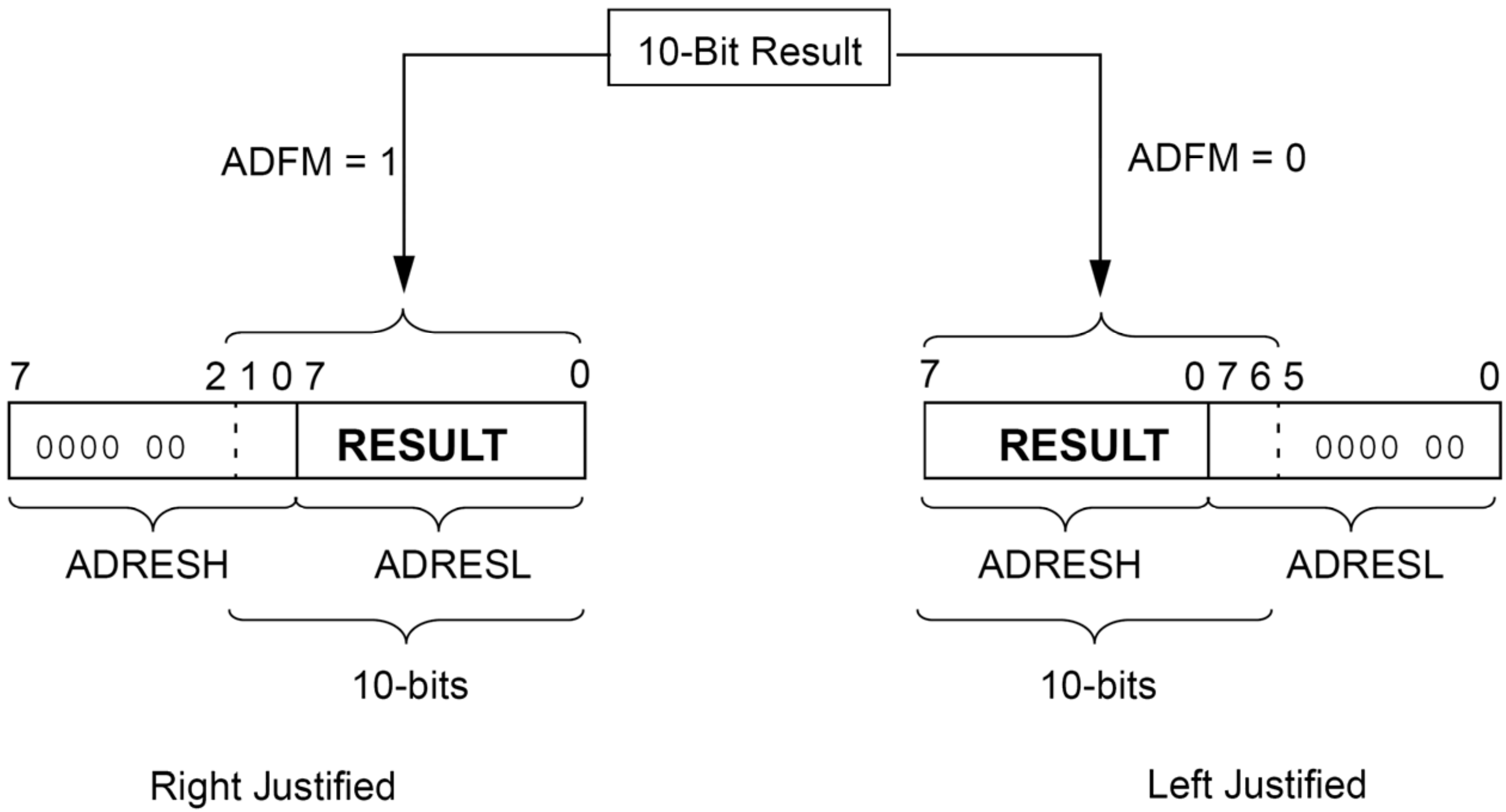
bit 0 **ADON:** A/D Conversion Status bit
1 = A/D converter module is operating
0 = A/D converter is shut-off and consumes no operating current

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

10-BIT A/D RESULT FORMAT



10-BIT A/D RESULT FORMAT



Step 1 (continued)
Configure the A/D module (ADCON1)

- **ADCON1** ... AD Control Register 1 (The Register is shown on the next slide)
  - Selects the A/D conversion clock
- Recall ... that the A/D conversion time per bit is defined as TAD

```
ADCON1 = 0b00010000;       //  Select the Clock as Fosc/8
```

REGISTER 9-3: ADCON1 – A/D CONTROL REGISTER 1 (ADDRESS: 9Fh)

U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	ADCS2	ADCS1	ADCS0	—	—	—	—
bit 7							bit 0

bit 7: **Unimplemented:** Read as '0'

bit 6-4: **ADCS<2:0>:** A/D Conversion Clock Select bits

000 = FOSC/2
001 = FOSC/8
010 = FOSC/32
x11 = FRC (clock derived from a dedicated internal oscillator = 500 kHz max)
100 = FOSC/4
101 = FOSC/16
110 = FOSC/64

bit 3-0: **Unimplemented:** Read as '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

TAD VS. DEVICE OPERATING FREQUENCIES

TABLE 9-1: TAD VS. DEVICE OPERATING FREQUENCIES

A/D Clock Source (TAD)		Device Frequency			
Operation	**ADCS2:ADCS0**	**20 MHz**	**5 MHz**	**4 MHz**	**1.25 MHz**
2 TOSC	000	100 ns(2)	400 ns(2)	500 ns(2)	1.6 μs
4 TOSC	100	200 ns(2)	800 ns(2)	1.0 μs(2)	3.2 μs
8 TOSC	001	400 ns(2)	1.6 μs	2.0 μs	6.4 μs
16 TOSC	101	800 ns(2)	3.2 μs	4.0 μs	12.8 μs(3)
32 TOSC	010	1.6 μs	6.4 μs	8.0 μs(3)	25.6 μs(3)
64 TOSC	110	3.2 μs	12.8 μs(3)	16.0 μs(3)	51.2 μs(3)
A/D RC	x11	2-6 μs(1,4)	2-6 μs(1,4)	2-6 μs(1,4)	2-6 μs(1,4)

Legend: Shaded cells are outside of recommended range.

Note 1: The A/D RC source has a typical TAD time of 4 μs for VDD > 3.0V.

2: These values violate the minimum required TAD time.

3: For faster conversion times, the selection of another clock source is recommended.

4: When the device frequency is greater than 1 MHz, the A/D RC clock source is only recommended if the conversion will be performed during Sleep.

Step 2
Configure the A/D Interrupt (`PIR1, PIE1, INTCON`)

- We will not use the interrupt at this time …
- If we were to use the interrupts … we would …
  - Clear ADIF bit (**`PIR1<6>`**) … A/D Interrupt Flag bit
  - Set ADIE bit (**`PIE1<6>`**) … ADC Interrupt Enable bit
  - Set PEIE and GIE bits (**`INTCON<7:6>`**) …
    - Peripheral Interrupt Enable bit
    - Global Interrupt Enable bit

```
// Not using interrupts
```

REGISTER 2-5: PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EEIF	ADIF	CCP1IF	C2IF	C1IF	OSFIF	TMR2IF	TMR1IF
bit 7							bit 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 7 **EEIF:** EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation has not completed or has not been started

bit 6 **ADIF:** A/D Interrupt Flag bit
1 = A/D conversion complete
0 = A/D conversion has not completed or has not been started

bit 5 **CCP1IF:** CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode

bit 4 **C2IF:** Comparator 2 Interrupt Flag bit
1 = Comparator 2 output has changed (must be cleared in software)
0 = Comparator 2 output has not changed

bit 3 **C1IF:** Comparator 1 Interrupt Flag bit
1 = Comparator 1 output has changed (must be cleared in software)
0 = Comparator 1 output has not changed

bit 2 **OSFIF:** Oscillator Fail Interrupt Flag bit
1 = System oscillator failed, clock input has changed to INTOSC (must be cleared in software)
0 = System clock operating

bit 1 **TMR2IF:** Timer2 to PR2 Match Interrupt Flag bit
1 = Timer2 to PR2 match occurred (must be cleared in software)
0 = Timer2 to PR2 match has not occurred

bit 0 **TMR1IF:** Timer1 Overflow Interrupt Flag bit
1 = Timer1 register overflowed (must be cleared in software)
0 = Timer1 has not overflowed

2.2.2.4 PIE1 Register

The PIE1 register contains the peripheral interrupt enable bits, as shown in Register 2-4.

Note:	Bit PEIE of the INTCON register must be set to enable any peripheral interrupt.

REGISTER 2-4: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EEIE	ADIE	CCP1IE	C2IE	C1IE	OSFIE	TMR2IE	TMR1IE
bit 7							bit 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 7 **EEIE:** EE Write Complete Interrupt Enable bit
1 = Enables the EE write complete interrupt
0 = Disables the EE write complete interrupt

bit 6 **ADIE:** A/D Converter (ADC) Interrupt Enable bit
1 = Enables the ADC interrupt
0 = Disables the ADC interrupt

bit 5 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 4 **C2IE:** Comparator 2 Interrupt Enable bit
1 = Enables the Comparator 2 interrupt
0 = Disables the Comparator 2 interrupt

bit 3 **C1IE:** Comparator 1 Interrupt Enable bit
1 = Enables the Comparator 1 interrupt
0 = Disables the Comparator 1 interrupt

bit 2 **OSFIE:** Oscillator Fail Interrupt Enable bit
1 = Enables the oscillator fail interrupt
0 = Disables the oscillator fail interrupt

bit 1 **TMR2IE:** Timer2 to PR2 Match Interrupt Enable bit
1 = Enables the Timer2 to PR2 match interrupt
0 = Disables the Timer2 to PR2 match interrupt

bit 0 **TMR1IE:** Timer1 Overflow Interrupt Enable bit
1 = Enables the Timer1 overflow interrupt
0 = Disables the Timer1 overflow interrupt

REGISTER 2-3: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GIE	PEIE	T0IE	INTE	RAIE	T0IF	INTF	RAIF
bit 7							bit 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 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE:** Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **T0IE:** Timer0 Overflow Interrupt Enable bit
1 = Enables the Timer0 interrupt
0 = Disables the Timer0 interrupt

bit 4 **INTE:** RA2/INT External Interrupt Enable bit
1 = Enables the RA2/INT external interrupt
0 = Disables the RA2/INT external interrupt

bit 3 **RAIE:** PORTA Change Interrupt Enable bit(1)
1 = Enables the PORTA change interrupt
0 = Disables the PORTA change interrupt

bit 2 **T0IF:** Timer0 Overflow Interrupt Flag bit(2)
1 = Timer0 register has overflowed (must be cleared in software)
0 = Timer0 register did not overflow

bit 1 **INTF:** RA2/INT External Interrupt Flag bit
1 = The RA2/INT external interrupt occurred (must be cleared in software)
0 = The RA2/INT external interrupt did not occur

bit 0 **RAIF:** PORTA Change Interrupt Flag bit
1 = When at least one of the PORTA <5:0> pins changed state (must be cleared in software)
0 = None of the PORTA <5:0> pins have changed state

Note 1: IOCA register must also be enabled.

2: T0IF bit is set when TMR0 rolls over. TMR0 is unchanged on Reset and should be initialized before clearing T0IF bit.

Step 3
Wait the required acquisition time

- After the A/D module has been configured as desired ...
  - the selected channel must be acquired before the conversion is started
- The analog input channels must have their corresponding TRIS bits selected as inputs
- After this sample time has elapsed ...
  - the A/D conversion can be started

Step 4
Start conversion (ADCON0<1>)

- Set GO/DONE bit (**ADCON0<1>**)

  "1" --- Setting this bit starts an A/D conversion cycle
  This bit is automatically cleared when complete

  "0" --- A/D conversion completed/not in progress

Note:	The GO/$\overline{\text{DONE}}$ bit should not be set in the same instruction that turns on the A/D.

```
GODONE = 1;                    // Start the conversion
```

GODONE

- `GODONE` is defined in pic16f684.h which was called by
    - #include pic.h

REGISTER 9-2: ADCON0 – A/D CONTROL REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	VCFG	—	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
bit 7							bit 0

bit 7 **ADFM:** A/D Result Formed Select bit
1 = Right justified
0 = Left justified

bit 6 **VCFG:** Voltage Reference bit
1 = VREF pin
0 = VDD

bit 5 **Unimplemented:** Read as '0'

bit 4-2 **CHS<2:0>:** Analog Channel Select bits
000 = Channel 00 (AN0)
001 = Channel 01 (AN1)
010 = Channel 02 (AN2)
011 = Channel 03 (AN3)
100 = Channel 04 (AN4)
101 = Channel 05 (AN5)
110 = Channel 06 (AN6)
111 = Channel 07 (AN7)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle.
This bit is automatically cleared by hardware when the A/D conversion has completed.
0 = A/D conversion completed/not in progress

bit 0 **ADON:** A/D Conversion Status bit
1 = A/D converter module is operating
0 = A/D converter is shut-off and consumes no operating current

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

Step 5
Wait for A/D conversion to complete

- By either …
  - Polling for the **GO/DONE** bit to be cleared (with interrupts disabled) … OR …
  - Waiting for the A/D interrupt (if enabled)
    - Sets the ADIF flag (**PIR1<6>**)
- We are not using the interrupt … therefore … our Code looks like the following …

```
if (GODONE == 0);                    // ADC Complete
```

Step 6
Results

- Read A/D Result register pair (`ADRESH:ADRESL`)
- If interrupts were enabled …
    - Clear bit ADIF

```
ADC_Value = ADRESH;    // Save Sample Value in "ADC_Value"
```

Step 7
The Next Conversion

- For the next conversion, go back to step 1 or step 2 as required
- A minimum wait of 2 TAD is required before the next acquisition starts

Aborting
a Conversion

- If the conversion must be aborted, the `GO/DONE` bit can be cleared in software
- The `ADRESH`:`ADRESL` registers will not be updated with the partially complete A/D conversion sample …
  - Instead, the `ADRESH`:`ADRESL` registers will retain the value of the previous conversion
- After an aborted conversion … a 2 TAD delay is required before another acquisition can be initiated
  - Following the delay, an input acquisition is automatically started on the selected channel

Now … Let's Walk Through the ADC Process via an Example

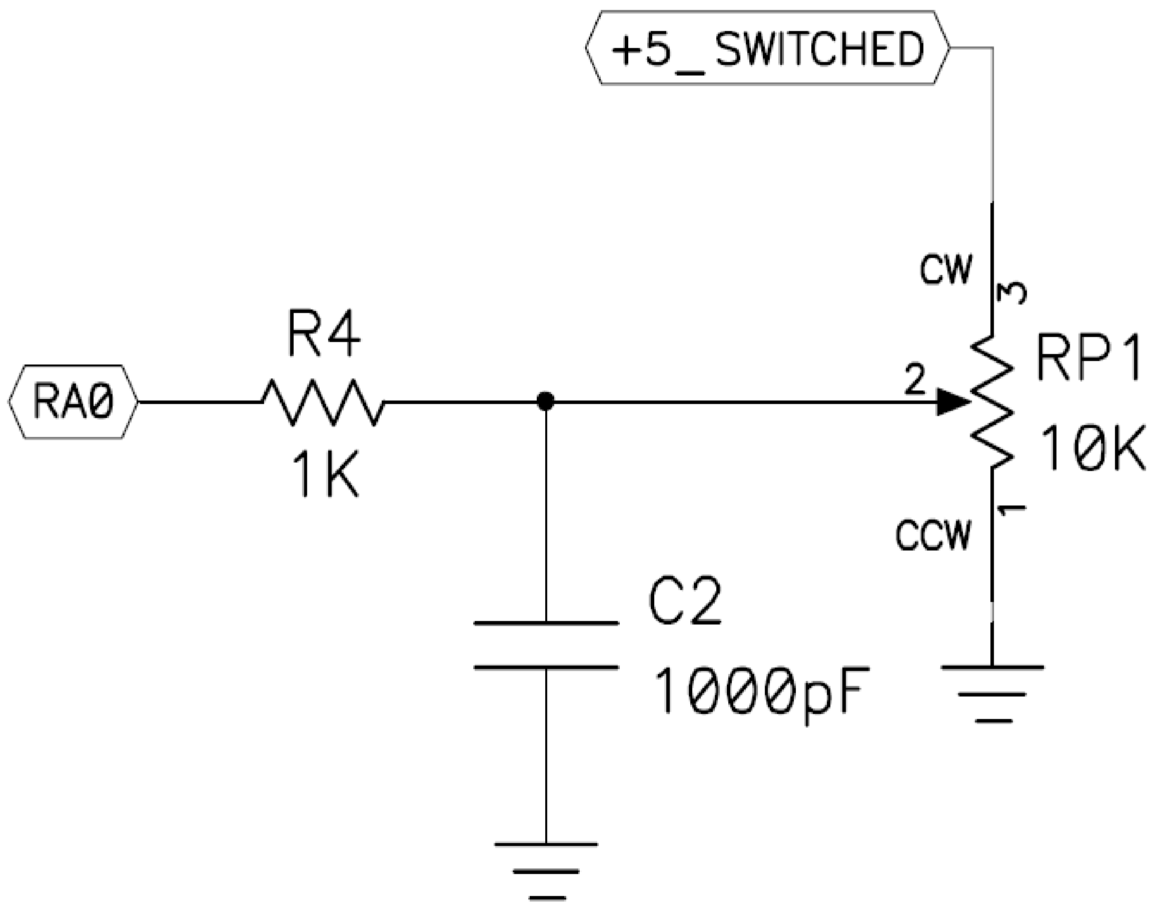
- We shall develop the required software needed for Lab 4
- Lab 4 requires …
    - Analog to Digital Conversion using the PIC16F684 … and …
    - The circuitry of the PICkit 1 Starter Kit
        - LEDs will display the result of the ADC operation
        - Potentiometer `RP1`, connected to `RA0`, will be used to vary the input voltage to the ADC

Let’s Understand the Circuit

- We will use `RA0` as the analog input channel
- The analog voltage is the output/wiper of Potentiometer `RP1`
- LEDs `D0` through `D7` will be used for the results display

- The next slide contains the analog input circuit …

RA0 Analog Input



What the Code is doing …

- Set the initial conditions

- Lights corresponding LEDs (represents the binary equivalent of the analog voltage)

- While in an Endless Loop …
    - Checks to see if the A/D Conversion is complete
        - If it is … Then …
            - » Get the new conversion value
            - » Display the new value
            - » Start a new conversion
        - If still in the conversion
            - » Display the last result

A/D Conversion – "`main`" Program

```
main()
{

   PORTA_init();

   ANSEL = 1;                              //  Just RA0 is an Analog Input
   TRISA0 = 1;                             //  Corresponding TRIS bit is set as input

   ADCON0 = 0b00000001;                    //  Turn on the ADC
                                                      //   Bit 7         - Left Justified Sample
                                                      //   Bit 6         - Use VDD
                                                      //   Bit 4:2       - Channel 0
                                                      //   Bit 1         - Do not Start
                                                      //   Bit 0         - Turn on ADC

   ADCON1 = 0b00010000;                    //  Select the Clock as Fosc/8


   ADC_Disp();
   GODONE = 1;                             // Start A/D Conversion


   while(1 == 1)                           //  Loop Forever
   {

      if (GODONE == 0)                     // Is A/D Conversion complete?
               {  ADC_Disp();              // Display A/D Conversion Results
                  ADC_Value = ADRESH;      // Get new A/D value
                  GODONE = 1;              // Start the next A/D Conversion
                 }
      else                                 // A/D Conversion still in progress
               ADC_Disp();

   }
}
```

PORTA_init();

```
void PORTA_init(void)

{
    PORTA = 0;              //  All PORTA Pins are low
    CMCON0 = 7;             //  Turn off Comparators
    ANSEL = 0;              //  Turn off ADC

    return;

}
```

Configure the A/D module

```
ANSEL = 1;                  //  Just RA0 is an Analog Input
TRISA0 = 1;                 //  Corresponding TRIS bit is set as input

ADCON0 = 0b00000001;        //  Turn on the ADC
                            //   Bit 7          - Left Justified Sample
                            //   Bit 6          - Use VDD
                            //   Bit 4:2        - Channel 0
                            //   Bit 1          - Do not Start
                            //   Bit 0          - Turn on ADC

ADCON1 = 0b00010000;        //  Select the Clock as Fosc/8

ADC_Disp();
GODONE = 1;                 // Start A/D Conversion
```

PORTA and TRISA

```
const char PORTA_Value[8] = {
          0b010000,        // D0
          0b100000,        // D1
          0b010000,        // D2
          0b000100,        // D3
          0b100000,        // D4
          0b000100,        // D5
          0b000100,        // D6
          0b000010};       // D7

const char TRISA_Value[8] = {
          0b001111,        // D0
          0b001111,        // D1
          0b101011,        // D2
          0b101011,        // D3
          0b011011,        // D4
          0b011011,        // D5
          0b111001,        // D6
          0b111001};       // D7
```

while(1 == 1) // Loop Forever

```
while(1 == 1)                     //  Loop Forever
 {

    if (GODONE == 0)              // Is A/D Conversion complete?
    {
        ADC_Disp();               // Display A/D Conversion Results
        ADC_Value = ADRESH;       // Get new A/D value
        GODONE = 1;               // Start the next A/D Conversion
    }

    else                          // A/D Conversion still in progress
        ADC_Disp();

 }
```

ADC_Disp();

```
void ADC_Disp(void)
{
    int i;

    for (i = 0; i < 8; i++ )
       {                                    // Loop through Each of the 8 LEDS

             Delay_LED_On();    // Allows time for individual LEDs to light

             if ((ADC_Value & (1 << i)) == 0)
                 PORTA = 0;
             else
                 PORTA = PORTA_Value[i];
                TRISA = TRISA_Value[i];
       }
    return;
}
```

Programming / Software

Programming

- Commands/instructions that we will encounter tonight
  - C commands - ***No new commands tonight***
  - PIC16F684 control – ***See prior section***

Lab

Lab

- Finish Lab #3
- How are you making out?
- What is your approach?

Next Class ...

Next Class Topics

- The class of 11/02/10 …

    - LCD Display Interface and Assembly Language

HomeWork

Homework

1. Complete Lab #3

2. Lab #3 Report … due November 09, 2010

3. Read the sections on the PIC16F684 ADC module in both the …

    PICmicro Mid-Range MCU Family Reference Manual

    PIC16F684 Data Sheet

Time To ....

Start the Lab ....

References

1. PIC16F684 Data Sheet 41202F