Analog to Digital World

TM4C123GH Tiva C Board

Dr. Munesh Singh

Indian Institute of Information Technology Design and Manufacturing
Kancheepuram, Chennai, Tamil Nadu

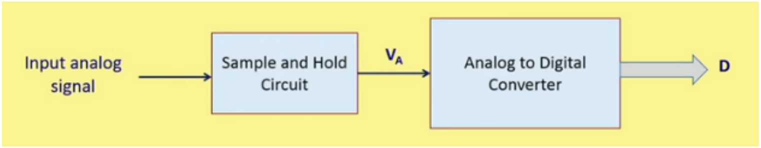
February 10, 2020

Introduction

- An analog-to-digital converter (ADC) takes an analog input V_A as input, and generates a digital word D as output, where $D \propto V_A$
- Various types of ADC are possible:
    - Flash-type ADC
    - Counter-type ADC
    - Tracking-type ADC
    - Successive approximation ADC
    - Dual-slope ADC

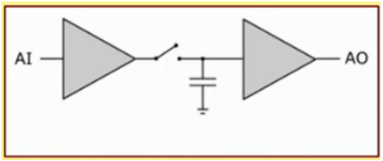
Overall Schematic

- If the input signal changes during the conversion, the final value D may be erroneous.
- We need a sample-and-hold circuit to sample the input voltage, and hold it to a constant value while the conversion is in progress.



Sample-and-Hold Circuit

- The input voltage AI is sampled by closing the switch; the capacitor charges to the voltage.
- During conversion, the switch is opened again, so that AO remain reasonably constant.



How Fast We Must Smaple?

- A very important decision.
  - Guided by Nyquist Theorem.
- What is Nyquist Theorem?
  - A band-limited analog signal that has been sampled can be perfectly reconstructed from an infinite sequence of samples if the sampling rate f_s exceeds $2f_{max}$ samples per second, where f_{max} is the highest frequency in the original signal.
    - If the analog signal does contain frequency components larger that $f_s/2$, then there will be an aliasing error.
    - Aliasing is when the digital signal appears to have a different frequency than the original analog signal.

Resolution

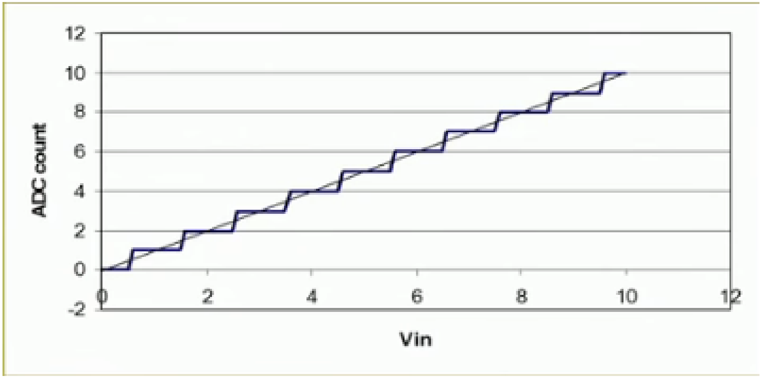
- The least significant bit of an analog-to-digital converter (ADC) gives the resolution.
- Related to full scale if the ADC is linear.
  - $\mathrm{LSB}=A/2^n$ for n-bit ADC.
  - For a linear 8-bit ADC with a 1V full scale input,
    Resolution $= 1.0/2^8 = 3.9$ mV
    percentage Resolution= Resolution/(Full-scale voltage)*100%

Conversion Time and Bandwidth

- How often can a conversion be done?
  - A few ns to a few ms depending on the technology.
- Input Bandwidth
  - Maximum input signal bandwidth
    - Sample and hold input circuitry
    - Conversion Frequency

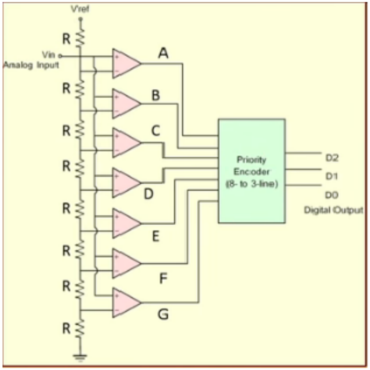
ADC Transfer Characterisics

- This curve is for ideal ADC
- Possible errors:
  - Offset
  - Non-linearity



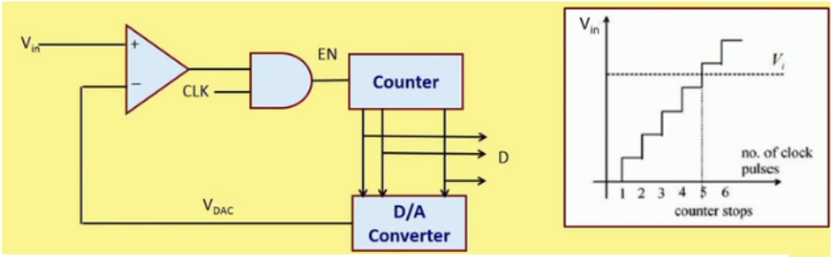
Flash-type ADC

- This type of ADC provides the fastest conversion.
- Requires large amount of hardware.
    - For an n-bit converter, we require $2^n - 1$ comparators, 2^n resistors, and one 2^n -input priority encoder.
- We illustrate the design for a 3-bit ADC



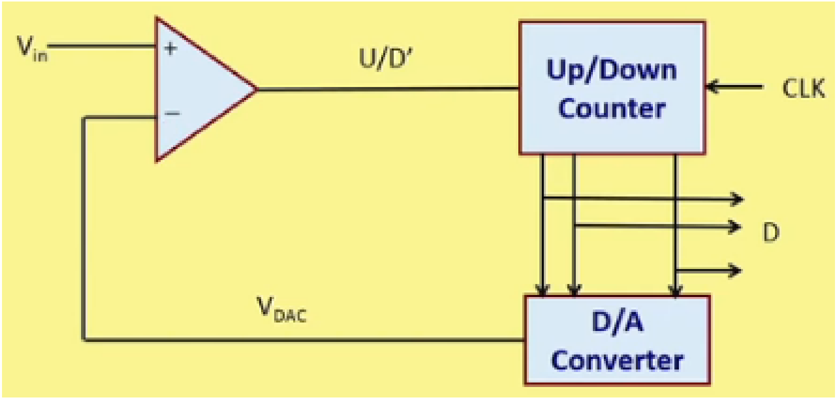
A	B	C	D	E	F	G	D2	D1	D0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	X	0	1	0
0	0	0	0	1	X	X	0	1	1
0	0	0	1	X	X	X	1	0	0
0	0	1	X	X	X	X	1	0	1
0	1	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	1	1	1

Counter-Type ADC



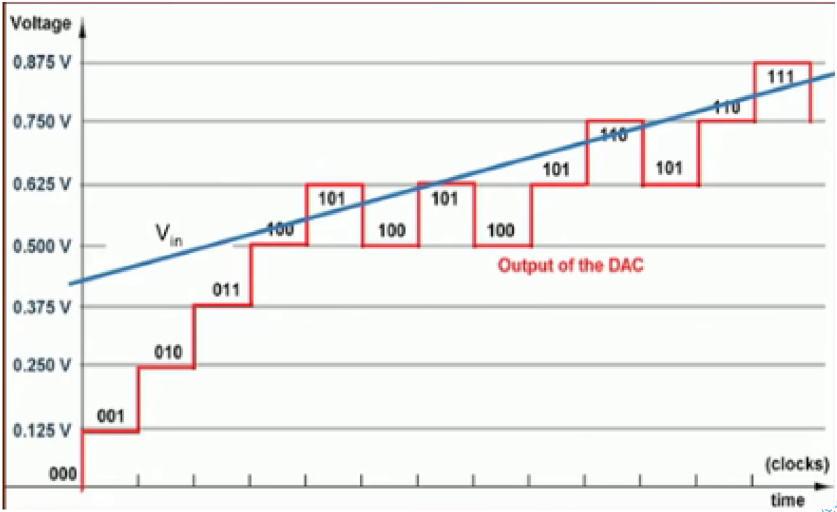
- We use a D/A converter, a counter and a comparator.
  - Before a conversion starts, the counter is reset to zero.
  - The counter output is fed to the input of the DAC.
  - The DAC output V_{DAC} is compared with the input analog voltage V_{in}.
  - if $V_{in} > V_{DAC}$, then the counter is incremented by 1.
  - if $V_{in} < V_{DAC}$, the conversion is complete, and the counter output D gives the required digital output.
- The worst-case conversion time is equal to 2^n

Tracking-Type ADC



Tracking-Type ADC

- Here we use a binary up-down counter.
  - if the output of the comparator is 1 ($V_{in} > V_{DAC}$), the counter is incremented by 1.)
  - if the output of the comparator is 0 ($V_{in} < V_{DAC}$), the counter is decremented by 1.
  - The counter output D continuously tracks the input analog voltage.
- Suitable for continuous conversion of slowly-varying waveforms.
  - if V_{in} changes too fast, the counter may not be able to track the changes.
  - Here also, the worst-case conversion time is 2^n.


Voltage
0.875 V
0.750 V
0.625 V
0.500 V
0.375 V
0.250 V
0.125 V
000
001
010
011
100
101
100
101
100
101
110
101
110
111
V_{in}
Output of the DAC
(clocks)
time

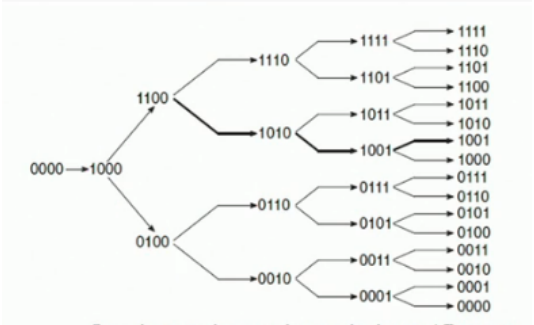
Sucessive Approximation Type ADC

- The principle of operation is analogous to **binary search**.
  - Suppose we are searching for a number (here V_{in}) in a sorted list.
  - We look into the middle of the list-effectively the size of the list reduces by half.
  - For 2^n steps, the number of the steps required is only n.

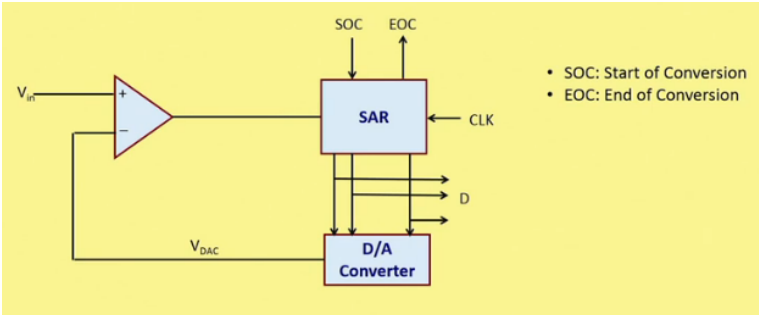
Sucessive Approximation Type ADC

- The principle of operation is analogous to **binary search**.
  - Suppose we are searching for a number (here V_{in}) in a sorted list.
  - We look into the middle of the list-effectively the size of the list reduces by half.
  - For 2^n steps, the number of the steps required is only n.
- What modifications are required?
  - We use a successive approximation register (SAR) instead of a counter.
  - For example, in 4 bits, it start with 1000 (i.e 8).
    - If the input is smaller, the next value is set as 0100 (i.e. 4).
    - If the input is larger, the next value is set as 1100 (i.e. 12).

- An example for a 4-bit ADC, where the expected digital output is 1001.
- Only 4 comparison steps are required, once per bit position.



Conversion process in a successive approximation type A/D converter.

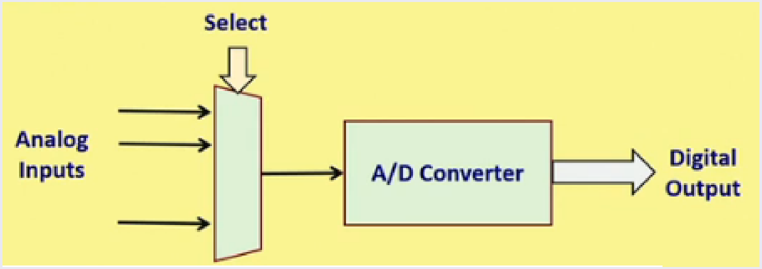


- SOC: Start of Conversion
- EOC: End of Conversion

A/D Conversion in ARM Microcontrollers

- ADC in TM4C123GH ARM Microcontroller
  - Built-in 10-bit successive-approximation ADC.
  - With V_{REF}=3.3V, step size = 3.3/($2^1$2-1) =0.85mV
  - This determines the accuracy of the measurement.
- Some specific details:
  - Two built-in ADC modules, ADC0 and ADC1.
  - ADCO and ADC1 has 12 analog input channels.

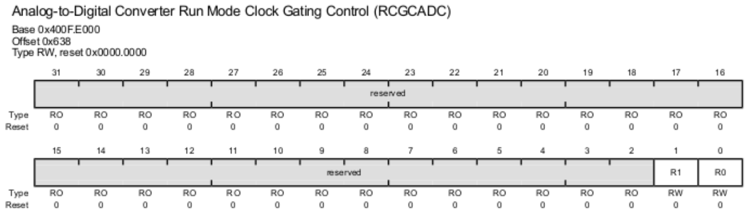
- There is a Control Register that must be written to determine the operating mode.
- Interrupt is generated after A/D conversion is complete.
- An analog multiplexer is used to select one of the input analog channel at the input of an ADC.



- ADC in ARM Cortex-M4
  - TM4C123GH micro-controllers have up to 2 in-buit 12-bit A/D converters, each of which has 12 channels.
  - With $V_{REF} = 3.3V$, step size =3.3/($2^1$2-1)=0.85mV
  - These two ADC modules provide the following features:
    - 12 shared analog input channels
    - 12-bit precision ADC
    - Single-ended and differential-input configurations
    - On-chip internal temperature sensor
    - Maximum sample rate of 1 million samples/sec (MSPS)
    - Four programmable sample conversion sequencers
    - Five flexible trigger controls (software trigger control (default), Timers, Analog Comparators, PWM, GPIO)
    - Hardware averaging of up to 64 samples
    - 2 Analog Comparators

ADC Programming with the Tiva TM4C123G

Enabling Clock to ADC



Analog-to-Digital Converter Run Mode Clock Gating Control (RCGCADC)

Base 0x400F.E000
Offset 0x638
Type RW, reset 0x0000.0000

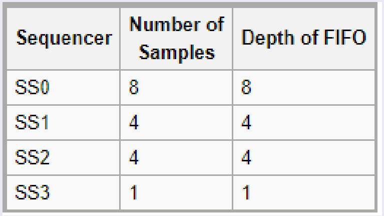
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

bit	Name	Description
0	R0	0: ADC Module 0 is disabled, 1: Enable and provide a clock to ADC module 0
1	R1	0: ADC Module 1 is disabled, 1: Enable and provide a clock to ADC module 1

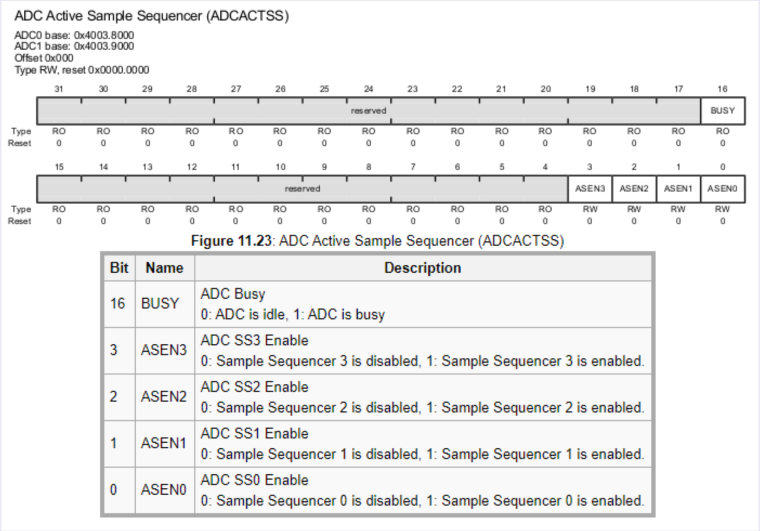
The Sample Sequencer

- The Sample Sequencer is a part of the ADC module that moves the conversion result of the ADC to one of the FIFOs.



Sequencer	Number of Samples	Depth of FIFO
SS0	8	8
SS1	4	4
SS2	4	4
SS3	1	1

The Sample Sequencer Enabling



ADC Active Sample Sequencer (ADCACTSS)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x000
Type RW, reset 0x0000.0000

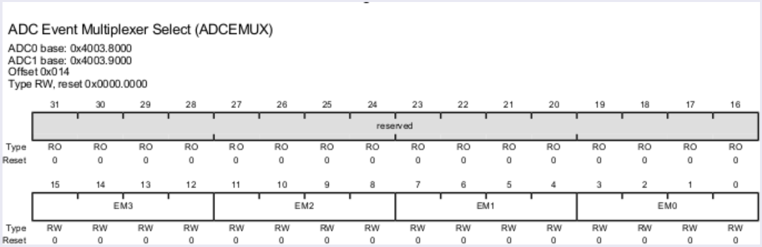
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															BUSY
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												ASEN3	ASEN2	ASEN1	ASEN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11.23: ADC Active Sample Sequencer (ADCACTSS)

Bit	Name	Description
16	BUSY	ADC Busy 0: ADC is idle, 1: ADC is busy
3	ASEN3	ADC SS3 Enable 0: Sample Sequencer 3 is disabled, 1: Sample Sequencer 3 is enabled.
2	ASEN2	ADC SS2 Enable 0: Sample Sequencer 2 is disabled, 1: Sample Sequencer 2 is enabled.
1	ASEN1	ADC SS1 Enable 0: Sample Sequencer 1 is disabled, 1: Sample Sequencer 1 is enabled.
0	ASEN0	ADC SS0 Enable 0: Sample Sequencer 0 is disabled, 1: Sample Sequencer 0 is enabled.

Start Conversion trigger options


ADC Event Multiplexer Select (ADCEMUX)
ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x014
Type RW, reset 0x0000.0000

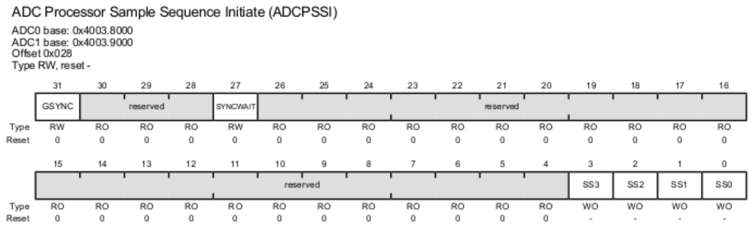
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EM3				EM2				EM1				EM0			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- EMx bits select the trigger source for Sample Sequencer x

EMx Value	Trigger source
0x0	Processor (default)
0x1	Analog Comparator 0
0x2	Analog Comparator 1
0x3	reserved
0x4	External (GPIO Pins)
0x5	Timer
0x6	PWM Generator 0
0x7	PWM Generator 1
0x8	PWM Generator 2
0x9	PWM Generator 3
0xF	Always (continuously sample)

The Sample Sequence Intiate



ADC Processor Sample Sequence Initiate (ADCPSSI)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x028
Type RW, reset -

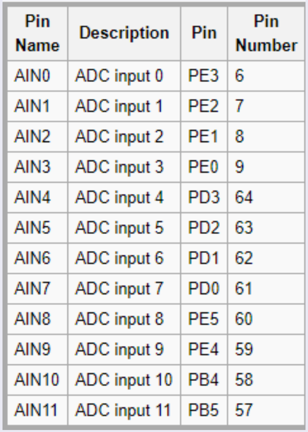
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	GSYNC	reserved			SYNCWAIT	reserved										
Type	RW	RO	RO	RO	RW	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												SS3	SS2	SS1	SS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-

SSx Value	Trigger source
0	No Effect
1	Begin sampling on sample sequence x

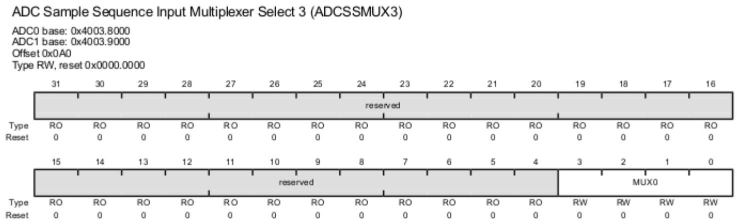
Choosing Vin input channel

- The channel selection is done through the ADCSSMUXn (n=0, 1, 2, 3) registers
- For the SS3, the ADCSSMUX3 is used



Pin Name	Description	Pin	Pin Number
AIN0	ADC input 0	PE3	6
AIN1	ADC input 1	PE2	7
AIN2	ADC input 2	PE1	8
AIN3	ADC input 3	PE0	9
AIN4	ADC input 4	PD3	64
AIN5	ADC input 5	PD2	63
AIN6	ADC input 6	PD1	62
AIN7	ADC input 7	PD0	61
AIN8	ADC input 8	PE5	60
AIN9	ADC input 9	PE4	59
AIN10	ADC input 10	PB4	58
AIN11	ADC input 11	PB5	57

Choosing Vin input channel



ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0A0
Type RW, reset 0x0000.0000

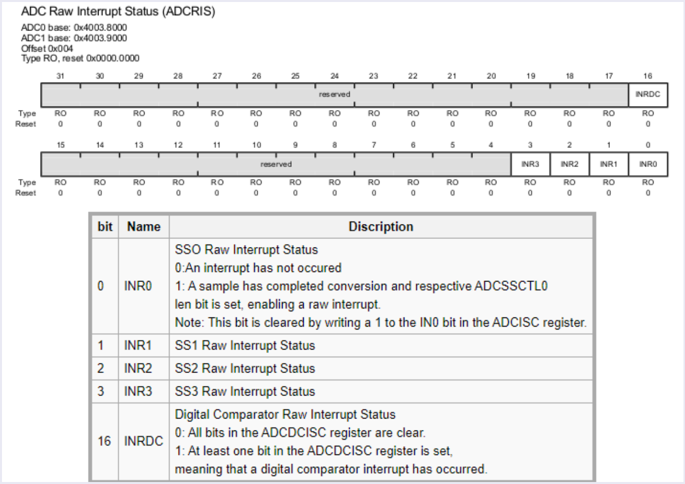
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												MUX0			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

bit	Name	Desription
0-3	MUX0	Sample Input Select

Polling or interrupt

- The end-of-conversion is indicated by a flag bit in the ADCRIS (ADC Raw Interrupt) register



ADC Raw Interrupt Status (ADCRIS)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x004
Type RO, reset 0x0000.0000

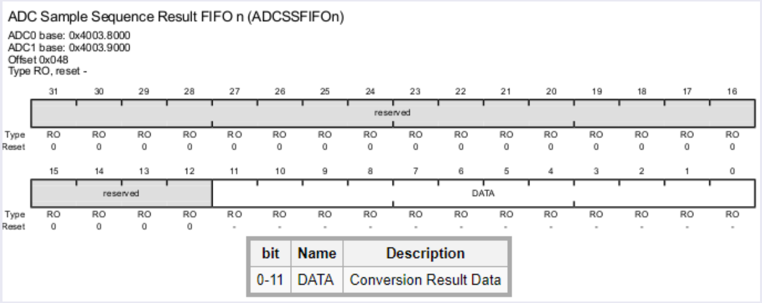
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								reserved								INRDC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						reserved							INR3	INR2	INR1	INR0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

bit	Name	Discription
0	INR0	SSO Raw Interrupt Status 0:An interrupt has not occured 1: A sample has completed conversion and respective ADCSSCTL0 Ien bit is set, enabling a raw interrupt. Note: This bit is cleared by writing a 1 to the IN0 bit in the ADCISC register.
1	INR1	SS1 Raw Interrupt Status
2	INR2	SS2 Raw Interrupt Status
3	INR3	SS3 Raw Interrupt Status
16	INRDC	Digital Comparator Raw Interrupt Status 0: All bits in the ADCDCISC register are clear. 1: At least one bit in the ADCDCISC register is set, meaning that a digital comparator interrupt has occurred.

ADC Data result

- Upon the completion of conversion, the binary result is placed in the ADCSSFIFOn register
- Since we are using SS3, we need to read the result from ADCSSFIFO3 register.



ADC Sample Sequence Result FIFO n (ADCSSFIFOn)
ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x048
Type RO, reset -

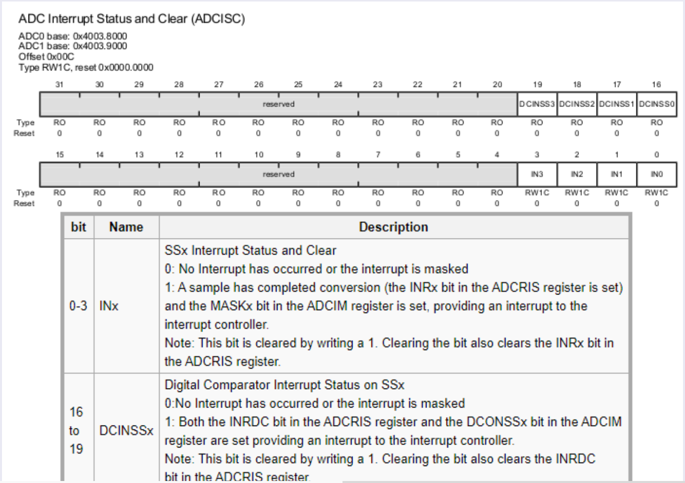
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				DATA											
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-

bit	Name	Description
0-11	DATA	Conversion Result Data

Clearing end-of-conversion flag

- After reading the data from the ADCSSFIFOx register, we must clear the INR3 flag bit in ADCRIS register so that we may detect another conversion complete.



ADC Interrupt Status and Clear (ADCISC)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x00C
Type RW1C, reset 0x0000.0000

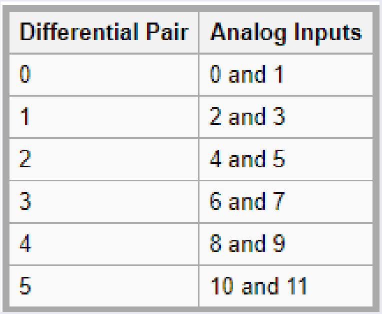
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												DCINSS3	DCINSS2	DCINSS1	DCINSS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												IN3	IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW1C	RW1C	RW1C	RW1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

bit	Name	Description
0-3	INx	SSx Interrupt Status and Clear 0: No Interrupt has occurred or the interrupt is masked 1: A sample has completed conversion (the INRx bit in the ADCRIS register is set) and the MASKx bit in the ADCIM register is set, providing an interrupt to the interrupt controller. Note: This bit is cleared by writing a 1. Clearing the bit also clears the INRx bit in the ADCRIS register.
16 to 19	DCINSSx	Digital Comparator Interrupt Status on SSx 0:No Interrupt has occurred or the interrupt is masked 1: Both the INRDC bit in the ADCRIS register and the DCONSSx bit in the ADCIM register are set providing an interrupt to the interrupt controller. Note: This bit is cleared by writing a 1. Clearing the bit also clears the INRDC bit in the ADCRIS register

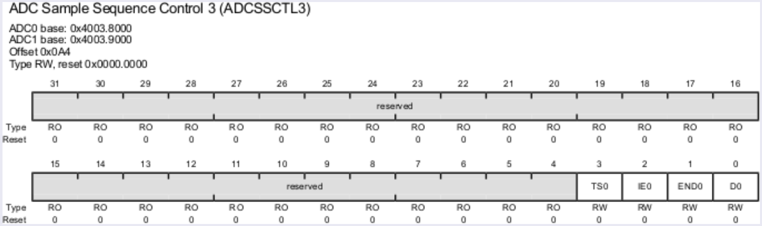
Differential versus Single-Ended

- In some applications, our interest is in the differences between two analog signal voltages (the differential voltages)
- The bit 0 of ADCSSCTL3 (ADC Sample sequence Control 3) register allows us to enable the differential option.



Differential Pair	Analog Inputs
0	0 and 1
1	2 and 3
2	4 and 5
3	6 and 7
4	8 and 9
5	10 and 11

Differential versus Single-Ended

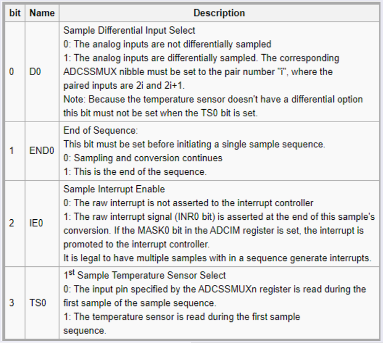


ADC Sample Sequence Control 3 (ADCSSCTL3)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x0A4
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

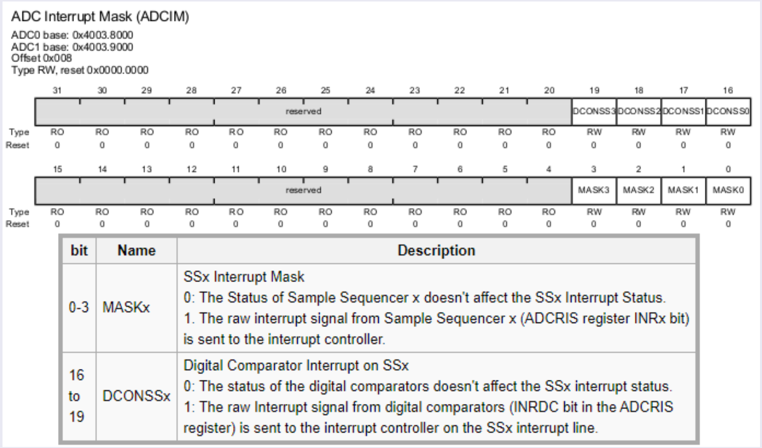
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												TS0	IE0	END0	D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



bit	Name	Description
0	D0	Sample Differential Input Select 0: The analog inputs are not differentially sampled 1: The analog inputs are differentially sampled. The corresponding ADCSSMUX nibble must be set to the pair number "i", where the paired inputs are 2i and 2i+1. Note: Because the temperature sensor doesn't have a differential option this bit must not be set when the TS0 bit is set.
1	END0	End of Sequence: This bit must be set before initiating a single sample sequence. 0: Sampling and conversion continues 1: This is the end of the sequence.
2	IE0	Sample Interrupt Enable 0: The raw interrupt is not asserted to the interrupt controller 1: The raw interrupt signal (INR0 bit) is asserted at the end of this sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller. It is legal to have multiple samples with in a sequence generate interrupts.
3	TS0	1st Sample Temperature Sensor Select 0: The input pin specified by the ADCSSMUXn register is read during the first sample of the sample sequence. 1: The temperature sensor is read during the first sample sequence.

Masking interrupt for SS3

- Since we are using polling for the end-of-conversion, we must mask the interrupt option for the SS3 to prevent it from interrupting us via NVIC.



ADC Interrupt Mask (ADCIM)

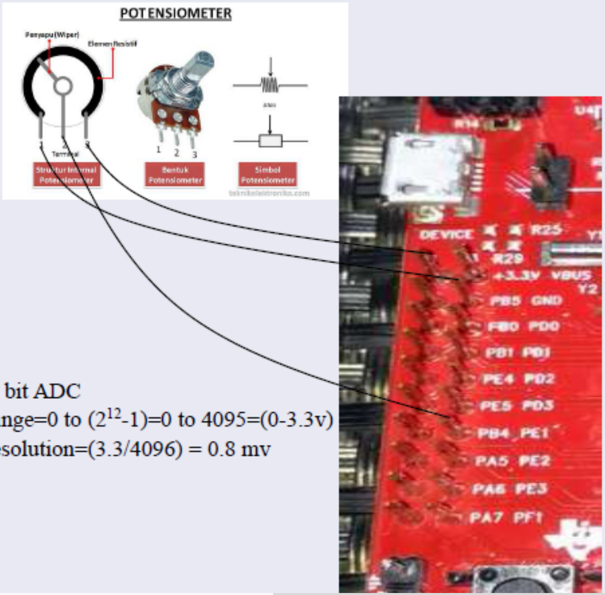
ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x008
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												DCONSS3	DCONSS2	DCONSS1	DCONSS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												MASK3	MASK2	MASK1	MASK0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

bit	Name	Description
0-3	MASKx	SSx Interrupt Mask 0: The Status of Sample Sequencer x doesn't affect the SSx Interrupt Status. 1. The raw interrupt signal from Sample Sequencer x (ADCRIS register INRx bit) is sent to the interrupt controller.
16 to 19	DCONSSx	Digital Comparator Interrupt on SSx 0: The status of the digital comparators doesn't affect the SSx interrupt status. 1: The raw Interrupt signal from digital comparators (INRDC bit in the ADCRIS register) is sent to the interrupt controller on the SSx interrupt line.

Connection Diagram

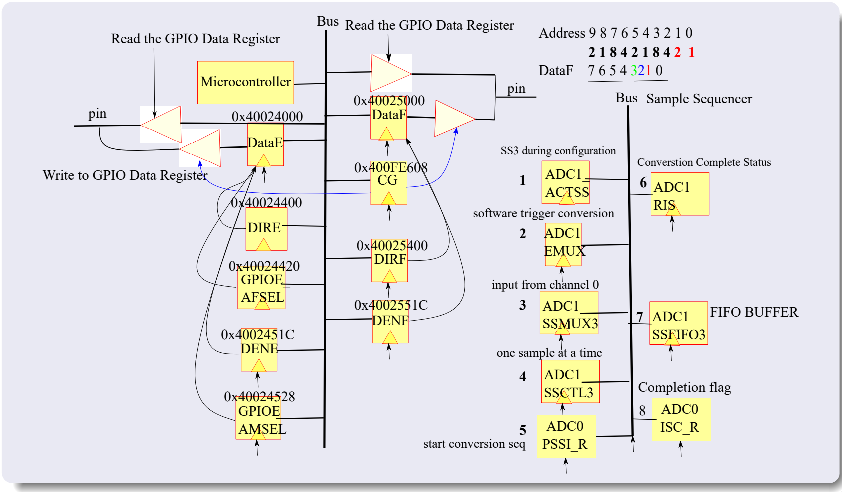


12 bit ADC

Range=0 to (2^{12}-1)=0 to 4095=(0-3.3v)

Resolution=(3.3/4096) = 0.8 mv

Initialization of Analog to Digital Convertor



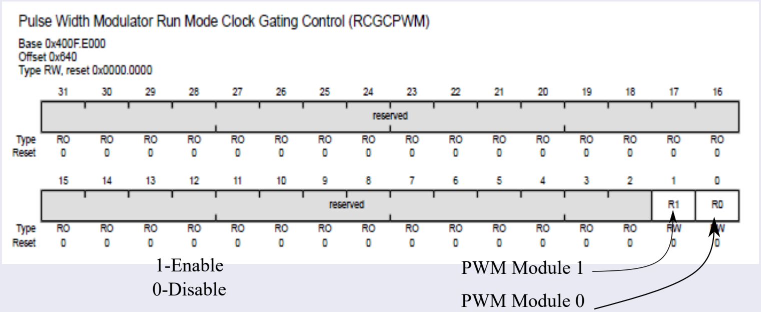
Analog to Digital Control

```
#include "tm4c123gh6pm.h"
int main(){
  volatile int result;
  *((volatile unsigned int *)0x400FE608)=(1<<4)|(1<<5);  /* clock gating portE*/
  *((volatile unsigned int *)0x400FE638)|=1; /* analog clock gating */
  *((volatile unsigned int *)0x40025400)=0xFFU; /* portF direction register*/
  *((volatile unsigned int *)0x4002551C)=0xFFU; /* portF digital function register*/
  GPIO_PORTE_AFSEL_R |= 8;      /* enable alternate function */
  GPIO_PORTE_DIR_R &= ~8;       /* disable digital function */
  GPIO_PORTE_AMSEL_R |= 8;      /* enable analog function */
  ADC0_ACTSS_R &=  ~8;       /* disable SS3 during configuration */
  ADC0_EMUX_R &= ~0xF000;    /* software trigger conversion */
  ADC0_SSMUX3_R = 0;        /* get input from channel 0 */
  ADC0_SSCTL3_R |= 6;       /* take one sample at a time, set flag at 1st sample */
  ADC0_ACTSS_R |= 8;        /* enable ADC0 sequencer 3 */

   while(1)   {
      ADC0_PSSI_R |= 8;       /* start a conversion sequence 3 */
      if((ADC0_RIS_R & 8) == 0)    /* wait for conversion complete */
      {
      result = ADC0_SSFIFO3_R; /* read conversion result */
      if(result > 4000)
       *((voltaile unsigned int *)0x400253FC) |= (1<<1);
      else
        *((voltaile unsigned int *)0x400253FC)&= ~(1<<1);
      }
     ADC0_ISC_R = 8;         /* clear completion flag */
   }
  return 0;
}
```

PWM Initialization & Configuration

Enable the clock signal for PWM module by setting its corresponding bit in the SYSCTL_RCGCPWM register



Pulse Width Modulator Run Mode Clock Gating Control (RCGCPWM)

Base 0x400F.E000
Offset 0x640
Type RW, reset 0x0000.0000

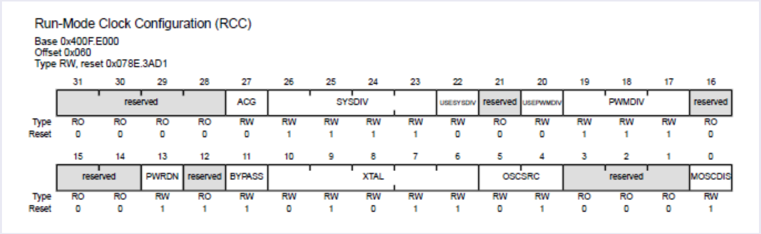
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1-Enable
0-Disable
PWM Module 1
PWM Module 0

PWM Initialization & Configuration

- Enable the PWM clock by writing a value of 0x0010.0000 to the RCGC0 register in the System Control module.
- Configure the Run-Mode Clock Configuration (RCC) register in the System Control module to use the PWM divide (USEPWMDIV) and set the divider (PWMDIV) to divide by 2 (000).



Run-Mode Clock Configuration (RCC)

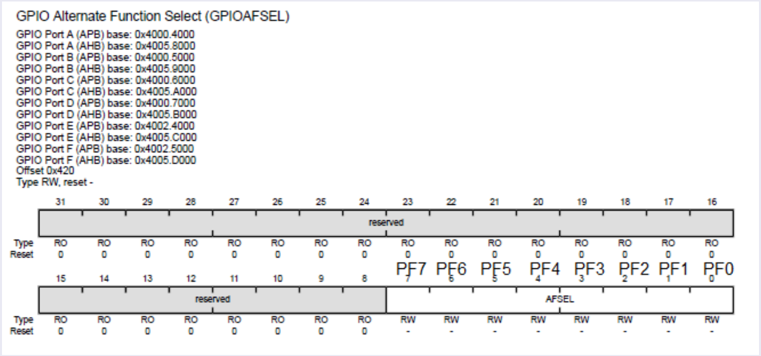
Base 0x400F.E000
Offset 0x060
Type RW, reset 0x078E.3AD1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				ACG	SYSDIV				USESYSDIV	reserved	USEPWMDIV	PWMDIV			reserved
Type	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RO	RW	RW	RW	RW	RO
Reset	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		PWRDN	reserved	BYPASS	XTAL					OSCSRC		reserved			MOSCDIS
Type	RO	RO	RW	RO	RW	RW	RW	RW	RW	RW	RW	RW	RO	RO	RO	RW
Reset	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1

PWM Initialization & Configuration

- Configure the GPIOAFSEL register to program each bit as a GPIO or alternate pin. If an alternate pin is chosen for a bit, then the PMCx field must be programmed in the GPIOPCTL register for the specific peripheral required.



GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x420
Type RW, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
	reserved								AFSEL							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

PWM Initialization & Configuration

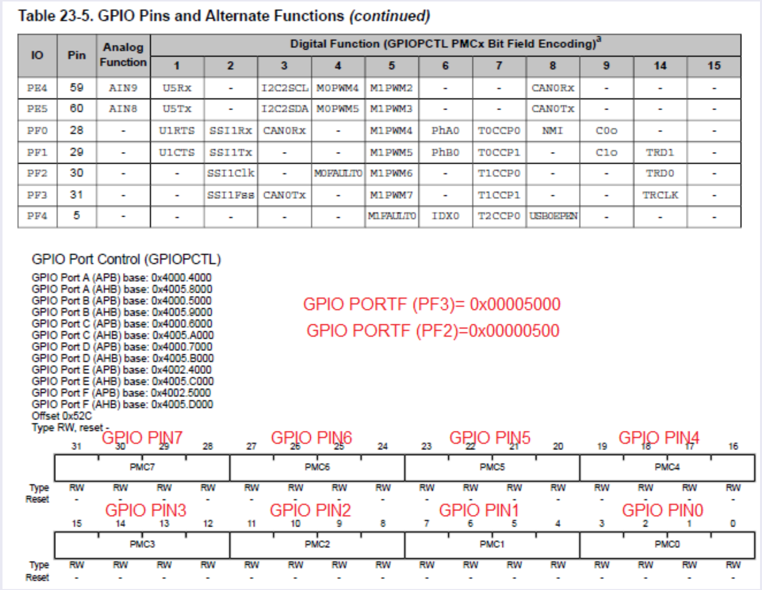


Table 23-5. GPIO Pins and Alternate Functions *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PE4	59	AIN9	U5Rx	-	I2C2SCL	M0PWM4	M1PWM2	-	-	CAN0Rx	-	-	-
PE5	60	AIN8	U5Tx	-	I2C2SDA	M0PWM5	M1PWM3	-	-	CAN0Tx	-	-	-
PF0	28	-	U1RTS	SSI1Rx	CAN0Rx	-	M1PWM4	PhA0	T0CCP0	NMI	C0o	-	-
PF1	29	-	U1CTS	SSI1Tx	-	-	M1PWM5	PhB0	T0CCP1	-	C1o	TRD1	-
PF2	30	-	-	SSI1Clk	-	M0FAULT0	M1PWM6	-	T1CCP0	-	-	TRD0	-
PF3	31	-	-	SSI1Fss	CAN0Tx	-	M1PWM7	-	T1CCP1	-	-	TRCLK	-
PF4	5	-	-	-	-	-	M1FAULT0	IDX0	T2CCP0	USB0EPEN	-	-	-

GPIO Port Control (GPIOPCTL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x52C
Type RW, reset -

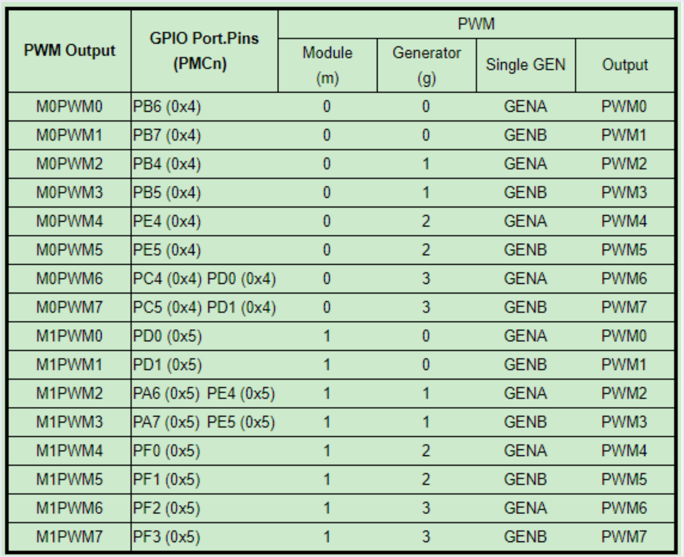
GPIO PORTF (PF3)= 0x00005000

GPIO PORTF (PF2)=0x00000500

	GPIO PIN7				GPIO PIN6				GPIO PIN5				GPIO PIN4			
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PMC7				PMC6				PMC5				PMC4			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

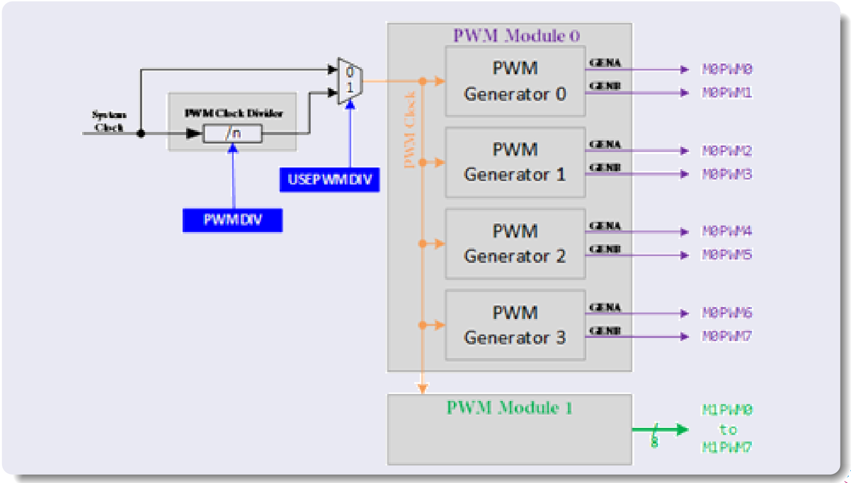
	GPIO PIN3				GPIO PIN2				GPIO PIN1				GPIO PIN0			
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PMC3				PMC2				PMC1				PMC0			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

PWM Initialization & Configuration

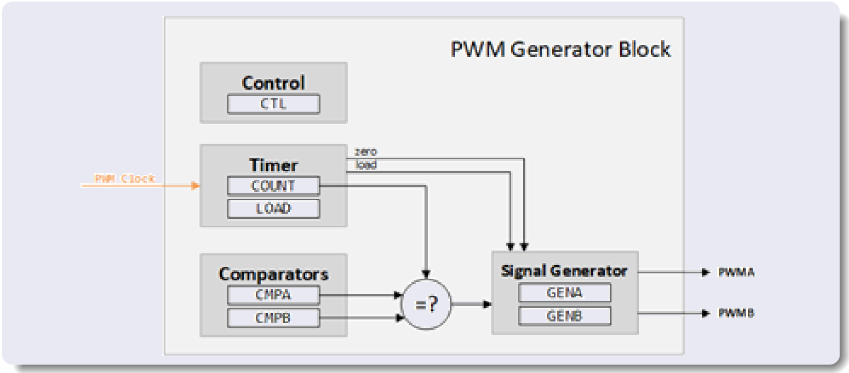


PWM Output	GPIO Port.Pins (PMCn)	PWM			
		Module (m)	Generator (g)	Single GEN	Output
M0PWM0	PB6 (0x4)	0	0	GENA	PWM0
M0PWM1	PB7 (0x4)	0	0	GENB	PWM1
M0PWM2	PB4 (0x4)	0	1	GENA	PWM2
M0PWM3	PB5 (0x4)	0	1	GENB	PWM3
M0PWM4	PE4 (0x4)	0	2	GENA	PWM4
M0PWM5	PE5 (0x4)	0	2	GENB	PWM5
M0PWM6	PC4 (0x4) PD0 (0x4)	0	3	GENA	PWM6
M0PWM7	PC5 (0x4) PD1 (0x4)	0	3	GENB	PWM7
M1PWM0	PD0 (0x5)	1	0	GENA	PWM0
M1PWM1	PD1 (0x5)	1	0	GENB	PWM1
M1PWM2	PA6 (0x5) PE4 (0x5)	1	1	GENA	PWM2
M1PWM3	PA7 (0x5) PE5 (0x5)	1	1	GENB	PWM3
M1PWM4	PF0 (0x5)	1	2	GENA	PWM4
M1PWM5	PF1 (0x5)	1	2	GENB	PWM5
M1PWM6	PF2 (0x5)	1	3	GENA	PWM6
M1PWM7	PF3 (0x5)	1	3	GENB	PWM7

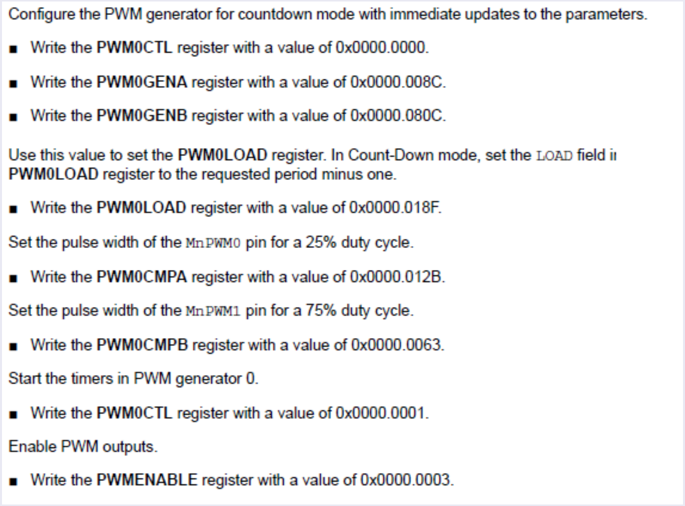
PWM Initialization & Configuration



PWM Initialization & Configuration



PWM Initialization & Configuration



Configure the PWM generator for countdown mode with immediate updates to the parameters.

- Write the **PWM0CTL** register with a value of 0x0000.0000.
- Write the **PWM0GENA** register with a value of 0x0000.008C.
- Write the **PWM0GENB** register with a value of 0x0000.080C.

Use this value to set the **PWM0LOAD** register. In Count-Down mode, set the `LOAD` field ir **PWM0LOAD** register to the requested period minus one.

- Write the **PWM0LOAD** register with a value of 0x0000.018F.

Set the pulse width of the `MnPWM0` pin for a 25% duty cycle.

- Write the **PWM0CMPA** register with a value of 0x0000.012B.

Set the pulse width of the `MnPWM1` pin for a 75% duty cycle.

- Write the **PWM0CMPB** register with a value of 0x0000.0063.

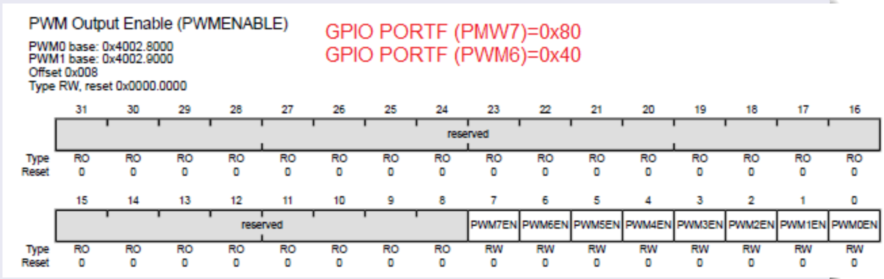
Start the timers in PWM generator 0.

- Write the **PWM0CTL** register with a value of 0x0000.0001.

Enable PWM outputs.

- Write the **PWMENABLE** register with a value of 0x0000.0003.

PWM Initialization & Configuration



PWM Output Enable (PWMENABLE)

PWM0 base: 0x4002.8000
PWM1 base: 0x4002.9000
Offset 0x008
Type RW, reset 0x0000.0000

GPIO PORTF (PMW7)=0x80
GPIO PORTF (PWM6)=0x40

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PWM7EN	PWM6EN	PWM5EN	PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PWM Program to Control Brightness of LED

```
#include <stdint.h>
#include "tm4c123gh6pm.h"

void delayMs(int n);
int main(void){
   int x = 15999;

   /* Enable Peripheral Clocks */
   SYSCTL_RCGCPWM_R |= 2;       /* enable clock to PWM1 */
   SYSCTL_RCGCGPIO_R |= 0x20;   /* enable clock to PORTF */
   SYSCTL_RCC_R &= ~0x00100000; /* no pre-divide for PWM clock */

   /* Enable port PF2 for PWM1 M1PWM7 */
   GPIO_PORTF_AFSEL_R = 0x04;      /* PF2 uses alternate function */
   GPIO_PORTF_PCTL_R |= 0x00000500;
   GPIO_PORTF_DEN_R = 0x04;       /* pin digital */

   PWM1_3_CTL_R = 0;            /* stop counter */
   PWM1_3_GENA_R = 0x0000008C;  /* M1PWM6 output set when reload, */
  /* clear when match PWMCMPA */
   PWM1_3_LOAD_R = 16000;       /* set load value for 1kHz (16MHz/16000) */
```

PWM Program to Control Brightness of LED

```
PWM1_3_CMPA_R = 15999;      /* set duty cycle to min */
   PWM1_3_CTL_R = 1;          /* start timer */
   PWM1_ENABLE_R = 0x40;       /* start PWM1 ch7 */

   for(;;) {
      x = x - 50;
      if (x <= 0)
        for(;;){
          x = x + 50;
          if (x>=15999)
            break;
      PWM1_3_CMPA_R = x;
      delayMs(20);
        }
      PWM1_3_CMPA_R = x;
      delayMs(20);
   }
}
/* delay n milliseconds (16 MHz CPU clock) */
void delayMs(int n){
   int i, j;
   for(i = 0 ; i < n; i++)
      for(j = 0; j < 3180; j++)
         {}  /* do nothing for 1 ms */
}
```

Thank You