CET 241: Day 13
ADC on TM4C123

Dr. Noori Kim

Recap

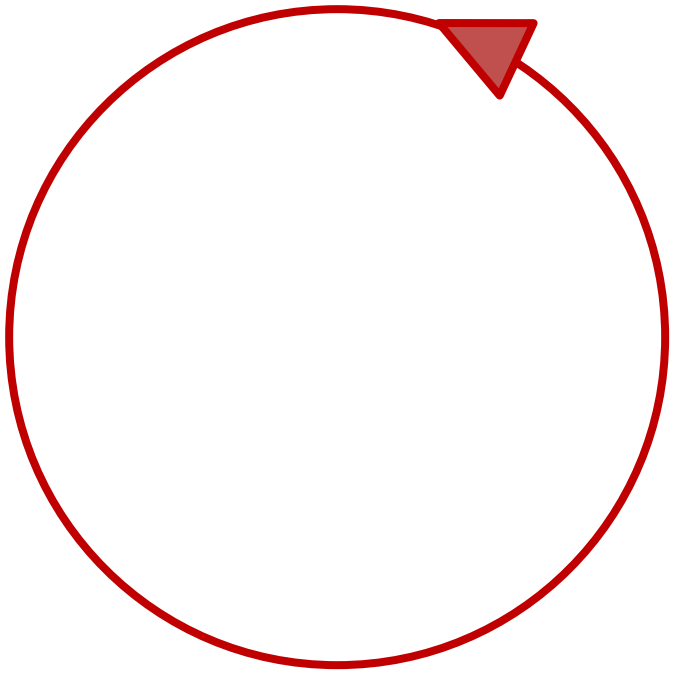
- We learnt about DAC theory.. Including NST, resolution, precision, range, types of DAC

Agenda

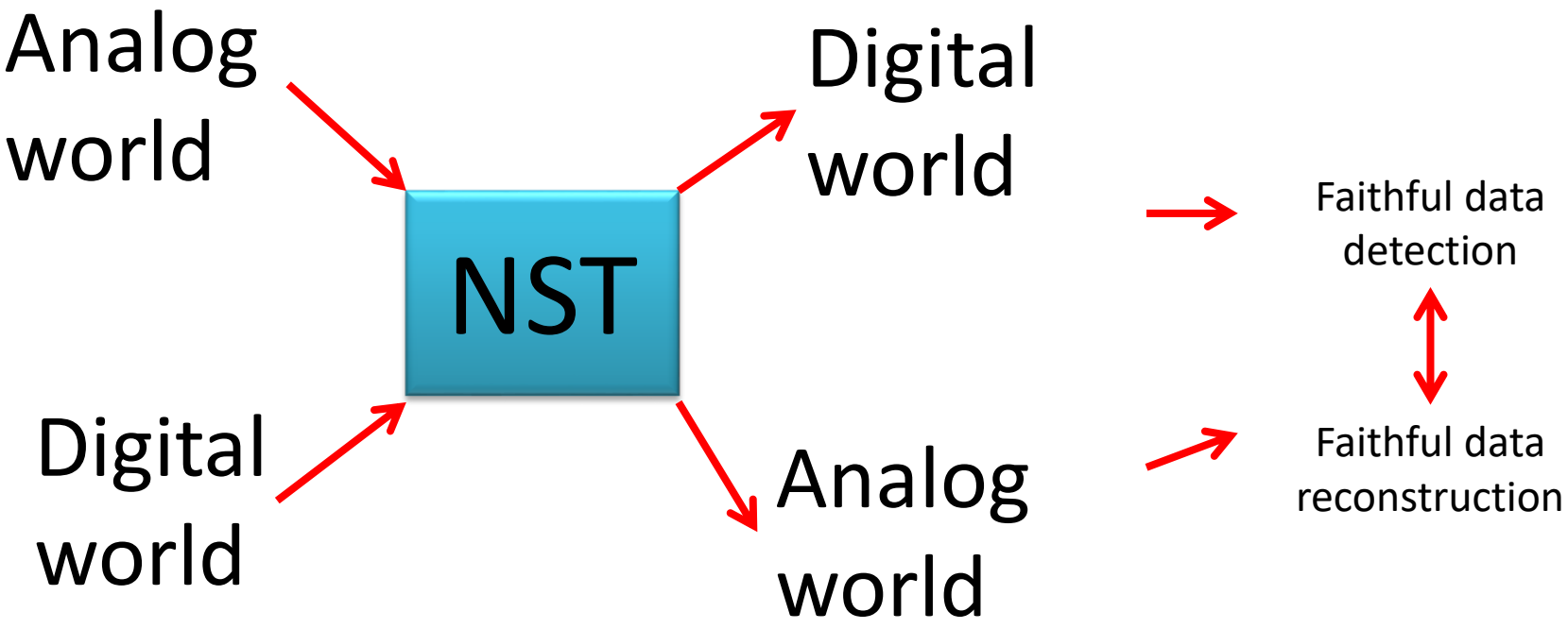
- Successive approximation ADC
- ADC Features (dedicated registers) on our board
- See an Example

Nyquist Theorem

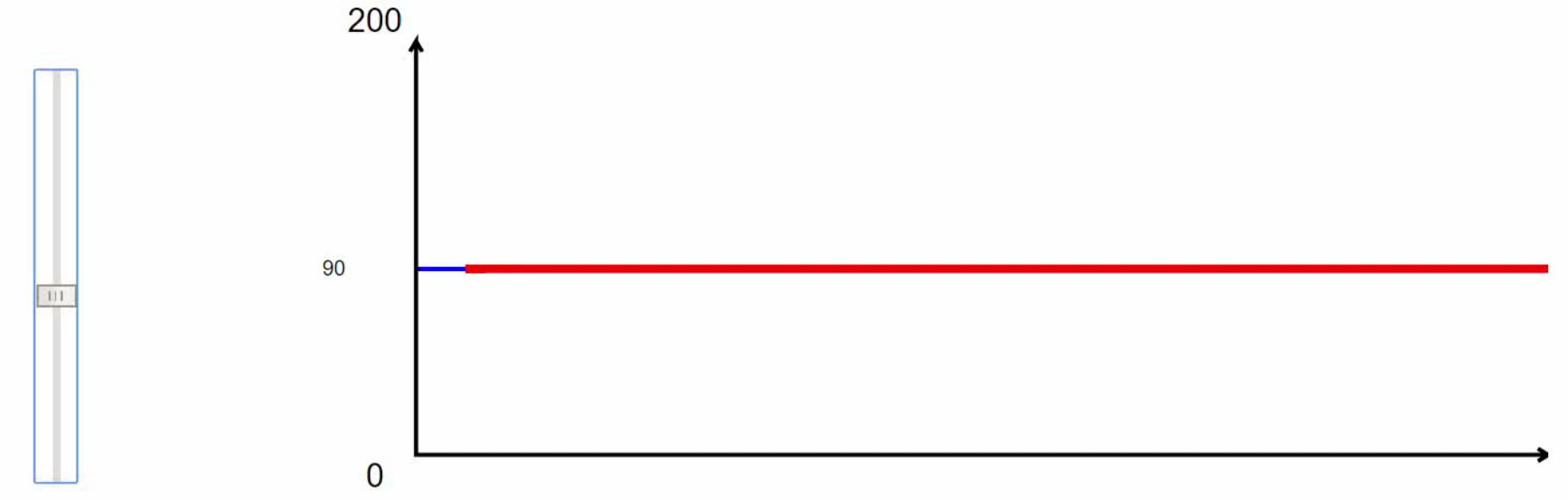
- The Nyquist Theorem says that if a signal is oscillating at frequency F, in order to capture it faithfully, we must sample at a frequency that is strictly <u>larger than two times of F.</u>



How does the NST apply to each case?

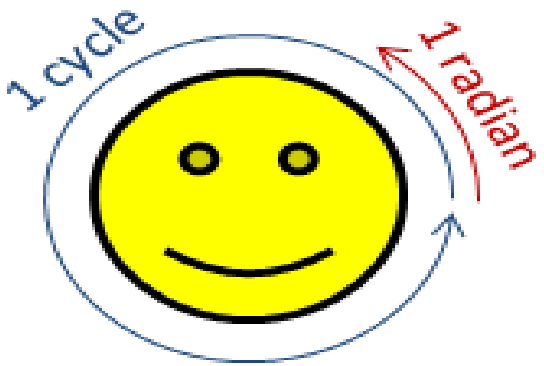


- Dragging the handle to generate analog signal
  - Analog wave (blue): My angry hand
  - Digital sampled wave (red): sampled at 1Hz



BTW, what is the frequency?

- Frequency (f): A measure of "how frequent a cycle is" per unit time
- Angular frequency ***ω (****unit: [rad]****)****:*** *a scalar measure of phase,* 2π*[rad] = 6.28 = 360º*
  - $\omega = 2\pi f$



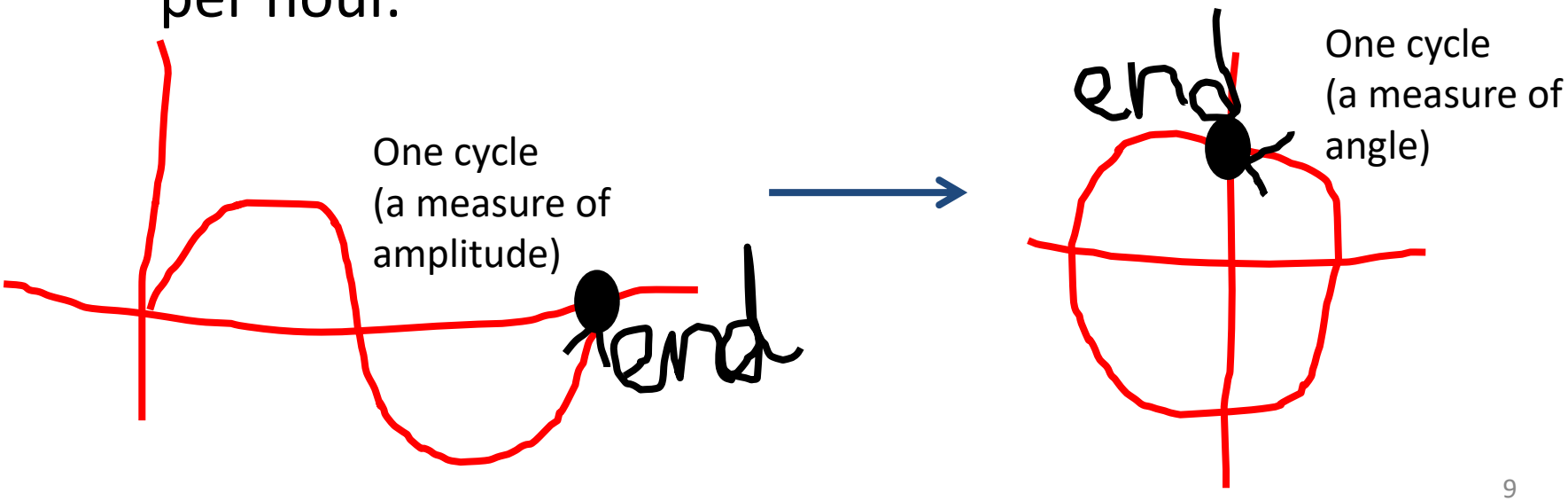
Time (in seconds) = 0.00 s
Rotation (in radians) = 0.00 rad
Rotation (in cycles) = 0.00 cycle

$$\omega = \frac{0.00\ \text{rad}}{0.00\ \text{s}} =$$

$$\nu = \frac{0.00\ \text{cycle}}{0.00\ \text{s}} =$$

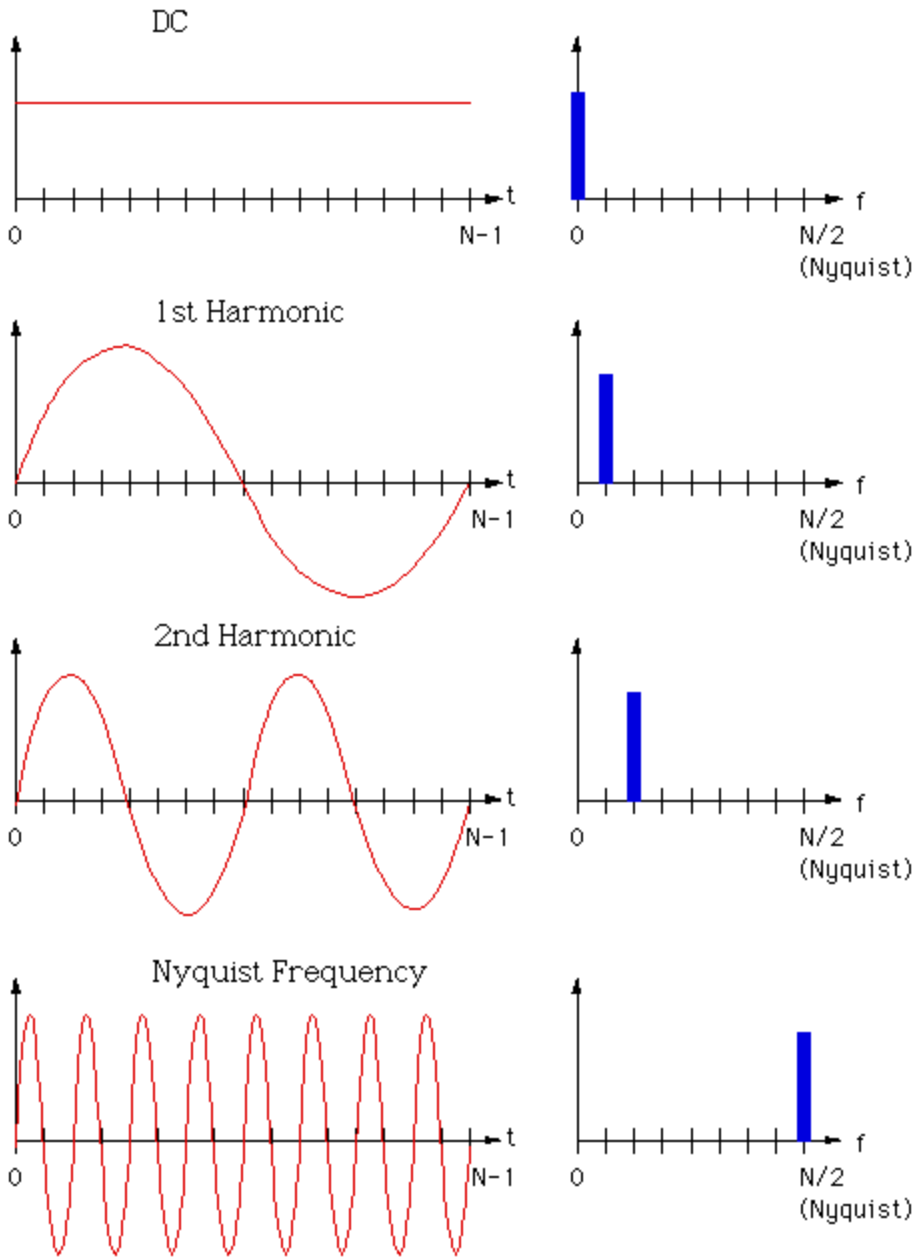
Frequency:
A cyclic version of velocity

- Frequency describes the number of waves that pass a fixed place (i.e., a point to complete one cycle) in a given amount of time (i.e., second)
  - If ½ sec takes to pass one cycle of wave, the frequency is 2 cycles per second or 2Hz
  - If it takes 1/100 of an hour, the frequency is 100 per hour.



- Frequency is measured in the hertz unit, named in honor of the 19th-century German physicist Heinrich Rudolf Hertz.
- The hertz measurement, abbreviated Hz, is the number of waves that pass by per second.
- For example, an **"A"** note on a violin string vibrates at about 440 Hz = 440 vibrations (cycles) per second

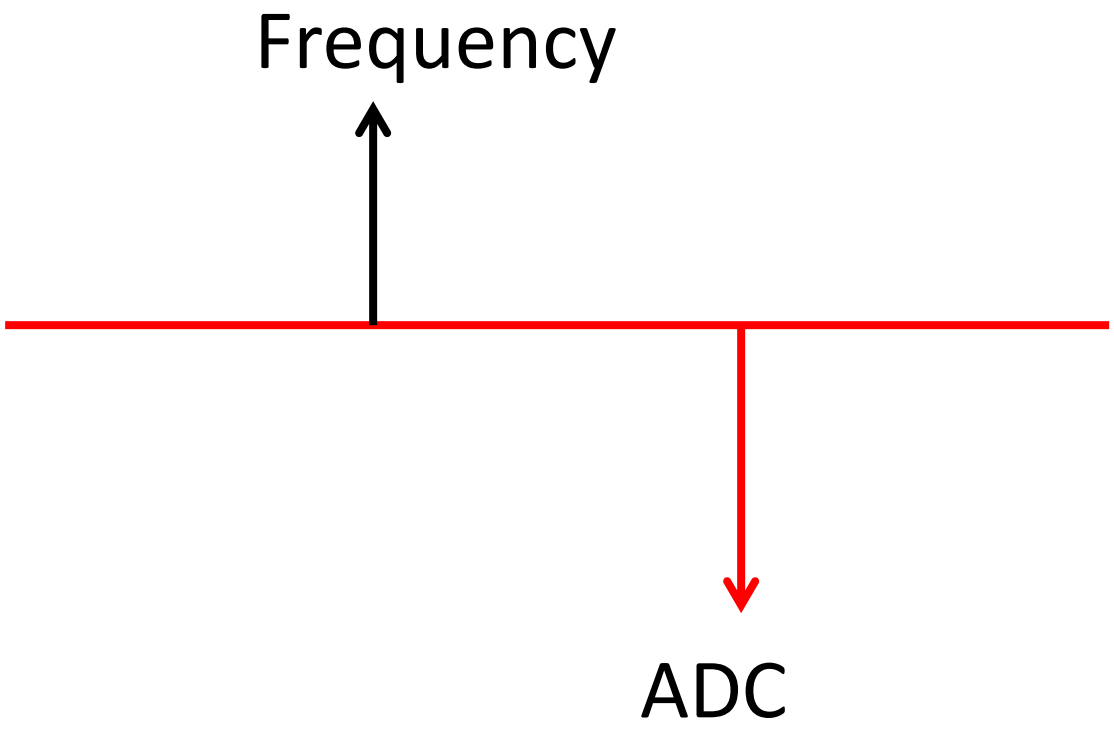
- DC in terms of frequency?
- What is the period of DC voltage?


DC
t
0
N-1
f
0
N/2
(Nyquist)
1st Harmonic
t
0
N-1
f
0
N/2
(Nyquist)
2nd Harmonic
t
0
N-1
f
0
N/2
(Nyquist)
Nyquist Frequency
t
0
N-1
f
0
N/2
(Nyquist)

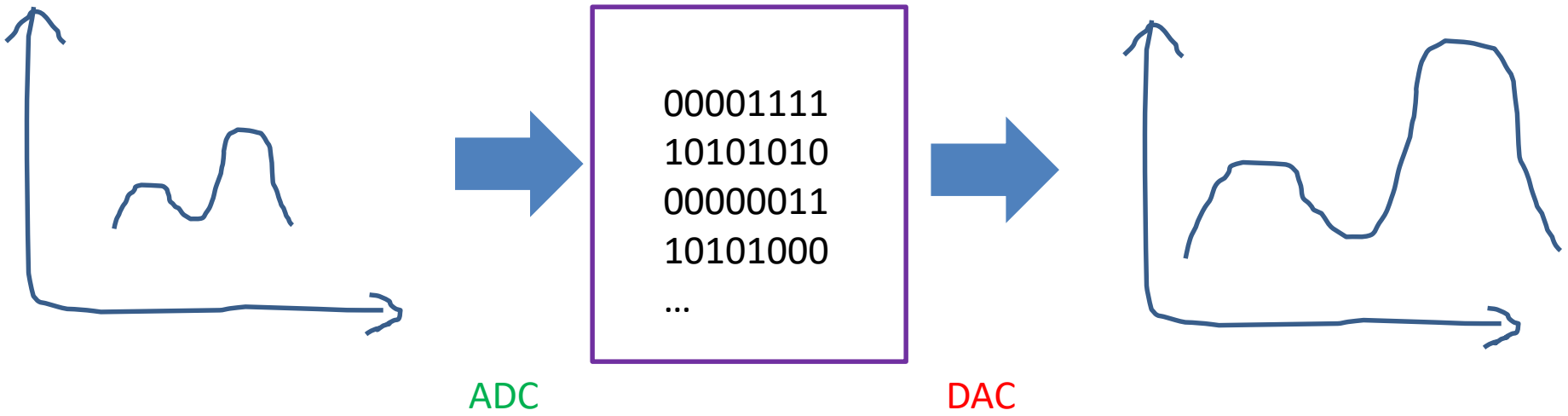
FFT

- Therefore, f=100Hz means
  - 100 cycles per a second: “velocity” similar to [m/s]
  - Equals to 1/100 = 0.01 second **(per one cycle)**
    - Usually, we don’t say **‘per one cycle’** as like we don’t reference 24hrs (1 cycle) when we talk about times
- **Displacement=velocity*time**
- **Wavelength=speed of wave*(1/f)**
  - Wavelength of 1kHz acoustic sine wave?

343m/sec *0.001sec = 1ft


Frequency
ADC

Digital and Analog world



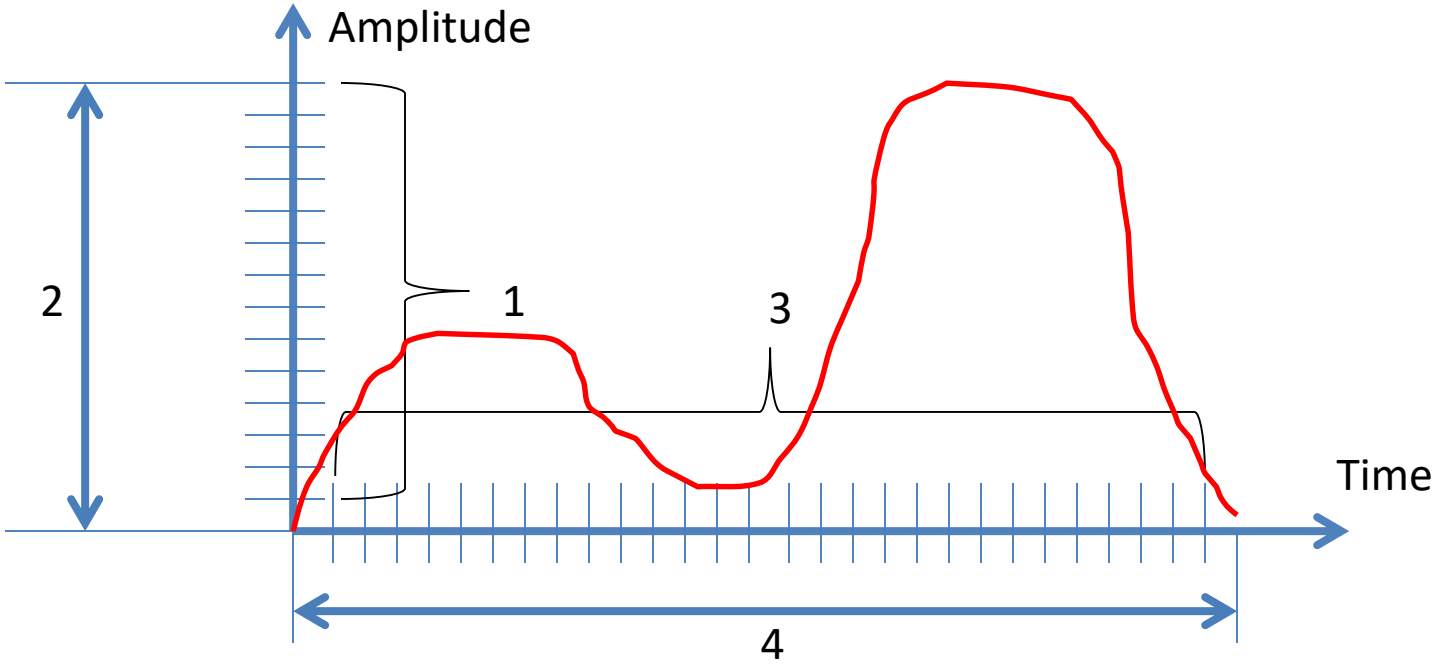
Our voice through a microphone

Our voice amplification/ noise filtering via computer processing

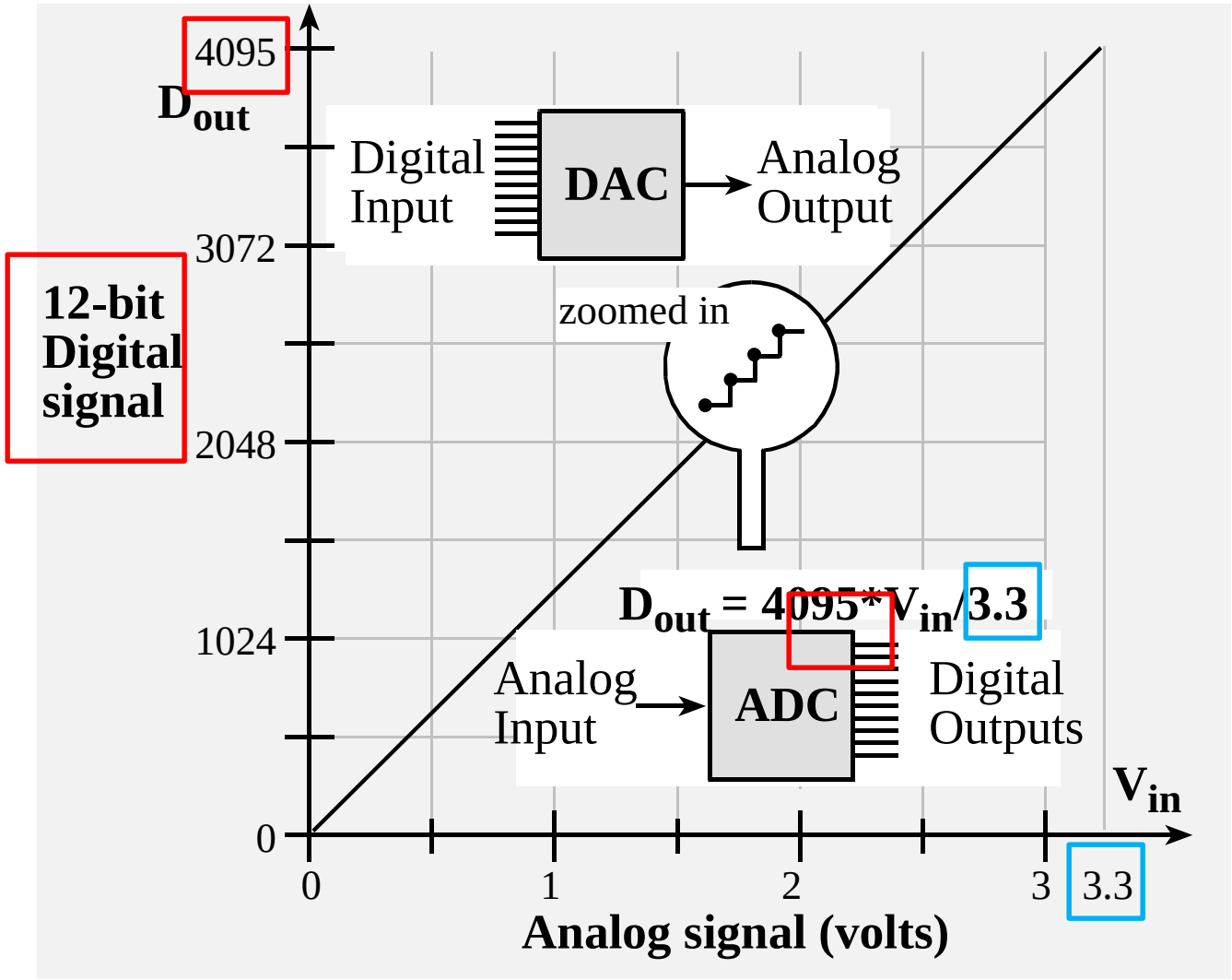
Our processed voice via speaker

Difference between A and D signals

	Analog signal	Digital Signal
1. Amplitude quantization	Continuous	Discrete
2. Amplitude range	Infinite	Finite
3. Time quantization	Continuous	Discrete
4. Time range	Infinite	Finite



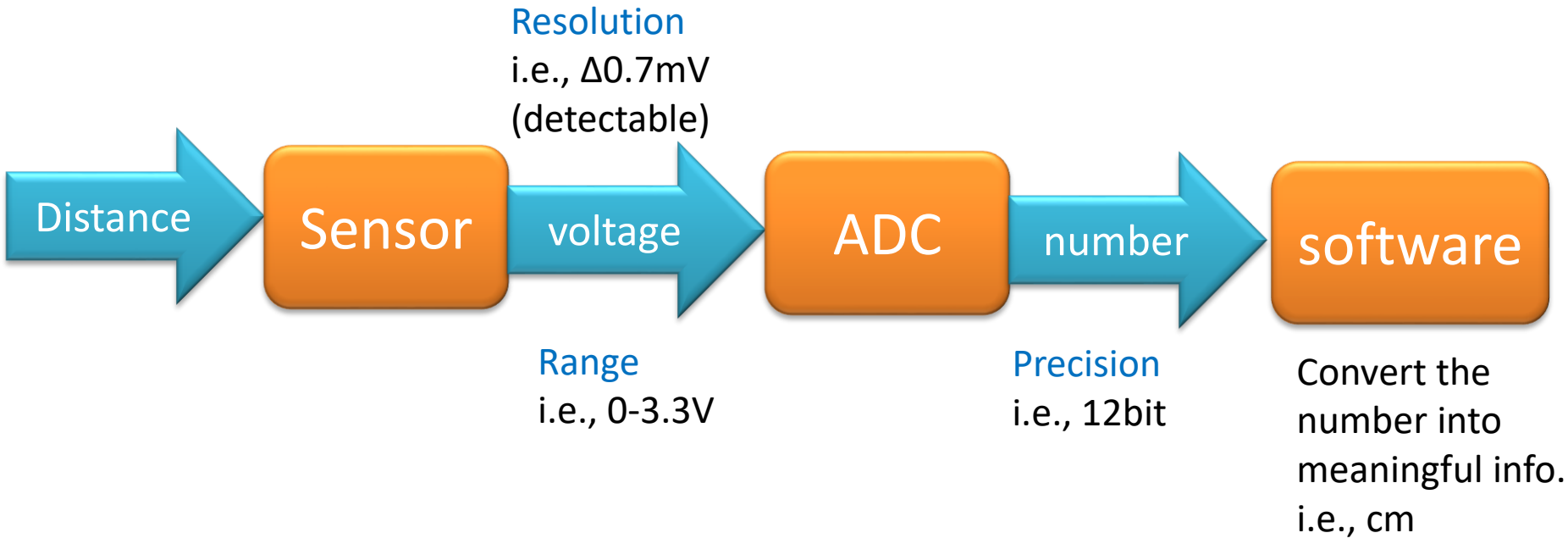
Analog to Digital Conversion



ADC parameters

1. Precision: # of different alternatives (i.e., 2^{12}=4096, 12bit)
2. Range: min-max voltage (i.e., 0V-3V)
3. Resolution (ΔV): Smallest change in voltage that can be reliably detectable
   - $\Delta V = \frac{Range}{Precision} = \frac{3V-0V}{4096} \approx 0.7mV$
     $= \frac{Range}{Precision-1} = \frac{3V-0V}{4095} \approx 0.7mV$
   - Higher resolution, higher price

i.e. Data acquisition system



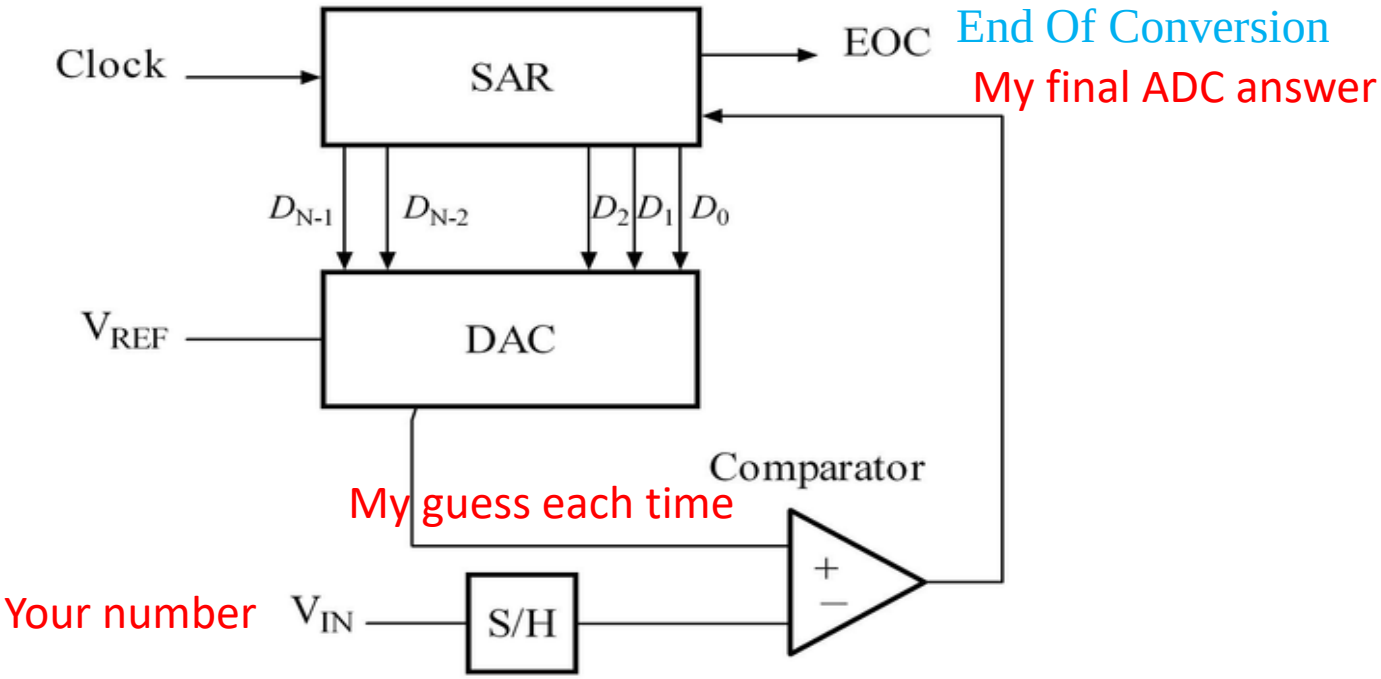
- But how does the ADC work?
  - Successive approximation (SAR)
- If our ADC system precision is n bit, we can discretize our analog inputs via n times comparison

- Let’s play
  - Our ADC has 4 bit precision
  - You think (don’t tell me) one number between 0-15
  - I will guess and tell you a number
    - You say yes: if the value is greater than or same with your number
    - You say no: if the value is less than your number
  - I will guess 4 times and get your number

- N bit ADC
- Secret number (from 0 to (2^N-1))
- Start to guess from $2^N/2=2^{N-1}$
- Q: Is the secret number greater or equal to the guessing number?
  - Yes: 1, No:0
- Increase or decrease (+ or -) 2^{N-1-n}
- n=trial: 0 to (N-1)

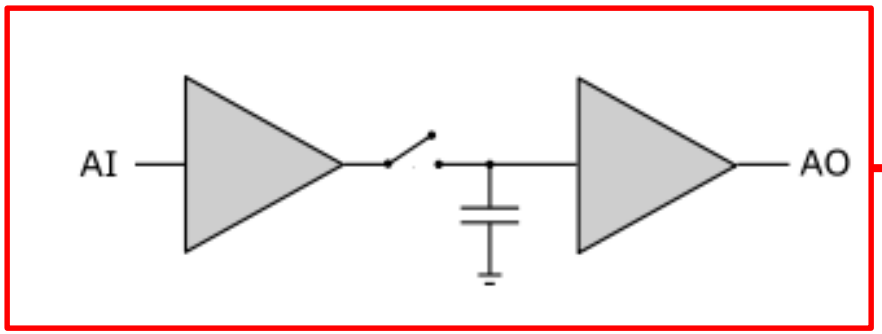
- 4 bit
- Secret number: 11
- $2^4=16$, $16/2=8$
- $11 \geq 8$? Yes: 1
- $(8+2^2)$ $11 \geq 12$? No: 0
- $(12-2^1)$ $11 \geq 10$: Yes: 1
- $(10+2^0)$ $11 \geq 11$:Yes: 1
- Ans: $1011_2=11$

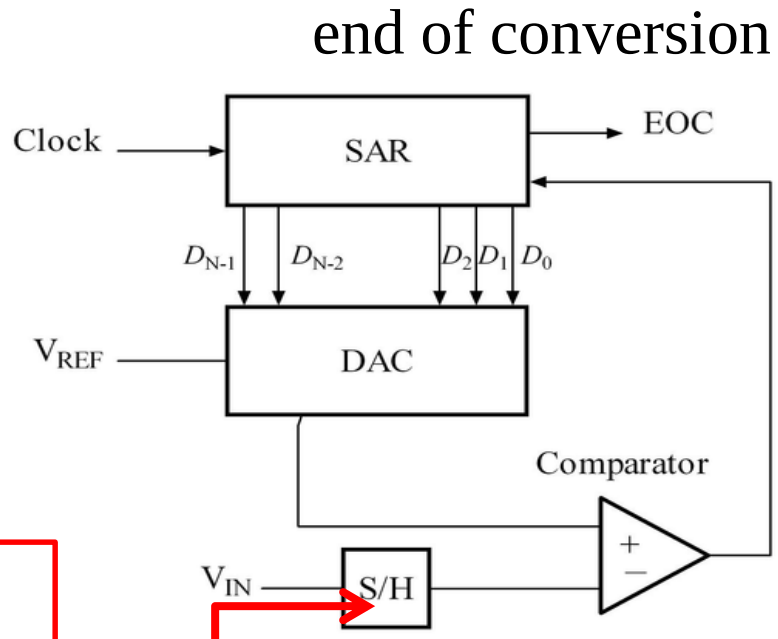
Successive approximation ADC



- SAR is a counter that increments each *clock* as long as it is enabled by the *comparator*
- The output of the SAR is fed to a DAC that generates a voltage for comparison with V_{IN}
- When the output of the DAC = V_{IN} the value of SAR is the digital representation of V_{IN}

- V_{IN} is approximated as a static value in a *sample and hold* (S/H) circuit
- Analog Input (AI) is sampled when the switch is closed and its value is held on the capacitor where it becomes the Analog Output (AO)





- ADC general concept are studied enough
- Let’s talk about ADC on our board
- Slides that I will go over from now is a summary of our datasheet ADC pp.799-892

General features ADC on TM4C123

- Sampling Range/Resolution
  - 3.3V internal reference voltage
    - 0x000 at 0 V input
    - 0xFFF(=$1111.1111.1111_2$:12bit ADC) at 3.3 V
  - resolution = range/precision

    = 3V/(**4096** alternatives or 4095) < 1mV

    (0.7mV?)
    - 2^{12}=4096
  - Actual resolution dominated by noise

- Improve signal to noise ratio (SNR): we don't worry about this. It's our board's features
  - Slow down ADC (take longer to sample)
  - Analog filtering, ground shield
  - Digital filtering (averaging of up to 64 samples for improved accuracy)

Specific features ADC on TM4C123

- Two modules, twelve analog input channels
  - Like GPIO port A to F, each port has 5-8 pins
  - ADC0 and ADC1 modules, each module has 12 pins (0-11): sharing
- On-chip internal temperature sensor
- Sample rate
  - Min: 125k samples/sec
  - Up to one million samples/second
- Flexible, configurable analog-to-digital modules: i.e., GIPO ← →ADC

13.1 Block Diagram

The TM4C123GH6PM microcontroller contains two identical Analog-to-Digital Converter modules. These two modules, ADC0 and ADC1, share the same 12 analog input channels. Each ADC module operates independently and can therefore execute different sample sequences, sample any of the analog input channels at any time, and generate different interrupts and triggers. Figure 13-1 on page 800 shows how the two modules are connected to analog inputs and the system bus.

Figure 13-1. Implementation of Two ADC Blocks

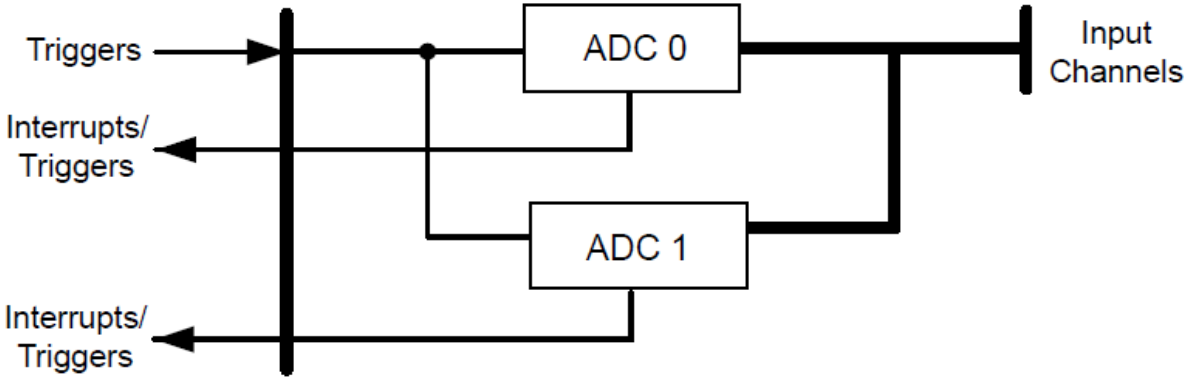


Figure 13-2 on page 801 provides details on the internal configuration of the ADC controls and data registers.

13.5 Register Map

Table 13-4 on page 818 lists the ADC registers. The offset listed is a hexadecimal increment to the register's address, relative to that ADC module's base address of:

- ADC0: 0x4003.8000
- ADC1: 0x4003.9000

Note that the ADC module clock must be enabled before the registers can be programmed (see page 352). There must be a delay of 3 system clocks after the ADC module clock is enabled before any ADC module registers are accessed.

Table 13-4. ADC Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	RW	0x0000.0000	ADC Active Sample Sequencer	821
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	823
0x008	ADCIM	RW	0x0000.0000	ADC Interrupt Mask	825

- Four programmable sample conversion sequences (SS0-SS3), with corresponding conversion result FIFOs (ADC data read)
- Flexible trigger control
  - Controller (software initiated, default and simple)
  - Timers
  - Analog Comparators
  - PWM
  - GPIO

We will use software initiated trigger

Digital Sample = (Analog Input volts / 3.3V) • 4095

- Max Digital Value: DS = Max Analog value : AIV
  - 4095 : DS = 3.3 : AIV
- If the input voltage is 1.5V, what value will the TM4C 12-bit ADC return?

- If the input voltage is 0.5V, what value will the TM4C 12-bit ADC return?

ADC registers on TM4C123

Address	31-2	1	0	Name
$400F.E638		ADC1	ADC0	SYSCTL_RCGCADC_R

	31-14	13-12	11-10	9-8	7-6	5-4	3-2	1-0	
$4003.8020		SS3		SS2		SS1		SS0	ADC0_SSPRI_R

	31-16	15-12	11-8	7-4	3-0	
$4003.8014		EM3	EM2	EM1	EM0	ADC0_EMUX_R

	31-4	3	2	1	0	
$4003.8000		ASEN3	ASEN2	ASEN1	ASEN0	ADC0_ACTSS_R
$4003.80A0		MUX0				ADC0_SSMUX3_R
$4003.80A4		TS0	IE0	END0	D0	ADC0_SSCTL3_R
$4003.8028		SS3	SS2	SS1	SS0	ADC0_PSSI_R
$4003.8004		INR3	INR2	INR1	INR0	ADC0_RIS_R
$4003.8008		MASK3	MASK2	MASK1	MASK0	ADC0_IM_R
$4003.8FC4		Speed				ADC0_PC_R

	31-12	11-0	
$4003.80A8		DATA	ADC0_SSFIFO3_R

1. RCGCADC: Analog-to-Digital Converter Run Mode Clock Gating Control (p.352)
2. ADCSSPRI: ADC Sample Sequencer PRIority (p.841)
3. ADCACTSS: ADC ACTive Sample Sequencer (p.821)

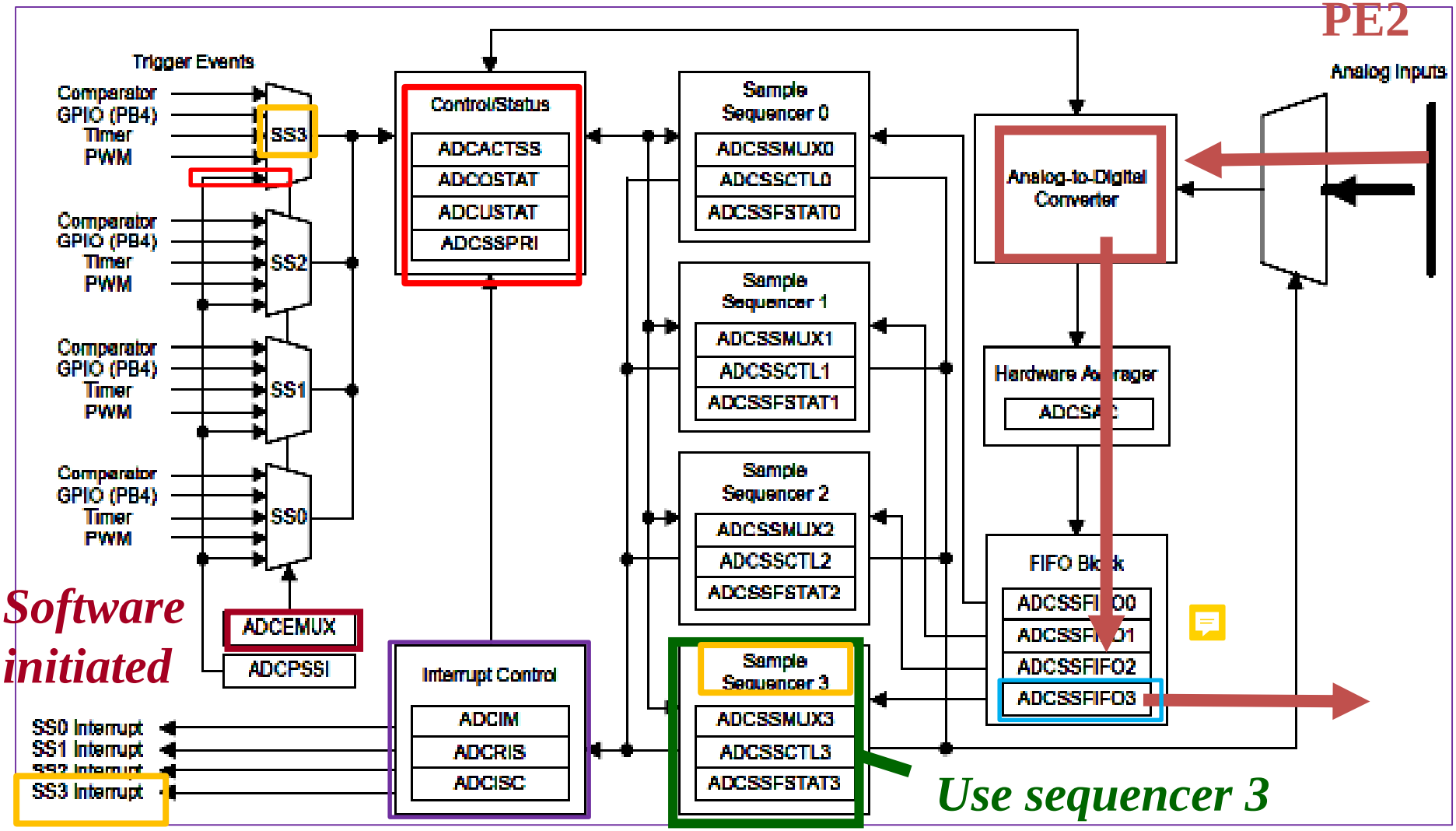
Control/ Status

4. ADCEMUX: ADC Event Multiplexer Select (p.833)
5. ADCSSMUX3: ADC Sample Sequence Input Multiplexer Select 3 (p.875)
6. ADCSSCTL3: ADC Sample Sequence Control 3 (p.876)
7. ADCPSSI: ADC Processor Sample Sequence Initiate (p.845)
8. ADCRIS: ADC Raw Interrupt Status (p.823)
9. ADCIM: ADC Interrupt Mask (p.825)

Interrupt control

10. ADCPC: ADC Peripheral Configuration (p.891)
11. ADCSSFIFO3: ADC Sample Sequence Result FIFO 3 (p.860)

Ain1
PE2



Software initiated

Use sequencer 3

ADC on TM4C123 (PCTL table)

IO	Ain	0	1	2	3	4	5	6	7	8	9	14
PB4	Ain10	Port		SSI2Clk		M0PWM2			T1CCP0	CAN0Rx		
PB5	Ain11	Port		SSI2Fss		M0PWM3			T1CCP1	CAN0Tx		
PD0	Ain7	Port	SSI3Clk	SSI1Clk	I_2C3SCL	M0PWM6	M1PWM0		WT2CCP0			
PD1	Ain6	Port	SSI3Fss	SSI1Fss	I_2C3SDA	M0PWM7	M1PWM1		WT2CCP1			
PD2	Ain5	Port	SSI3Rx	SSI1Rx		M0Fault0			WT3CCP0	USB0epen		
PD3	Ain4	Port	SSI3Tx	SSI1Tx				IDX0	WT3CCP1	USB0pflt		
PE0	Ain3	Port	U7Rx									
PE1	Ain2	Port	U7Tx									
PE2	Ain1	Port										
PE3	Ain0	Port										
PE4	Ain9	Port	U5Rx		I_2C2SCL	M0PWM4	M1PWM2			CAN0Rx		
PE5	Ain8	Port	U5Tx		I_2C2SDA	M0PWM5	M1PWM3			CAN0Tx		

PE2=Ain1 used

ADC Init step

0. Proper GPIO port related setting
1. ADC clock
   - Enable ADC clock: set bit 0 in **SYSCTL_RCGCADC_R**
2. ADC speed
   - Set 125kHz ADC conversion speed: write 0x01 to **ADC0_PC_R**

Value	*Description*
0x7	1M samples/second
0x5	500K samples/second
0x3	250K samples/second
0x1	125K samples/second

3. Set sequencer priority
   - Use SEQ3 (highest priority 0 setting): set 1,2,3 on the other seqs: **ADC0_SSPRI_R**
   - Using just one sequencer, priorities are irrelevant, except for the fact that no two sequencers should have the same priority.
4. Disable selected sequence 3 while configuring ADC (last step is enable this bit)
   - zero bit 3 of **ADC0_ACTSS_R**
5. Select trigger modes (Trigger ADC==Start ADC)
   - Set software start trigger event
   - Zero bits 15-12 of **ADC0_EMUX_R**

Value	*Event*
0x0	Software start
0x1	Analog Comparator 0
0x2	Analog Comparator 1
0x3	Analog Comparator 2
0x4	External (GPIO PB4)
0x5	Timer
0x6	PWM0
0x7	PWM1
0x8	PWM2
0xF	Always (continuously sample)

initiates sampling for each sample sequencer

EM3 (bit15-12), EM2, EM1, and EM0 bits in ADC_EMUX_R

6. Set software start trigger event
   - Set input source (0-11) among 12 possible channels
   - write channel number in bits 3-0 of **ADC0_SSMUX3_R**
   - **(i.e., channel 9 is PE4, channel 1 is PE2)**
7. Select sample mode
   - **ADC0_SSCTL3_R = 0x06 (0110_2);**
     - Bit 3: 0 not temperature: disable temp measurement
     - Bit 2: 1 set completion flag: notify on sample complete

       (IE0 bit: set because we want to the **RIS** bit to be set when the sample is complete)
     - Bit 1: 1 end sequence: indicate single sample in sequence

       (END0 bit: Sequencer 3 has only one sample)
     - Bit 0: 0 not differential mode

Register 36: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for a sample executed with Sample Sequencer 3. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSCTL0** register on page 853 for detailed bit descriptions.

Note: When configuring a sample sequence in this register, the END0 bit must be set.

Pp 876

← →pp 868

Bit	Name	Type	Reset	Description
3	TS0	RW	0	1st Sample Temp Sensor Select Value Description 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 of the sample sequence.
2	IE0	RW	0	Sample Interrupt Enable Value Description 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 within a sequence generate interrupts.
1	END0	RW	0	End of Sequence This bit must be set before initiating a single sample sequence. Value Description 0 Sampling and conversion continues. 1 This is the end of sequence.
0	D0	RW	0	Sample Differential Input Select Value Description 0 The analog inputs are not differentially sampled. 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". Because the temperature sensor does not have a differential option, this bit must not be set when the TS0 bit is set.

Register 29: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

Register 30: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084

These registers contain the configuration information for each sample for a sequence executed with Sample Sequencer 1 or 2. When configuring a sample sequence, the END bit must be set for the final sample, whether it be after the first sample, fourth sample, or any sample in between. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSCTL0** register on page 853 for detailed bit descriptions. The **ADCSSCTL1** register configures Sample Sequencer 1 and the **ADCSSCTL2** register configures Sample Sequencer 2.

ADC Sample Sequence Control n (ADCSSCTLn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x064
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
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
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

8. Disable interrupts: we will check a flag

(busy-wait not an interrupt)

   - zero bit 3 of **ADC0_IM_R**
   - In other words, not using interrupt handler we just check a flag

9. Enable selected sequencer 3 after configuring ADC
   - 1 to bit 3 of **ADC0_ACTSS_R**

Example: Channel 9=PE4

```
void ADC0_InitSWTriggerSeq3_Ch9(void){
 SYSCTL_RCGCGPIO_R |= 0x10;       // 1) activate clock for Port E
  while((SYSCTL_PRGPIO_R&0x10) == 0){}; //Peripheral Ready register
  GPIO_PORTE_DIR_R &= ~0x10;      // 2) make PE4 input
  GPIO_PORTE_AFSEL_R |= 0x10;     // 3) enable alternate fun on PE4
  GPIO_PORTE_DEN_R &= ~0x10;      // 4) disable digital I/O on PE4
  GPIO_PORTE_AMSEL_R |= 0x10;     // 5) enable analog fun on PE4
  SYSCTL_RCGCADC_R |= 0x01;       // 6) activate ADC0
  delay = SYSCTL_RCGCADC_R;       // extra time to stabilize
  delay = SYSCTL_RCGCADC_R;       // extra time to stabilize
  delay = SYSCTL_RCGCADC_R;       // extra time to stabilize
  delay = SYSCTL_RCGCADC_R;
  ADC0_PC_R = 0x01;               // 7) configure for 125K
  ADC0_SSPRI_R = 0x0123;          // 8) Seq 3 is highest priority
  ADC0_ACTSS_R &= ~0x0008;        // 9) disable sample sequencer 3
  ADC0_EMUX_R &= ~0xF000;         // 10) seq3 is software trigger
  ADC0_SSMUX3_R = (ADC0_SSMUX3_R&0xFFFFFFF0)+9;  // 11) Ain9 (PE4)
  ADC0_SSCTL3_R = 0x0006;         // 12) no TS0 D0, yes IE0 END0
  ADC0_IM_R &= ~0x0008;           // 13) disable SS3 interrupts
  ADC0_ACTSS_R |= 0x0008;         // 14) enable sample sequencer 3
}
```

Register 108: General-Purpose Input/Output Peripheral Ready (PRGPIO), offset 0xA08

The **PRGPIO** register indicates whether the GPIO modules are ready to be accessed by software following a change in status of power, Run mode clocking, or reset. A Run mode clocking change is initiated if the corresponding **RCGCGPIO** bit is changed. A reset change is initiated if the corresponding **SRGPIO** bit is changed from 0 to 1.

The **PRGPIO** bit is cleared on any of the above events and is not set again until the module is completely powered, enabled, and internally reset.

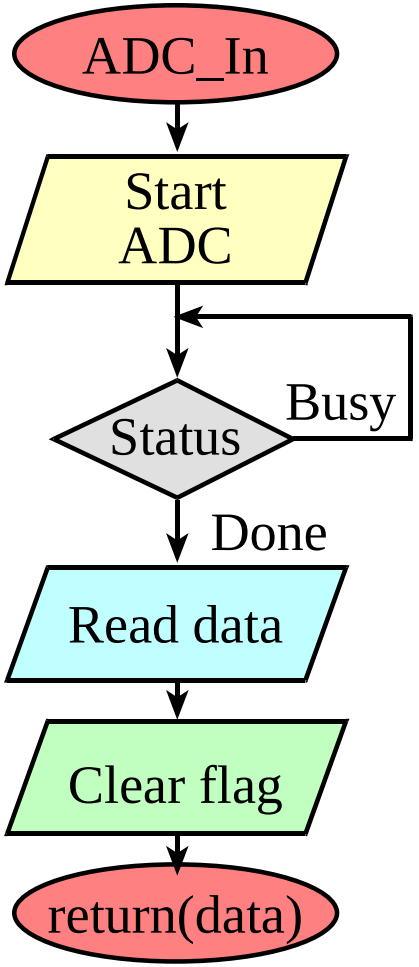
4	R4	RO	0	GPIO Port E Peripheral Ready

Value	Description
0	GPIO Port E is not ready for access. It is unclocked, unpowered, or in the process of completing a reset sequence.
1	GPIO Port E is ready for access.

IO	Ain	0	1	2	3	4	5	6	7	8	9	14
PA2		Port		SSI0Clk								
PA3		Port		SSI0Fss								
PA4		Port		SSI0Rx								
PA5		Port		SSI0Tx								
PA6		Port			I_2C1SCL		M1PWM2					
PA7		Port			I_2C1SDA		M1PWM3					
PB0		Port	U1Rx						T2CCP0			
PB1		Port	U1Tx						T2CCP1			
PB2		Port			I_2C0SCL				T3CCP0			
PB3		Port			I_2C0SDA				T3CCP1			
PB4	Ain10	Port		SSI2Clk		M0PWM2			T1CCP0	CAN0Rx		
PB5	Ain11	Port		SSI2Fss		M0PWM3			T1CCP1	CAN0Tx		
PB6		Port		SSI2Rx		M0PWM0			T0CCP0			
PB7		Port		SSI2Tx		M0PWM1			T0CCP1			
PC4	C1-	Port	U4Rx	U1Rx		M0PWM6		IDX1	WT0CCP0	U1RTS		
PC5	C1+	Port	U4Tx	U1Tx		M0PWM7		PhA1	WT0CCP1	U1CTS		
PC6	C0+	Port	U3Rx					PhB1	WT1CCP0	USB0epen		
PC7	C0-	Port	U3Tx						WT1CCP1	USB0pflt		
PD0	Ain7	Port	SSI3Clk	SSI1Clk	I_2C3SCL	M0PWM6	M1PWM0		WT2CCP0			
PD1	Ain6	Port	SSI3Fss	SSI1Fss	I_2C3SDA	M0PWM7	M1PWM1		WT2CCP1			
PD2	Ain5	Port	SSI3Rx	SSI1Rx		M0Fault0			WT3CCP0	USB0epen		
PD3	Ain4	Port	SSI3Tx	SSI1Tx				IDX0	WT3CCP1	USB0pflt		
PD6		Port	U2Rx			M0Fault0		PhA0	WT5CCP0			
PD7		Port	U2Tx					PhB0	WT5CCP1	NMI		
PE0	Ain3	Port	U7Rx									
PE1	Ain2	Port	U7Tx									
PE2	Ain1	Port										
PE3	Ain0	Port										
PE4	Ain9	Port	U5Rx		I_2C2SCL	M0PWM4	M1PWM2			CAN0Rx		
PE5	Ain8	Port	U5Tx		I_2C2SDA	M0PWM5	M1PWM3			CAN0Tx		
PF0		Port	U1RTS	SSI1Rx	CAN0Rx		M1PWM4	PhA0	T0CCP0	NMI	C0o	
PF1		Port	U1CTS	SSI1Tx			M1PWM5	PhB0	T0CCP1		C1o	TRD1
PF2		Port		SSI1Clk		M0Fault0	M1PWM6		T1CCP0			TRD0
PF3		Port		SSI1Fss	CAN0Tx		M1PWM7		T1CCP1			TRCLK
PF4		Port					M1Fault0	IDX0	T2CCP0	USB0epen		

An ADC example on TM4C123

- Analog to digital conversion
  - Set software trigger
    - Write to PSSI bit 3
  - Busy-Wait
    - Raw Interrupt Status = RIS bit 3
    - Poll until sample complete
  - Read sample
    - Read from SSFIFO3
  - Clear sample complete flag
    - Write to ISC bit 3



Register 11: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028

Pp 846

This register provides a mechanism for application software to initiate sampling in the sample sequencers. Sample sequences can be initiated individually or in any combination. When multiple sequences are triggered simultaneously, the priority encodings in **ADCSSPRI** dictate execution order.

This register also provides a means to configure and then initiate concurrent sampling on all ADC modules. To do this, the first ADC module should be configured. The **ADCPSSI** register for that module should then be written. The appropriate `SS` bits should be set along with the `SYNCWAIT` bit. Additional ADC modules should then be configured following the same procedure. Once the final ADC module is configured, its **ADCPSSI** register should be written with the appropriate `SS` bits set along with the `GSYNC` bit. All of the ADC modules then begin concurrent sampling according to their configuration.

Bit/Field	Name	Type	Reset	Description
3	SS3	WO	-	SS3 Initiate

Value	Description
0	No effect.
1	Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the **ADCACTSS** register.

Only a write by software is valid; a read of this register returns no meaningful data.

Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

Pp 828

This register provides the mechanism for clearing sample sequencer interrupt conditions and shows the status of interrupts generated by the sample sequencers and the digital comparators which have been sent to the interrupt controller. When read, each bit field is the logical AND of the respective INR and MASK bits. Sample sequencer interrupts are cleared by writing a 1 to the corresponding bit position. Digital comparator interrupts are cleared by writing a 1 to the appropriate bits in the **ADCDCISC** register. If software is polling the **ADCRIS** instead of generating interrupts, the sample sequence INRn bits are still cleared via the **ADCISC** register, even if the INn bit is not set.

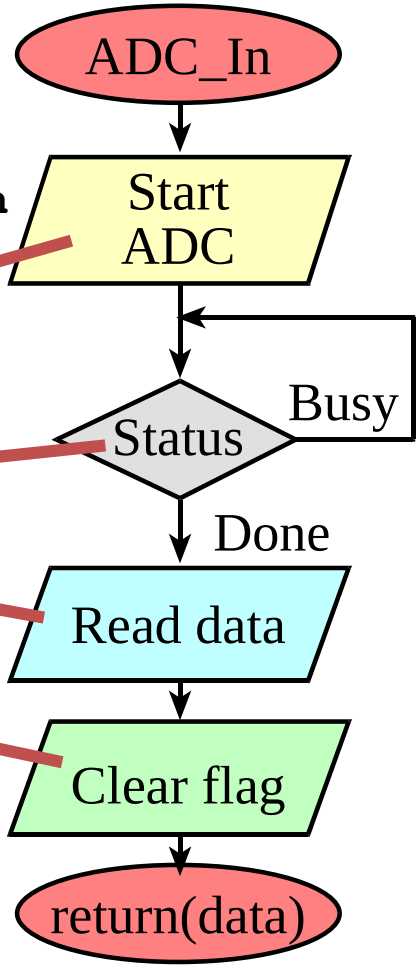
3	IN3	RW1C	0	SS3 Interrupt Status and Clear

Value	Description
0	No interrupt has occurred or the interrupt is masked.
1	Both the INR3 bit in the **ADCRIS** register and the MASK3 bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller.

This bit is cleared by writing a 1. Clearing this bit also clears the INR3 bit in the **ADCRIS** register.

One of the synchronization methods

```
//------------ADC_InSeq3------------
// Busy-wait analog to digital conversion
// Input: none
// Output: 12-bit result of ADC conversion
uint32_t ADC0_InSeq3(void){
uint32_t data;
  ADC0_PSSI_R = 0x0008;
  while((ADC0_RIS_R&0x08)==0){};
  data = ADC0_SSFIFO3_R&0xFFF;
  ADC0_ISC_R = 0x0008;
  return data;
}
```



What is a FIFO?

- A buffer that stores data in a first in first out manner
  - Data are passed from the ADC hardware to the software.
- How is the FIFO used on this ADC?
  - When you read **ADC0_SSFIFO3_R** you get the oldest data.
  - Once data read from the FIFO , they are removed

Why do we use a FIFO when implementing input/output?

- The processing for producer can be separated from the processing of consumer
- FIFO to even out variations in rate between a producer and a consumer.
  - i.e., The buffer allows the software to temporarily run slow, while the hardware continuous to sample.
  - As long as:
    - Maximum producer rate >latency
    - Average producer rate < average consumer rate
  - If not, FIFO full

Another way to initialize ADC: datasheet page 817

- The concept of ADC (and other functions) is general
- Steps to activate ADC of every MCU can be different
- We must refer to its datasheet to capture the specific features of its ADC module

```
1  //(Page 817-818)
2  //13.4.1 Module Initialization
3  //  Initialization of the ADC module is a simple process with very few steps: enabling the clock to
4  //  the ADC, disabling the analog isolation circuit associated with all inputs that are to be used,
5  //  and reconfiguring the sample sequencer priorities (if needed).
6  //
7  /*********************************
8   * Driver functions for an ADC
9   *
10  * Runs on LM4F120/TM4C123
11  * Referenced by Datasheet page 817-818
12  *********************************/
13
14 #include "ADC.h"
15 #include "..//tm4c123gh6pm.h"
16 //The initialization sequence for the ADC is as follows:
17 //
18 /* Intialises ADC on PE2*/
19 void ADC0_Init(void){
20   unsigned long volatile delay;
21
22 //  1. Enable the ADC clock using the RCGCADC register (see page 352).
23 //  or RCGC0ADC register (page 456)
24   SYSCTL_RCGC0_R |= 0x00010000;   // activate ADC0
25   delay = SYSCTL_RCGC2_R;
26
27 //  2. Enable the clock to the appropriate GPIO modules via the RCGCGPIO register (see page 340).
28 //  To find out which GPIO ports to enable, refer to "Signal Description" on page 801.
29   SYSCTL_RCGC2_R |= 0x10;   // activate port E
30   delay = SYSCTL_RCGC2_R;
31 //  Configure I/O pin
32   GPIO_PORTE_DIR_R &= ~0X04;      // make PE2 in
```

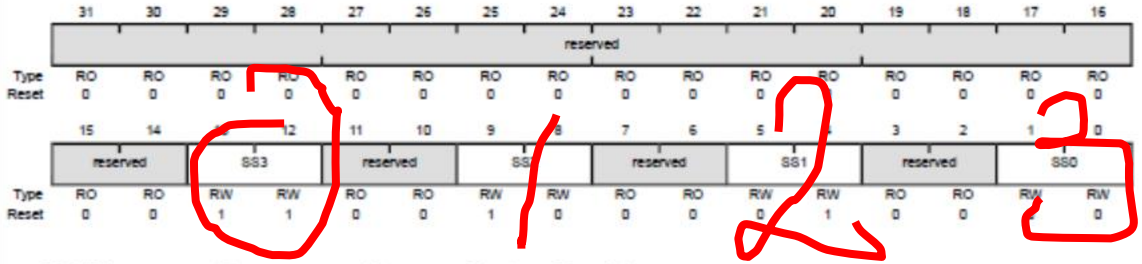
```
34  //  3. Set the GPIO AFSEL bits for the ADC input pins (see page 671). To determine which GPIOs
35  //  to configure, see Table 23-4 on page 1344.
36    GPIO_PORTE_AFSEL_R |= 0X04;   // enable alt funct on PE2
37
38  //  4. Configure the AINx signals to be analog inputs by clearing the corresponding DEN bit in
39  //  the GPIO Digital Enable (GPIODEN) register (see page 682).
40    GPIO_PORTE_DEN_R &= ~0X04;      // disable digital I/O on PE2
41
42  //  5. Disable the analog isolation circuit for all ADC input pins that are to be used by writing a
43  //  1 to the appropriate bits of the GPIOAMSEL register (see page 687) in the associated GPIO block.
44    GPIO_PORTE_AMSEL_R |= 0x04;     // enable analog on PE2
45
46  //  6. If required by the application, reconfigure the sample sequencer priorities in the ADCSSPRI
47  //  register. The default configuration has Sample Sequencer 0 with the highest priority and Sample
48  //  Sequencer 3 as the lowest priority.
49  //  (datasheet page 841)ADCSSPRI
50    SYSCTL_RCGC0_R &= ~0x00000300;  // 125K samplig rate, RCGC0ADC register (page 456)
51    ADC0_SSPRI_R = 0x0123;          // SS3 = highest priority SS2=second p,...
52           // as we are using SS3 (we want to sample 1 at a time not multiple
53  /*
54  //13.4.2 Sample Sequencer Configuration
55  //  Configuration of the sample sequencers is slightly more complex than the module initialization
56  //  because each sample sequencer is completely programmable.
57  //
58  //The configuration for each sample sequencer should be as follows:
59  //
60  //  1. Ensure that the sample sequencer is disabled by clearing the corresponding ASENn bit in
61  //  the ADCACTSS register. Programming of the sample sequencers is allowed without having them
62  //  enabled. Disabling the sequencer during programming prevents erroneous execution if a trigger
63  //  event were to occur during the configuration process.
64    ADC0_ACTSS_R &= ~0x0008;        // disable SS3
```

Register 9: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the sample sequencers. Out of reset, Sequencer 0 has the highest priority, and Sequencer 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority for the ADC to operate properly.

ADC Sample Sequencer Priority (ADCSSPRI)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x020
Type RW, reset 0x0000.3210



Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:12	SS3	RW	0x3	SS3 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
11:10	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be

- We use sequencer 3 which can only sample one channel at a time.
- However sequencer 0 can sample from 1 to 8 channels at a time.
  - i.e., Acceleration x,y or x,y,z, Joystick x,y, Temperature and humidity, Force x,y

```
34  //  3. Set the GPIO AFSEL bits for the ADC input pins (see page 671). To determine which GPIOs
35  //  to configure, see Table 23-4 on page 1344.
36    GPIO_PORTE_AFSEL_R |= 0X04;   // enable alt funct on PE2
37
38  //  4. Configure the AINx signals to be analog inputs by clearing the corresponding DEN bit in
39  //  the GPIO Digital Enable (GPIODEN) register (see page 682).
40    GPIO_PORTE_DEN_R &= ~0X04;      // disable digital I/O on PE2
41
42  //  5. Disable the analog isolation circuit for all ADC input pins that are to be used by writing a
43  //  1 to the appropriate bits of the GPIOAMSEL register (see page 687) in the associated GPIO block.
44    GPIO_PORTE_AMSEL_R |= 0x04;     // enable analog on PE2
45
46  //  6. If required by the application, reconfigure the sample sequencer priorities in the ADCSSPRI
47  //  register. The default configuration has Sample Sequencer 0 with the highest priority and Sample
48  //  Sequencer 3 as the lowest priority.
49  //  (datasheet page 841)ADCSSPRI
50    SYSCTL_RCGC0_R &= ~0x00000300;  // 125K samplig rate, RCGC0ADC register (page 456)
51    ADC0_SSPRI_R = 0x0123;          // SS3 = highest priority SS2=second p,...
52          // as we are using SS3 (we want to sample 1 at a time not multiple
53  //
54  //13.4.2 Sample Sequencer Configuration
55  //  Configuration of the sample sequencers is slightly more complex than the module initialization
56  //  because each sample sequencer is completely programmable.
57  //
58  //The configuration for each sample sequencer should be as follows:
59  //
60  //  1. Ensure that the sample sequencer is disabled by clearing the corresponding ASENn bit in
61  //  the ADCACTSS register. Programming of the sample sequencers is allowed without having them
62  //  enabled. Disabling the sequencer during programming prevents erroneous execution if a trigger
63  //  event were to occur during the configuration process.
64    ADC0_ACTSS_R &= ~0x0008;        // disable SS3
```

tm4c123gh6pm Datasheet.pdf - Adobe Acrobat Pro

File Edit View Document Comments Forms Tools Advanced Window Help

821 / 1409

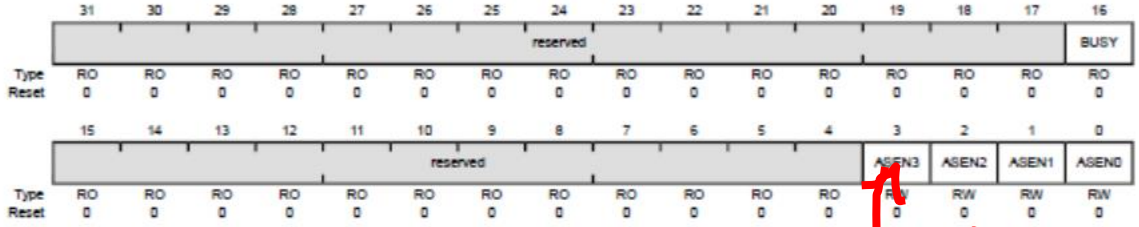
75%

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the sample sequencers. Each sample sequencer can be enabled or disabled independently.

ADC Active Sample Sequencer (ADCACTSS)

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



Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	BUSY	RO	0	ADC Busy Value Description 0 ADC is idle 1 ADC is busy
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ASEN3	RW	0	ADC SS3 Enable Value Description 0 Sample Sequencer 3 is disabled. 1 Sample Sequencer 3 is enabled

8.50 x 11.00 in

```
66  //  2. Configure the trigger event for the sample sequencer in the ADCEMUX register.
67  //  (datasheet page 833) ADCEMUX
68    ADC0_EMUX_R &= ~0xF000;          // software trigger field for SS3, default
69
70  //  3. When using a PWM generator as the trigger source, use the ADC Trigger Source Select
71  //  (ADCTSSEL) register to specify in which PWM module the generator is located. The default
72  //  register reset selects PWM module 0 for all generators.
73  //
74  //  (skip) Not relevant for our use
75
76  //  4. For each sample in the sample sequence, configure the corresponding input source in
77  //  the ADCSSMUXn register.
78    ADC0_SSMUX3_R &= ~0x000F;        // clear SS3
79    ADC0_SSMUX3_R += 1;              // set channel to Ain1/PE2
80
81  //  5. For each sample in the sample sequence, configure the sample control bits in the
82  //  corresponding nibble in the ADCSSCTLn register. When programming the last nibble, ensure that
83  //  the END bit is set. Failure to set the END bit causes unpredictable behavior.
84    ADC0_SSCTL3_R = 0x0006;          // no TS0 D0, yes IE0 END0
85
86  //  6. If interrupts are to be used, set the corresponding MASK bit in the ADCIM register.
87  //
88    ADC0_IM_R &= ~0x0008;            //disable SS3 interrupts
89
90
91  //  7. Enable the sample sequencer logic by setting the corresponding ASENn bit in the
92  //  ADCACTSS register.
93    ADC0_ACTSS_R |= 0x0008;          // enable SS3
94
95  }
```

The end of ADC0_Init

tm4c123gh6pm Datasheet.pdf - Adobe Acrobat Pro

File Edit View Document Comments Forms Tools Advanced Window Help

835 / 1409

75%

Select text and images for copying and pasting with Select tool; hold Spacebar to change to Hand tool f scrolling

Tiva™

Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The ADCEMUX selects the event (trigger) that initiates sampling for each sample sequencer. Each sample sequencer can be configured with a unique trigger source. When using a PWM generator as the trigger source, the ADCEMUX register selects which generator within a PWM module is used as a trigger and the PSn field in the ADC Trigger Source Select (ADCTSSEL) register specifies the PWM module instance in which the generator is located.

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															
RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
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			
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCD

Bit/Field	Name	Type	Reset	Description
15:12	EM3	RW	0x0	SS3 Trigger Select This field selects the trigger source for Sample Sequencer 3. The valid configurations for this field are:

Value	Event
0x0	Processor (default) The trigger is initiated by setting the SSn bit in the **ADCPSSI** register.
0x1	Analog Comparator 0 This trigger is configured by the **Analog Comparator Control 0 (ACCTL0)** register (page 1227).
0x2	Analog Comparator 1 This trigger is configured by the **Analog Comparator Control 1 (ACCTL1)** register (page 1227).
0x3	reserved
0x4	External (GPIO Pins) This trigger is connected to the GPIO interrupt for the corresponding GPIO (see "ADC Trigger Source" on page 655). **Note:** GPIOs that have AINx signals as alternate functions can be used to trigger the ADC. However, the pin cannot be used as both a GPIO and an analog input.
0x5	Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (page 737).
0x6	PWM generator 0 The PWM generator 0 trigger can be configured with the **PWM0 Interrupt and Trigger Enable (PWM0INTEN)** register (page 1271).
0x7	PWM generator 1 The PWM generator 1 trigger can be configured with the **PWM1INTEN** register (page 1271).
0x8	PWM generator 2 The PWM generator 2 trigger can be configured with the **PWM2INTEN** register (page 1271).
0x9	PWM generator 3 The PWM generator 3 trigger can be configured with the **PWM3INTEN** register (page 1271).
0xA-0xE	reserved
0xF	Always (continuously sample)

ADC0_In function

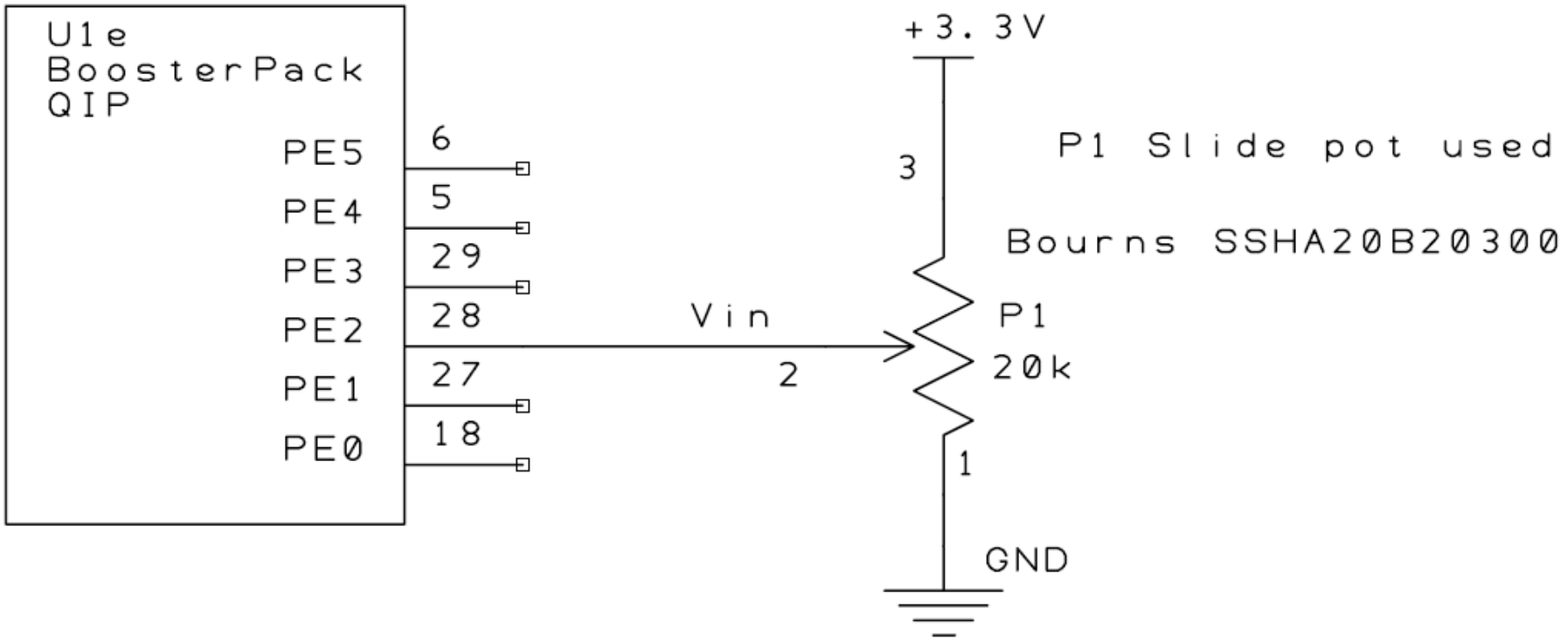
ADC0_In

```
// Busy-wait reading of ADC
// Returns: 12-bit result of ADC conversion
unsigned long ADC0_In(void){
// Busy-wait reading of ADC
// Returns: 12-bit result of ADC conversion
//
  unsigned long result;
//ADC Processor Sample Sequence Initiate datasheet 845
//Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the ADCACTSS register.
  ADC0_PSSI_R = 0x0008;
//
//ADC Raw Interrupt Status (ADCRIS), page 823
//bit3 is for SS3: A sample has completed conversion and the respective ADCSSCTL3 IEn bit is set,
//enabling a raw interrupt.
  while((ADC0_RIS_R&0x08)==0){};
//
//data sheet 860,ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3)
//ADCSSFIFO3 for Sequencer 3).
//Reads of this register return conversion result data in the order sample 0, sample 1, and so on,
// until the FIFO is empty. If the FIFO is not properly handled by software,
// overflow and underflow conditions are registered in the ADCOSTAT and ADCUSTAT registers.
  result = ADC0_SSFIFO3_R&0xFFF;

//data sheet 828, ADC Interrupt Status and Clear (ADCISC),
//bit 3:SS3 Interrupt Status and Clear
//Both the INR3 bit in the ADCRIS register and the MASK3 bit in the ADCIM register are set,
//providing a level-based interrupt to the interrupt controller.
// set it to 1: This bit is cleared by writing a 1.
// Clearing this bit also clears the INR3 bit in the ADCRIS register.
  ADC0_ISC_R = 0x0008;
  return result;
```

Example: Measure Distance

- Use SysTick interrupts to periodically initiate a software triggered ADC conversion, convert the sample to a fixed point decimal distance, and store the result in a mailbox (one of the Inter Thread Communication methods).
- The foreground thread takes the result from the mailbox, converts the result to a string, and prints it


U1e
BoosterPack
QIP
PE5
PE4
PE3
PE2
PE1
PE0
6
5
29
28
27
18
Vin
2
+3.3V
3
P1
20k
1
GND
P1 Slide pot used
Bourns SSHA20B20300

```
unsigned long Size = 6; // potentiomter range in cm
unsigned char String[10]; // string to output to screen
unsigned long Distance; // potentiometer distance in units of 0.001 cm
unsigned long ADCdata;  // 12-bit 0 to 4095 sample
unsigned long Flag;     // 1 means valid Distance, 0 means Distance is empty
```

```
unsigned long Convert(unsigned long sample){
  // Converts ADC input to actual distance in units of 0.001cm
  // Input: sample  12-bit ADC sample
  // Output: 32-bit distance (resolution 0.001cm)
  return (int)sample/4095.0 * Size * 1000;
}
```

```
int main(void){
  // main function
  volatile unsigned long delay;
  ADC0_Init();
  SysTick_Init(2000000); // 40Hz (assuming 80MHz PLL)
  EnableInterrupts();

  while(1){
    if(Flag){
      Flag = 0;
      ADCdata = ADC0_In();
      Distance = Convert(ADCdata)     ;
    }
  }
}
```

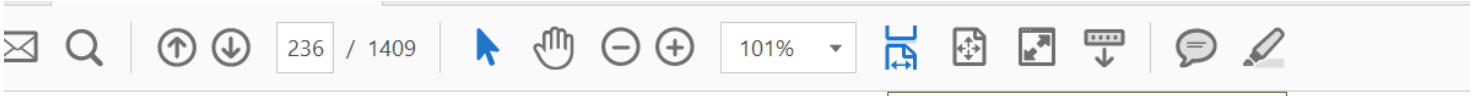
```
// Slide pot pin 3 connected to +3.3V
// Slide pot pin 2 connected to PE2
// Slide pot pin 1 connected to ground
```

- Length ∝ Voltage receiving
- Based on voltage, MCU converts into digital value

Reading

Vol.1	Vol.2
Ch.10 (10.1, 10.2, 10.3, 10.4)	Ch.6 (6.4) Ch.8 (8.4, 8.5) Ch.10 (10.2)

*Useful registers: Peripheral Ready registers.



Fit to window width and enable scrolling

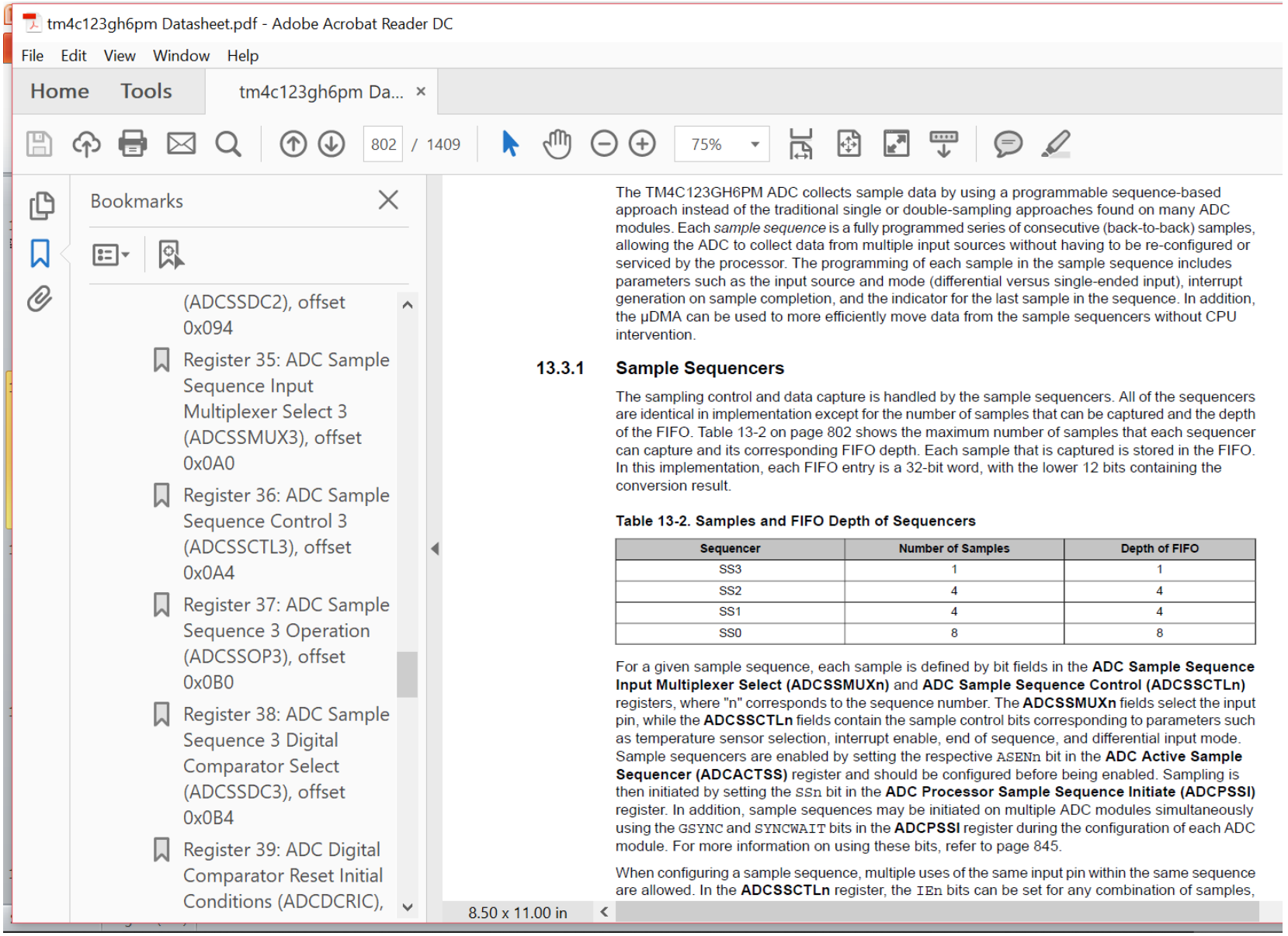
Table 5-7. System Control Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xA04	PRTIMER	RO	0x0000.0000	16/32-Bit General-Purpose Timer Peripheral Ready	404
0xA08	PRGPIO	RO	0x0000.0000	General-Purpose Input/Output Peripheral Ready	406
0xA0C	PRDMA	RO	0x0000.0000	Micro Direct Memory Access Peripheral Ready	408
0xA14	PRHIB	RO	0x0000.0001	Hibernation Peripheral Ready	409
0xA18	PRUART	RO	0x0000.0000	Universal Asynchronous Receiver/Transmitter Peripheral Ready	410
0xA1C	PRSSI	RO	0x0000.0000	Synchronous Serial Interface Peripheral Ready	412
0xA20	PRI2C	RO	0x0000.0000	Inter-Integrated Circuit Peripheral Ready	414
0xA28	PRUSB	RO	0x0000.0000	Universal Serial Bus Peripheral Ready	416
0xA34	PRCAN	RO	0x0000.0000	Controller Area Network Peripheral Ready	417
0xA38	PRADC	RO	0x0000.0000	Analog-to-Digital Converter Peripheral Ready	418
0xA3C	PRACMP	RO	0x0000.0000	Analog Comparator Peripheral Ready	419
0xA40	PRPWM	RO	0x0000.0000	Pulse Width Modulator Peripheral Ready	420
0xA44	PRQEI	RO	0x0000.0000	Quadrature Encoder Interface Peripheral Ready	421
0xA58	PREEPROM	RO	0x0000.0000	EEPROM Peripheral Ready	422
0xA5C	PRWTIMER	RO	0x0000.0000	32/64-Bit Wide General-Purpose Timer Peripheral Ready	423
System Control Legacy Registers					
0x008	DC0	RO	0x007F.007F	Device Capabilities 0	425
0x010	DC1	RO	0x1333.2FFF	Device Capabilities 1	427

More on ADC Sample sequencers

FIFOS for each Sample Sequencer

➔Have different depths based on number of samples that each can take



The TM4C123GH6PM ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approaches found on many ADC modules. Each *sample sequence* is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the processor. The programming of each sample in the sample sequence includes parameters such as the input source and mode (differential versus single-ended input), interrupt generation on sample completion, and the indicator for the last sample in the sequence. In addition, the µDMA can be used to more efficiently move data from the sample sequencers without CPU intervention.

13.3.1 Sample Sequencers

The sampling control and data capture is handled by the sample sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 13-2 on page 802 shows the maximum number of samples that each sequencer can capture and its corresponding FIFO depth. Each sample that is captured is stored in the FIFO. In this implementation, each FIFO entry is a 32-bit word, with the lower 12 bits containing the conversion result.

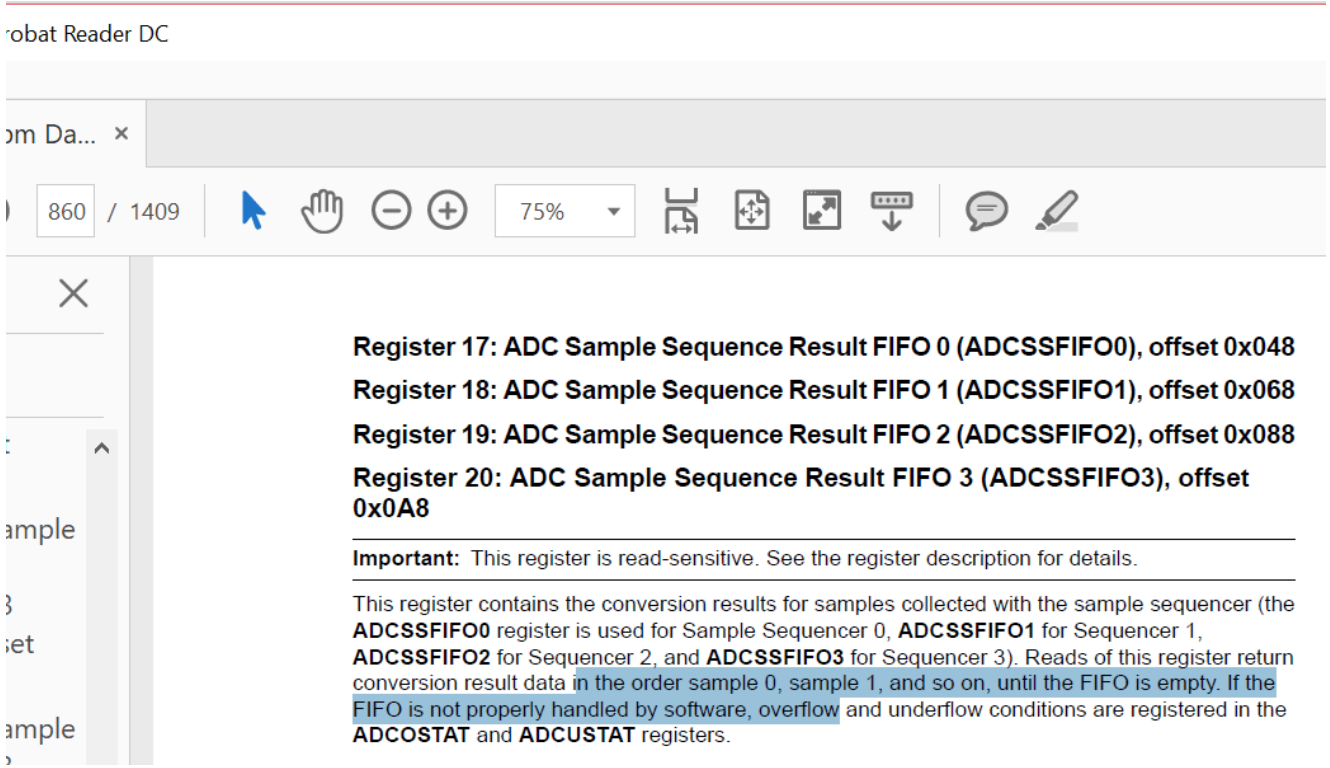
Table 13-2. Samples and FIFO Depth of Sequencers

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

For a given sample sequence, each sample is defined by bit fields in the **ADC Sample Sequence Input Multiplexer Select (ADCSSMUXn)** and **ADC Sample Sequence Control (ADCSSCTLn)** registers, where "n" corresponds to the sequence number. The **ADCSSMUXn** fields select the input pin, while the **ADCSSCTLn** fields contain the sample control bits corresponding to parameters such as temperature sensor selection, interrupt enable, end of sequence, and differential input mode. Sample sequencers are enabled by setting the respective `ASENn` bit in the **ADC Active Sample Sequencer (ADCACTSS)** register and should be configured before being enabled. Sampling is then initiated by setting the `SSn` bit in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register. In addition, sample sequences may be initiated on multiple ADC modules simultaneously using the `GSYNC` and `SYNCWAIT` bits in the **ADCPSSI** register during the configuration of each ADC module. For more information on using these bits, refer to page 845.

When configuring a sample sequence, multiple uses of the same input pin within the same sequence are allowed. In the **ADCSSCTLn** register, the `IEn` bits can be set for any combination of samples,

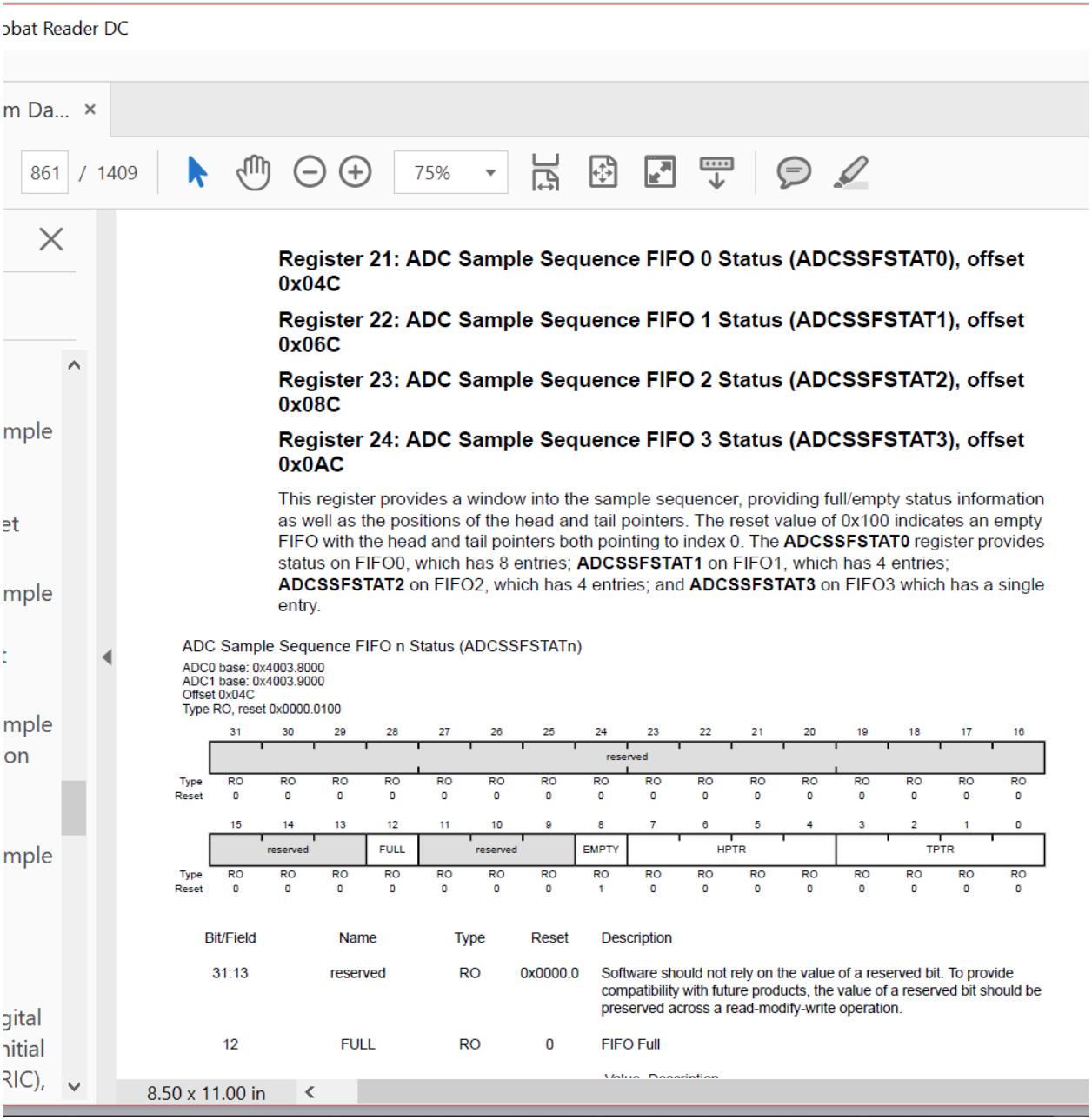
The order of reading is sample 0 -> 1-> 2 and so on. Therefore LSB chunk can't be the end of sequence except for the 1 sample case (SS3)




```
void ADC_In89(uint32_t data[2]){
  ADC0_PSSI_R = 0x0004;          // 1) initiate SS2
  while((ADC0_RIS_R&0x04)==0){};  // 2) wait for conversion done
  data[1] = ADC0_SSFIFO2_R&0xFFF;  // 3A) read first result
  data[0] = ADC0_SSFIFO2_R&0xFFF;  // 3B) read second result
  ADC0_ISC_R = 0x0004;           // 4) acknowledge completion
}
```

An example of FIFO reading with 2 samples per sequence

(A sidebar) These registers allow you to check condition of FIFOs while reading data from them.



obat Reader DC

m Da... ×

861 / 1409

75%

Register 21: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C

Register 22: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C

Register 23: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C

Register 24: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC

This register provides a window into the sample sequencer, providing full/empty status information as well as the positions of the head and tail pointers. The reset value of 0x100 indicates an empty FIFO with the head and tail pointers both pointing to index 0. The **ADCSSFSTAT0** register provides status on FIFO0, which has 8 entries; **ADCSSFSTAT1** on FIFO1, which has 4 entries; **ADCSSFSTAT2** on FIFO2, which has 4 entries; and **ADCSSFSTAT3** on FIFO3 which has a single entry.

ADC Sample Sequence FIFO n Status (ADCSSFSTATn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x04C
Type RO, reset 0x0000.0100

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			FULL	reserved			EMPTY	HPTR				TPTR			
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	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	FULL	RO	0	FIFO Full

Value Description

8.50 x 11.00 in

Here is the register that you define sample order from each ADC channel

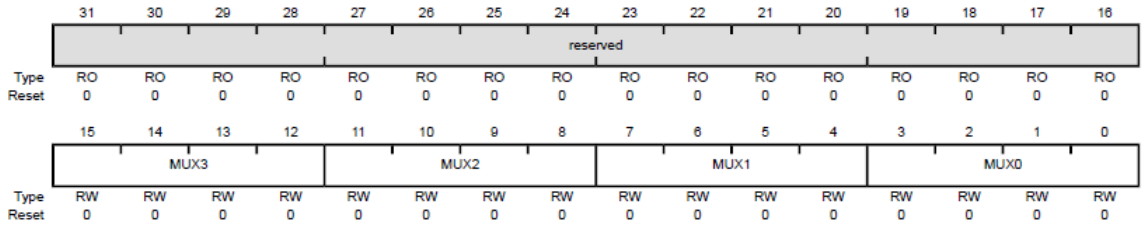
Register 27: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

Register 28: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 1 or 2. These registers are 16 bits wide and contain information for four possible samples. See the **ADCSSMUX0** register on page 851 for detailed bit descriptions. The **ADCSSMUX1** register affects Sample Sequencer 1 and the **ADCSSMUX2** register affects Sample Sequencer 2.

ADC Sample Sequence Input Multiplexer Select n (ADCSSMUXn)

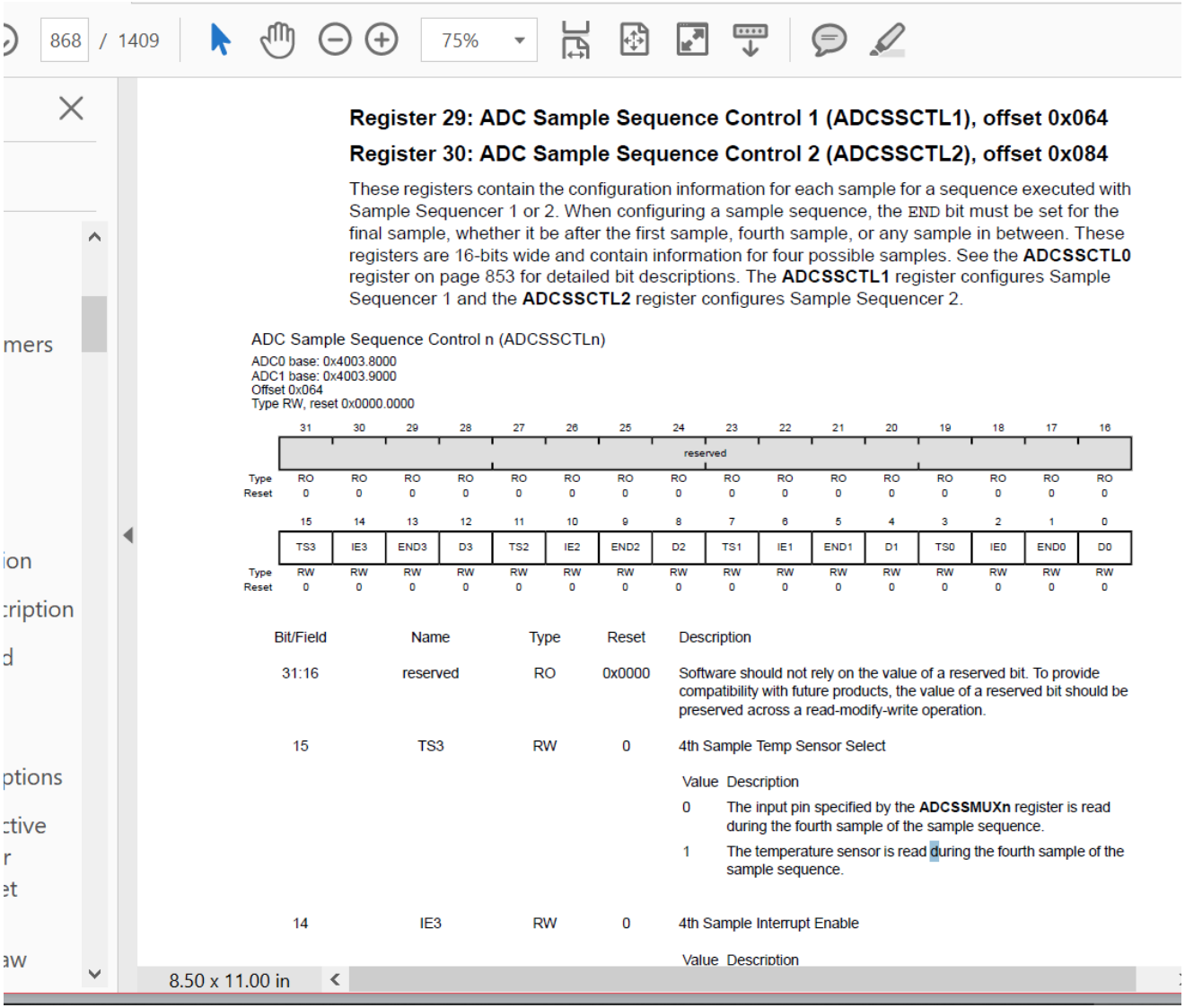
ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x060
Type RW, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	MUX3	RW	0x0	4th Sample Input Select
11:8	MUX2	RW	0x0	3rd Sample Input Select
7:4	MUX1	RW	0x0	2nd Sample Input Select
3:0	MUX0	RW	0x0	1st Sample Input Select

This register defines the end of your sample sequence and more. Assuming that you have two samples in your ADC sequence,

- **For 1st sample: “zero” on TS0 D0 IE0 END0,**
- **For 2nd sample: “one” on IE1 END1 “zero” on TS1 D1**



868 / 1409 75%

Register 29: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

Register 30: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084

These registers contain the configuration information for each sample for a sequence executed with Sample Sequencer 1 or 2. When configuring a sample sequence, the END bit must be set for the final sample, whether it be after the first sample, fourth sample, or any sample in between. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSCTL0** register on page 853 for detailed bit descriptions. The **ADCSSCTL1** register configures Sample Sequencer 1 and the **ADCSSCTL2** register configures Sample Sequencer 2.

ADC Sample Sequence Control n (ADCSSCTLn)

ADC0 base: 0x4003.8000
ADC1 base: 0x4003.9000
Offset 0x064
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
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	TS3	RW	0	4th Sample Temp Sensor Select Value Description 0 The input pin specified by the **ADCSSMUXn** register is read during the fourth sample of the sample sequence. 1 The temperature sensor is read during the fourth sample of the sample sequence.
14	IE3	RW	0	4th Sample Interrupt Enable Value Description

8.50 x 11.00 in

An example

- ADC0 sample sequencer 2 is used which takes up to four samples
- In this example, two samples are needed.
- Sample sequencer 2 generates a raw interrupt when the second conversion is complete

ADC Init snapshot

```
// Initializes ADC8 and ADC9 sampling
// 125k max sampling
// SS2 triggering event: software trigger, busy-wait sampling
// SS2 1st sample source: Ain9 (PE4)
// SS2 2nd sample source: Ain8 (PE5)
// SS2 interrupts: enabled after 2nd sample but not promoted to controller
void ADC_Init89(void){
  volatile uint32_t delay;
//  SYSCTL_RCGC0_R |= 0x00010000; // 1) activate ADC0 (legacy code)
  SYSCTL_RCGCADC_R |= 0x00000001; // 1) activate ADC0
  SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R4; // 1) activate clock for Port E
  delay = SYSCTL_RCGCGPIO_R;      // 2) allow time for clock to stabilize
  delay = SYSCTL_RCGCGPIO_R;
  GPIO_PORTE_DIR_R &= ~0x30;      // 3) make PE4 PE5 input
  GPIO_PORTE_AFSEL_R |= 0x30;     // 4) enable alternate function on PE4 PE5
  GPIO_PORTE_DEN_R &= ~0x30;      // 5) disable digital I/O on PE4 PE5
                                  // 5a) configure PE4 as ?? (skip this line because PCTL is for digital only)
  GPIO_PORTE_PCTL_R = GPIO_PORTE_PCTL_R&0xFF00FFFF;
  GPIO_PORTE_AMSEL_R |= 0x30;     // 6) enable analog functionality on PE4 PE5
  ADC0_PC_R &= ~0xF;              // 8) clear max sample rate field
  ADC0_PC_R |= 0x1;               //    configure for 125K samples/sec
  ADC0_SSPRI_R = 0x3210;          // 9) Sequencer 3 is lowest priority
  ADC0_ACTSS_R &= ~0x0004;        // 10) disable sample sequencer 2
  ADC0_EMUX_R &= ~0x0F00;         // 11) seq2 is software trigger
  ADC0_SSMUX2_R = 0x0089;         // 12) set channels for SS2
  ADC0_SSCTL2_R = 0x0060;         // 13) no TS0 D0 IE0 END0 TS1 D1, yes IE1 END1
  ADC0_IM_R &= ~0x0004;           // 14) disable SS2 interrupts
  ADC0_ACTSS_R |= 0x0004;         // 15) enable sample sequencer 2
}
```

ADC reading code snapshot

```
//------------ADC_In89------------
// Busy-wait Analog to digital conversion
// Input: none
// Output: two 12-bit result of ADC conversions
// Samples ADC8 and ADC9
// 125k max sampling
// software trigger, busy-wait sampling
// data returned by reference
// data[0] is ADC8 (PE5) 0 to 4095
// data[1] is ADC9 (PE4) 0 to 4095
void ADC_In89(uint32_t data[2]){
  ADC0_PSSI_R = 0x0004;             // 1) initiate SS2
  while((ADC0_RIS_R&0x04)==0){};    // 2) wait for conversion done
  data[1] = ADC0_SSFIFO2_R&0xFFF;   // 3A) read first result
  data[0] = ADC0_SSFIFO2_R&0xFFF;   // 3B) read second result
  ADC0_ISC_R = 0x0004;              // 4) acknowledge completion
}
```

More on ADC differential mode (pp.810+)

- Differential input ADC is required two channels to configure it. For our board paring is as follows:

13.3.5 Differential Sampling

In addition to traditional single-ended sampling, the ADC module supports differential sampling of two analog input channels. To enable differential sampling, software must set the Dn bit in the **ADCSSCTL0n** register in a step's configuration nibble.

When a sequence step is configured for differential sampling, the input pair to sample must be configured in the **ADCSSMUXn** register. Differential pair 0 samples analog inputs 0 and 1; differential pair 1 samples analog inputs 2 and 3; and so on (see Table 13-3 on page 810). The ADC does not support other differential pairings such as analog input 0 with analog input 3.

Table 13-3. Differential Sampling Pairs

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

To use a differential mode, therefore, you need to use two ADC channels accordingly.

- Note that we have used so called single-end ADC, which takes a value from direct ADC conversion.
  - This method can't take negative value (as reference is grounded)
- Taking a difference between + and – voltages from two channel (so called differential mode) allows us to configure negative value.
  - This method is also good for reducing noise as one can even out common noise taken from + and – terminals.