CPE 323
Intro to Embedded Computer Systems
Analog-to-Digital Conversion

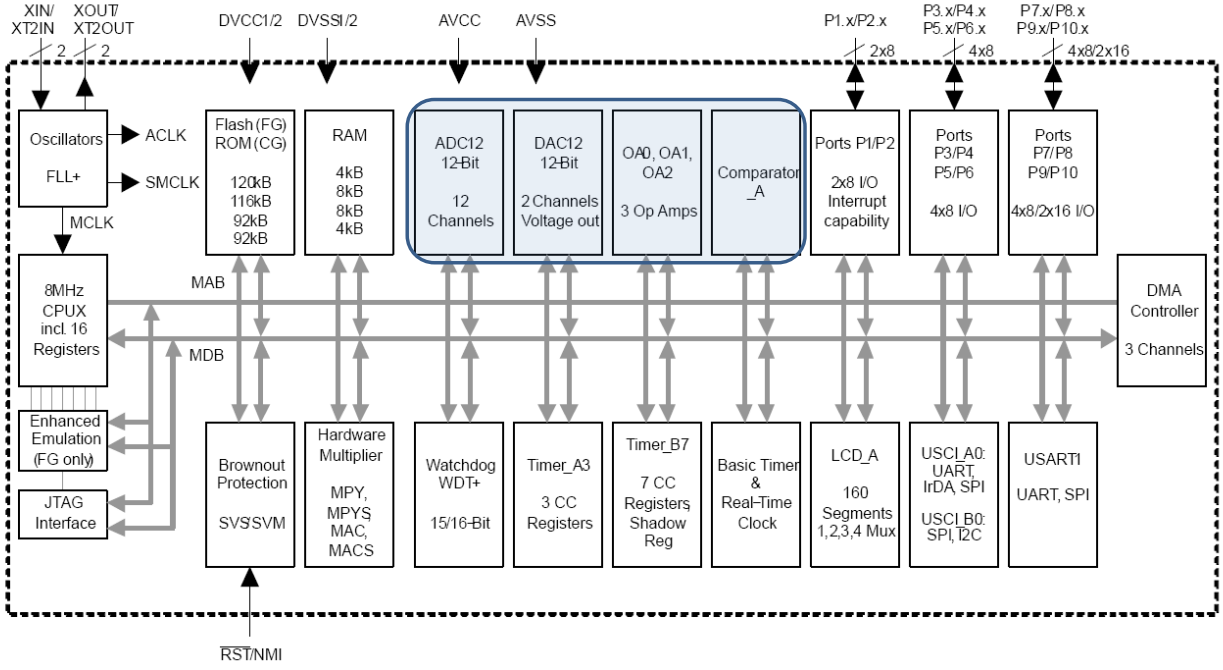
Aleksandar Milenkovic

milenka@uah.edu

Admin

1. Practice quiz
   Clocks, WDT, TimerB, UART
2. Next quiz will be next week on Tuesday
3. HW#4
5. HW.5 will be coming soon
6. Missing Panopto (from Wednesday => use Zoom recording)

MSP430FG4618 Block Diagram



MSP430F5529 Block Diagram

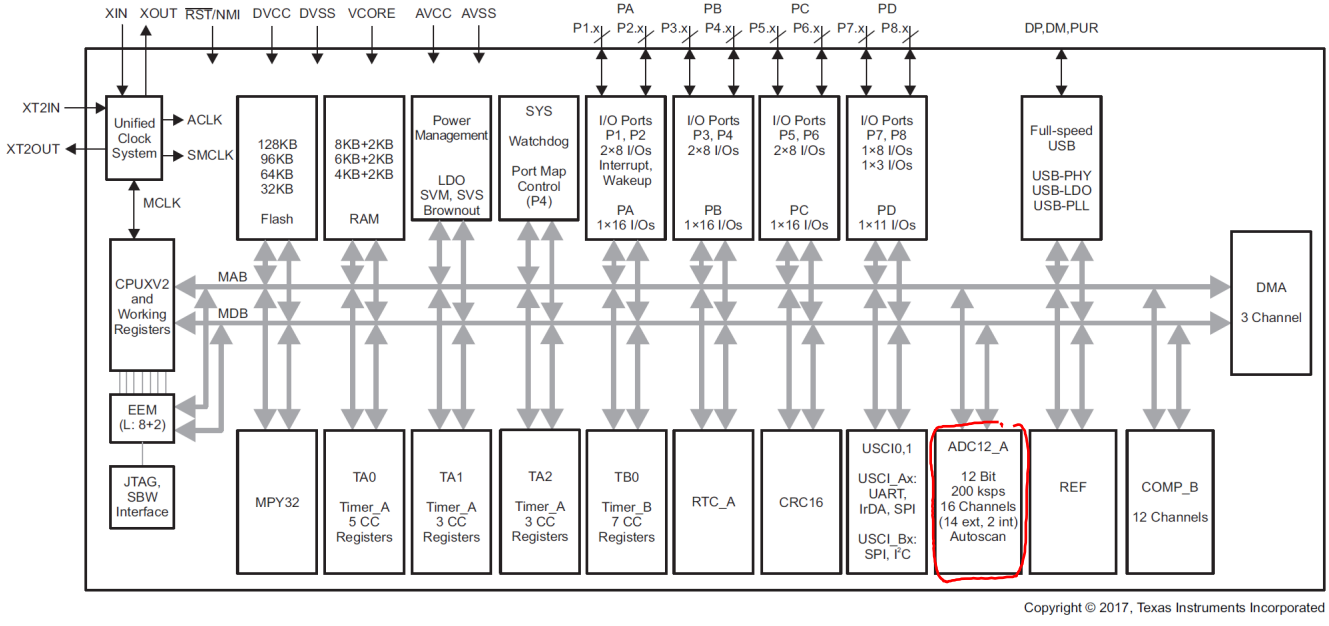
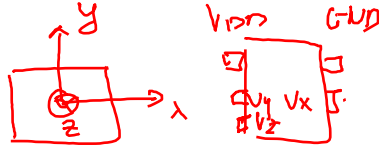


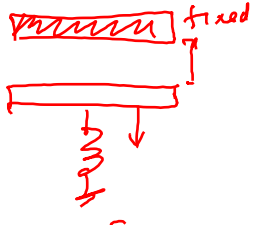
Figure 1-1. Functional Block Diagram – MSP430F5529IPN, MSP430F5527IPN, MSP430F5525IPN, MSP430F5521IPN

Interfacing Physical World: From Analog Signals to Digital Values

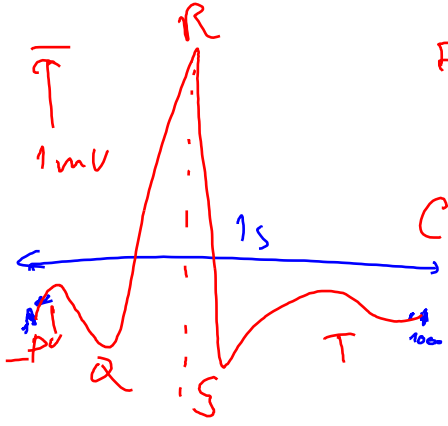
- Sensors/Transducers
  - convert physical quantity into an electrical signals
- Signal Conditioning
  - isolation from dangerous voltages due to static discharges
  - amplification of signals
  - bandwidth limiting: filters
- Analog-to-Digital Converters
  - convert analog signals to digital values



[-2g, 2g]

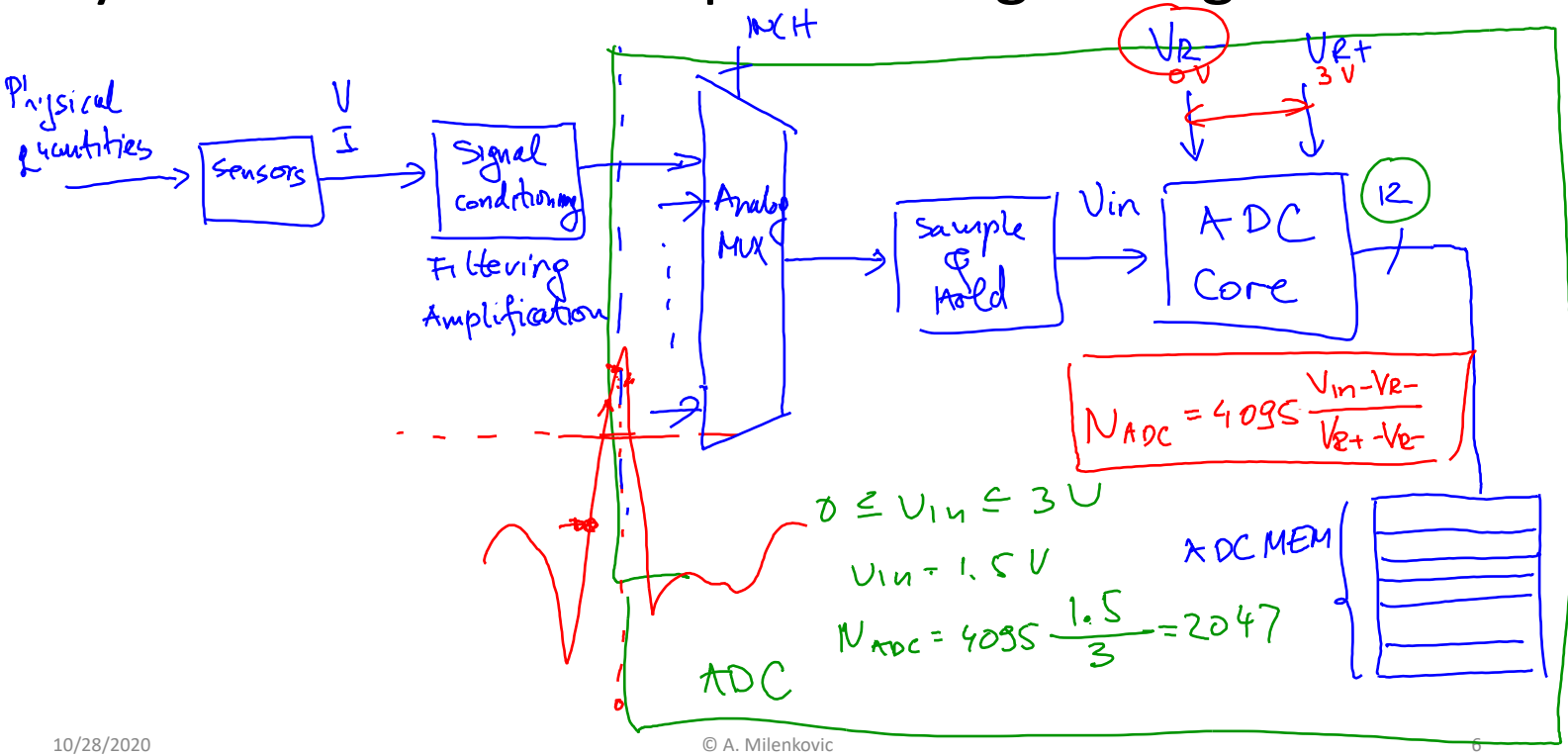


$C = \varepsilon \cdot \frac{S}{d}$



$F_S = 1,000\ Hz$

System View: From Input Analog Voltage to Bits



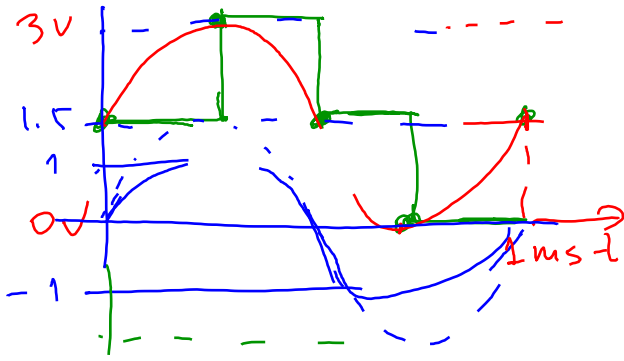
ADC Modules

- Analog Multiplexer
- Sample-and-hold
- AD Conversion Core
- Buffers

Definitions

- Resolution
- Accuracy
- Transfer Function
- Aperture Time / Sample time
- Conversion Time – time the core needs to produce the digital counterpart
- Sampling Frequency – how many samples you want to get

$V_{LSB} = \frac{V_{FS}}{2^n}$



$y = \sin(2 \cdot \pi \cdot f \cdot t)$

$f = \frac{1}{1ms} = 1,000\ Hz$

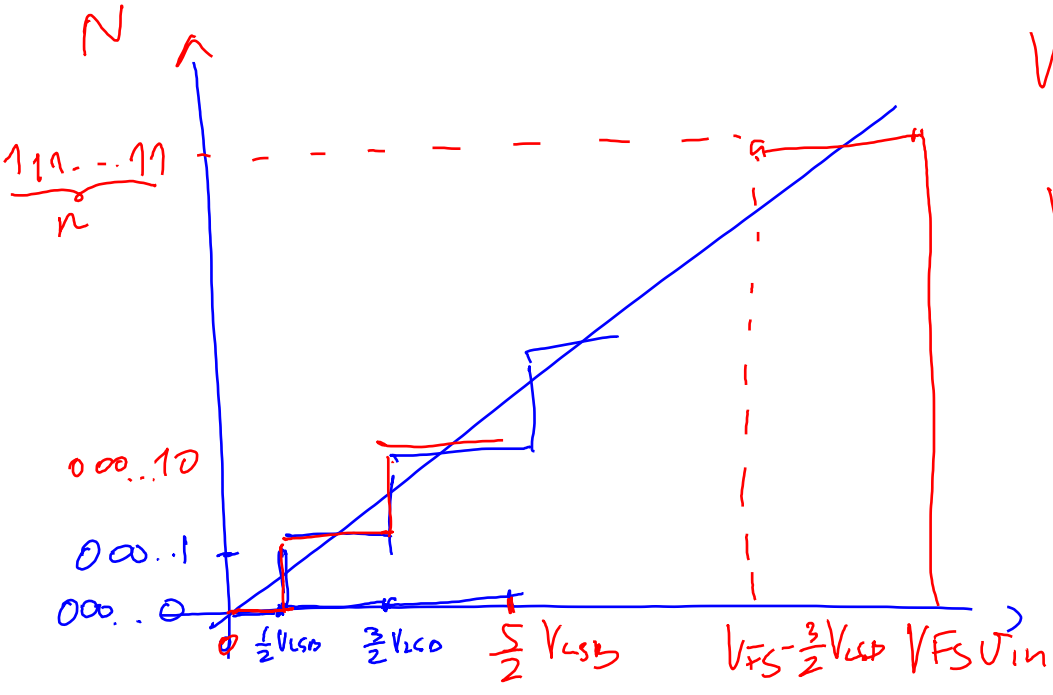
$Input = 1.5 \cdot (1 + \sin(2\pi f t))$, $f = 1,000\ Hz$

$F_S = 4,000\ Hz$

$\Delta t_S = \frac{1}{F_S} = 0.25\ ms$

0
0.25
0.5
0.75
1.0

AD Transfer Function

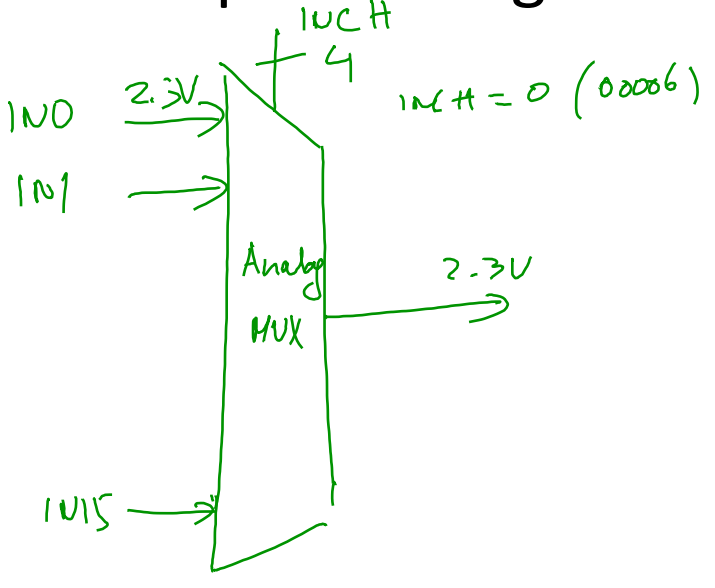


$$V_{LSB} = \frac{V_{FS}}{2^n}$$

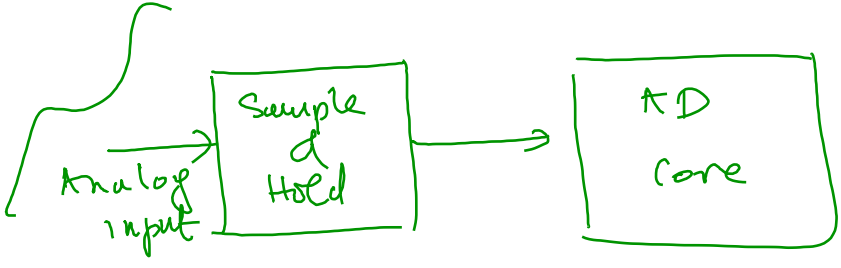
$$V_{FS} = V_{R+} - V_{R-}$$

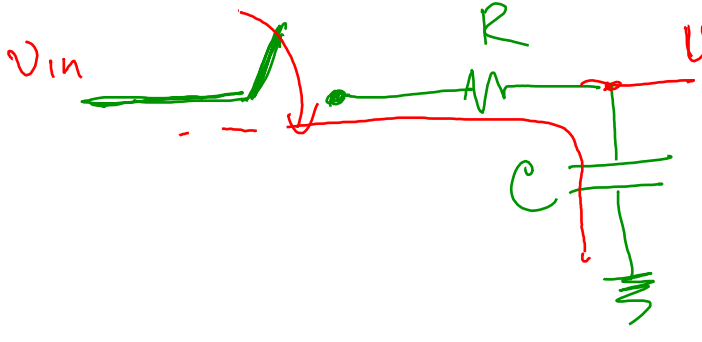
$$V_{R-} = 0\,V$$

Input Analog MUX

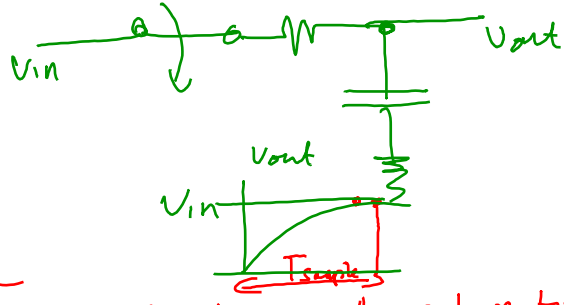


Sample&Hold





Close switch:

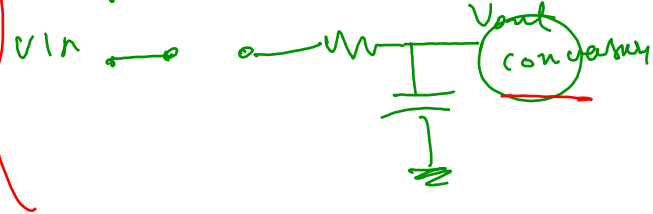


Sample

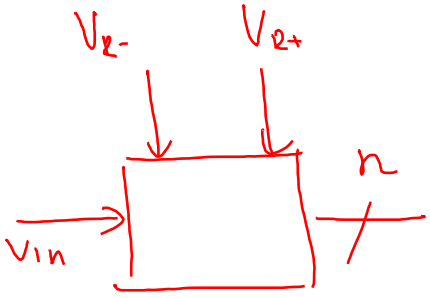
Sample time = Aperature time

Open switch

hold



ADC Core

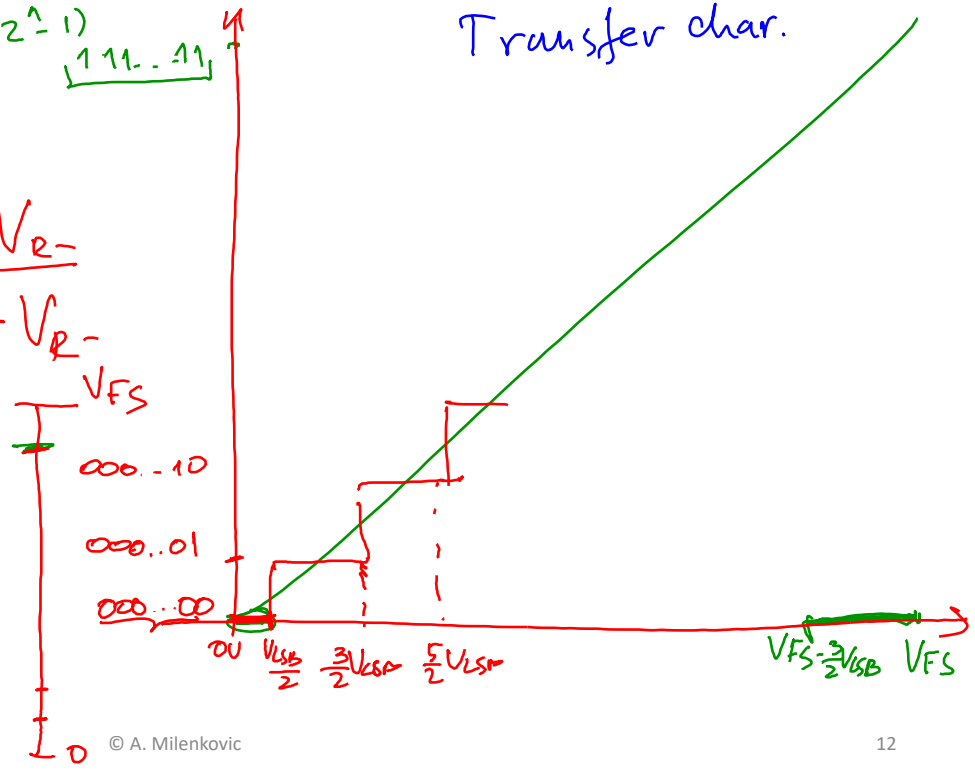


$$N_{ADC} = (2^n - 1) \cdot \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}}$$

$$V_{FS} = V_{R+} - V_{R-}$$

$V_{LSB} = ?$

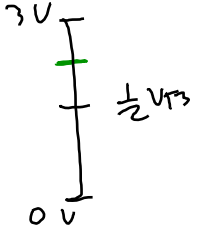
$$V_R = V_{LSB} = \frac{V_{FS}}{2^n}$$



Successive Approximation Register (SAR) ADC Core

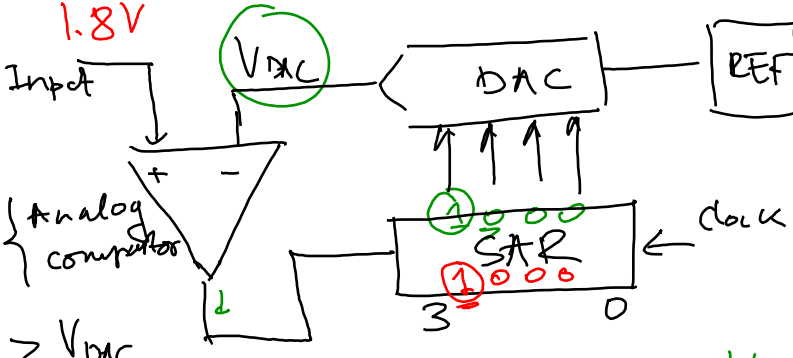
n = 4-bit converter

$V_{FS} = 3V$



$V_{LSB} = V_R = \frac{V_{FS}}{2^n} = \frac{3V}{2^4} = 0.1875V$

SAR: 1 0 0 1



Input > V_{DAC} => 1

otherwise => Ø

ADC12

$T_{conv} = 13 \cdot T_{ADC}$

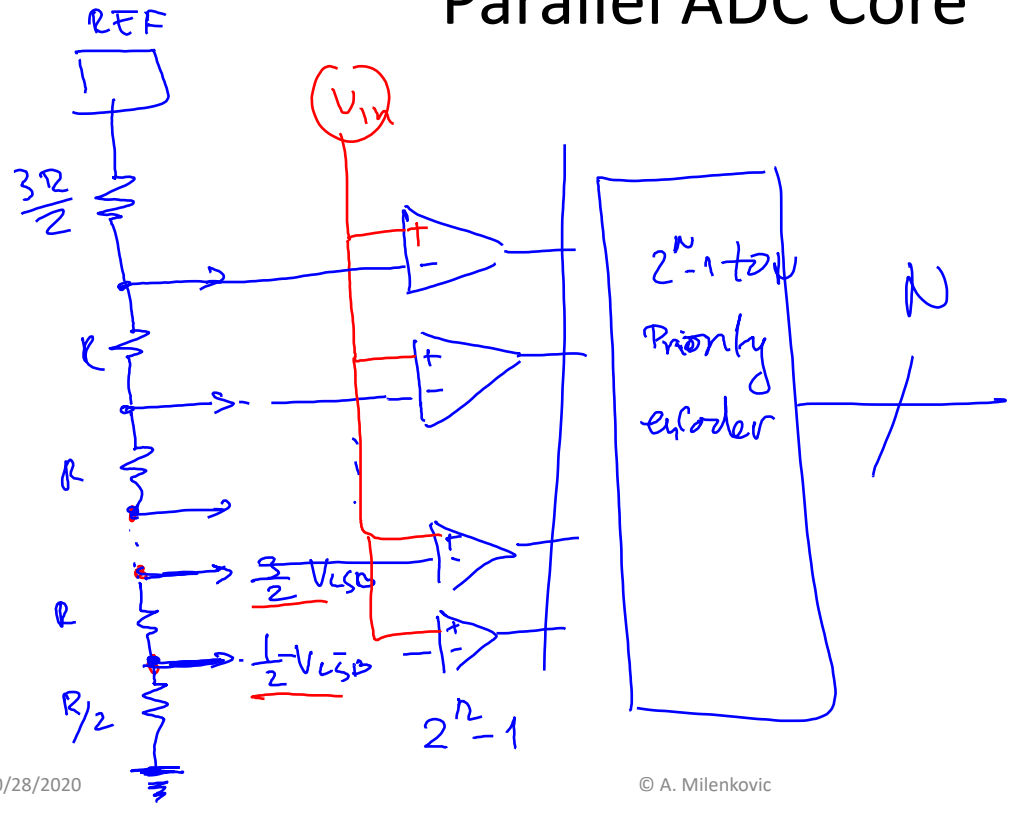
step 1: SAR: 1000 => $V_{DAC} = 8 \cdot V_{LSB} = 1.5V$
$V_{in} > V_{DAC}$ => keep 1 as the MSB

step 2: SAR: 1100 => $V_{DAC} = 12 \cdot V_{LSB} = 2.25V$
$V_{in} < V_{DAC}$ => bit 2 is Ø

step 3: SAR: 1010 => $V_{DAC} = 10 \cdot V_{LSB} = 1.875V$
$V_{in} < V_{DAC}$ => bit 1 is Ø

step 4: SAR: 1001 => $V_{DAC} = 9 \cdot V_{LSB} = 1.6875V$
$V_{in} > V_{DAC}$ => bit Ø is 1

Parallel ADC Core



$T_{CONV} = 1\ T_{ADCLK}$

ADC12

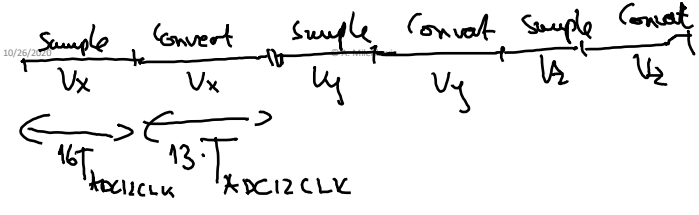
CPE 323

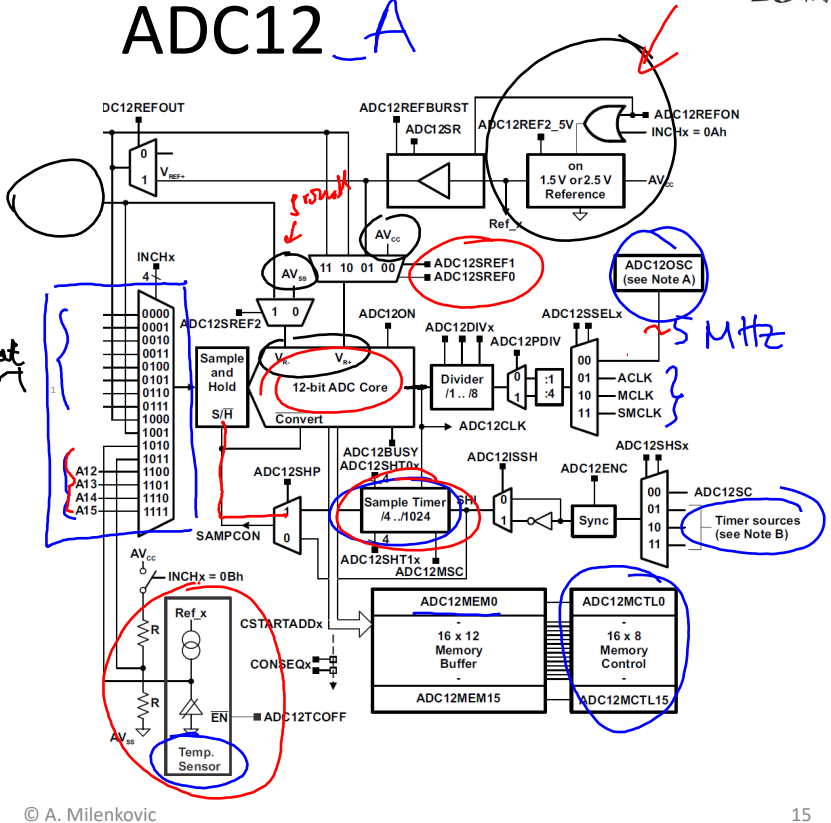
Intro to Embedded Computer Systems

Analog-to-Digital Conversion

Aleksandar Milenkovic

milenka@uah.edu





Walk Through

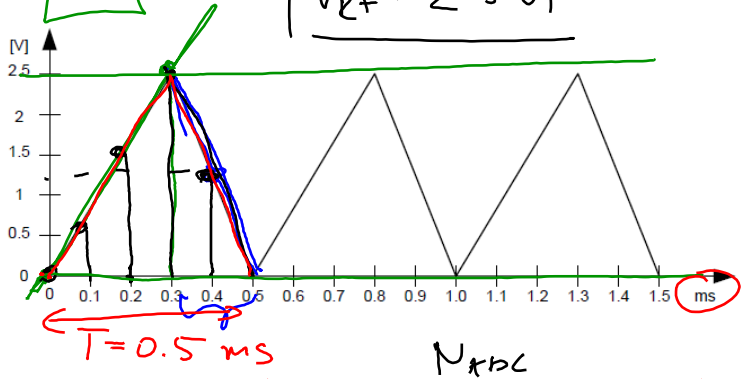
ADC12

$V_{R-} = 0\ V$

$V_{R+} = 2.5\ V$

$N_{ADC} = 4095 \frac{V_{in}}{V_{R+}}$

- Fs=10,000 Hz (sampling rate)
- $T_{s_{grad}}$ = 0.5 ms = 5 · Δt_s
- Vmin= 0 V
- Vmax= 2.5 V
- Find samples for one period



N_{ADC}

$F_s = 10{,}000\ Hz \Rightarrow 10{,}000$ samples in one second

$\Delta t_s = \frac{1}{10000} = 0.1\ ms$

0, Δt_s, $2\Delta t_s$, ...

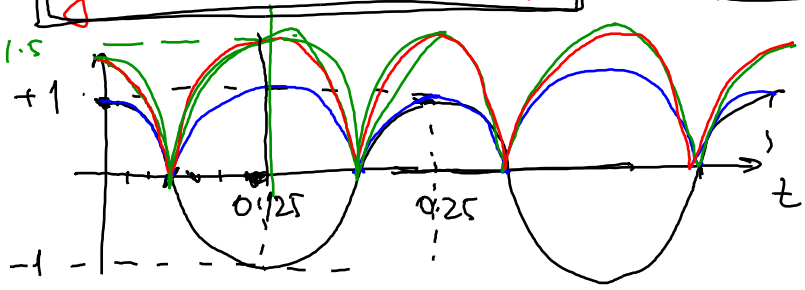
$$N_{ADC} = nint\left(4095 \cdot \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}}\right)$$

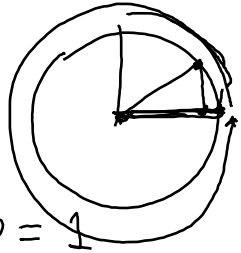
t	V	N_{ADC}
t=0	0 V	0
t=0.1ms	$\frac{2.5}{3} = 0.833$	.
t=0.2ms	$2 \cdot \frac{2.5}{3} = 1.666$	.
t=0.3ms	2.5 V	0x0FFF
t=0.4ms	1.25V	0x0800

$y = 1.5 \cdot |\cos(2\pi f t)|$ $f = 4{,}000$ Hz

$T = \frac{1}{4{,}000} = 0.25 \cdot 10^{-3}$

14-bit ADC





$\cos(2\pi f t)$

$t = 0 \rightarrow \cos 0 = 1$

$t = 0.0625$ eq $\frac{T}{2} = 0$

$N_{ADC} = (2^{14}-1) \cdot \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}}$

$|\cos(2\pi \cdot f t)|$

1) $V_{min} = 0$ V, $V_{max} = 1.5$ V } $V_{R-} = 0$ V, $V_{R+} = 1.5$ V

2) $T_y = \frac{0.25\text{ms}}{2} = 0.125$ ms $F_y = 2 \cdot f = 8{,}000$ Hz $F_y = \frac{1}{T_y}$

3) 5 samples per one period

$\Delta t_s = \frac{T_y}{5} = \frac{0.125\text{ms}}{5} = 0.025$ ms

t	y_y	N_{ADC}
0	1.5 V	$2^{14}-1$
$\Delta t_s = 0.025$ ms	$1.5\|\cos(2\pi \cdot 4000 \cdot 0.025 \cdot 10^{-3})\| = 1.21$	
$2\Delta t_s = 0.050$ ms	0.46	
$3\Delta t_s = 0.075$ ms	0.46	
$4\Delta t_s = 0.100$ ms	1.21	

Measuring Temperature (on-chip temperature sensor on MSP430F5529)

- Input channel INCHx=1010 (10)
- Temperature sensor equations
- Sample time > 30 μs
- Calibration may be needed
- TCsensor – slope (mV/°C),
- Vsensor – intercept (mV)

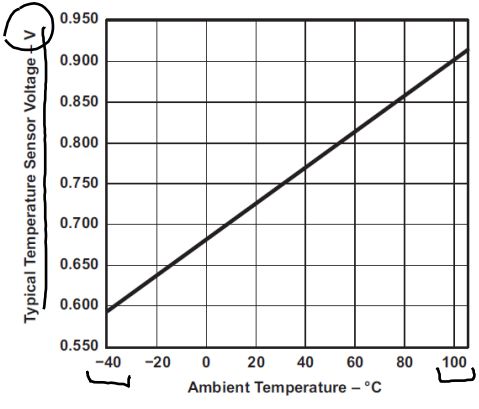


Figure 28-11. Typical Temperature Sensor Transfer Function

$$V_{sense} = TC_{sensor} \cdot \text{Temp} + V_{sensor}$$

$$TEMPC = (ADC.raw - CAL\_ADC\_T30) \times \frac{85-30}{CAL\_ADC\_T85 - CAL\_ADC\_T30} + 30$$

Demo

```
/*-------------------------------------------------------------------------------
 * File:         Lab10_D1.c (CPE 325 Lab10 Demo code)
 *
 * Function:     Measuring the temperature (MPS430F5529)
 *
 * Description: This C program samples the on-chip temperature sensor and
 *              converts the sampled voltage from the sensor to temperature in
 *              degrees Celsius and Fahrenheit. The converted temperature is
 *              sent to HyperTerminal over the UART by using serial UART.
 *
 * Clocks:       ACLK = LFXT1 = 32768Hz, MCLK = SMCLK = DCO = default (~1MHz)
 *               An external watch crystal between XIN & XOUT is required for ACLK
 *
 * Instructions:Set the following parameters in HyperTerminal
 *                   Port :        COM1
 *                   Baud rate :   115200
 *                   Data bits:    8
 *                   Parity:       None
 *                   Stop bits:    1
 *                   Flow Control: None
 *
 *                         MSP430F5529
 *                      -------------------
 *                  /|\|              XIN|-
 *                   | |                 | 32kHz
 *                   --|RST          XOUT|-
 *                     |                 |
 *                     |     P3.3/UCA0TXD|------------>
 *                     |                 | 115200 - 8N1
 *                     |     P3.4/UCA0RXD|<------------
 *                     |                 |
 * Input:        Character Y or y or N or n
 *
 * Output:       Displays Temperature in Celsius and Fahrenheit in HyperTerminal
 * Author:       Aleksandar Milenkovic, milenkovic@computer.org
 *               Prawar Poudel
 *-------------------------------------------------------------------------------*/
```

```
#include  <msp430.h>
#include  <stdio.h>

#define CALADC12_15V_30C  *((unsigned int *)0x1A1A)   // Temperature Sensor
Calibration-30 C
                                                      //See device datasheet for TLV
table memory mapping
#define CALADC12_15V_85C  *((unsigned int *)0x1A1C)   // Temperature Sensor
Calibration-85 C

char ch;                    // Holds the received char from UART
unsigned char rx_flag;      // Status flag to indicate new char is received

char gm1[] = "Hello! I am an MSP430. Would you like to know my temperature? (Y|N)";
char gm2[] = "Bye, bye!";
char gm3[] = "Type in Y or N!";

long int temp;                      // Holds the output of ADC
long int IntDegF;                   // Temperature in degrees Fahrenheit
long int IntDegC;                   // Temperature in degrees Celsius

char NewTem[25];

void UART_setup(void) {

    P3SEL |= BIT3 + BIT4;   // Set USCI_A0 RXD/TXD to receive/transmit data
    UCA0CTL1 |= UCSWRST;    // Set software reset during initialization
    UCA0CTL0 = 0;           // USCI_A0 control register
    UCA0CTL1 |= UCSSEL_2;   // Clock source SMCLK

    UCA0BR0 = 0x09;         // 1048576 Hz  / 115200 lower byte
    UCA0BR1 = 0x00;         // upper byte
    UCA0MCTL = 0x02;        // Modulation (UCBRS0=0x01, UCOS16=0)

    UCA0CTL1 &= ~UCSWRST;   // Clear software reset to initialize USCI state machine
    UCA0IE |= UCRXIE;                       // Enable USCI_A0 RX interrupt
}
```

Demo (cont'd)

```
void UART_putCharacter(char c) {
    while (!(UCA0IFG&UCTXIFG));  // Wait for previous character to transmit
    UCA0TXBUF = c;               // Put character into tx buffer
}

void sendMessage(char* msg, int len) {
    int i;
    for(i = 0; i < len; i++) {
        UART_putCharacter(msg[i]);
    }
    UART_putCharacter('\n');        // Newline
    UART_putCharacter('\r');        // Carriage return
}

void ADC_setup(void) {
    REFCTL0 &= ~REFMSTR;                    // Reset REFMSTR to hand over control
to
                                            // ADC12_A ref control registers
    ADC12CTL0 = ADC12SHT0_8 + ADC12REFON + ADC12ON;
                                            // Internal ref = 1.5V
    ADC12CTL1 = ADC12SHP;                   // enable sample timer
    ADC12MCTL0 = ADC12SREF_1 + ADC12INCH_10; // ADC i/p ch A10 = temp sense i/p
    ADC12IE = 0x001;                        // ADC_IFG upon conv result-ADCMEMO
    __delay_cycles(100);                    // delay to allow Ref to settle
    ADC12CTL0 |= ADC12ENC;
}
```

```
void main(void) {
    WDTCTL = WDTPW | WDTHOLD;           // Stop watchdog timer
    UART_setup();                       // Setup USCI_A0 module in UART mode
    ADC_setup();                        // Setup ADC12

    rx_flag = 0;                        // RX default state "empty"
    _EINT();                            // Enable global interrupts
    while(1) {
        sendMessage(gm1, sizeof(gm1));// Send a greetings message

        while(!(rx_flag&0x01));         // Wait for input
        rx_flag = 0;                    // Clear rx_flag
        sendMessage(&ch, 1);            // Send received char

        // Character input validation
        if ((ch == 'y') || (ch == 'Y')) {

            ADC12CTL0 &= ~ADC12SC;
            ADC12CTL0 |= ADC12SC;                   // Sampling and conversion start

            _BIS_SR(CPUOFF + GIE);      // LPM0 with interrupts enabled

            //in the following equation,
            // ..temp is digital value read
            //..we are using double intercept equation to compute the
            //.. .. temperature given by temp value
            //.. .. using observations at 85 C and 30 C as reference
            IntDegC = (float)(((long)temp - CALADC12_15V_30C) * (85 - 30)) /
                    (CALADC12_15V_85C - CALADC12_15V_30C) + 30.0f;

            IntDegF = IntDegC*(9/5.0) + 32.0;

            // Printing the temperature on HyperTerminal/Putty
            sprintf(NewTem, "T(F)=%ld\tT(C)=%ld\n", IntDegF, IntDegC);
            sendMessage(NewTem, sizeof(NewTem));
        }
        else if ((ch == 'n') || (ch == 'N')) {
            sendMessage(gm2, sizeof(gm2));
            break;                      // Get out
        }
        else {
            sendMessage(gm3, sizeof(gm3));
        }
    }                                   // End of while
    while(1);                           // Stay here forever
}
```

Demo (cont'd)

```
#pragma vector = USCI_A0_VECTOR
__interrupt void USCIA0RX_ISR (void) {
    ch = UCA0RXBUF;                 // Copy the received char
    rx_flag = 0x01;                 // Signal to main
    LPM0_EXIT;
}

#pragma vector = ADC12_VECTOR
__interrupt void ADC12ISR (void) {
    temp = ADC12MEM0;               // Move results, IFG is cleared
    _BIC_SR_IRQ(CPUOFF);            // Clear CPUOFF bit from 0(SR)
}
```