EEET2481

EMBEDDED SYSTEMS D&I

ADC & DAC

RMIT UNIVERSITY

Agenda

- Introduction to Analog to Digital Conversion
- Introduction to NUC140 MCU’s ADC Controller
- NUC140 MCU’s ADC Controller - Registers

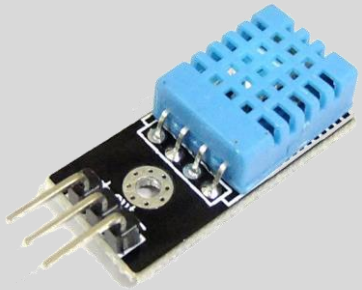
Introduction to Analog to Digital Conversion

RMIT UNIVERSITY

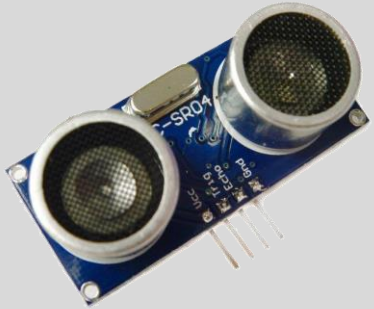
Acquiring Data - Sensors

Most embedded systems are designed to acquire data from the outside world. Acquiring data from the external world requires a sensor.

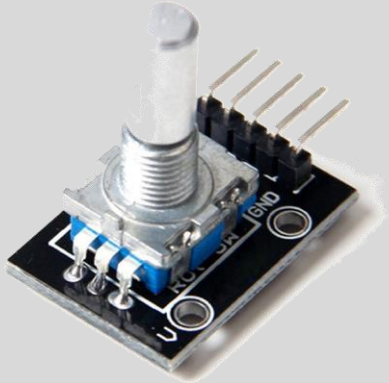
A sensor is a device which measures a physical quantity and converts it into another form – typically an electrical signal.



DHT11 Temp Sensor



HC-SR04 Ultrasonic Sensor



KY-040 Rotary Encoder

Nature of Sensors & Signals

- A lot of sensors and input devices are inherently digital and, therefore, are suitable to be directly interfaced with an embedded system

  e.g. switch, pushbutton, linear or rotary encoders

- Some sensors come packaged with their own processing unit to convert a raw signal into a digital signal that an embedded system will read

  e.g. DHT11 Temp & Humidity sensor

- Other sensors produce a pure analogue signal, which is directly proportional to the measured quantity

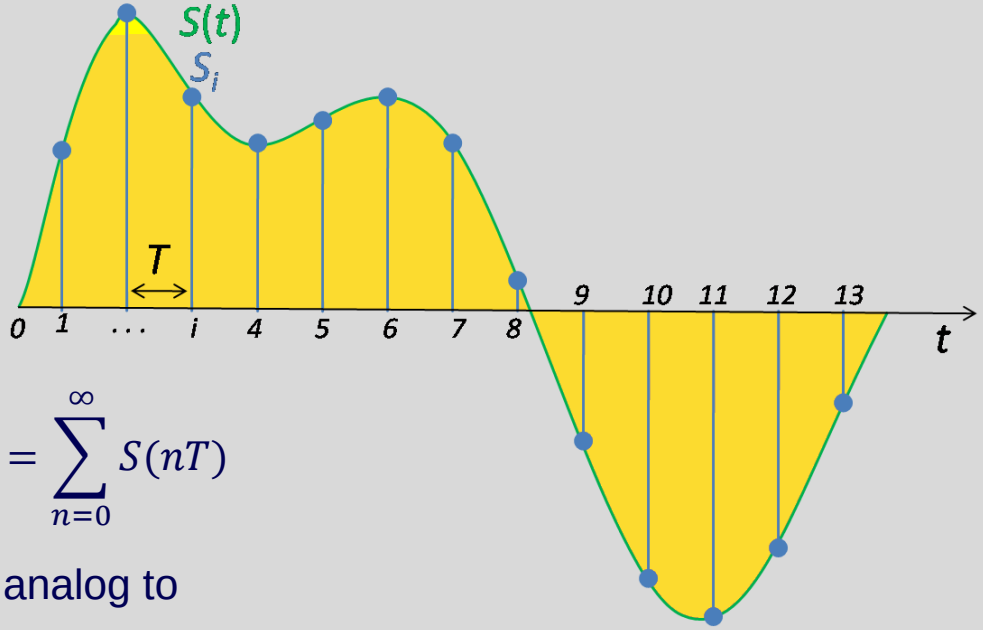
  To connect such sensors to an embedded system, analogue-to-digital (ADC) converts are required!

Analog Data sampling

Consider an arbitrary analogue signal $S(t)$ – green curve, e.g. $S(t)$ could represent the temperature variation over time

- Sampling: The process of taking discrete samples of a continuous signal, here at a uniform rate T
- Samples are denoted as s_i, where a finite sequence of samples is given as

$$S[n] = \sum_{n=0}^{\infty} S(nT)$$

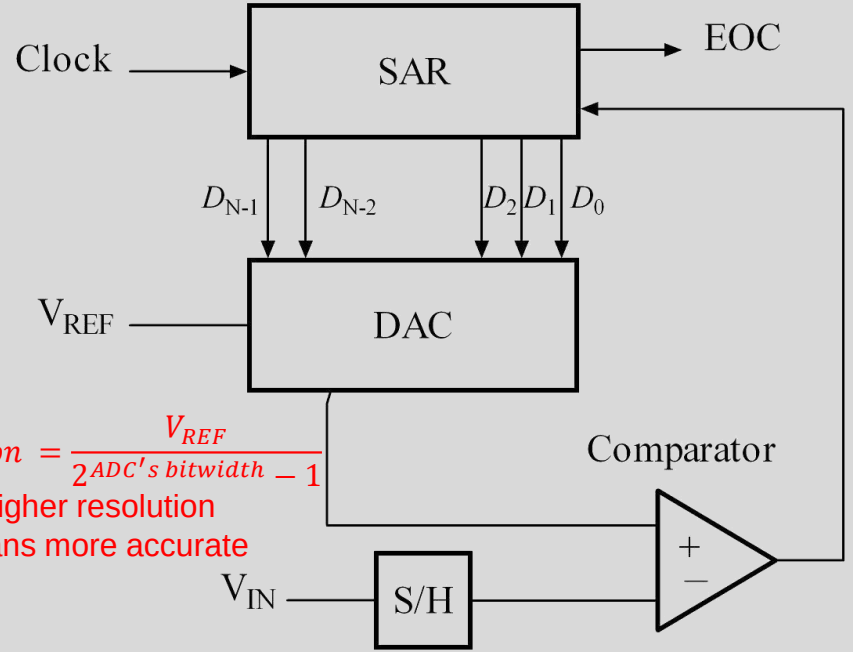


Sampled continuous signal or quantity

- Analog signal must be sampled before analog to digital conversion process takes place

ADC approach: Successive Approximation

- Type of ADC conversion where an analogue value is successively represented by an approximate binary number
- Input voltage V_{IN} – voltage to be converted to binary value
- Reference voltage V_{REF} – voltage level corresponding to maximum binary value
- End of Conversion EOC – flag which notifies the conversion process is finished
- The converted binary value only represents an approximate value of the analog signal. Accuracy is fully dependent on resolution of the ADC device



$$Resolution = \frac{V_{REF}}{2^{ADC's\ bitwidth} - 1}$$

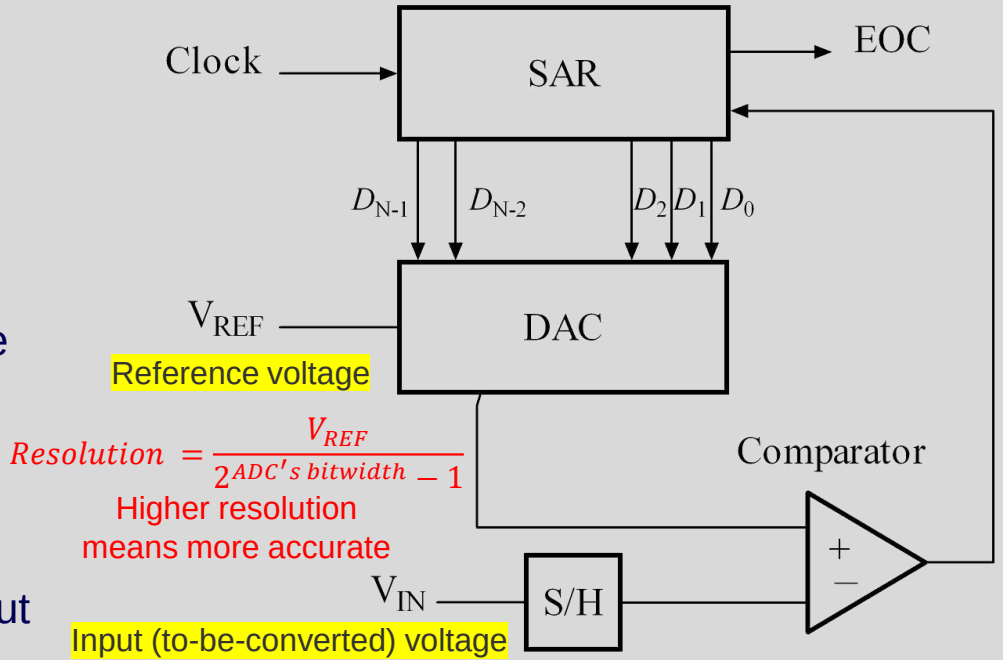
Higher resolution means more accurate

Successive approximation Block Diagram

Successive Approximation (cont.)

Successive Approximation workflow:

1. Set V_{REF} to maximum voltage that V_{IN} will reach
2. The DAC will initialize itself with the binary value 1000_2 (assuming 4-bit converter), and convert it to analogue
3. Input signal V_{IN} is sampled and held
4. Comparator outputs
   - 1 if V_{IN} is less than DAC output
   - 0 if V_{IN} is greater than DAC output
5. Respective bit is updated in the DAC
6. Step 2 is repeated for the next binary bit



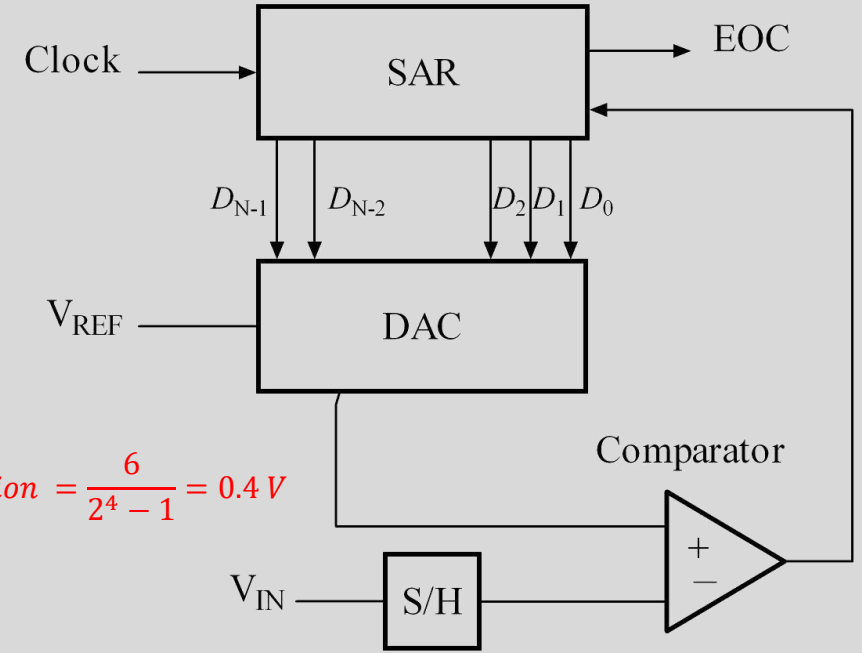
Successive approximation Block Diagram

Successive Approximation (cont.)

Example: $V_{REF} = 6V$ & V_{IN} is sampled and held at $4.6V$. Assume 4-bit ADC

1. SAR sets DAC input to 1000_2, where the DAC converts it to analogue (i.e. $8 \times 0.4V = 3.2V$)
2. Comparator outputs 1 (since $4.6 - 3.2 > 0$)
3. SAR keeps MSB to 1. SAR sets next bit to 1 in DAC input – 1100_2. Converts to $4.8V$
4. Comparator outputs 0 (since $4.6 - 4.8 < 0$)
5. SAR changes current bit to 0. SAR sets next bit to 1 in DAC input – 1010_2. Converts to $4V$.
6. Comparator outputs 1 (since $4.6 - 4.0 > 0$)
7. SAR keeps current bit at 1. SAR sets next bit to 1 in DAC input – 1011_2. Converts to $4.4V$
8. Comparator outputs 1 (since $4.6 - 4.4 > 0$)
9. EOC flag is enabled. Final result is 1011 (this is approx. equal to $1011_2 \times 0.4V = 4.4V$. Conversion error is 0.2V)

$$Resolution = \frac{6}{2^4 - 1} = 0.4\ V$$

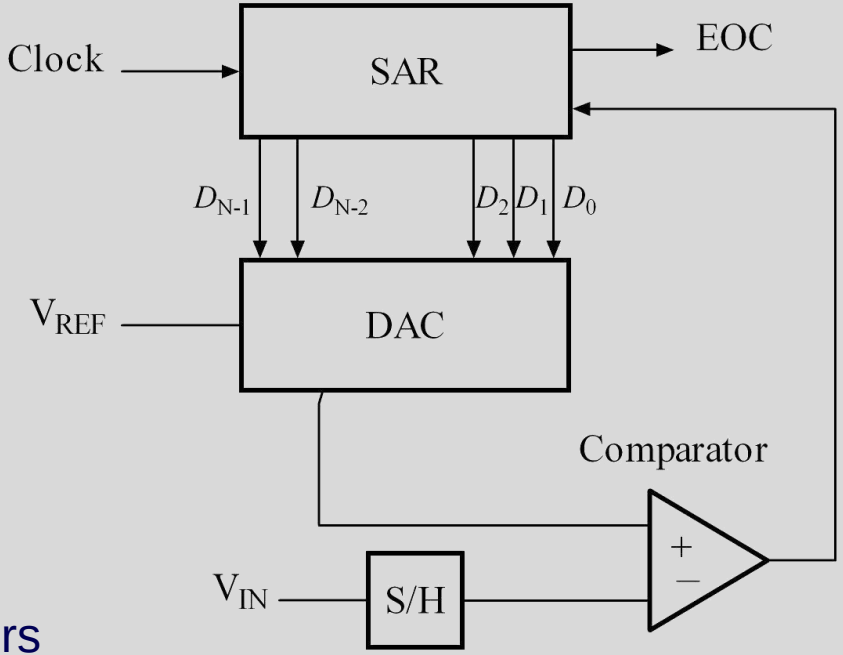


Successive approximation Block Diagram

Successive Approximation (cont.)

Example converter problems

1. Consider an 8-bit converter, where $V_{REF} = 10V$ and V_{IN} is sampled and held at $7.3V$, what is the resulting binary number from the SAR conversion?
2. Consider a 10-bit converter, where $V_{REF} = 7V$ and V_{IN} is sampled and held at $3.2V$, what is the resulting binary number from the SAR conversion?
3. Calculate the resolution of both converters in 1 & 2. What happens if $V_{IN} > V_{REF}$?



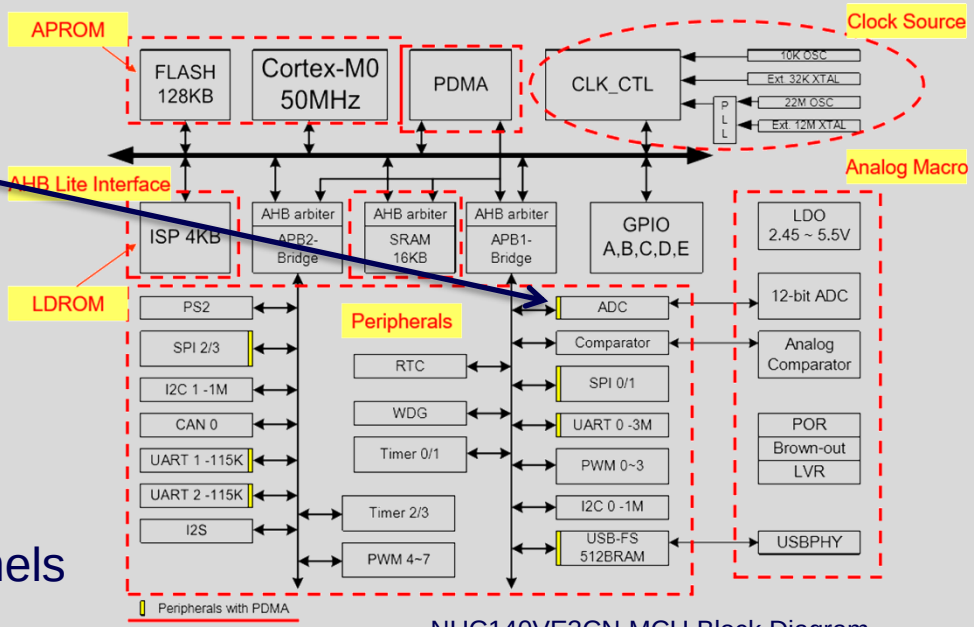
Successive approximation Block Diagram

Introduction to NUC140 MCU's ADC Controller

RMIT UNIVERSITY

NUC140VE3CN ADC Controller

- Successive Approx. Converter
- Located on the APB1
- Maximum of 12-bits resolution
- Run up to 16MHz
- Convert up to 700k samples/sec
- Three operating modes
- Up to 8 single-end analogue channels
  or 4 differential analogue channels
- Can be started by software or external pin (STADC)



NUC140VE3CN MCU Block Diagram

ADC Controller' Block Diagram & operating modes

ADC registers (Control, Status & Data)

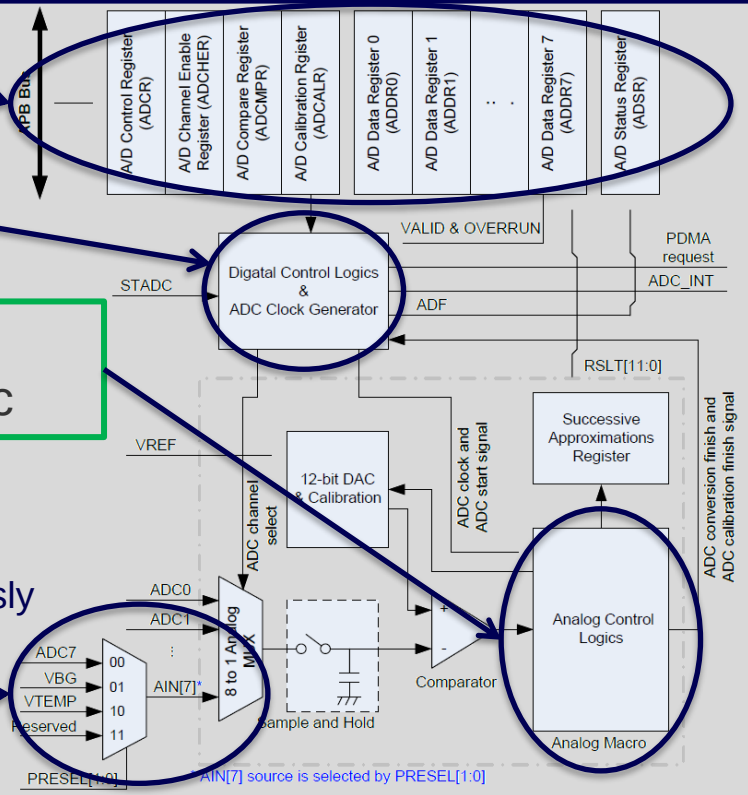
Control logic & Clock Generator

Three operating modes

- **Single Mode** – conversion is performed on one channel once
- **Single-Cycle Scan Mode** – conversion is sequentially performed on all specified channels once
- **Continuous Scan Mode** – conversion is sequentially performed on all specified channels continuously

Analogue Control Logic

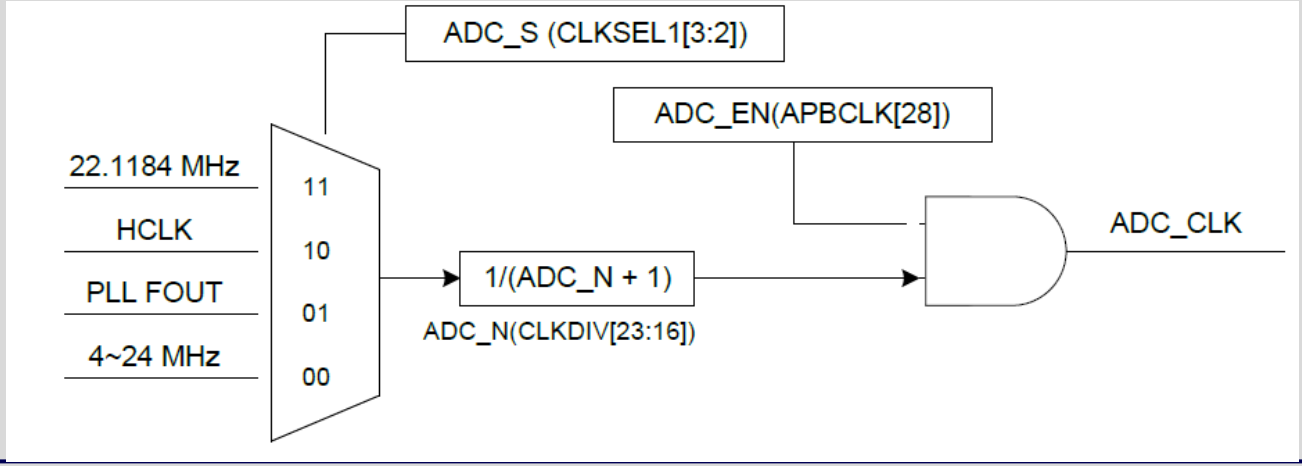
For channel 7, Only 1 ADC input can operate at a time



ADC Controller' clock configuration

- 4 clock sources to select from (by setting ADC_S bits in the CLKSEL1 register)
- Max operating clock frequency: 16 MHz
- ADC clock frequency can be further divided by setting ADC_N bits in the CLKDIV register. Note that if the clock source is HCLK then ADC_N can't be 0

$$ADC\ clock\ frequency\ (ADC\_CLK) = \frac{ADC\ clock\ source\ frequency}{ADC\_N + 1}$$



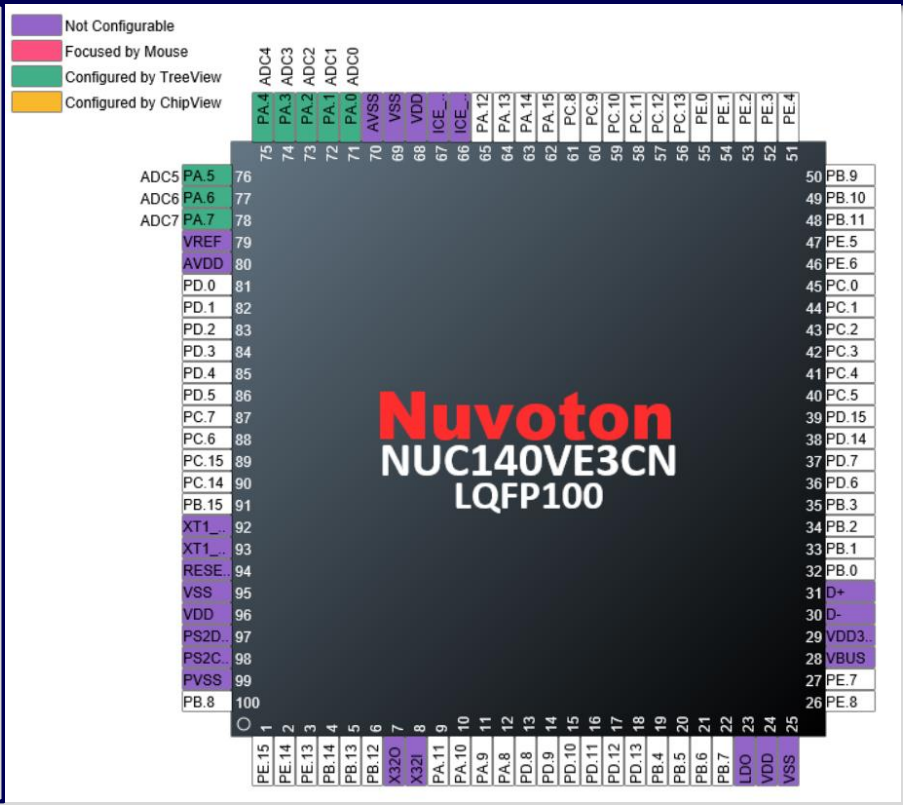
ADC Controller’ pin configuration

NUC140 MCU has one ADC Controller with 8 different channels: 0~7

Each ADC channel is allocated a pin:

ADC channel	0	1	2	3	4	5	6	7
MCU pin	GPA .0	GPA. 1	GPA. 2	GPA. 3	GPA. 4	GPA. 5	GPA. 6	GPA. 7

Pin alternative function must be configured before using these pins with ADC (since by default, these are GPIO pins)



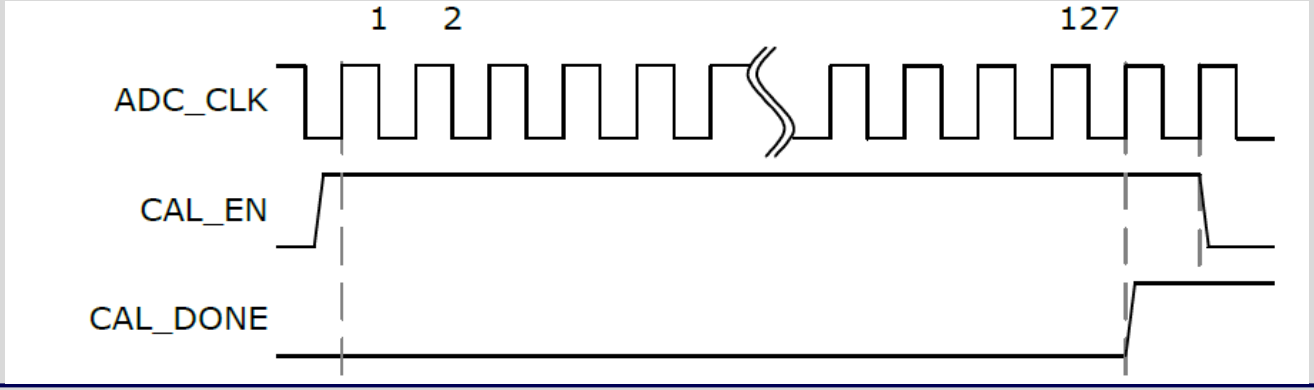
ADC Controller' Input Channel Setting

- ADC channels can be configured in either single-end input mode (i.e. conversion is done separately in each selected channel) or differential input mode (i.e. conversion is done on the difference in input values, $V_{plus} - V_{minus}$, of two selected channels)

Differential Input Paired Channel	ADC Analog Input	
	V_{plus}	V_{minus}
0	ADC0	ADC1
1	ADC2	ADC3
2	ADC4	ADC5
3	ADC6	ADC7

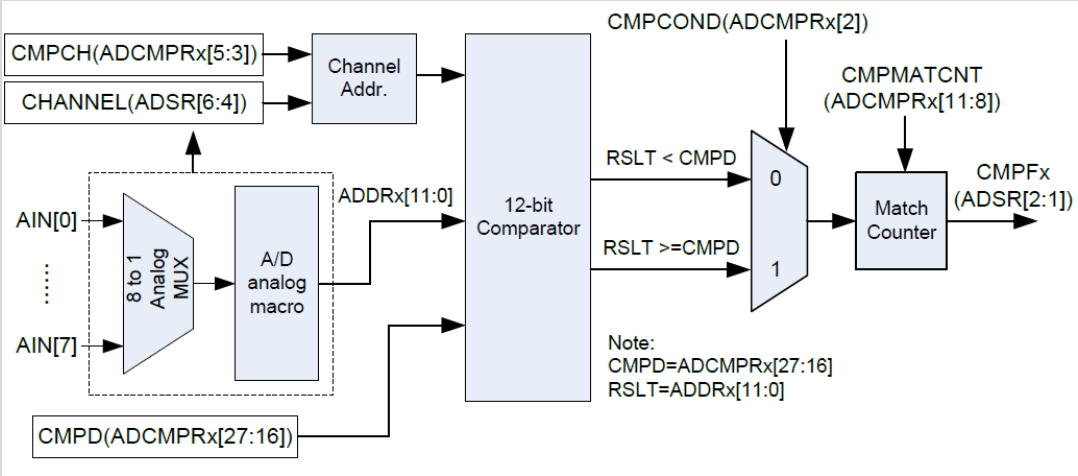
ADC Controller’ Self-calibration

- Self-calibration can be performed to minimize conversion errors occurring in some special cases (e.g. after power on or switching input modes)
- Self calibration is started when user writes 1 to CALEN bit in ADCALR register.
- When self calibration is completed, CALDONE bit in ADCALR is set to 1. It takes the ADC 127 ADC clocks to complete the self calibration process.



ADC Controller' conversion result monitor and comparison

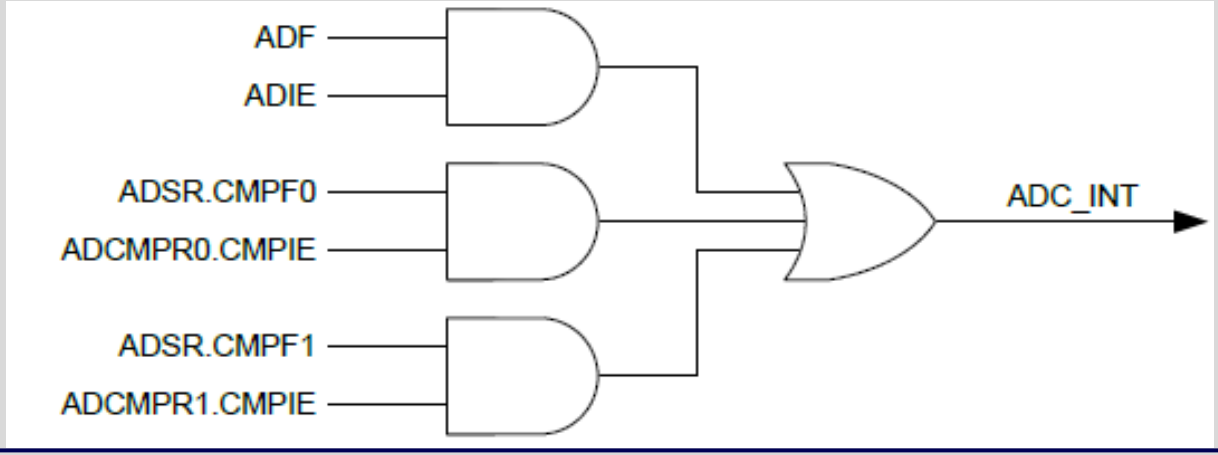
- The A/D conversion results of up to 2 ADC channels (0~7) can be constantly monitored and compared with a pre-defined value.
- When the comparison result matches a certain condition (i.e. greater than or equal to, or less than the pre-defined value), follow-up actions (e.g. interrupt) can be triggered automatically



ADC Controller’ interrupts

ADC interrupts can be generated in different scenarios:

- Conversion is completed (ADF=1) and conversion interrupt is enabled (ADIE=1)
- Conversion data comparison condition is matched (CMPF0=1) and conversion data comparison interrupt is enabled (CMPIE=1)



NUC140 MCU's ADC Controller - Registers

RMIT UNIVERSITY

ADC Registers

There are fifteen registers for the ADC Module

1. A/D Data Register x – *ADDRx*
2. A/D Control Register – *ADCR*
3. A/D Channel Enable Register – *ADCHER*
4. A/D Compare Register 0 – *ADCMPR0*
5. A/D Compare Register 1 – *ADCMPR1*
6. A/D Status Register – *ADSR*
7. A/D Calibration Register – *ADCALR*
8. A/D PDMA Current Transfer Data Register - *ADPDMA*

x denotes the ADC channel, 0 – 7

ADC is the pointer defined in the header file to access the registers. ADC has the type *ADC_T*

ADC_T is a typedef structure containing the ADC registers

The 8 *ADDRx* registers are accessed as an array, e.g. *ADDR2* is accessed by *ADC->ADDR[2]*

ADC Data Register ADDR0~7

VALID: indicate whether data stored in ADDR is valid or not
- 1: valid, 0: invalid
- This bit is set to 1 by hardware after conversion is completed.
- This bit is cleared to 0 by hardware after ADDR is read

OVERRUN: indicate whether previous data in ADDR is read or not
- 1: Data in RSLT of ADDR is not read and is overwritten
- 0: Data in RSLT of ADDR is the recent conversion result
- This bit is cleared to 0 by hardware after ADDR is read

RSLT: indicate the conversion result of ADC
- Note that the conversion result can be signed/unsigned binary
- If ADCR[31]=0: RSLT is unsigned binary integer
- If ADCR[31]=1: RSLT is 2's complement signed binary integer

Register	Offset	R/W	Description	Reset Value
ADDR0	ADC_BA+0x00	R	A/D Data Register 0	0x0000_0000
ADDR1	ADC_BA+0x04	R	A/D Data Register 1	0x0000_0000
ADDR2	ADC_BA+0x08	R	A/D Data Register 2	0x0000_0000
ADDR3	ADC_BA+0x0C	R	A/D Data Register 3	0x0000_0000
ADDR4	ADC_BA+0x10	R	A/D Data Register 4	0x0000_0000
ADDR5	ADC_BA+0x14	R	A/D Data Register 5	0x0000_0000
ADDR6	ADC_BA+0x18	R	A/D Data Register 6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	A/D Data Register 7	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						VALID	OVERRUN
15	14	13	12	11	10	9	8
RSLT [15:8]							
7	6	5	4	3	2	1	0
RSLT[7:0]							

ADC Control Register ADCR

DMOF: A/D differential input mode output format
1: A/D conversion result (stored in RSLT of ADDR register) is in 2's complement format
0: A/D conversion result (stored in RSLT of ADDR register) is in unsigned format

ADST: A/D conversion start (1: start, 0: stop)
This bit can be set to 1 by software or external pin (STADC)
This bit is cleared to 0 by hardware at the end of single-mode and single-cycle scan-mode.
In continuous scan-mode, A/D conversion is performed continuously until this bit is cleared to 0

Register	Offset	R/W	Description	Reset Value
ADCR	ADC_BA+0x20	R/W	A/D Control Register	0x0000_0000

31	30	29	28	27	26	25	24
DMOF	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				ADST	DIFFEN	PTEN	TRGEN
7	6	5	4	3	2	1	0
TRGCOND		TRGS		ADMD		ADIE	ADEN

Differential Input Paired Channel	ADC Analog Input	
	V_{plus}	V_{minus}
0	ADC0	ADC1
1	ADC2	ADC3
2	ADC4	ADC5
3	ADC6	ADC7

ADC Control Register ADCR (cont.)

DIFFEN: Differential Input Mode enable
- 1: differential analog input mode
- 0: single-end analog input mode

Differential input voltage $V_{diff} = V_{plus} - V_{minus}$
Also please note that in differential input mode, only the even-number channel needs to be enabled

PTEN: PDMA transfer enable (1: enable, 0: disable)

TRGEN: external trigger enable (1: enable, 0: disable)
This bit is used to enable/disable the triggering of A/D conversion by external pin STADC

TRGCOND: external triggering condition
00: low level, 01: high level
10: falling edge, 11: rising edge

Register	Offset	R/W	Description	Reset Value
ADCR	ADC_BA+0x20	R/W	A/D Control Register	0x0000_0000

31	30	29	28	27	26	25	24
DMOF	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				ADST	DIFFEN	PTEN	TRGEN
7	6	5	4	3	2	1	0
TRGCOND		TRGS		ADMD		ADIE	ADEN

Differential Input Paired Channel	ADC Analog Input	
	V_{plus}	V_{minus}
0	ADC0	ADC1
1	ADC2	ADC3
2	ADC4	ADC5
3	ADC6	ADC7

ADC Control Register ADCR (cont.)

ADMD: A/D converter operation mode
- 00: single conversion (with the enabled channel)
- 10: single-cycle scan (with all enabled channels)
- 11: continuous scan (with all enabled channels)
- 01: reserved

ADIE: A/D interrupt enable (1: enable, 0: disable)
A/D conversion end interrupt request is generated if the ADIE bit is set to 1

ADEN: A/D converter enable (1: enable, 0: disable)
Before starting A/D conversion, set this bit to 1
Clear this bit to 0 to completely disable ADC

Register	Offset	R/W	Description	Reset Value
ADCR	ADC_BA+0x20	R/W	A/D Control Register	0x0000_0000

31	30	29	28	27	26	25	24
DMOF	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				ADST	DIFFEN	PTEN	TRGEN
7	6	5	4	3	2	1	0
TRGCOND		TRGS		ADMD		ADIE	ADEN

Differential Input Paired Channel	ADC Analog Input	
	V_{plus}	V_{minus}
0	ADC0	ADC1
1	ADC2	ADC3
2	ADC4	ADC5
3	ADC6	ADC7

ADC Channel Enable Register ADCHER

PRESEL: Analog input channel 7 selection

Channel 7 has 3 different sources to select from: external input, internal temperature sensor, internal band-gap voltage

00: external input, 01: internal band-gap voltage, 10: internal temperature sensor, 11: reserved

Note that when internal band-gap voltage source is selected, ADC clock must be lower than 300 kHz

CHEN0~7: enable/disable analog input (1: enabled, 0: disabled)

Register	Offset	R/W	Description	Reset Value
ADCHER	ADC_BA+0x24	R/W	A/D Channel Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						PRESEL[1:0]	
7	6	5	4	3	2	1	0
CHEN7	CHEN6	CHEN5	CHEN4	CHEN3	CHEN2	CHEN1	CHEN0

ADC Compare Register ADCMPR0~1

CMPD: comparison data
- If DMOF=0: comparison is done with unsigned format
- If DMOF=1: data is done with 2's complement format

CMPMATCNT: Compare match count
- When the A/D conversion result matches the compare condition defined by CMPCOND bit, the internal match counter increases by 1.
- When the internal match counter = CMPMATCNT+1, the CMPFx (x=0~1, in the ADC status register) bit is set to 1

CMPCH: compare channel selection (000~111: ADC0~7)

CMPCOND: define the compare condition
- 1: the compare condition is TRUE if A/D result >= CMPD
- 0: the compare condition is TRUE if A/D result < CMPD

CMPIE: compare interrupt enable (1: enable, 0: disable)
- If the CMPFx (x0~7) is set to 1, interrupt request is generated

CMPEN: enable/disable data comparison (1: enable, 0: disable)

In NUC140 MUC, conversion results of up to 2 selected ADC channels can be monitored, compared, and generate interrupt requests accordingly.
This can be configured in ADCMPR0~1 registers

Register	Offset	R/W	Description	Reset Value
ADCMPR0	ADC_BA+0x28	R/W	A/D Compare Register 0	0x0000_0000
ADCMPR1	ADC_BA+0x2C	R/W	A/D Compare Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				CMPD[11:8]			
23	22	21	20	19	18	17	16
CMPD[7:0]							
15	14	13	12	11	10	9	8
Reserved				CMPMATCNT			
7	6	5	4	3	2	1	0
Reserved		CMPCH			CMPCOND	CMPIE	CMPEN

ADC Status Register ADSR

OVERRUN: equivalent to OVERRUN bit in ADDR0~7

VALID: equivalent to VALID bit in ADDR0~7

BUSY: conversion state (1: conversion is in progress, 0: idle)
This bit is equivalent to ADST bit in ADCR register

CHANNEL: current conversion channel
BUSY=1: this bit indicate the current A/D conversion channel
BUSY=0: this bit indicate the next A/D conversion channel

CMPF0~1: compare flag 0~1
This bit is set to 1 whenever the selected channel A/D conversion result meets data comparison condition in ADCMPR0~1 registers
Writing 1 to this bit will clear it to 0

ADF: indicate the end of A/D conversion
This bit is set to 1 when:
- A/D conversion ends in single mode
- A/D conversion ends on all selected channels in scan mode
Writing 1 to this bit will clear it to 0

Operational status of ADC channels can be "observed" by checking value of corresponding bits in this register.

Register	Offset	R/W	Description	Reset Value
ADSR	ADC_BA+0x30	R/W	A/D Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
OVERRUN							
15	14	13	12	11	10	9	8
VALID							
7	6	5	4	3	2	1	0
Reserved	CHANNEL			BUSY	CMPF1	CMPF0	ADF

ADC Calibration Register ADCALR

ADC conversion error occur in some specific circumstances (e.g. after power-on or ADC input switching)
→ ADC self-calibration should be performed to minimize conversion errors.

CALEN: enable/disable self-calibration (1: enable, 0: disable)
CALDONE: calibration completion flag (1: self-calibration is completed, 0: not started or still on-going)

Register	Offset	R/W	Description	Reset Value
ADCALR	ADC_BA+0x34	R/W	A/D Calibration Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						CALDONE	CALEN

ADC PDMA Current Transfer Data Register ADPDMA

When PDMA is transferring, by reading this register (AD_PDMA bits) we can monitor the current PDMA transfer data

Register	Offset	R/W	Description	Reset Value
ADPDMA	ADC_BA+0x40	R	ADC PDMA current transfer data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				AD_PDMA[11:8]			
7	6	5	4	3	2	1	0
AD_PDMA[7:0]							

NUC140 MCU's ADC

- In the current programming environment, ADC is defined as ADC, a pointer to the C data structure **ADC_T** at **ADC_BASE** address
- We access and configure registers of each ADC channel (8 of them) by using pointer operator "->"
  - e.g. ADC->ADDR[0] to access ADC Channel 0's Data Register

```
// file: NUC100Series.h
typedef struct
{
    __I  uint32_t ADDR[8];          /* Offset: 0x00-0x1C  ADC Data Register x          */
    __IO uint32_t ADCR;             /* Offset: 0x20  ADC Control Register              */
    __IO uint32_t ADCHER;           /* Offset: 0x24  ADC Channel Enable Register       */
    __IO uint32_t ADCMPR[2];        /* Offset: 0x28  ADC Compare Register x            */
    __IO uint32_t ADSR;             /* Offset: 0x30  ADC Status Register               */
    __I  uint32_t RESERVE0[3];
    __I  uint32_t ADPDMA;           /* Offset: 0x40  ADC PDMA Current Transfer Data Reg.  */
} ADC_T;
```

```
#define APB1_BASE       ((    uint32_t)0x40000000)
#define ADC_BASE        (APB1_BASE     + 0xE0000)
#define ADC             ((ADC_T *) ADC_BASE)
```