

ACROPOLIS
The New
XC800-Family

Designed to Make the Difference

XC866 Product Presentation

November 2005

Part 1 – System & Core

Part 2 – System & Peripherals

Part 3 – Embedded Flash

Special – Analog to Digital Converter (ADC)

Infineon technologies

Never stop thinking.

Infineon

Table of contents

- Introduction – Why such an ADC?
- Special Features of XC866‘s ADC
- Questions and Answers
- Practical Usage – Application Examples
- Practical Usage – Software Examples
- Questions and Answers

Target Application for XC866 – Electronic Motor Control

- Drives need realtime performance
  - control loop must run faster than 2-4 PWM periods (e.g. 100-200us)
  - CPU performance is the most valuable good !!!
  - Question: How to offload the CPU?
  - Answer: Build intelligent and autonomous peripherals!
- ADC in a Drive application:
  - phase current and voltage
    - are fast changing signals, sample rate is high
    - accurate timing to sample these signals is mandatory
  - other signals have to be sampled (e.g. temperature)
    - sample rate is totally different
- ADC is used intensively – controlling the ADC steals CPU load !!!

stop thinking
never

Target Application for XC866 – Electronic Motor Control

- General problem with a standard ADCs
  - ADC is controlled through CPU only
    - start of conversion / read out result instructions are distributed over the whole program code
  - ADC hardware and CPU timing (software) are coupled
    - but sensed signals change asynchronously to CPU timing
    - CPU and ADC have to be synchronized by software
    - this is not accurate and takes CPU load (wait on event)
  - Reading the result needs CPU load
    - conversion time equals ~ interrupt response time (few microseconds)
    - it is not efficient to work with ADC-interrupts
    - polling is mostly used (wait until conversion is done) >> CPU load!
- New ADC approach is needed

never stop thinking

XC866-ADC – A New Approach

- ADC – Block Diagram

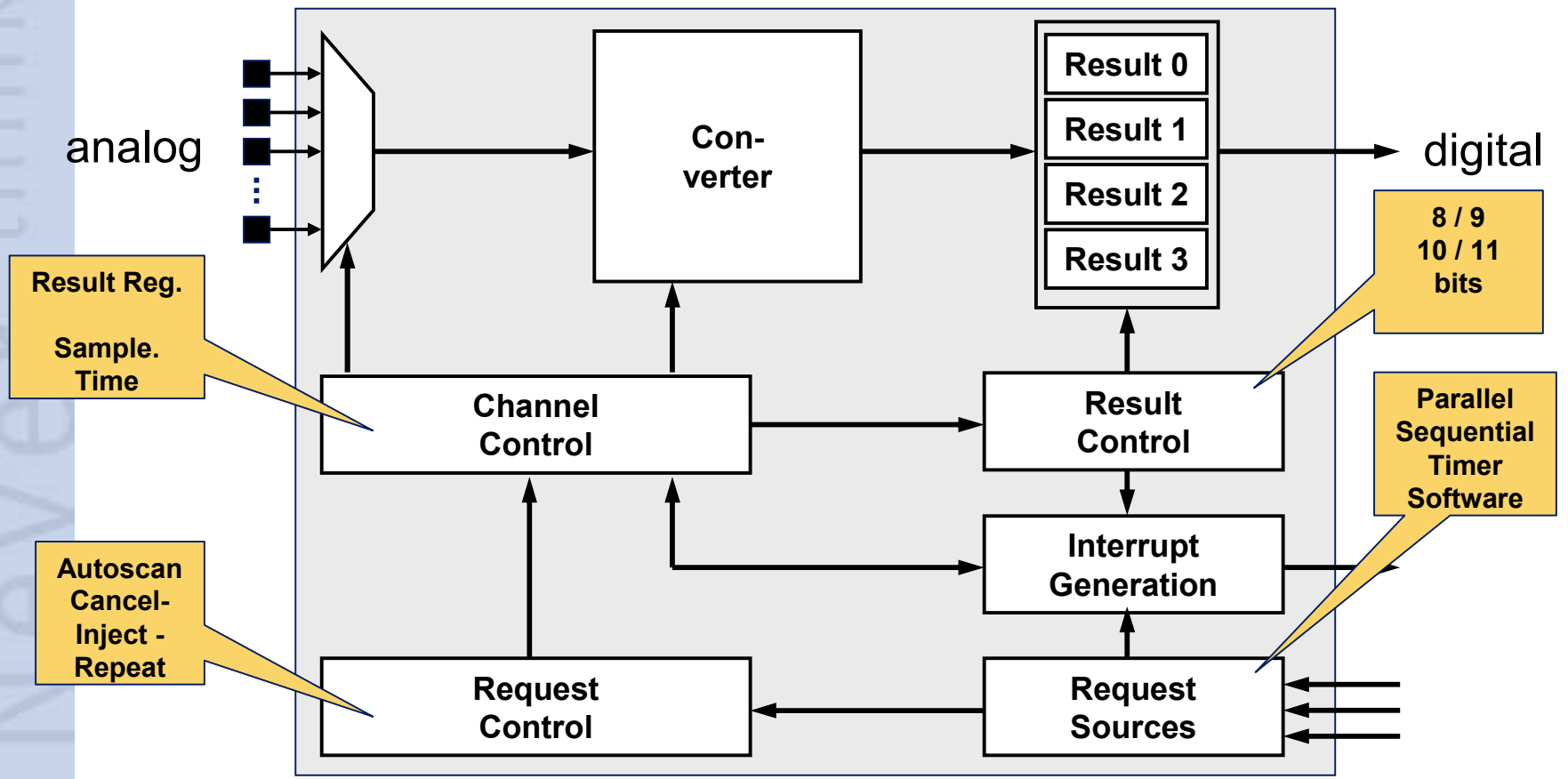


Table of contents

- Introduction – Why such an ADC?
- Special Features of XC866‘s ADC
- Practical Usage – Application Examples
- Practical Usage – Software Examples

XC866-ADC – The Register Map

- Standard ADCs have 3-4 SFRs
- This ADC has 50 SFRs in 7 SFR pages
- Switching between pages for configuration
- Running from a working page after configuration
  - just reading result registers

Address	Page 0	Page 1	Page 2	Page 3	Address	Page 4	Page 5	Page 6	Page 7
CA_H	GLOBCTR	CHCTR0	RESR0L	RESRA0L	CA_H	RCR0	CHINFR	CRCR1	–
CB_H	GLOBSTR	CHCTR1	RESR0H	RESRA0H	CB_H	RCR1	CHINCR	CRPR1	–
CC_H	PRAR	CHCTR2	RESR1L	RESRA1L	CC_H	RCR2	CHINSR	CRMR1	–
CD_H	LCBR	CHCTR3	RESR1H	RESRA1H	CD_H	RCR3	CHINPR	QMR0	–
CE_H	INPCR0	CHCTR4	RESR2L	RESRA2L	CE_H	VFCR	EVINFR	QSR0	–
CF_H	ETRCR	CHCTR5	RESR2H	RESRA2H	CF_H	–	EVINCR	Q0R0	–
$D2_H$	–	CHCTR6	RESR3L	RESRA3L	$D2_H$	–	EVINSR	QBUR0/QINR0	–
$D3_H$	–	CHCTR7	RESR3H	RESRA3H	$D3_H$	–	EVINPR	–	–

XC866-ADC – Channel & Result Control

Sample Time Control

- ts = (2 + STC)*CTC / fadc
  - fadc = 26.67MHz
  - STC = [0...255]
  - CTC = [2; 3; 4; 32]
- ts = 150ns....308us
- STC is valid for all channels

Conversion Time Control

- tconv = (1 + CTC*(3 + n + STC)) / fadc
  - fadc = 26.67MHz
  - STC = [0...255]
  - CTC = [2; 3; 4; 32]
  - n = 8; 10 (result bits)
- tconv = 1.5us...322us @10bits
- CTC is valid for all channels

Channel & Result

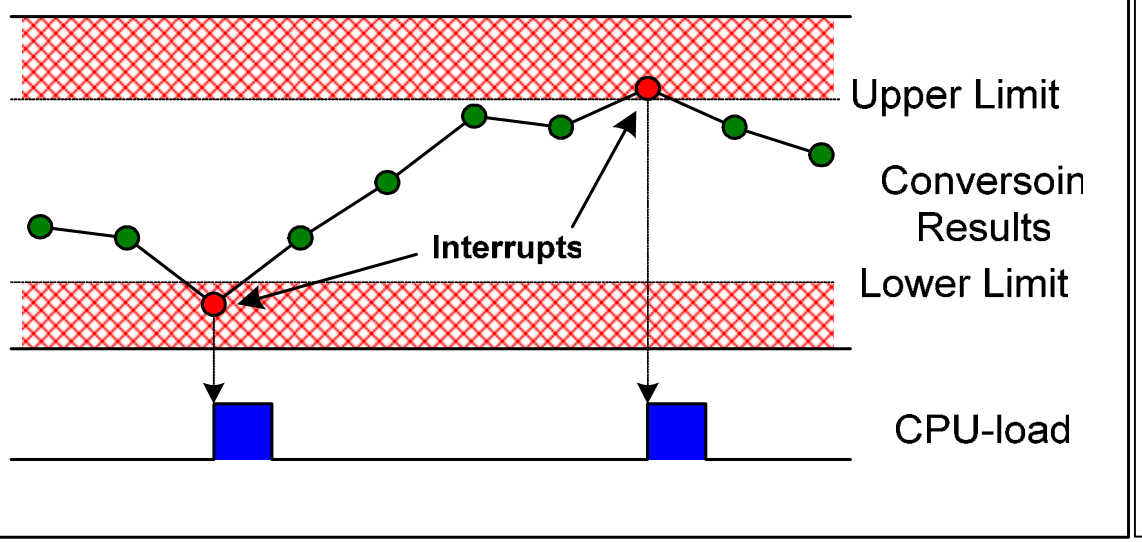
- 4 result registers for 8 channels
- Result register view (8, 9, 10, 11bits)
- Accumulation of 2 results (data reduction filter)
- Boundary Limit Checking – Comparator Mode

Result Control

never stop thinking

XC866-ADC – Limit and Boundary Checking

Comparator Mode



- Limit is individual per channel
- 4bits for upper limit
- 4bits for lower limit
- 8 combinations possible

Reduced CPU load

Software interaction for interesting results only

Never stop thinking

XC866-ADC – Trigger Request Control

Parallel Source

- for autoscan conversions
- predefined scan sequence (e.g. 7-6-4)
- background conversion
- write back result automatically
- wait-for-read mode

Sequential Source

- for timetriggered conversions
- channel is freely programmable
- 8 external events from CC6 can trigger a conversion request
  - injected mode
  - single or continues event trigger (refill)

Two Request Sources

- arbitration is needed for correct priotization between request sources
- complex & flexible control for automatic trigger requests

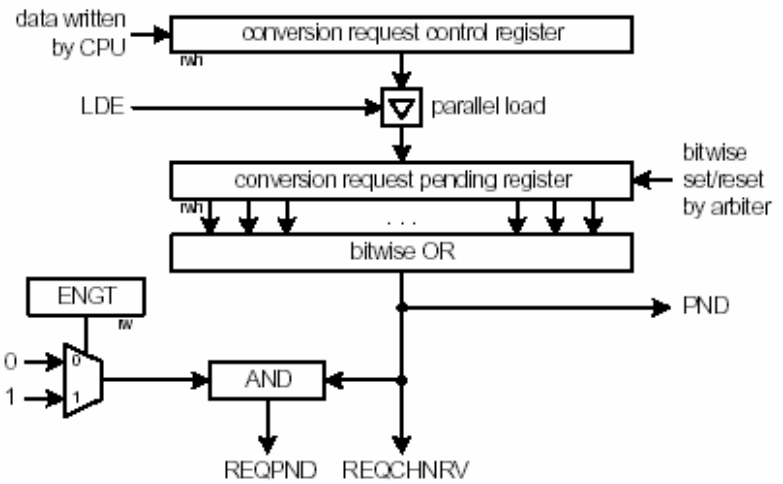
Decoupling ADC from CPU
Offload CPU

stop thinking never

XC866-ADC – Starting the ADC

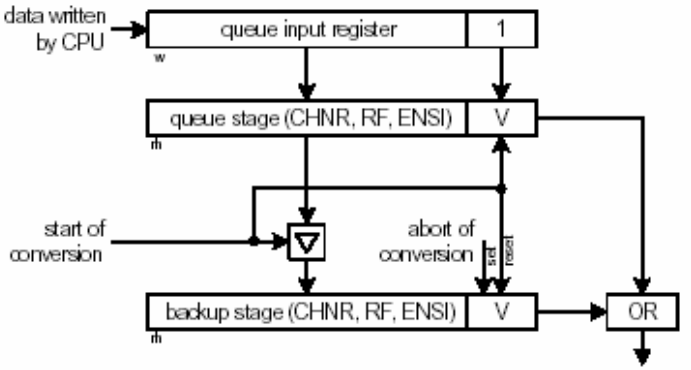
Parallel Source

- Write to register CRMR1 with
  - external trigger on/off (ENTR= 1/0)
  - autoscan on/off (SCAN = 1/0)
  - generate a load event (LDEV =1)

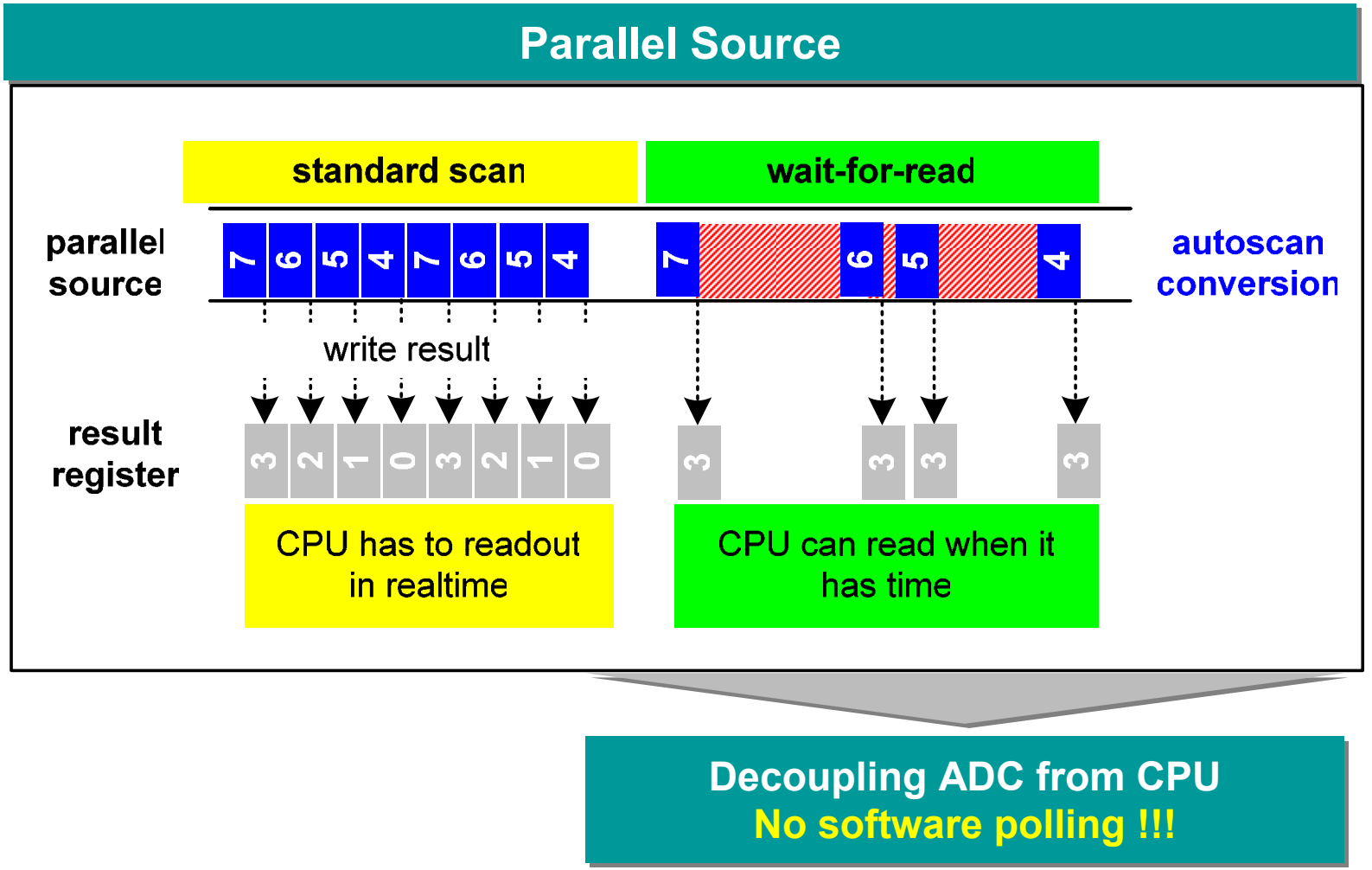


Sequential Source

- Write to queue input register QINR0 with
  - channel number (REQCHNR)
  - external trigger (EXTR = 1)
  - software trigger (EXTR = 0)
  - with automatic refill (RF = 1)
  - single trigger (RF = 0)
- XC866 has one queue stage

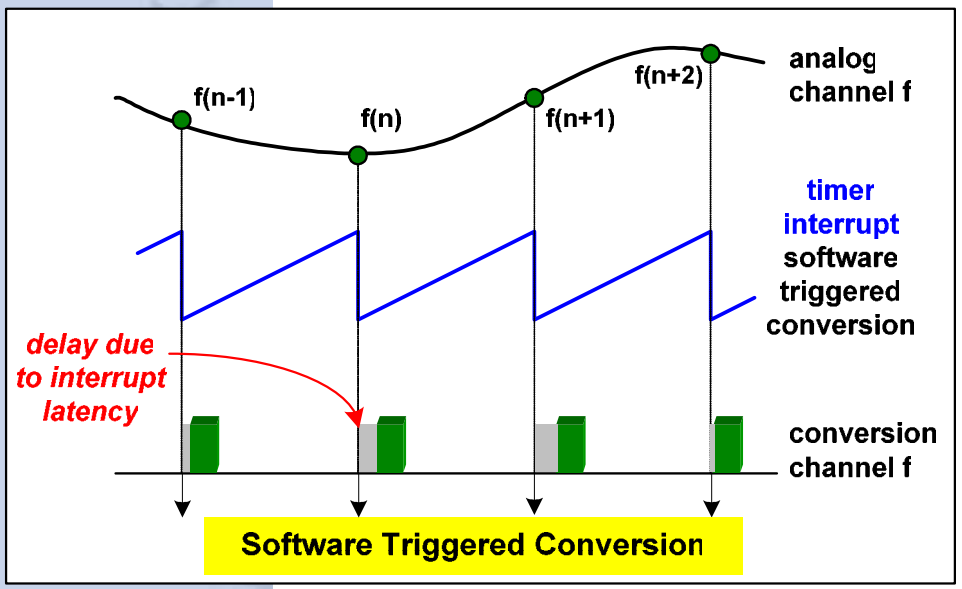


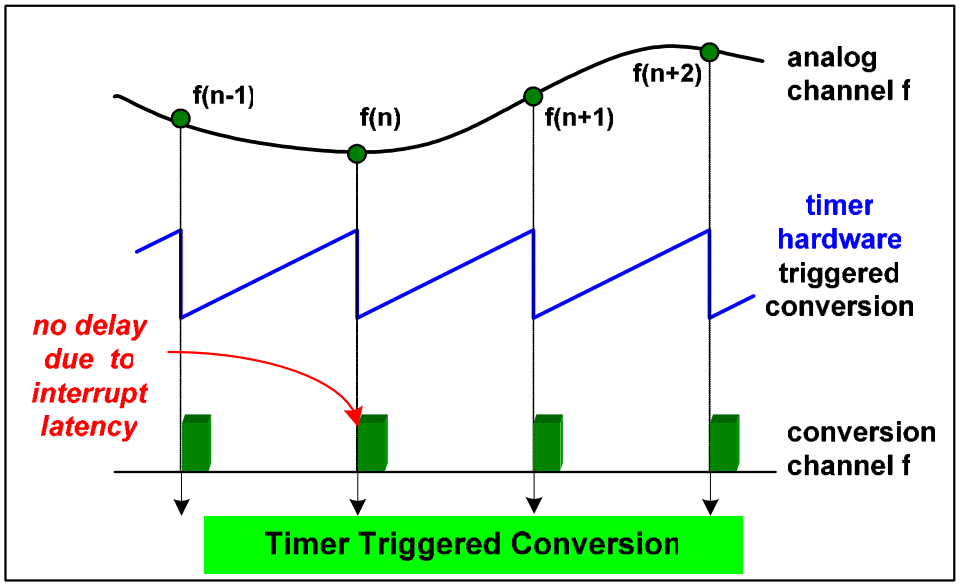
XC866-ADC – Parallel Source (Autoscan)



XC866-ADC – Sequential Source (Timetriggered)

Sequential Source

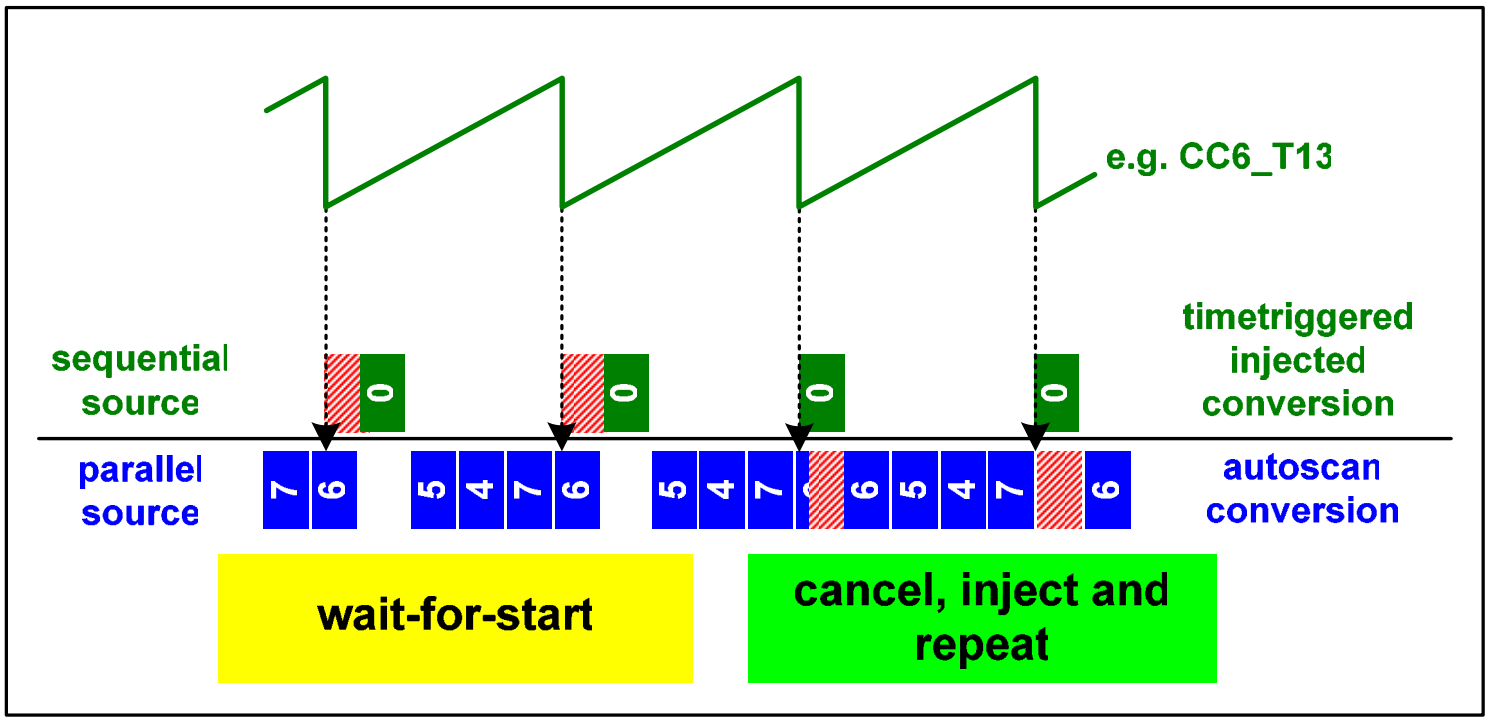




Accurate Time-Äquidistant Measurements
No software wait on trigger event !!!

XC866-ADC – Arbitration between Trigger Sources

Sequential and Parallel Source



Full hardware support on priotization
Software is fully decoupled from hardware

never stop thinking

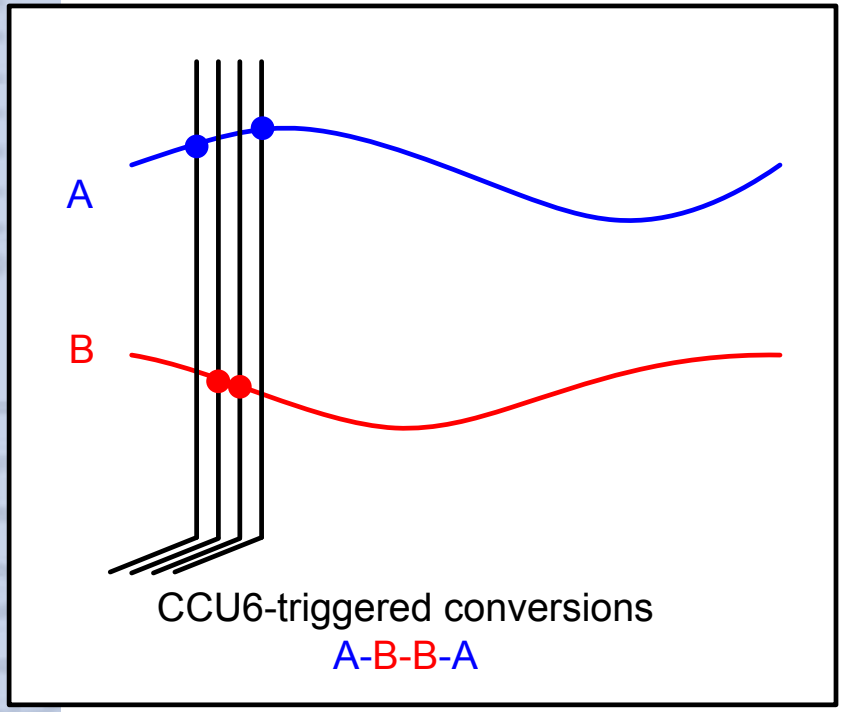

Any Questions?

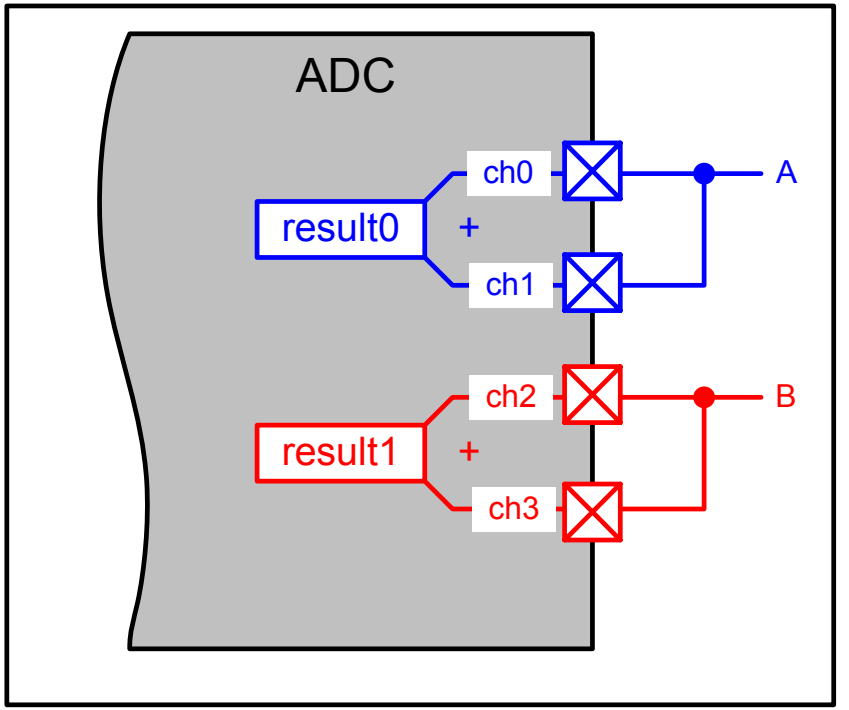
Table of contents

- Introduction – Why such an ADC?
- Special Features of XC866‘s ADC
- **Practical Usage – Application Examples**
- Practical Usage – Software Examples

XC866-ADC – Application Examples (1)

Quasi Synchronous Sampling



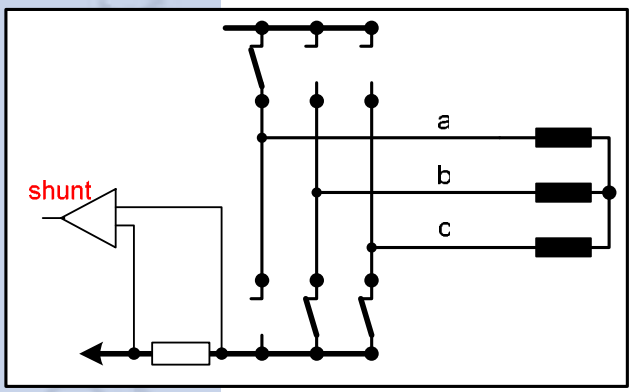


Sequential timetriggered conversion

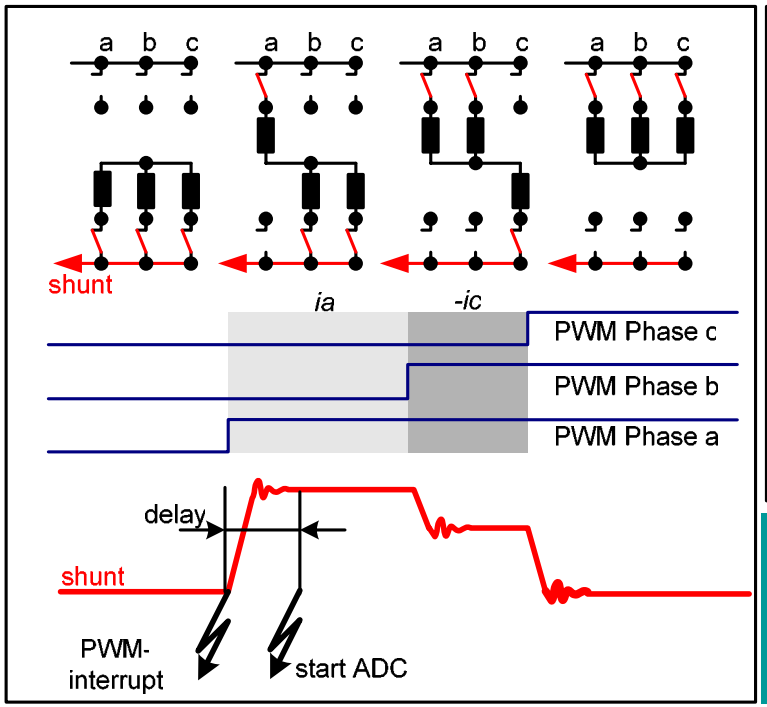
Accumulate results
Data reduction filter

XC866-ADC – Application Examples (2)

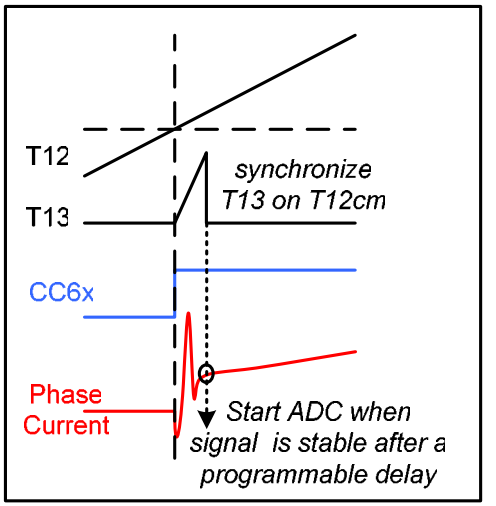
Single Shunt Current Measurement for SVM controlled Motors



Topology



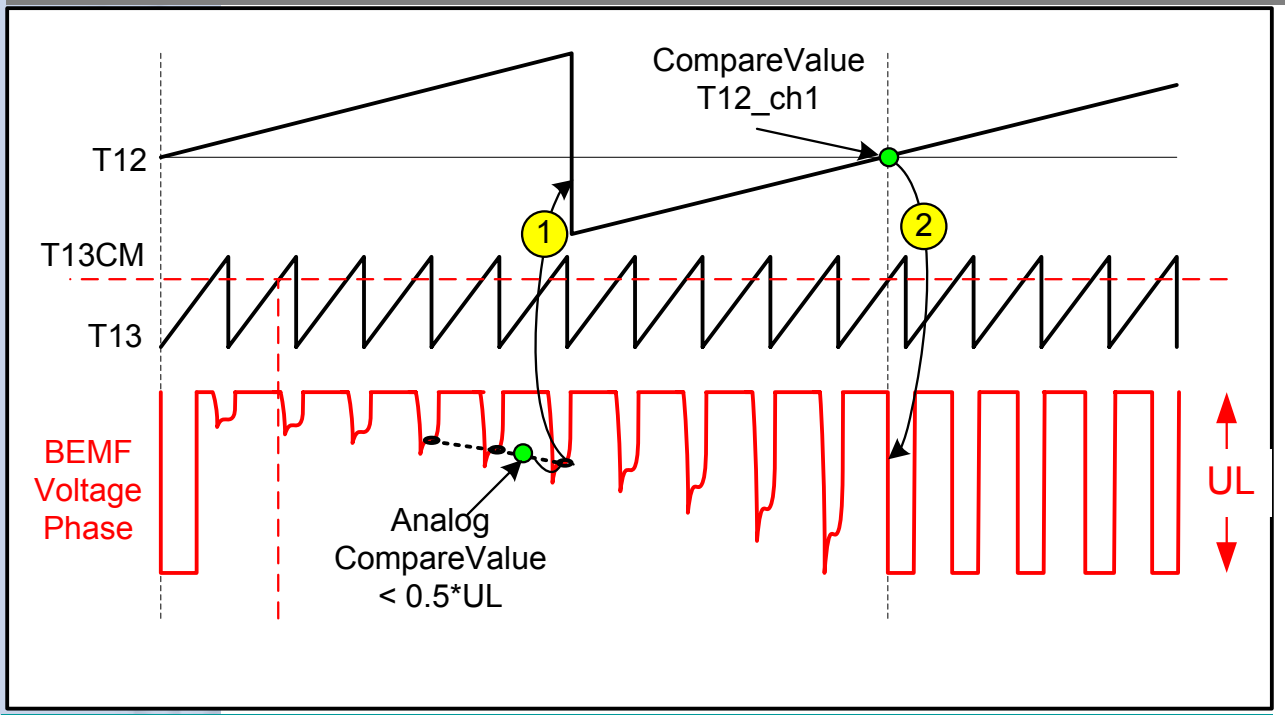
Space Vector Modulation (SVM)

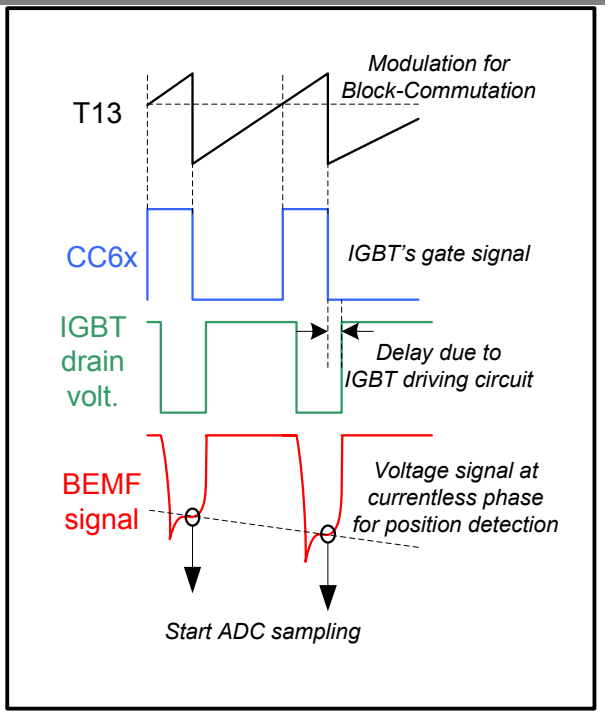


Noisefree sampling due to T13PM (delay after ON)

XC866-ADC – Application Examples (3)

Sensorless BLDC control through BEMF Zero-Crossing Detection





Classical Block Commutation with BEMF Signal
ADC samples BEMF and compares against UL/2

Noisefree sampling due to T13PM
(sample before OFF)

Table of contents

- Introduction – Why such an ADC?
- Special Features of XC866‘s ADC
- Practical Usage – Application Examples
- **Practical Usage – Software Examples**

never stop thinking

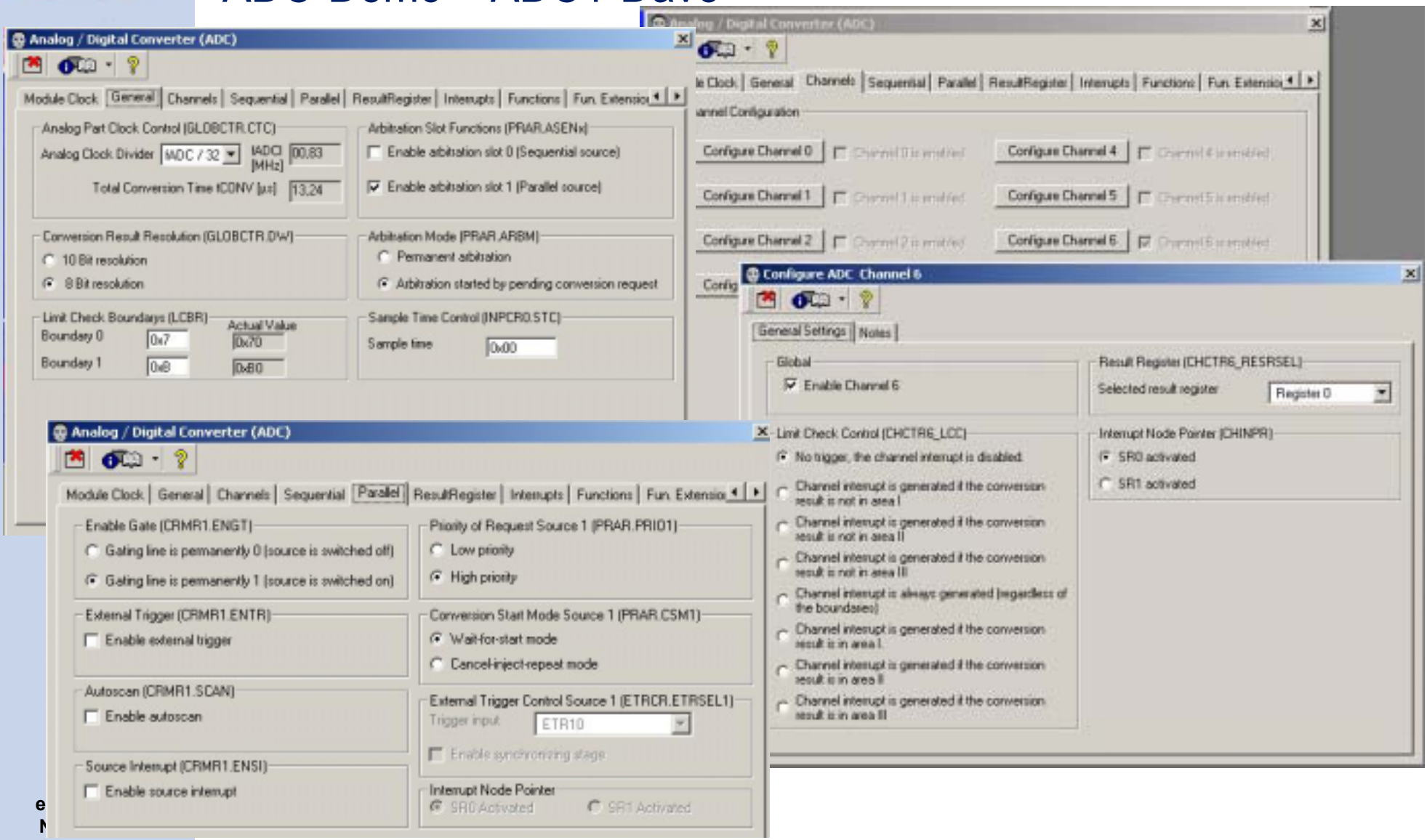
How to use the ADC – ADC-Demo

- ADC1 - Simple ADC using POT at AN6 and display on P3
  - parallel source
  - LoadEvent by SW
  - no interrupt, polling method
- Copy or type following code
  - main.c

```
// USER CODE END
 while(1)
 {
 // USER CODE BEGIN (MAIN_Main,4)
          ADC_vStartParReqChNum(0x40);
          ADC_vSetLoadEvent();
          while(ADC_ubBusy());
          IO_vWritePort(P3,ADC_uwGetResultData(0x00));
 // USER CODE END
```

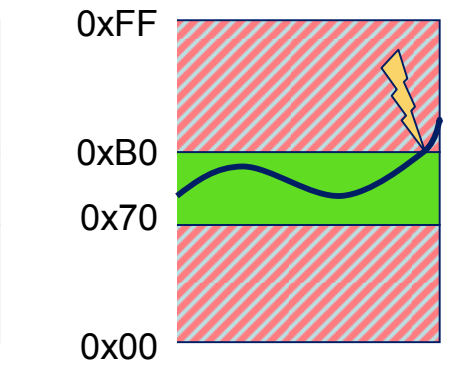

Never stop thinking

ADC-Demo – ADC1-Dave



How to use the ADC – ADC-Demo

- ADC2 - Comparator Mode and Interrupt
  - parallel source,
  - convert AN6, LoadEvent by SW
  - channel interrupt if result is out of boundary
  - lower boundary 0x70, upper boundary 0xB0
  - toggle P3.6 on if 0x70 > AN6 or AN6 > 0xB0



never stop thinking

How to use the ADC – ADC-Demo

- ADC2 - Comparator Mode and Interrupt

- Copy or type following code
  - main.c

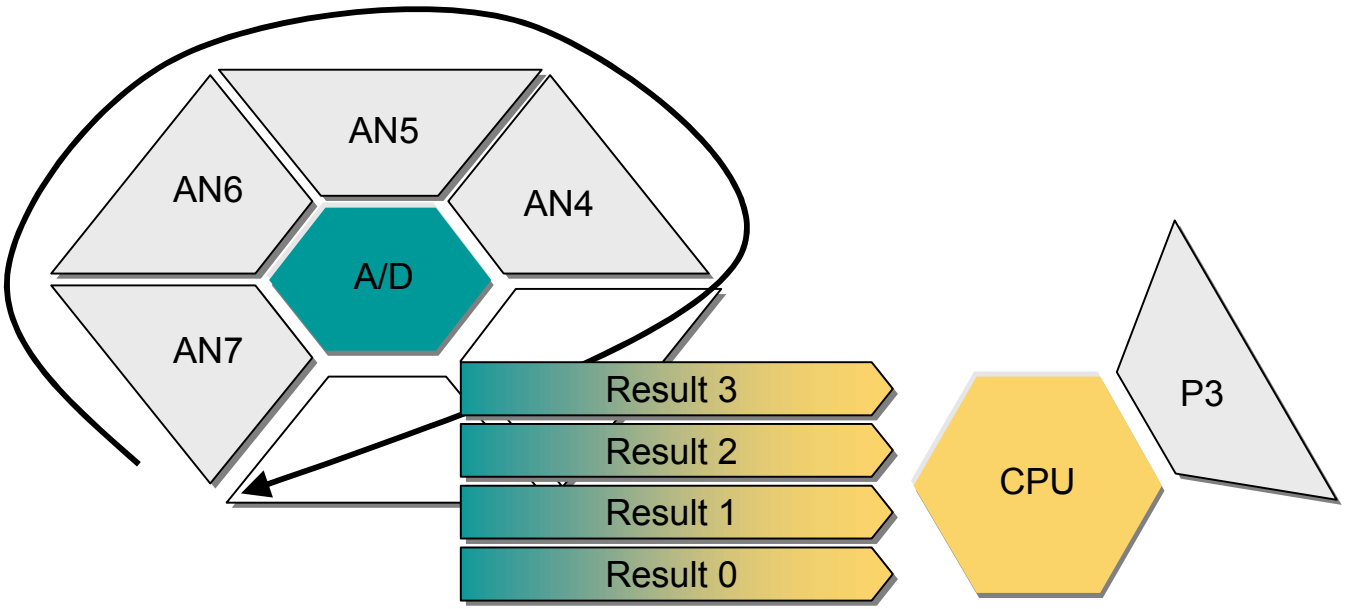
```
while(1)
{

// USER CODE BEGIN (MAIN_Main,4)
    ADC_vStartParReqChNum(0x40);
    ADC_vSetLoadEvent();
    while(ADC_ubBusy());
// USER CODE END

}
```

- Copy or type following code
  - adc.c

```
void ADC_vIsr(void) interrupt ADCINT
{

  // USER CODE BEGIN (ADC_Isr,2)
  unsigned char z;
  // USER CODE END

  SFR_PAGE(_su0, SST2);          // switch to page 0

  //   Check Interrupt Request 0 Flag
  if (((IRCON1 & 0x08) != 0))
  {
    IRCON1 &= ~(ubyte)0x08; //   clear ADCSRC0

    SFR_PAGE(_su0, RST2);        // restore the old SCU page

    // USER CODE BEGIN (ADC_Isr,3)
          z=ADC_uwGetResultData(0x00);
          IO_vTogglePin(P3_6);
          for (z=50;z>0;z--);
          IO_vTogglePin(P3_6);

    // USER CODE END
  }
```

How to use the ADC – ADC-Demo

- ADC3 - Autoscan Mode
  - Convert AN7, AN6, AN5, AN4 in a sequence automatically
  - Save to Result 3, 2, 1, 0
  - Display channel on P3 upon selection



How to use the ADC – ADC-Demo

- ADC3 - Autoscan Mode
- Copy or type following code
  - main.c

```
void main(void)
{
  // USER CODE BEGIN (MAIN_Main,2)
  // USER CODE END
  MAIN_vInit();
  // USER CODE BEGIN (MAIN_Main,3)
  ADC_vStartParReqChNum(0xF0);
  // USER CODE END

  while(1)
  {
  // USER CODE BEGIN (MAIN_Main,4)
  unsigned char z;

  // select one of the channel to be display on the L
  z=ADC_uwGetResultData(0);    //AN4
  //z=ADC_uwGetResultData(1);  //AN5
  //z=ADC_uwGetResultData(2);  //AN6
  //z=ADC_uwGetResultData(3);  //AN7
  P3_DATA=z;
  // USER CODE END

  }

} //  End of function main
```

or alternatively

```
// USER CODE BEGIN (MAIN_Main,4)
 unsigned char z;

 // Part II - Display each channel on 2 LEDs
 z=ADC_uwGetResultData(0);
 z=z>>6;
 P3_DATA=z;

 z=ADC_uwGetResultData(1);
 z=z>>6;
 z=z<<2;
 P3_DATA|=z;

 z=ADC_uwGetResultData(2);
 z=z>>6;
 z=z<<4;
 P3_DATA|=z;

 z=ADC_uwGetResultData(3);
 z=z>>6;
 z=z<<6;
 P3_DATA|=z;
// USER CODE END
```

How to use the ADC – ADC-Demo

- ADC4 – Timetriggerd injected conversion
  - trigger conversion on AN6 by T13PM
  - write to result 0, display on P3
  - optional T13PM interrupt
- Copy or type following code
  - main.c

```
// USER CODE BEGIN (MAIN_Main,3)
    //ADC_vStartSeqReqChNum(ubyte ExtTrg, ubyte EnIntr, ubyte RFill, ubyte ChannelNum)
    ADC_vStartSeqReqChNum(1, 0, 0, 6);
    //  SFR_PAGE(_ad6, SST1);          // switch to page 6
    //  ADC_QINR0 = ( 0x86 ) ;         // requested channel
    //  SFR_PAGE(_ad0, RST1);          // restore the old ADC page
// USER CODE END

while(1)
{

// USER CODE BEGIN (MAIN_Main,4)
    P3_DATA = ADC_uwGetResultData(0);
// USER CODE END

}

} //  End of function main
```


Never stop thinking

How to use the ADC – ADC-Demo

- ADC5 – Combination of ADC 2 to 4
  - Autoscan Mode
  - Comparator Mode and Interrupt
  - Time Triggered Injected Conversion

Port	Mode	Output Register	LEDs
P2.7	Autoscan	1	P3.6 - P3.7
P2.6	Time Triggered Injected Conversion	0	P3.0 - P3.3
P2.5	Autoscan	2	P3.4 - P3.5
P2.4	Autoscan + Comparator Mode and Interrupt	3	Toggle P3.6

Infineon technologies
never stop thinking

How to use the ADC – ADC-Demo

- ADC5 – Autoscan, Comparator & Injected Conversions
- Copy or type following code
  - main.c

```
// USER CODE BEGIN (MAIN_Main,3)
        ADC_vStartSeqReqChNum(1, 0, 0, 6);
        ADC_vStartParReqChNum(0xB0);
 // USER CODE END

 while(1)
 {
 // USER CODE BEGIN (MAIN_Main,4)
        unsigned char z;
        z = ADC_uwGetResultData(0);
        P3_DATA=z>>4;
        z = ADC_uwGetResultData(1);
        z=z>>6;
        z=z<<6;
        P3_DATA=z;
        z = ADC_uwGetResultData(2);
        z=z>>6;
        z=z<<4;
        P3_DATA=z;
 // USER CODE END
 }
} //  End of function main
```

- Copy or type following code
  - adc.c

```
void ADC_vIsr(void) interrupt ADCINT
{

 // USER CODE BEGIN (ADC_Isr,2)
 unsigned char z;
 // USER CODE END

 SFR_PAGE(_su0, SST2);        // switch to page 0

 //   Check Interrupt Request 0 Flag
 if (((IRCON1 & 0x08) != 0))
 {
  IRCON1 &= ~(ubyte)0x08; //   clear ADCSRC0

  SFR_PAGE(_su0, RST2);       // restore ôld page

  // USER CODE BEGIN (ADC_Isr,3)
        z=ADC_uwGetResultData(0x03);
        IO_vTogglePin(P3_6);
        for (z=50;z>0;z--);
        IO_vTogglePin(P3_6);
 // USER CODE END
```


stop thinking
Never


Infineon
technologies
Any Questions?
Never stop thinking
www.infineon.com