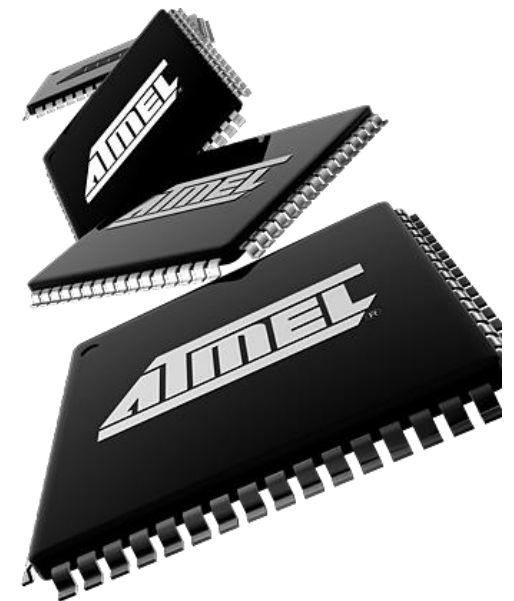# Principles and Applications of Microcontrollers

Yan-Fu Kuo

Dept. of Bio-industrial Mechatronics Engineering

National Taiwan University
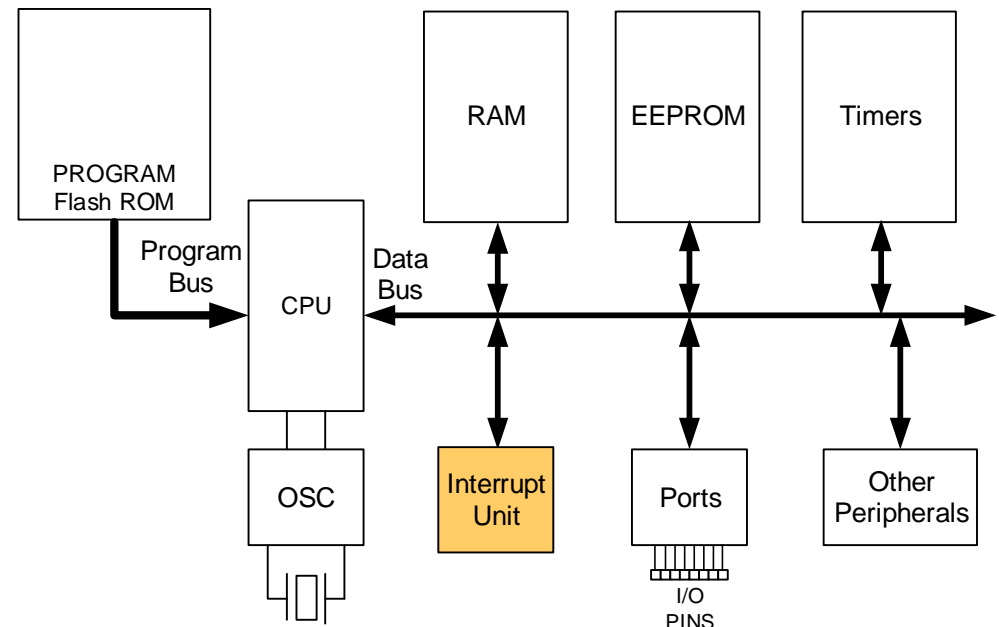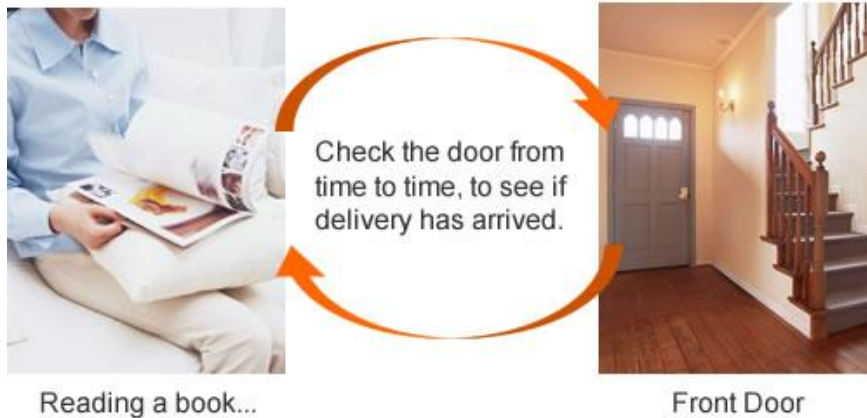
Today:

- Interrupt

Outline

- Polling vs. interrupt
- Interrupt procedure
- Enable interrupt
- Interrupt programming
- Some other issues
- Getting started

Polling vs. Interrupt

- **Polling mode**
  - MCU continuously monitors the status of a device

- **Interrupt mode**
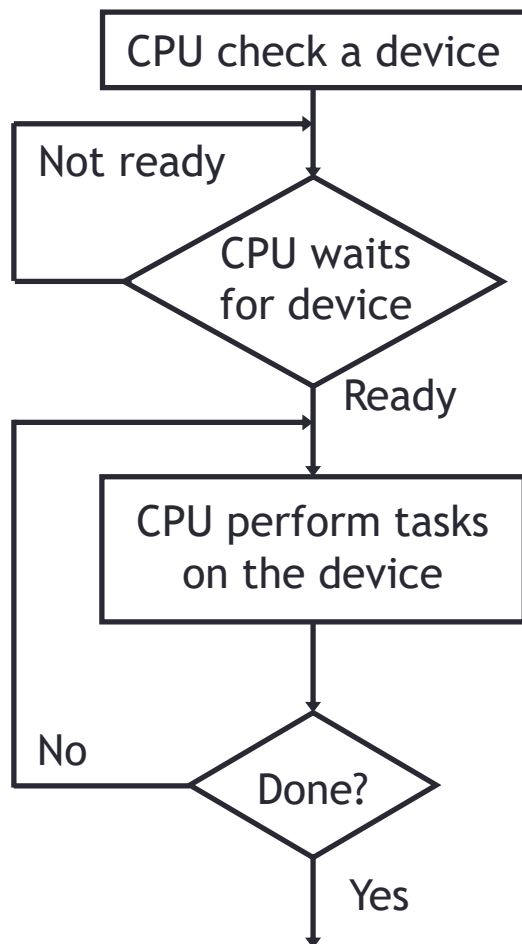 - Device notifies the MCU by sending an interrupt signal when it needs a service

Check the door from time to time, to see if delivery has arrived.

Reading a book...   Front Door

Delivery person rings the doorbell.

Reading a book...   Delivery
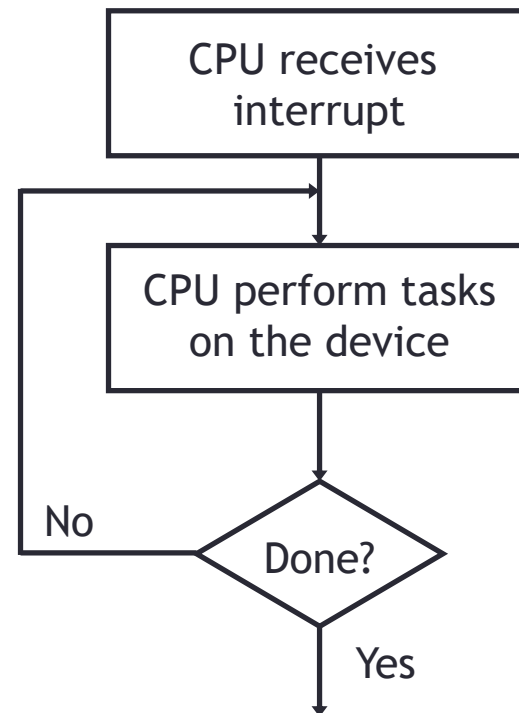
Polling vs. Interrupt

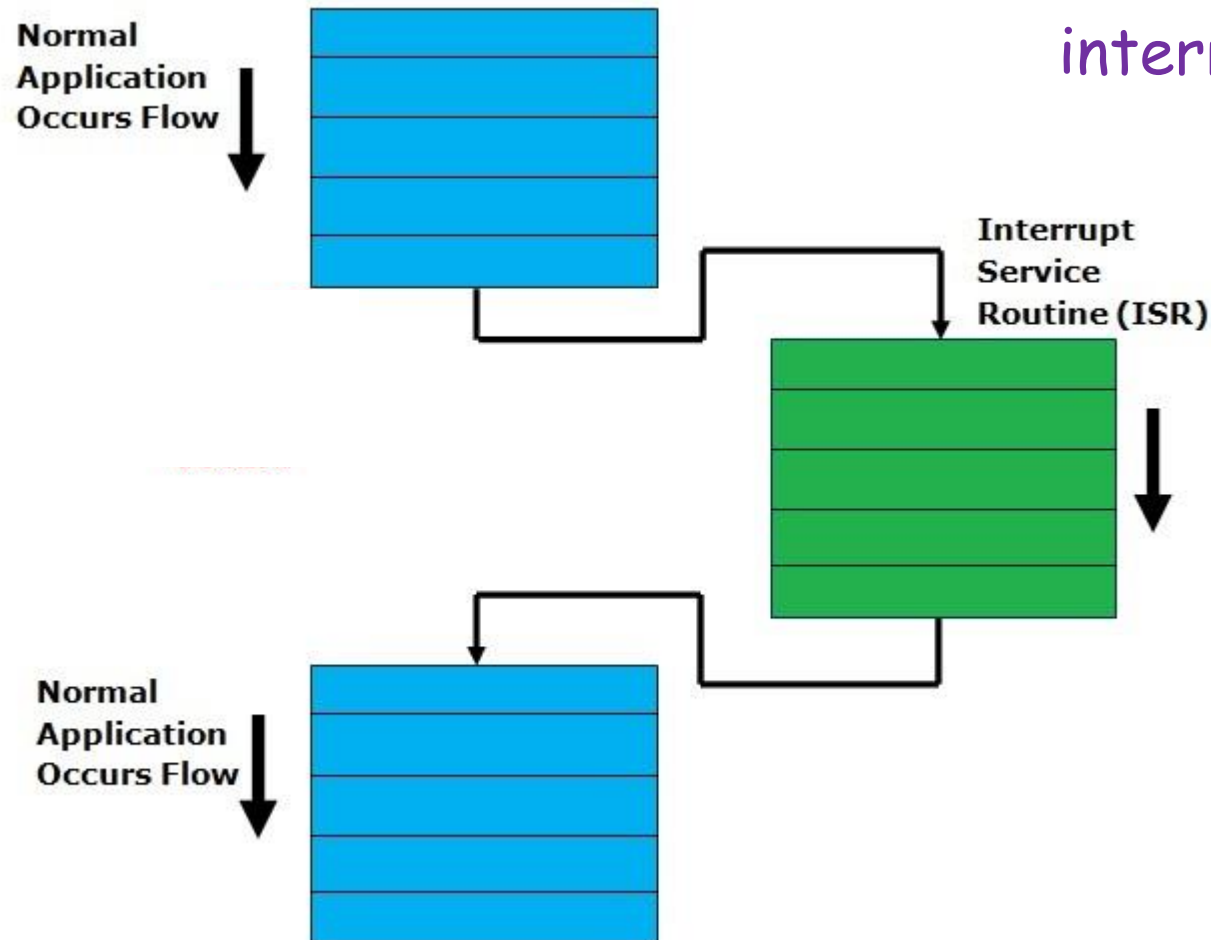- **Polling mode**
  - Ties down the CPU

- **Interrupt mode**
  - Efficient CPU use
  - Able to assign priority
  - Can be masked

Interrupt Program Flow

1. What do ISRs look like?
2. What events can trigger an interrupt?

Interrupt Program Illustration (C Language)

```c
#include <avr/io.h>
#include <avr/interrupt.h>

int main(void)
{
    sei();                 // enable global interrupts
    EIMSK = 0x01;          // enable INT0
    ...
}

 ...

ISR(INT0_vect)            // interrupt service routine
{
    ...
}
```
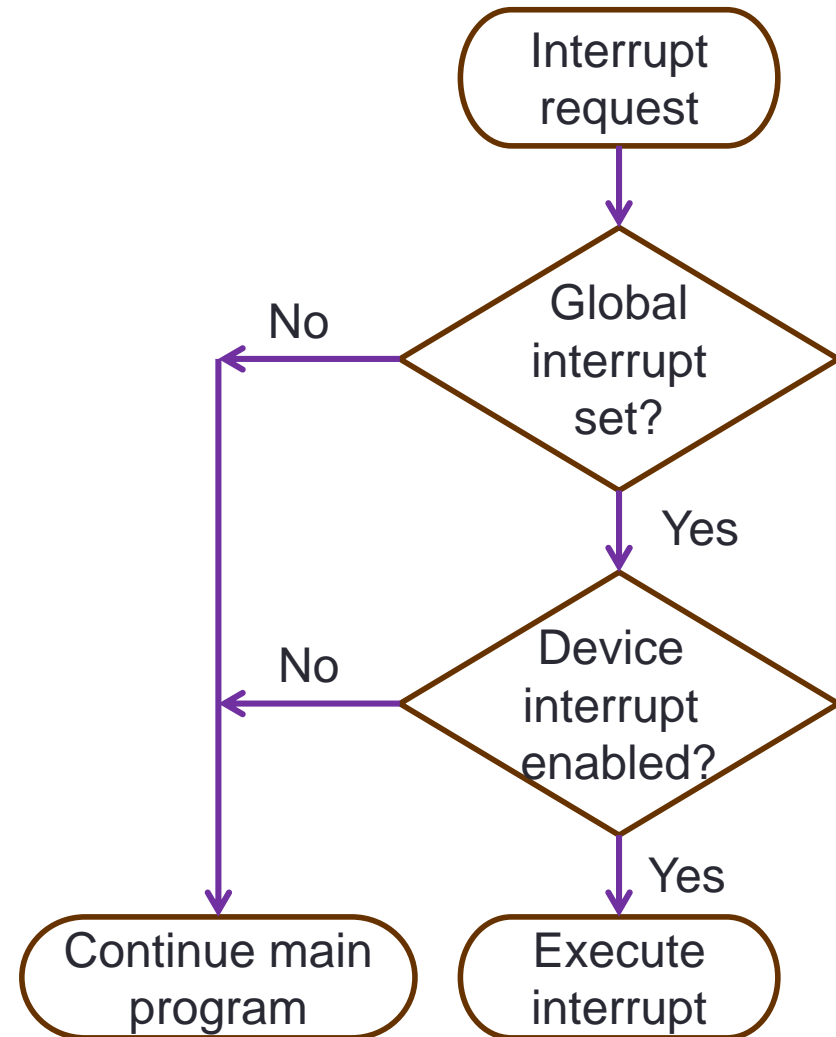
Common Interrupt Service Routine (ISR)

Resource/event	Interrupt service routine
External Interrupt Request 0	`INT0_vect`
External Interrupt Request 1	`INT1_vect`
Timer/Counter2 Compare Match A	`TIMER2_COMPA_vect`
Timer/Counter2 Compare Match B	`TIMER2_COMPB_vect`
Timer/Counter2 Overflow	`TIMER2_OVF_vect`
Timer/Counter1 Compare Match A	`TIMER1_COMPA_vect`
Timer/Counter1 Compare Match B	`TIMER1_COMPB_vect`
Timer/Counter1 Overflow	`TIMER0_OVF_vect`
Timer/Counter0 Compare Match A	`TIMER0_COMPA_vect`
Timer/Counter0 Compare Match B	`TIMER0_COMPB_vect`
Timer/Counter0 Overflow	`TIMER0_OVF_vect`
USART, Rx Complete	`USART_RX_vect`
USART, Tx Complete	`USART_TX_vect`
ADC Conversion Complete	`ADC_vect`

Enable Interrupt

1. Enable global interrupt:

```
sei();
```

2. Enable interrupt of a device (from its register)

```mermaid
flowchart TD
    A([Interrupt request]) --> B{Global interrupt set?}
    B -->|No| C([Continue main program])
    B -->|Yes| D{Device interrupt enabled?}
    D -->|No| C
    D -->|Yes| E([Execute interrupt])
```

Device Interrupt Enabling

- External interrupt register:

EIMSK	-	-	-	-	-	-	INT1	INT0

- Timer interrupt registers:

TIMSK0	-	-	-	-	-	OCIE0B	OCIE0A	TOIE0

TIMSK1	-	-	ICIE1	-	-	OCIE1B	OCIE1A	TOIE1

TIMSK2	-	-	-	-	-	OCIE2B	OCIE2A	TOIE2

Output compare   Timer overflow

- ADC control register:

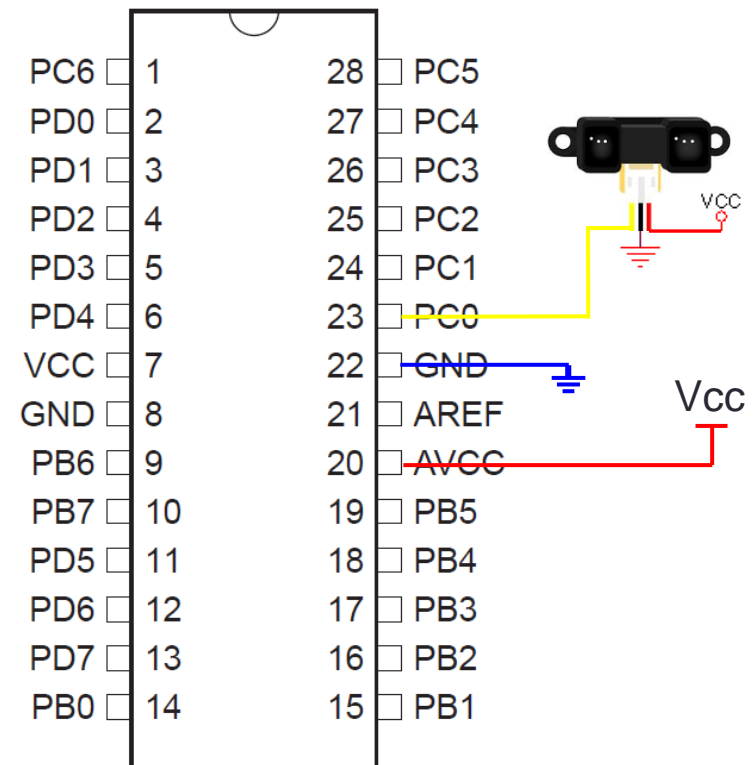| ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Outline (Cont'd)

- Polling vs. interrupt
- Interrupt procedure
- Enable interrupt
- Interrupt programming
- Some other issues
- Getting started

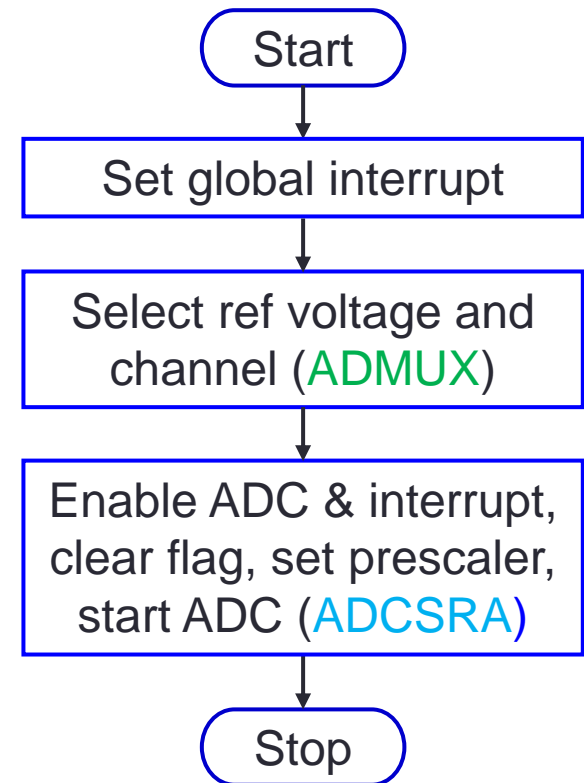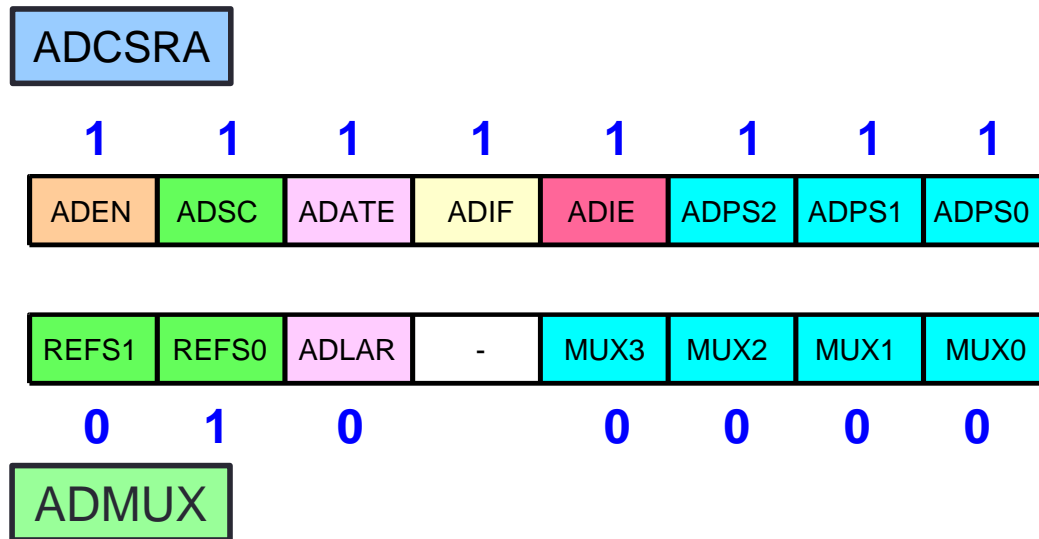"PLEASE FEEL FREE TO INTERRUPT IF YOU HAVE A QUESTION."

©1995 Tom Swanson
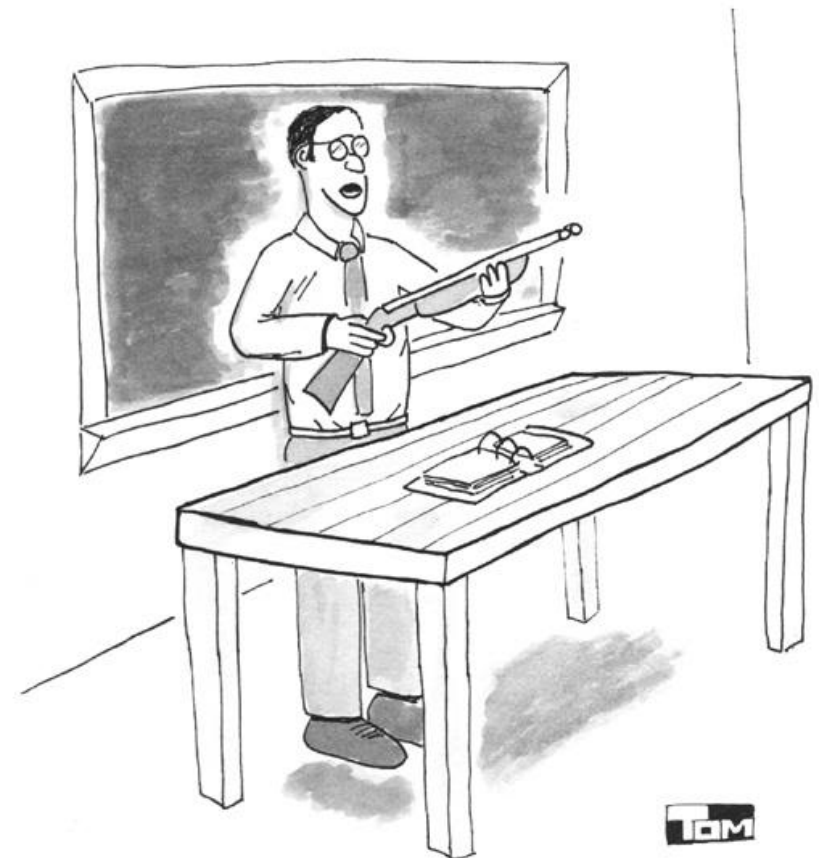
Example: ADC Interrupt

- Read from ADC0 (PC0) using interrupt
- Store the ADC reading to a variable "`value`"
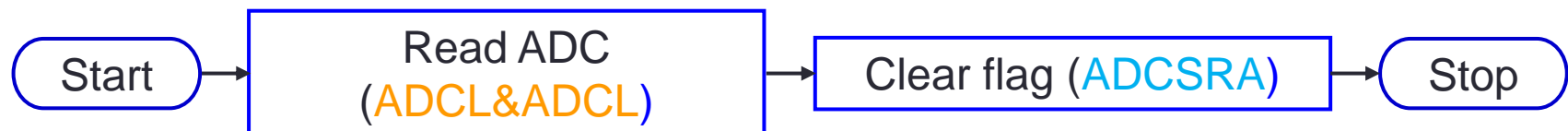- Free running mode
- ADC prescaler $p = 128$
- Vref=Avcc

Flowchart (ADC Interrupt)

- What value do we set the controller registers?

ADCSRA

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0
0	**1**	**0**		**0**	**0**	**0**	**0**

ADMUX

Start

Set global interrupt

Select ref voltage and channel (ADMUX)

Enable ADC & interrupt, clear flag, set prescaler, start ADC (ADCSRA)

Stop

- ISR:

Start → Read ADC (ADCL&ADCL) → Clear flag (ADCSRA) → Stop

ADC Interrupt

```c
#include <avr/io.h>
#include <avr/interrupt.h>

volatile unsigned int value;

int main(void)
{
    DDRC=0;                        // PORTC as input
    sei();                         // enable global interrupts
    ADMUX=0b01000000;              // ref volt + channel
    ADCSRA=0xFF;                   // free running & interrupt
    while(1);                      // idle
}

ISR(ADC_vect)
{
    unsigned char low, high;
    ADCSRA|=(1<<ADIF);             // clear ADIF
    low=ADCL;                      // read low byte first
    high=ADCH;
    value=(high<<8)+low;           // calculate integer value
}
```
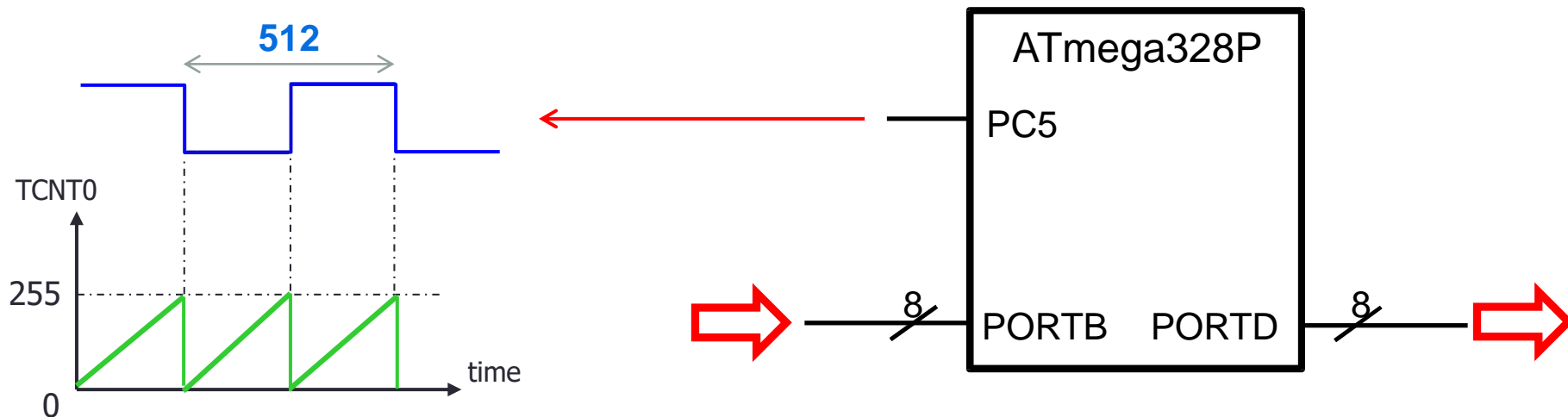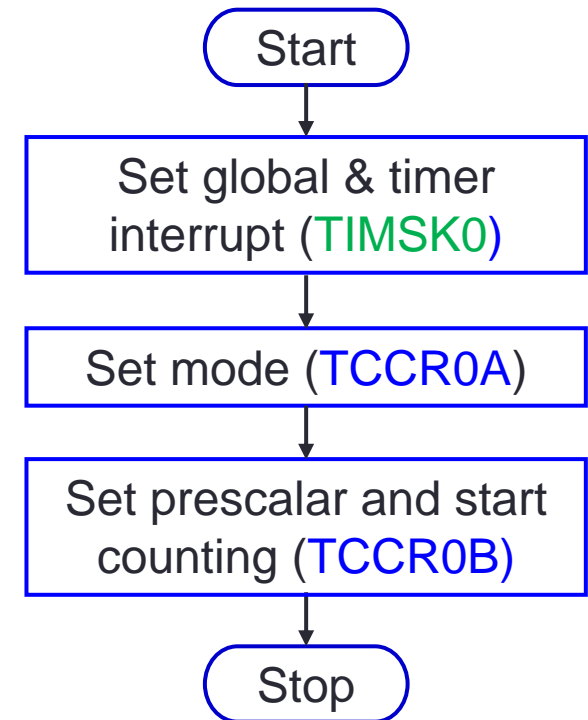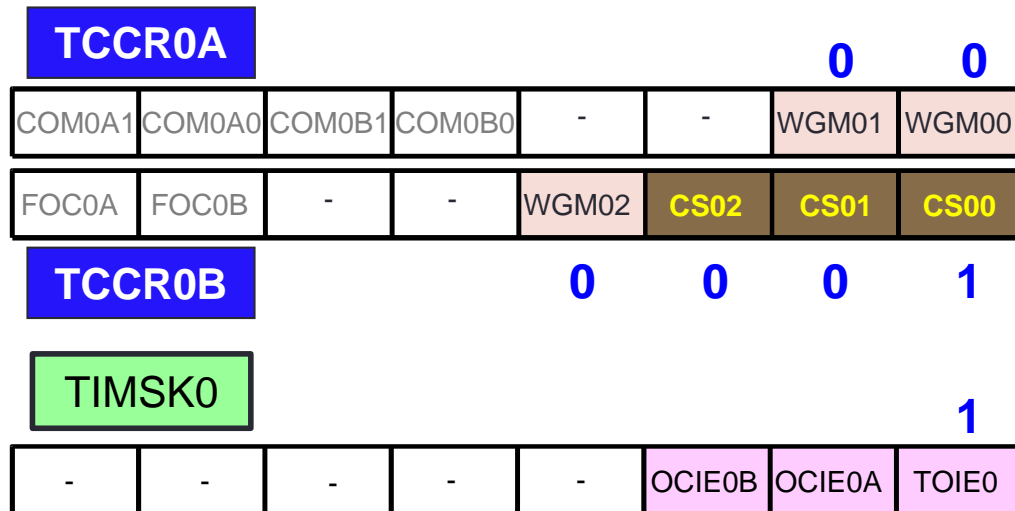
Example: Timer Overflow Interrupt

- Generate a square wave with a period of 512 system clock cycles on pin 5 of Port C (PC5)
- At the same time, transfer data from Port B to Port D
- Use Timer0 in normal mode with overflow interrupt

Flowchart (Timer Overflow Interrupt)

- What value do we set the controller registers?

TCCR0A

						0	**0**
COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00
FOC0A	FOC0B	-	-	WGM02	**CS02**	**CS01**	**CS00**
				0	**0**	**0**	**1**

TCCR0B

TIMSK0

							1
-	-	-	-	-	OCIE0B	OCIE0A	TOIE0

```
Start
  ↓
Set global & timer
interrupt (TIMSK0)
  ↓
Set mode (TCCR0A)
  ↓
Set prescalar and start
counting (TCCR0B)
  ↓
Stop
```

- ISR:

```
Start → Toggle PC5 (PORTC) → Stop
```

Timer0 Overflow Interrupt

```c
#include <avr/io.h>
#include <avr/interrupt.h>

volatile unsigned int value;

int main(void)
{
    DDRC|=0x20;                      // PC5 as input
    DDRB=0x00;                       // PORTB as input
    DDRD=0xFF;                       // PORTD as output
    sei();                           // enable global interrupts
    TIMSK0=(1<<TOIE0);               // timer overflow interrupt
    TCCR0B=0x01;                     // start counting
    while(1)
        PORTD=PINB;                  // transfer data
}

ISR(TIMER0_OVF_vect)
{
    PORTC^=0x20;                     // toggle PC5
}
```

Outline (Cont'd)

- Polling vs. interrupt
- Interrupt procedure
- Enable interrupt
- Interrupt programming
- Some other issues
- Getting started

Interrupt Priority

- The interrupt priorities in the ATmega series are fixed, and CANNOT be changed

Interrupt	ROM Location
Reset	0x0000
External Interrupt Request 0	0x0002
External Interrupt Request 1	0x0004
Time/Counter2 Compare Match A	0x000E
Time/Counter2 Compare Match B	0x0010
Time/Counter2 Overflow	0x0012
Time/Counter1 Compare Match A	0x0016
Time/Counter1 Compare Match B	0x0018
Time/Counter1 Overflow	0x001A
Time/Counter0 Compare Match A	0x001C
Time/Counter0 Compare Match B	0x001E
Time/Counter0 Overflow	0x0020

Highest priority

Lowest priority

External Interrupt Pins

External Interrupt Trigger

• Edge trigger versus level trigger

EICRA

-	-	-	-	ISC11	ISC10	ISC01	ISC00

ISC01	ISC00		Description
0	0		The low level of INT0 generates an interrupt request.
0	1		Any logical change on INT0 generates an interrupt request.
1	0		The falling edge of INT0 generates an interrupt request.
1	1		The rising edge of INT0 generates an interrupt request.

ISC11	ISC10		Description
0	0		The low level of INT1 generates an interrupt request.
0	1		Any logical change on INT1 generates an interrupt request.
1	0		The falling edge of INT1 generates an interrupt request.
1	1		The rising edge of INT1 generates an interrupt request.

Outline (Cont'd)

- Interrupt in assembly
  - Polling vs. interrupt
  - Interrupt procedure
  - Enable interrupt
- Interrupt in C programming
- Some other issues
- Example programs
- Getting started
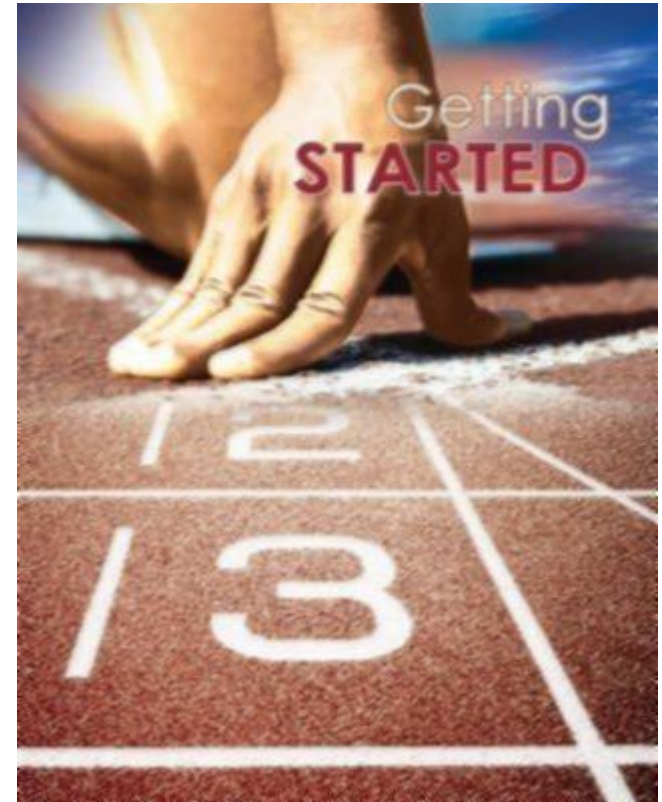
Outline (Cont'd)

- Interrupt in assembly
  - Polling vs. interrupt
  - Interrupt procedure
  - Enable interrupt
- Interrupt in C programming
- Some other issues
- Example programs
- Getting started

Reference

- ATmega328P data sheet
- AVR 8-bit instruction set
- AVR072: Accessing 16-bit I/O Registers
- AVR1200: Using External Interrupts for megaAVR Devices
- M. A. Mazidi, S. Naimi, and S. Naimi, *The AVR Microcontroller and Embedded Systems: Using Assembly and C*, Prentice Hall, 2010
- AVR GCC library help http://nongnu.org/avr-libc/user-manual/modules.html