Programming Microcontroller

Dipl.- Ing. Falk Salewski

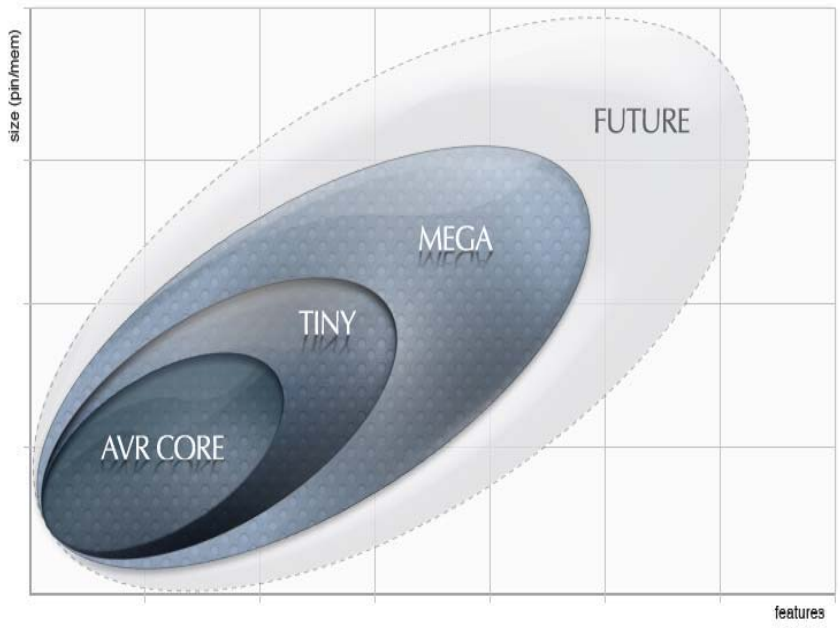
Lehrstuhl Informatik XI
RWTH Aachen
salewski@informatik.rwth-aachen.de

Summer term 2006

Microcontroller basics

- Microcontroller = CPU + Memory + internal Peripherals

AVR Microcontroller Family



- Devices range from 1 to 256KB
- Pin count range from 8 to 100
- Full code compatibility
- Pin/feature compatible families
- One set of development tools
- In-System Programming
- In-System Debugging

More on http://www.atmel.com/products/avr/overview.asp

ATmega16

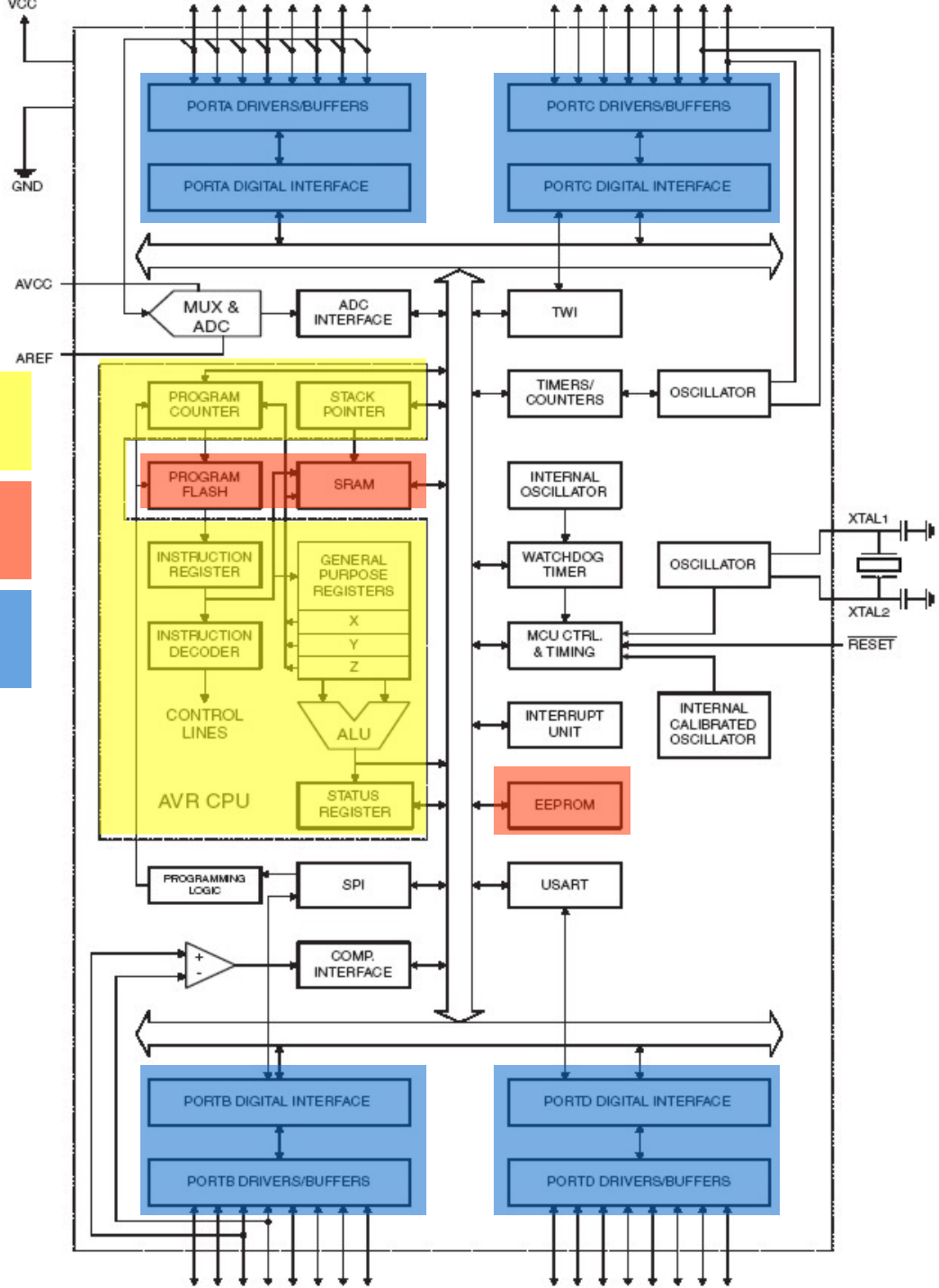
- We will use the ATMEL ATmega16
  - 8-bit Microcontroller with 16K Bytes In-System Programmable Flash
  - 512 Bytes EEPROM, 1K Byte Internal SRAM
  - Two 8-bit Timer/Counters, One 16-bit Timer/Counter
  - 8-channel, 10-bit ADC
  - Programmable Serial USART
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - JTAG Port for in-system programming and debugging
  - …
- More details: http://www.atmel.com/products/avr/

ATmega16

8bit CPU

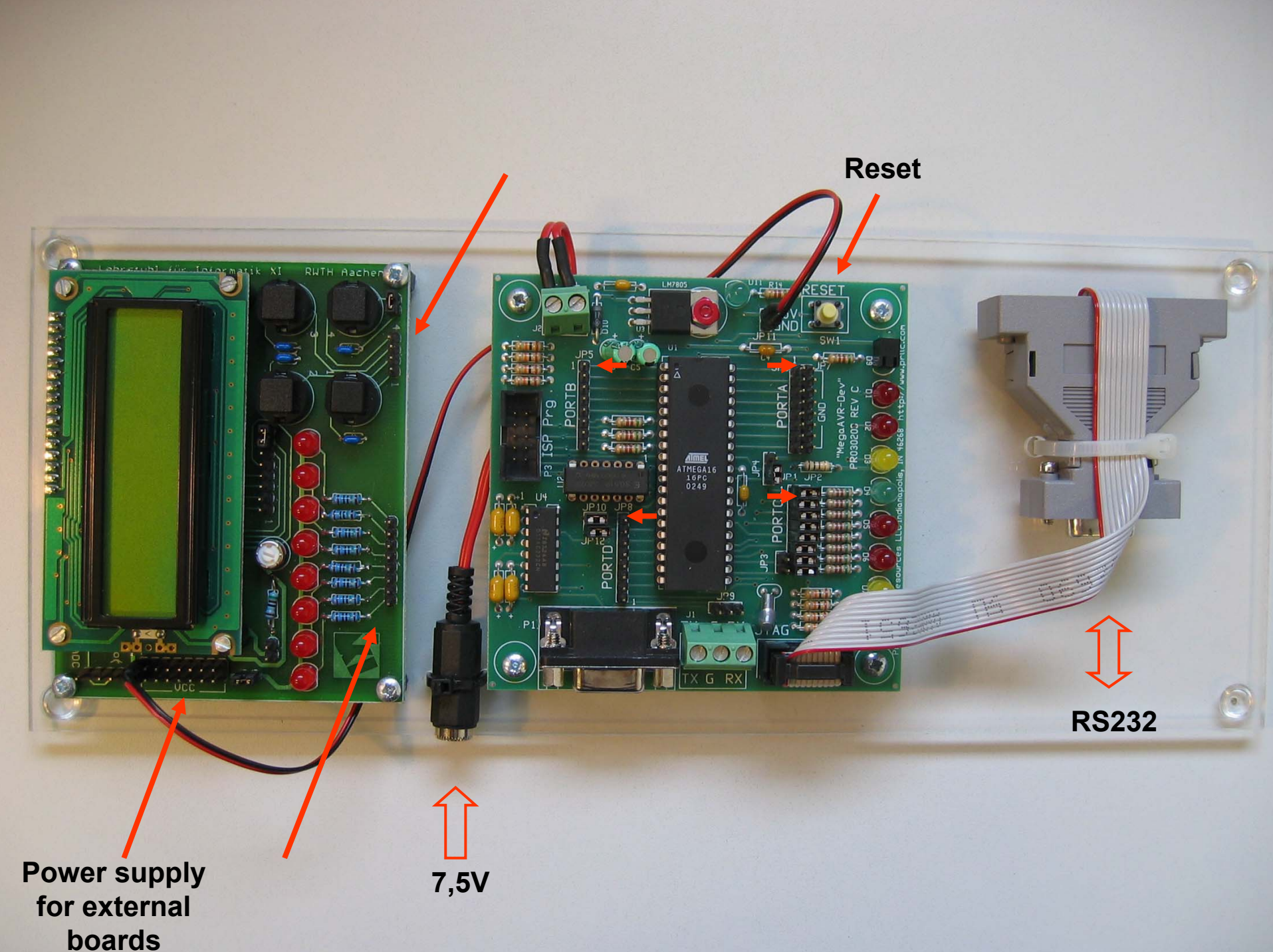
Memory

I/O Ports



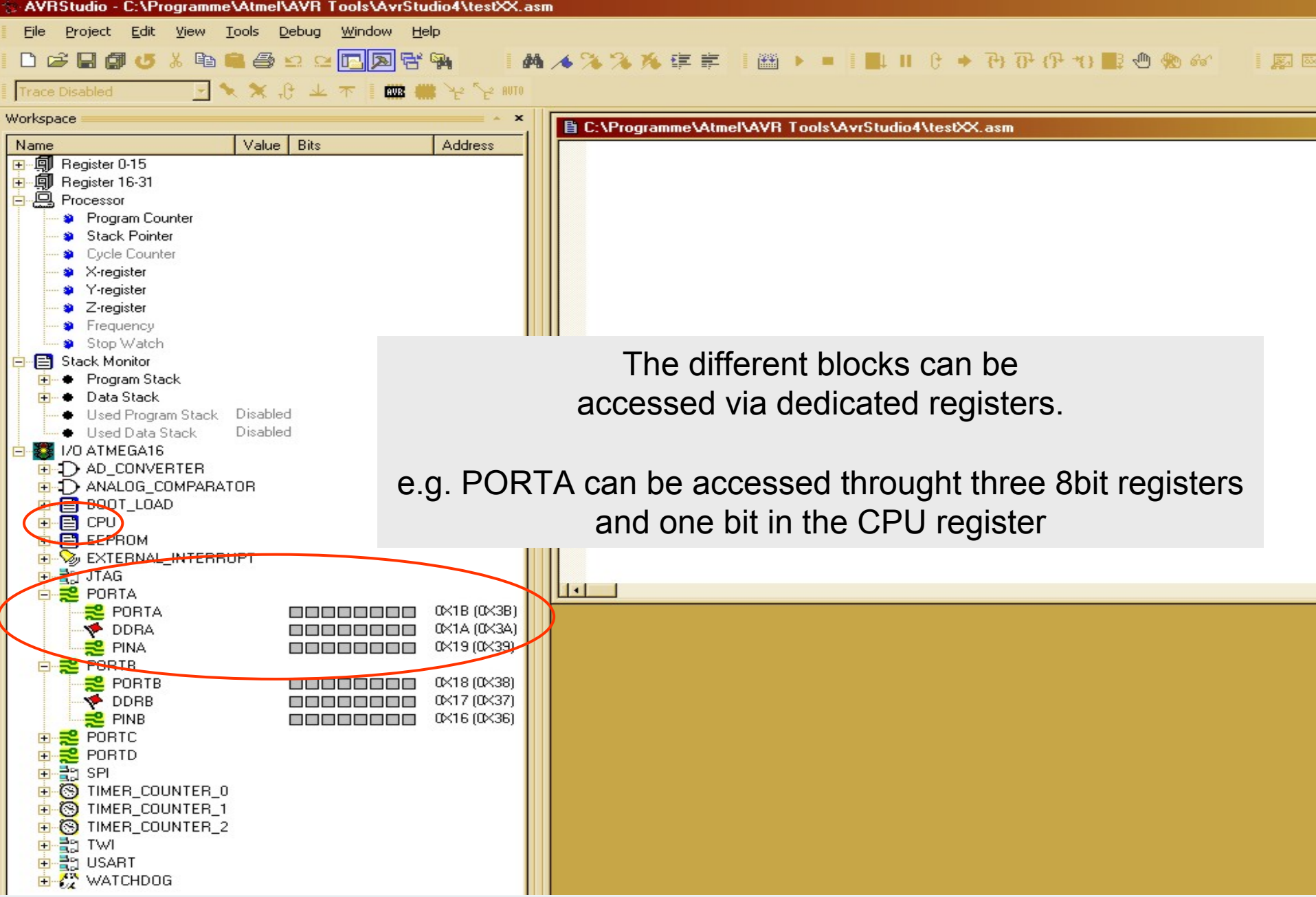
RWTH

LEHRSTUHL INFORMATIK XI SOFTWARE FÜR EINGEBETTETE SYSTEME


Reset
RS232
7,5V
Power supply
for external
boards

Question

- How can we access the
  - I/O Ports,
  - the Timers and
  - other internal peripherals?



The different blocks can be accessed via dedicated registers.

e.g. PORTA can be accessed throught three 8bit registers and one bit in the CPU register

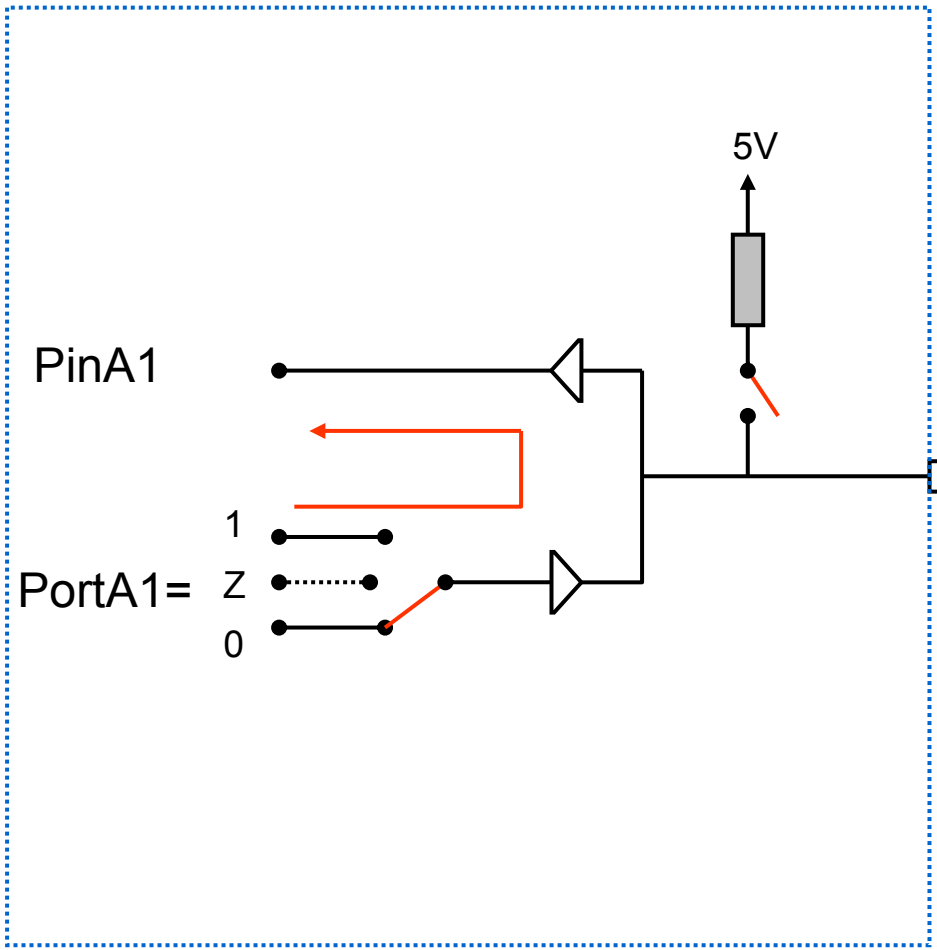
ATmega16 – I/O Ports

Port Pin Configurations PORTA:

DDRA	PORTA	PUD	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state
0	1	0	Input	Yes	
0	1	1	Input	No	Tri-state
1	0	X	Output	No	Output Low
1	1	X	Output	No	Output High

The port pin can always be read through the PINA Register bit.

Example: turn on an LED

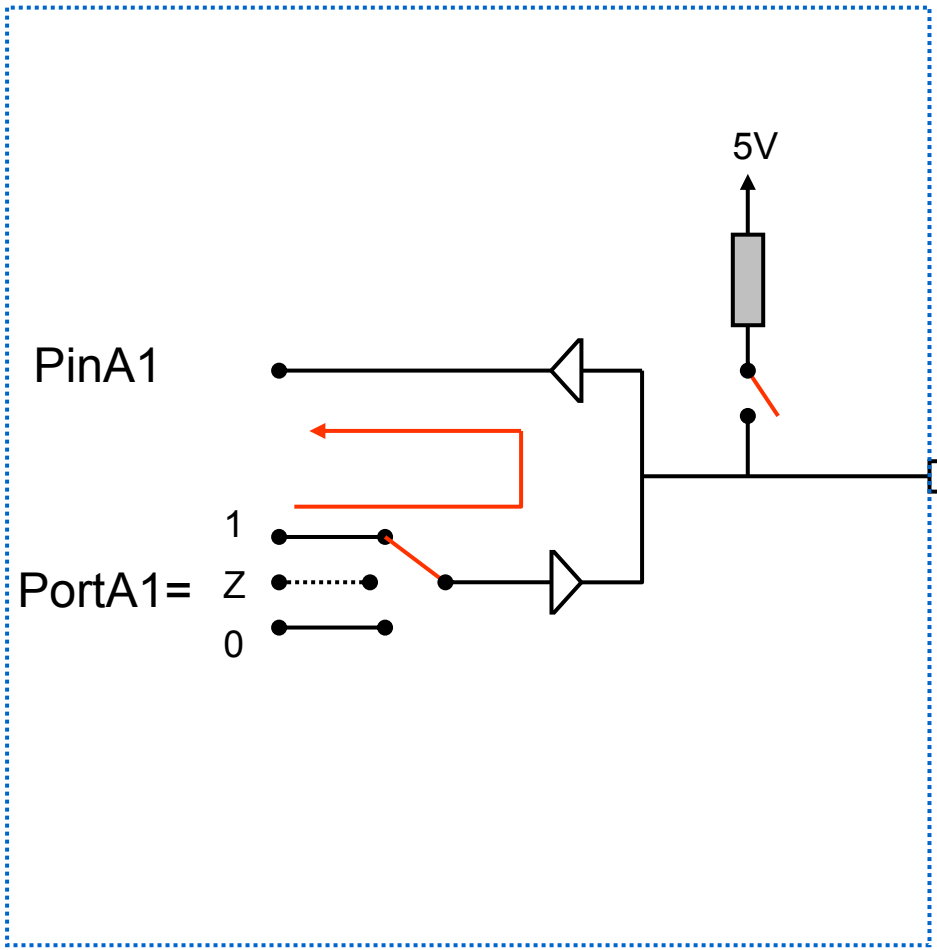


DDRA	PORTA	Signal on PINA1
0	0	1
0	1	1
1	0	PortA1
1	1	PortA1

X: don‘t care
U: unknown

LED is on !

Example: turn the LED off

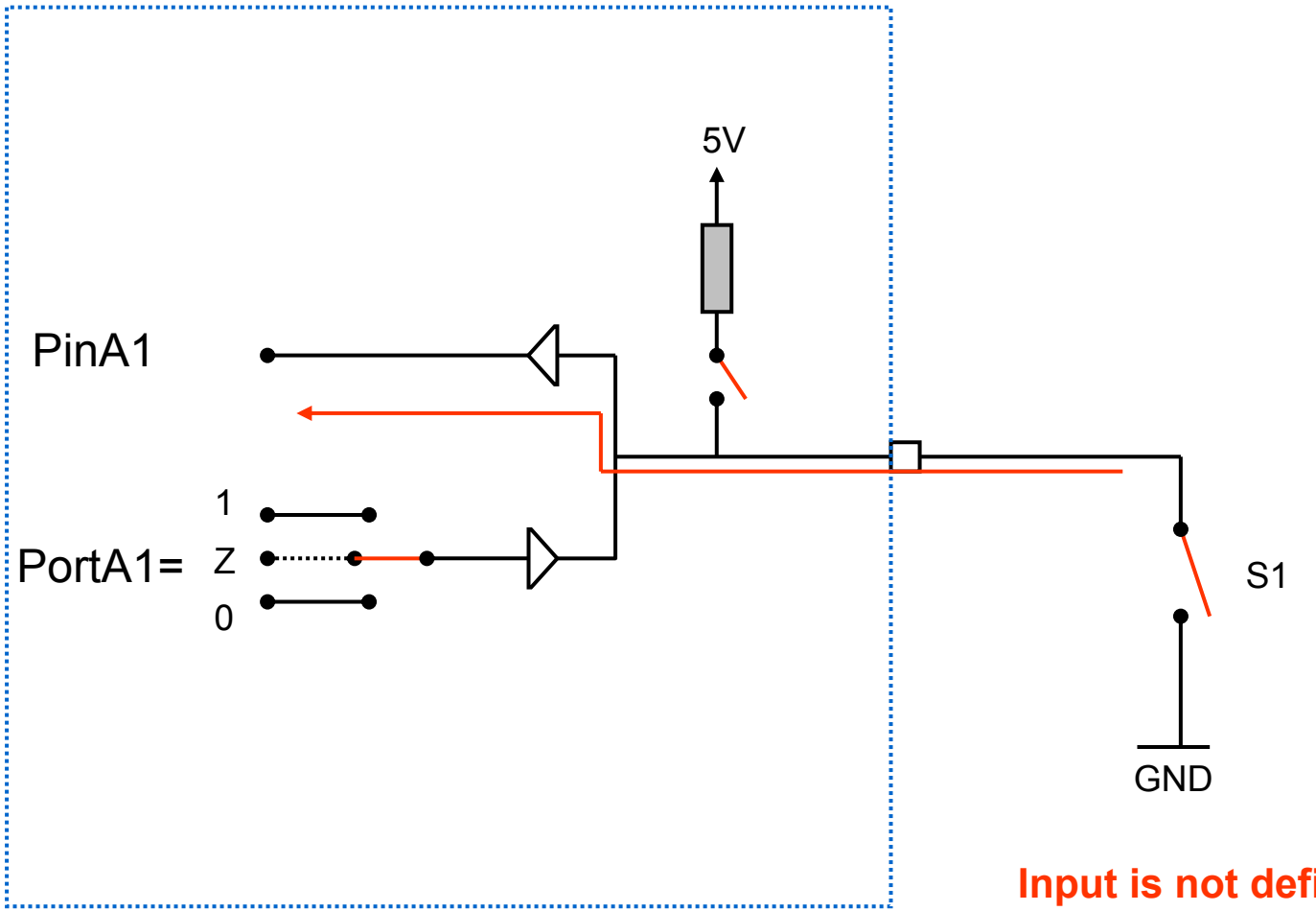


DDRA	PORTA	Signal on PINA1
0	0	1
0	1	1
1	0	PortA1
1	1	PortA1

X: don‘t care
U: unknown

LED is off !

Example: Reading Input

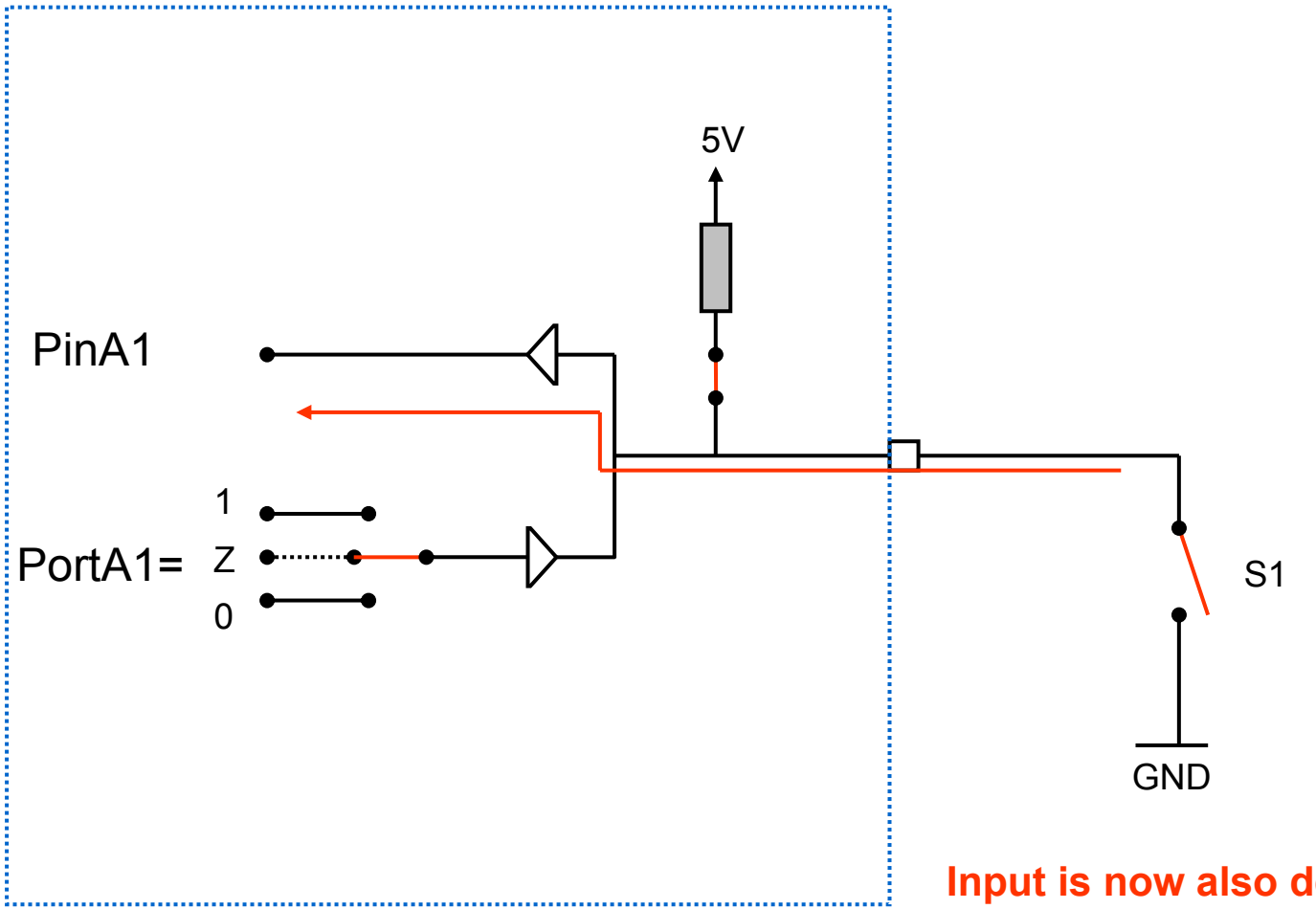


DDRA	PORTA	Signal on PINA1
0	0	S1 closed: 0 S1 open: U
0	1	S1 closed: 0 S1 open: 1
1	0	PortA1
1	1	PortA1

X: don‘t care
U: unknown

Input is not defined if S1 is open!

Example: Reading Input



DDRA	PORTA	Signal on PINA1
0	0	S1 closed: 0 S1 open: U
0	1	S1 closed: 0 S1 open: 1
1	0	PortA1
1	1	PortA1

X: don‘t care
U: unknown

Input is now also defined if S1 is open!

Register in C

	Lesen	Schreiben
Bitweise	`bit_is_set (<port>, <pin>);` `bit_is_clear (<port>, <pin>);`	`sbi (<register>, <bitnummer>);` `cbi (<register>, <bitnummer>);`
Rückgabewert	**bool** *(false=0, true=1)*	

Byteweise	`inp(<register>);`
	`Or`
	`Var = PINA;`
		`outp (<wert>, <register>);`
		`Or`
		`PORTA = 0xFF;`

Rückgabewert	**unsigned char**	

Include the following header: *#include <avr/io.h>*

Example in C

```
//LED to 5V at PINA1;  SWITCH to GND at PINA0

#include <avr/io.h>

int main (void)
{
 outp(0xFE,DDRA);      //PortA: Pin0: Input,  Pin1..7: Output
 outp(0xFF,PORTA);     //PortA: Pin0: pull up,  Pin1..7: high = LED off
 while(1)
 {
      if(bit_is_set (PINA,0))            //check if PinA0 is high
      {
            cbi(PORTA,1);                //clear PinA1 = LED on
      }
      else
      {
            sbi(PORTA,1);                //set PinA1 = LED off
      }
 }
}
```

Design steps

Create C source code in Programmers Notepad

Create Makefile with Mfile and safe it to the folder of your source code

- „Main File Name...“ (Name of the main program without the extension(.c))
- „MCU Type” (ATmega16)
- “Debug format” (AVR-ext-coff)

in Programmers Notepad execute

Tools -> [WinAVR] Make All

Connect MCU board to PC

Start Atmel AVR Studio

Click "open" and choose the created .cof-File

Download should start automatically

Execution & Debugging

Use the following programs:

- Programmers Notepad
- Mfile
- Atmel AVR Studio

Exercise

- Download and test LED Program
- Use a breakpoint to stop program if switch is pressed

Using Interrups (IR)

- ATmega16: 20 different IR sources ! (details in iom16.h)
- Each IR leads to a jump to an individual Interupt Service Routine (ISR)
- 4 external IR sources are available (IR by external event)
  - External IR0 (SIG_INTERRUPT0)
  - External IR1 (SIG_INTERRUPT1)
  - External IR2 (SIG_INTERRUPT2)
  - Input Capture 1 (SIG_INPUT_CAPTURE1)
- The following headers are needed:
  - #include <avr/interrupt.h>
  - #include <avr/signal.h>

Example with IR

```
//LEDs an PORTD gegen 5V & Schalter an PINB2(external IR2) gegen GND

#include <avr/io.h>             //für Zugriff auf I/O-Register
#include <avr/interrupt.h>      //für IR-Unterstützung
#include <avr/signal.h>         //für "SIGNAL" (entspricht ISR)

void delay(unsigned int j)
{
…some delay function
}

int main (void)
{
 outp(0xFF,DDRD);               //PORTD: output
 outp(0x00,DDRB);               //PORTB: input
 outp(0xFF,PORTB);              //PORTB: Pull-ups aktivated
 cbi(MCUCSR,6);                 //IR2 with falling edge (default)
 sbi(GICR,5);                   //ext. IR2 aktivieren          individual IR activation
 sbi(SREG,7);                   //Global IR enable             global IR activation

 while(1)
 {
       //some main program
 }
 return(1);
}

//ISR für externen IR2, wird aufgerufen sobald fallende Flanke an PINB2
SIGNAL (SIG_INTERRUPT2)   //alternative: SIGNAL(_VECTOR(18))
{
 outp(0x00,PORTD);              //Pin=low  => LEDs an
 delay(5);
 outp(0xFF,PORTD);              //Pin=high => LEDs aus
}
```

All interrupt sources: iom16.h

- /* Interrupt vectors, path: C:\Programme\WinAVR\avr\include\avr */

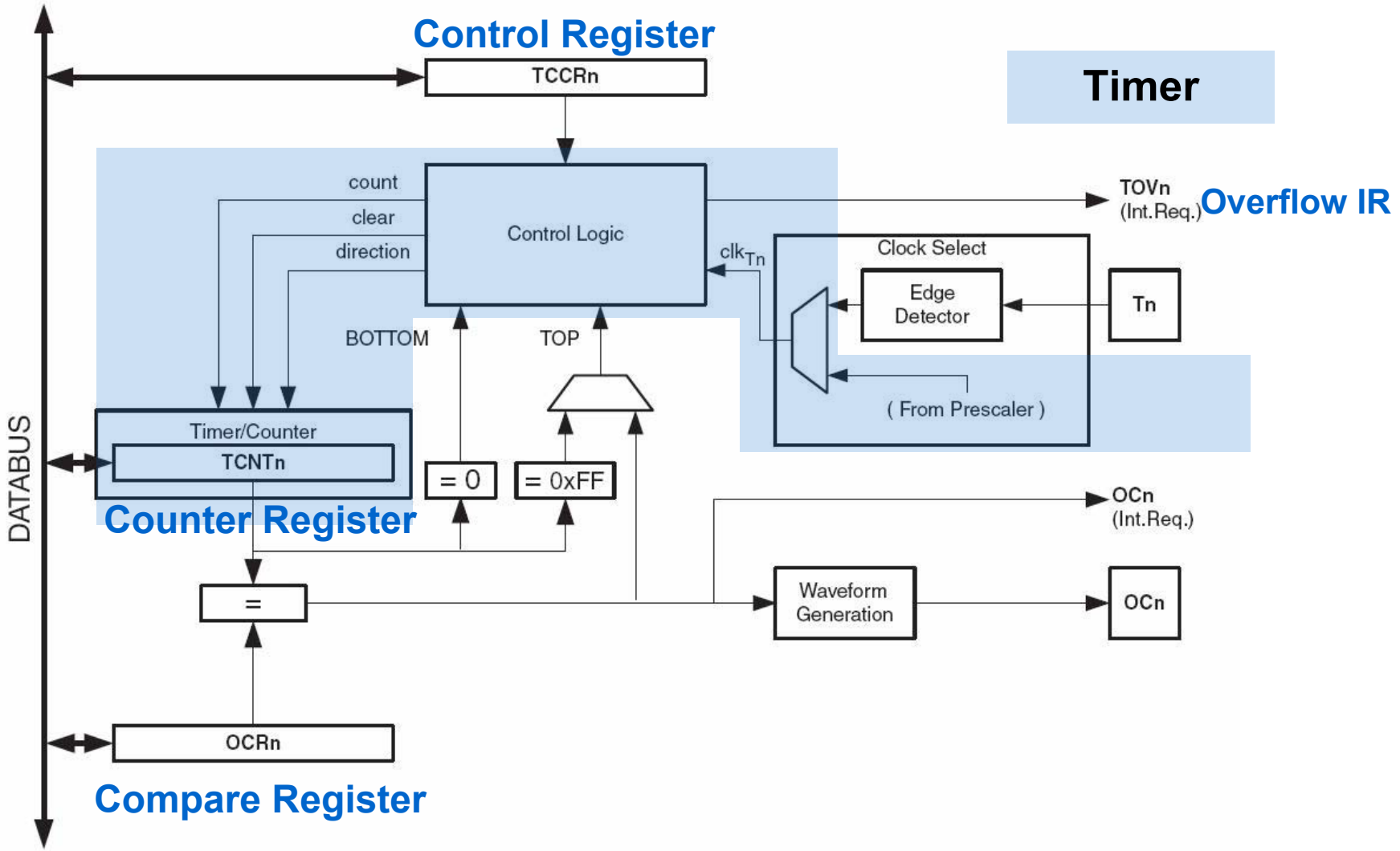
- #define SIG_INTERRUPT0 _VECTOR(1)
- #define SIG_INTERRUPT1 _VECTOR(2)
- #define SIG_OUTPUT_COMPARE2 _VECTOR(3)
- #define SIG_OVERFLOW2 _VECTOR(4)
- #define SIG_INPUT_CAPTURE1 _VECTOR(5)
- #define SIG_OUTPUT_COMPARE1A _VECTOR(6)
- #define SIG_OUTPUT_COMPARE1B _VECTOR(7)
- #define SIG_OVERFLOW1 _VECTOR(8)
- #define SIG_OVERFLOW0 _VECTOR(9)
- #define SIG_SPI _VECTOR(10)
- #define SIG_UART_RECV _VECTOR(11)
- #define SIG_UART_DATA _VECTOR(12)
- #define SIG_UART_TRANS _VECTOR(13)
- #define SIG_ADC _VECTOR(14)
- #define SIG_EEPROM_READY _VECTOR(15)
- #define SIG_COMPARATOR _VECTOR(16)
- #define SIG_2WIRE_SERIAL _VECTOR(17)
- #define SIG_INTERRUPT2 _VECTOR(18)
- #define SIG_OUTPUT_COMPARE0 _VECTOR(19)
- #define SIG_SPM_READY _VECTOR(20)

Atmega16 – Timer/Counter

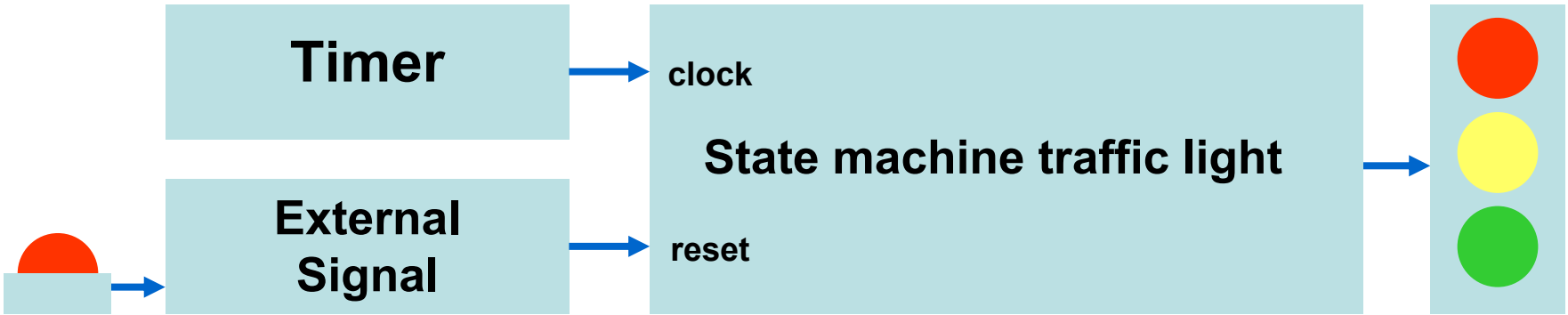
- Timer:
  - counting the system clock (6MHz in our case)
  - a prescaler can be used
- Counter:
  - counting an external signal

- The ATmega16 has 3 devices that can be used as Timer or Counter
  - Timer/Counter0: 8bit
  - Timer/Counter1: 16bit
  - Timer/Counter2: 8bit

ATmega16 – Timer/Counter0



New example: Traffic light



red(40%) => yellow(10%) => green(40%) => yellow(10%) => red…

If reset: blink yellow (~1Hz) for 4 seconds, then start cycle with red

The whole cycle should take 3 - 5 seconds

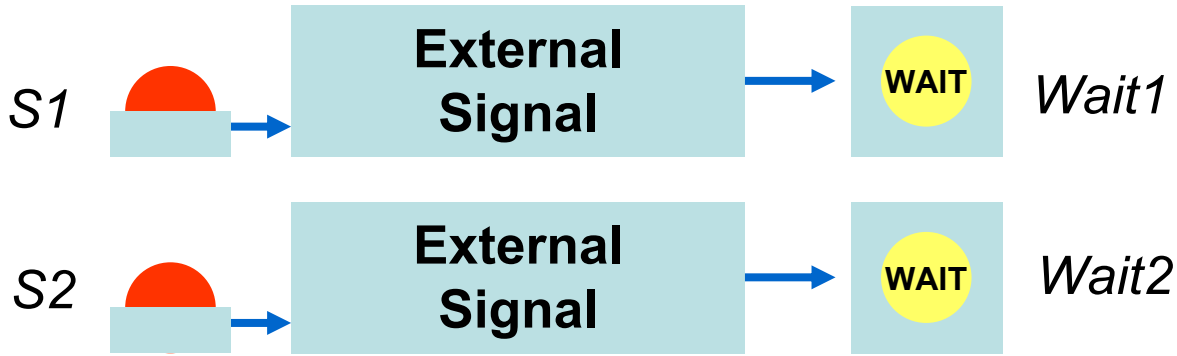
Use the standard MCU RESET on board

Example: Traffic Light (2)

- You need:
  - Describtion development board
    (CD \Data sheets\ATmega16 Development Board)
  - Describtion MCU-Peripheral
    (CD \Data sheets\ATmega16 Development Board)
  - WinAVR
    (already installed, free download from the web possible)
  - ATMEL Studio 4.0
    (already installed, free download from the web possible)

Example: Traffic Light (3)

- Add the following feature (pedestrian light):



- As soon as S1 has been pressed *Wait1* has to blink (~2Hz) until the traffic light is red. It has to be off until S1 is pressed again.
- As soon as S2 has been pressed *Wait2* has to blink (~2Hz) until the traffic light is red. It has to be off until S2 is pressed again.

Example: Traffic Light (4)

- How did you realize sequential and concurrent structures?
- Pro and cons of concurrent structures?
  ➔Do not overload ISRs!
- Pro and cons of sequential structures?
  ➔problems to deal quickly with external events

  ➔Find optimal combination!

WD Timer

- To enable an automatic reset in the case that the MCU „hangs“ the intgrated watchdog timer should be used.
- If enabled the WD has to be triggerd within a certain time
- If not triggered in time the WD resets the MCU

Useful Links

- http://www.mikrocontroller.net/wiki/AVR-GCC-Tutorial
- http://atmel.com/products/avr/
- Die Programmiersprache C. Ein Nachschlagewerk

  Regionales Rechenzentrum für Niedersachsen/Universität Hannover

  http://www.rz.rwth-aachen.de/computing/sw/rrzn/index.php
- C – Programmieren von Anfang an

  Helmut Erlenkötter, ISBN 3-499-60074-9
- Programming Embedded Systems in C and C++

  Michael Barr, ISBN 1-56592-354-5

➔use Windows calculater for 0x04 => 0b00000100 => 4