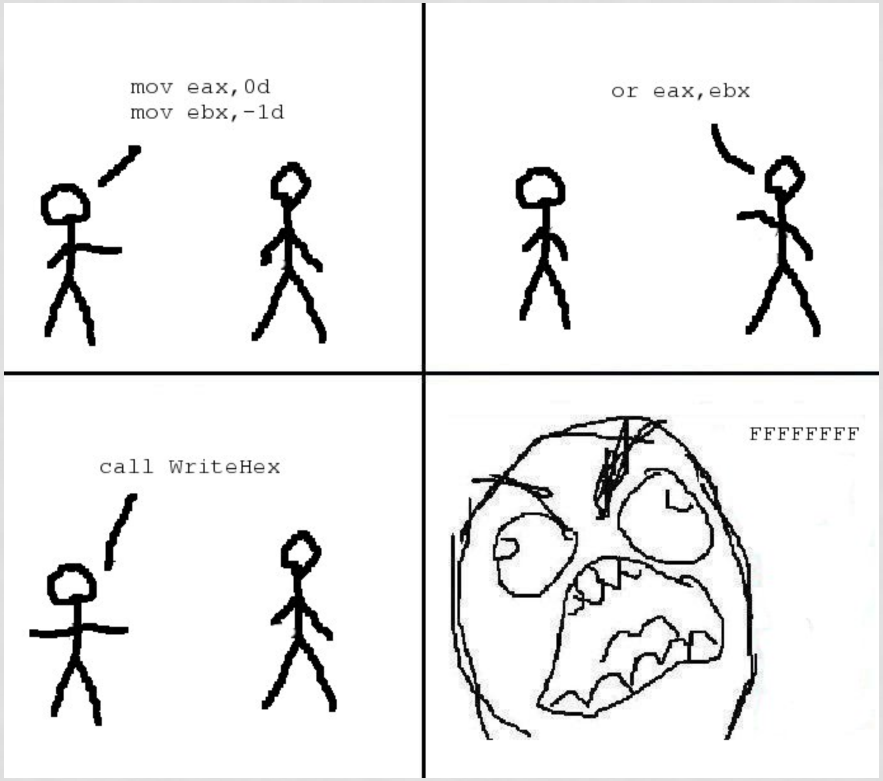
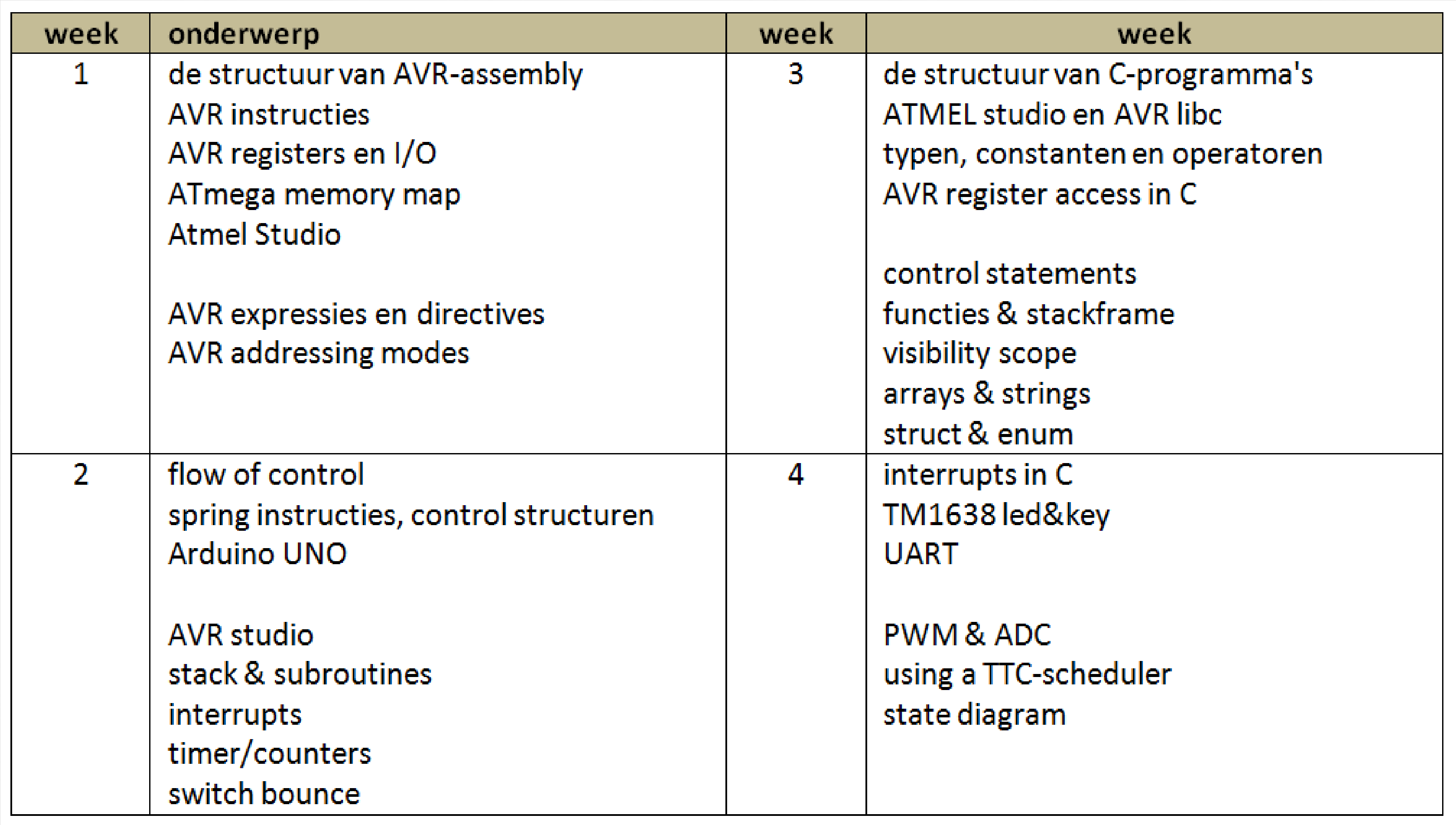


ASSEMBLY & C

WEEK 4-1

AGENDA



week	onderwerp	week	week
1	de structuur van AVR-assembly AVR instructies AVR registers en I/O ATmega memory map Atmel Studio AVR expressies en directives AVR addressing modes	3	de structuur van C-programma's ATMEL studio en AVR libc typen, constanten en operatoren AVR register access in C control statements functies & stackframe visibility scope arrays & strings struct & enum
2	flow of control spring instructies, control structuren Arduino UNO AVR studio stack & subroutines interrupts timer/counters switch bounce	4	interrupts in C TM1638 led&key UART PWM & ADC using a TTC-scheduler state diagram

AGENDA

- **interrupts in C**
- het volatile keyword
- TM1638 led&key
- UART

INTERRUPTS ATMEGA328P

reset and Interrupt vectors placement in code segment

-reset
-externe interrupts
-timer interrupts
-seriële interfaces

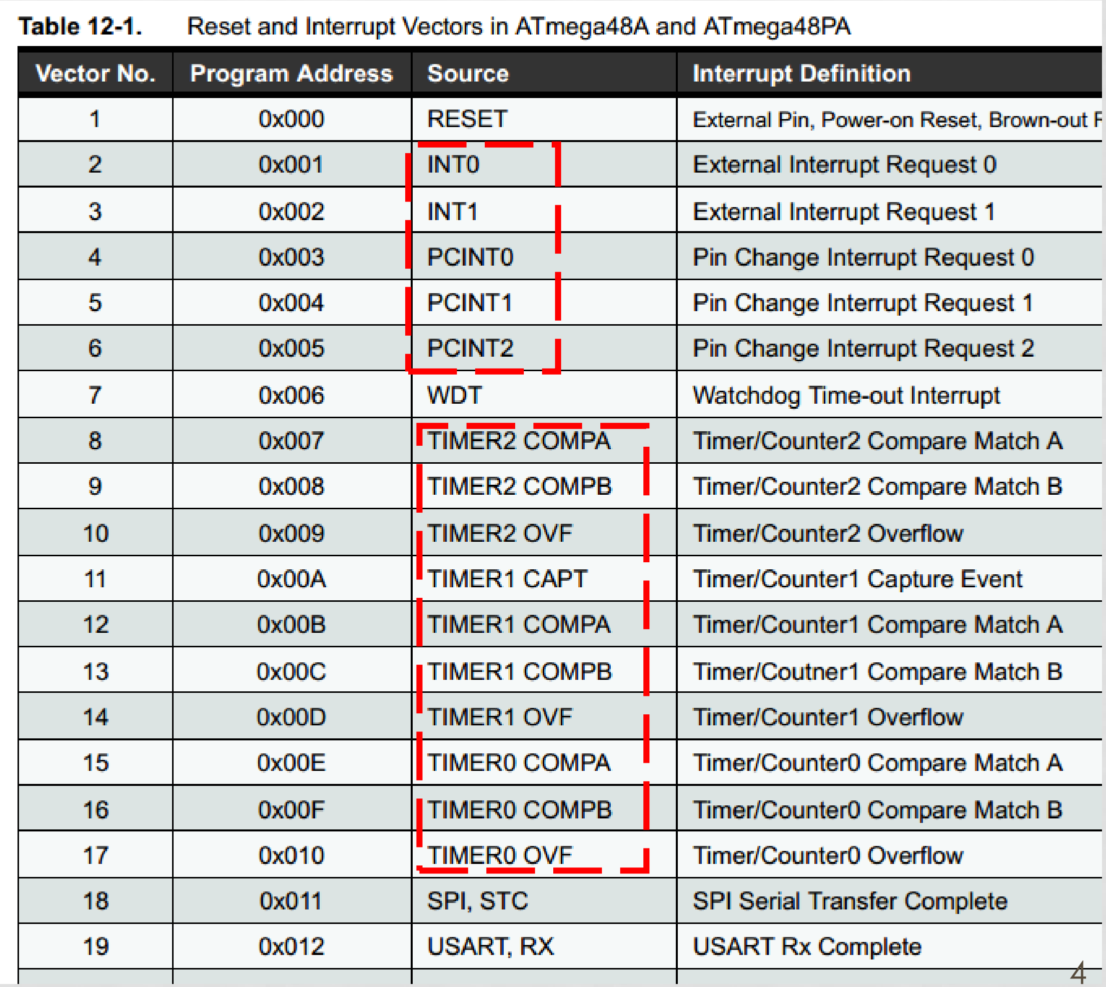
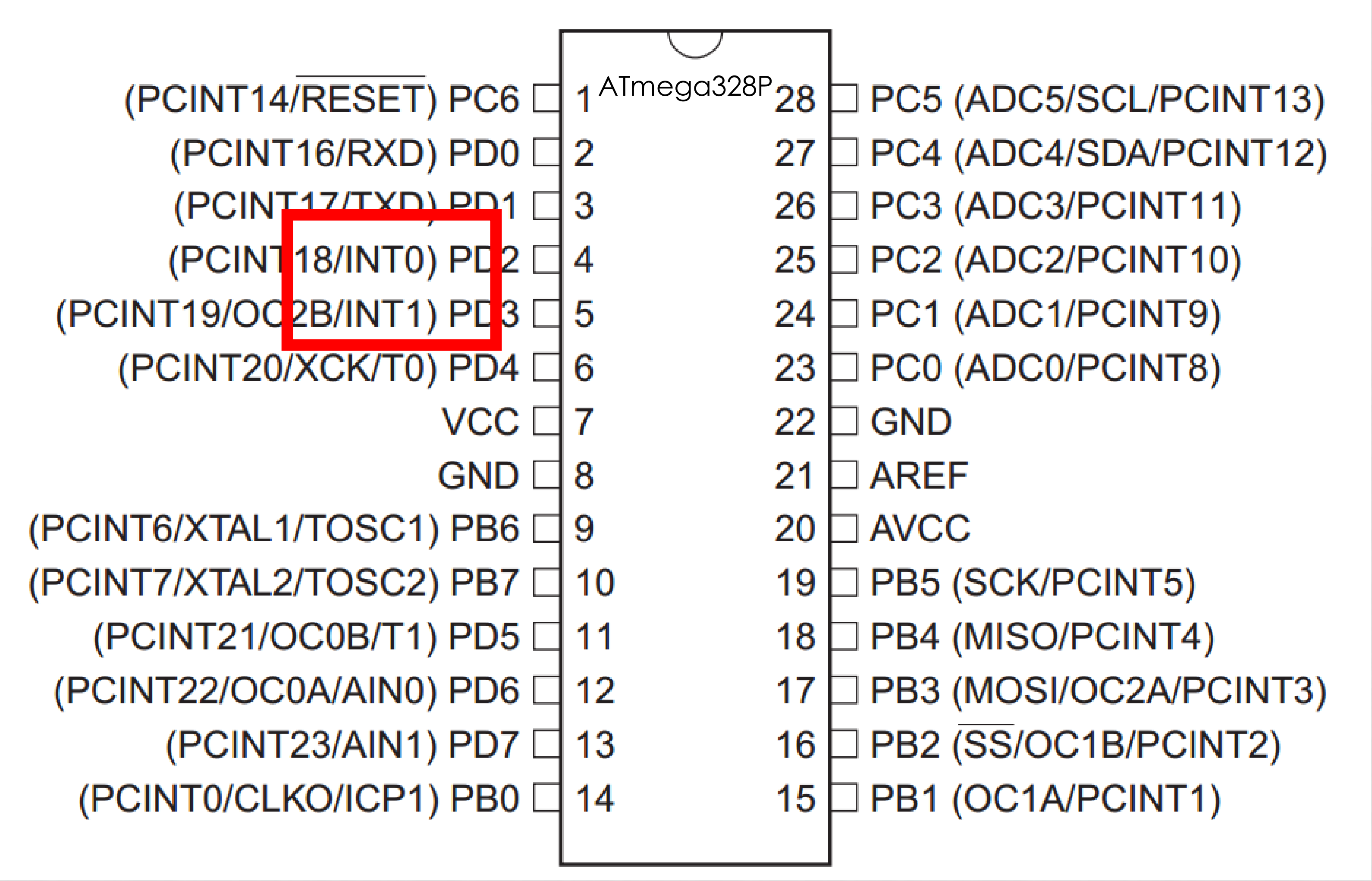


Table 12-1. Reset and Interrupt Vectors in ATmega48A and ATmega48PA

Vector No.	Program Address	Source	Interrupt Definition
1	0x000	RESET	External Pin, Power-on Reset, Brown-out R
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	PCINT0	Pin Change Interrupt Request 0
5	0x004	PCINT1	Pin Change Interrupt Request 1
6	0x005	PCINT2	Pin Change Interrupt Request 2
7	0x006	WDT	Watchdog Time-out Interrupt
8	0x007	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x008	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x009	TIMER2 OVF	Timer/Counter2 Overflow
11	0x00A	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x00B	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x00C	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x00D	TIMER1 OVF	Timer/Counter1 Overflow
15	0x00E	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x00F	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x010	TIMER0 OVF	Timer/Counter0 Overflow
18	0x011	SPI, STC	SPI Serial Transfer Complete
19	0x012	USART, RX	USART Rx Complete

INT0 EN INT1



ATmega328P
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)

MAPPING IO.H

- in project file :

```
<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build"
xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
    <SchemaVersion>2.0</SchemaVersion>
    ...
    <avrdevice>ATmega328P</avrdevice>
```

- #include <avr/io.h> is mapped to :

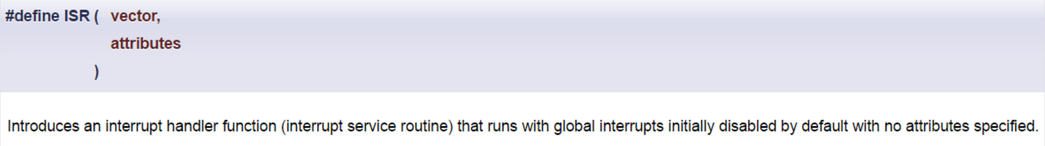
C:\Program Files (x86)\Atmel\Atmel Studio 6.0\extensions\Atmel\
AVRGCC\3.4.1.95\AVRToolchain\avr\include\avr\iom328p.h

iom328p.h

```
/* External Interrupt Request 0 */
#define INT0_vect                       _VECTOR(1)

/* Timer/Counter1 Compare Match A */
#define TIMER1_COMPA_vect               _VECTOR(11)

/* Timer/Counter1 Overflow */
#define TIMER1_OVF_vect  _VECTOR(13)
```

my_program.c

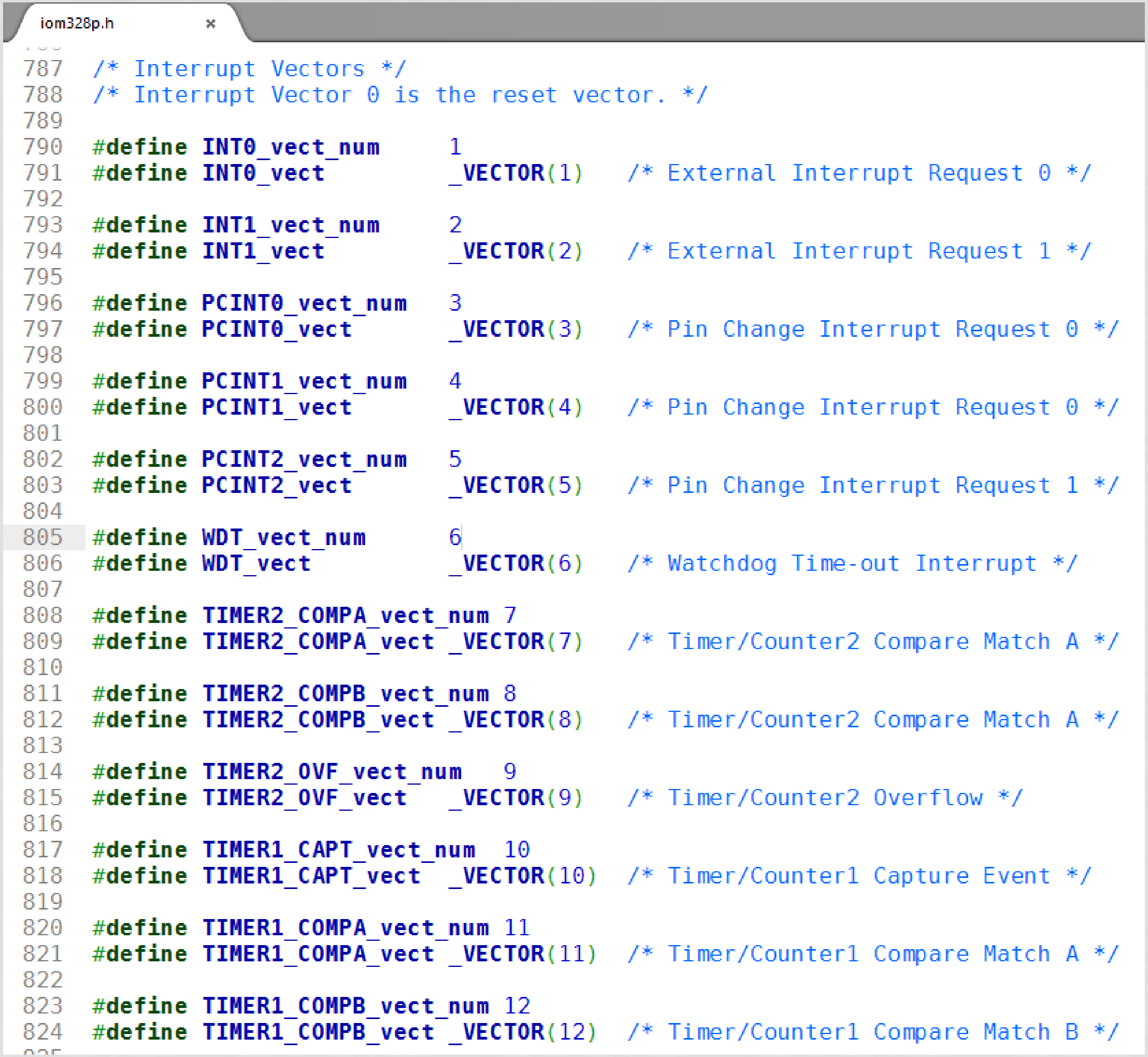
```
#include <avr/io.h>
#include <avr/interrupt.h>

ISR (TIMER1_COMPA_vect)
{
        // user code here
}
```



```
#define ISR (  vector,
               attributes
            )
```

Introduces an interrupt handler function (interrupt service routine) that runs with global interrupts initially disabled by default with no attributes specified.

see : http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html



```
/* Interrupt Vectors */
/* Interrupt Vector 0 is the reset vector. */

#define INT0_vect_num     1
#define INT0_vect         _VECTOR(1)   /* External Interrupt Request 0 */

#define INT1_vect_num     2
#define INT1_vect         _VECTOR(2)   /* External Interrupt Request 1 */

#define PCINT0_vect_num   3
#define PCINT0_vect       _VECTOR(3)   /* Pin Change Interrupt Request 0 */

#define PCINT1_vect_num   4
#define PCINT1_vect       _VECTOR(4)   /* Pin Change Interrupt Request 0 */

#define PCINT2_vect_num   5
#define PCINT2_vect       _VECTOR(5)   /* Pin Change Interrupt Request 1 */

#define WDT_vect_num      6
#define WDT_vect          _VECTOR(6)   /* Watchdog Time-out Interrupt */

#define TIMER2_COMPA_vect_num 7
#define TIMER2_COMPA_vect _VECTOR(7)   /* Timer/Counter2 Compare Match A */

#define TIMER2_COMPB_vect_num 8
#define TIMER2_COMPB_vect _VECTOR(8)   /* Timer/Counter2 Compare Match A */

#define TIMER2_OVF_vect_num   9
#define TIMER2_OVF_vect   _VECTOR(9)   /* Timer/Counter2 Overflow */

#define TIMER1_CAPT_vect_num  10
#define TIMER1_CAPT_vect  _VECTOR(10)  /* Timer/Counter1 Capture Event */

#define TIMER1_COMPA_vect_num 11
#define TIMER1_COMPA_vect _VECTOR(11)  /* Timer/Counter1 Compare Match A */

#define TIMER1_COMPB_vect_num 12
#define TIMER1_COMPB_vect _VECTOR(12)  /* Timer/Counter1 Compare Match B */
```

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

void init_ports(void){
DDRD = 0xFF ; //port D output
}

void init_timer (void){
// prescale op 256, top counter = value OCR1A (CTC mode)
TCCR1B = (1 << CS12) | (1 << WGM12);
TIMSK1 = 1 << OCIE1A; // Timer 1 Output Compare A Match Interrupt Enable
OCR1A = (uint16_t)62499; // 1 sec = (256/16.000.000)*62499
}

// iedere seconde wordt deze functie uitgevoerd
ISR (TIMER1_COMPA_vect){
PORTD = ~PORTD; // inverteer poort
}

int main(void){
init_ports();
init_timer();
sei();
while(1){};
return 0;
}
```

vergeet niet de haakjes !

interrupt library verzorgt zelf :

- vector table vullen
- stack pointer opzetten
- context switch : registers saven en restoren

AGENDA

- interrupts in C
- **het volatile keyword**
- TM1638 led&key
- UART

wat is er mis met dit programma ?

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

uint8_t counter = 0;

// hier : initialisatie poorten en interrupt

ISR(INT0_vect)
{
    counter++;
}

void main(void)
{
    while (counter == 0) {
        PORTB = $ff;
    }
}
```

compiler optimalisatie : in de lus wordt counter niet meer uit geheugen gehaald

VOLATILE KEYWORD

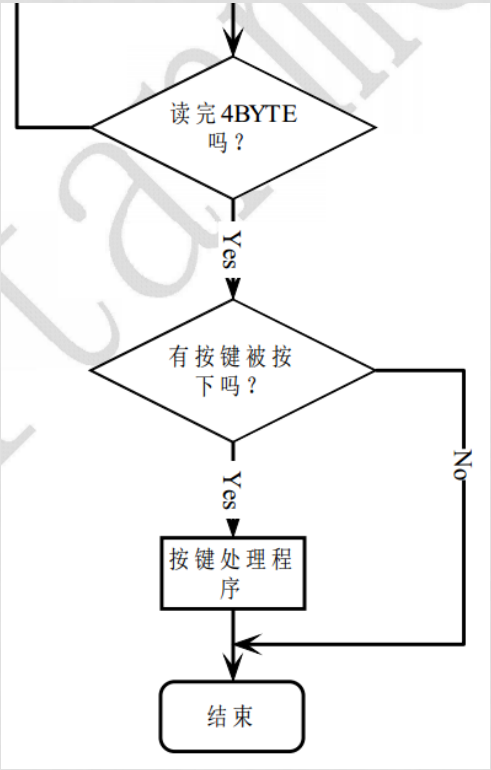
- volatile uint8_t counter = 0;
- gebruik volatile wanneer een interrupt (of ander proces of thread) de waarde van een variabele plotseling kan veranderen
- compiler houdt bij optimalisatie geen rekening met interrupts of andere threads
- "volatile" vertelt de compiler dat bij elke referentie de waarde opnieuw uit geheugen moet worden gehaald (i.p.v steeds een kopie gebruiken in CPU register)

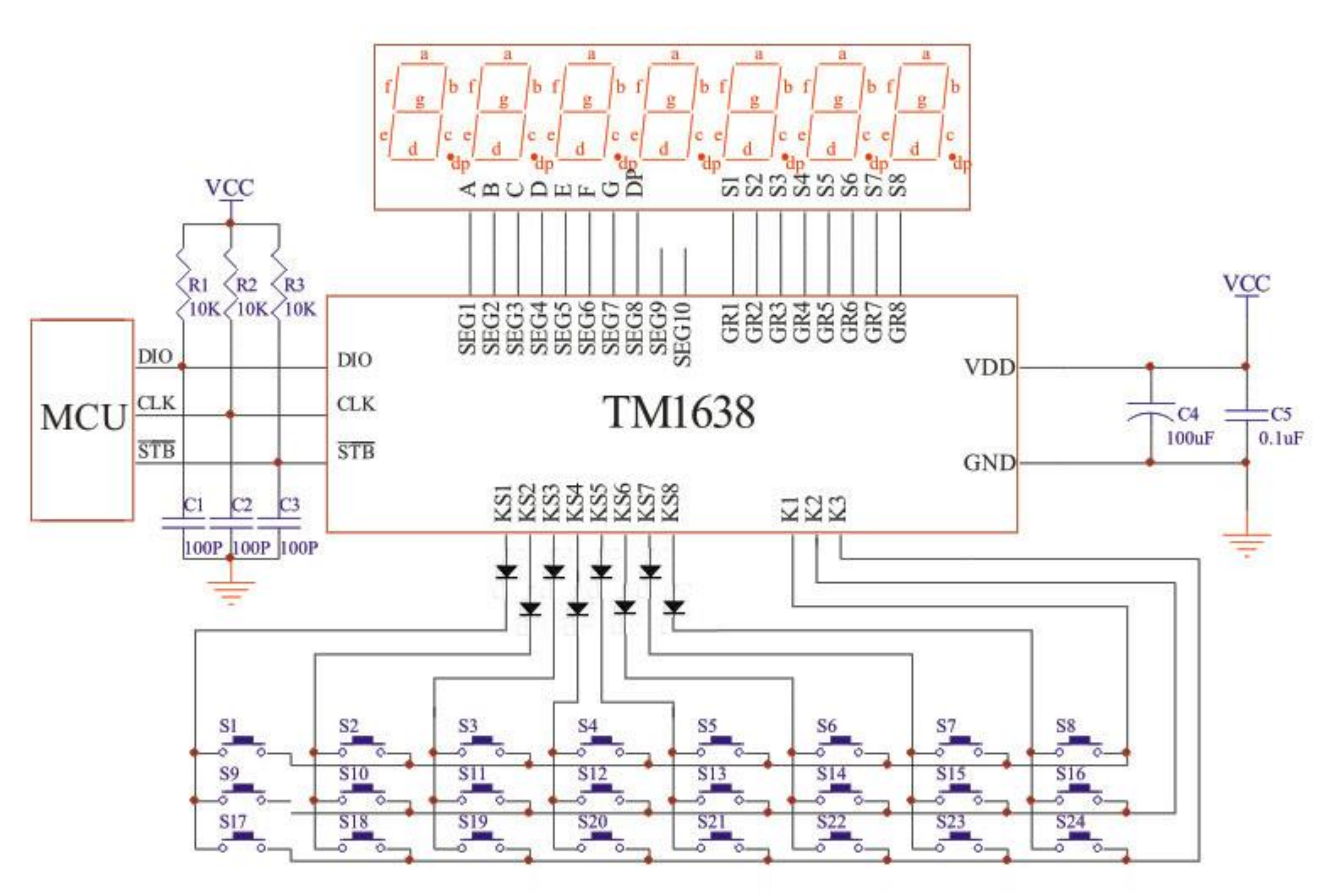
AGENDA

- interrupts in C
- het volatile keyword
- **TM1638 led&key**
- UART

TM1638

- a domestic Chinese product from “Titan Micro Electronics"
  - user manual : **我的中文不好**
- 8x
  - 7-segment red LED digits
  - red LEDs
  - push buttons
- 3 control pins plus power & ground
  - strobe = low when sending data
  - clock : bit is valid on rising edge
  - data = input & output



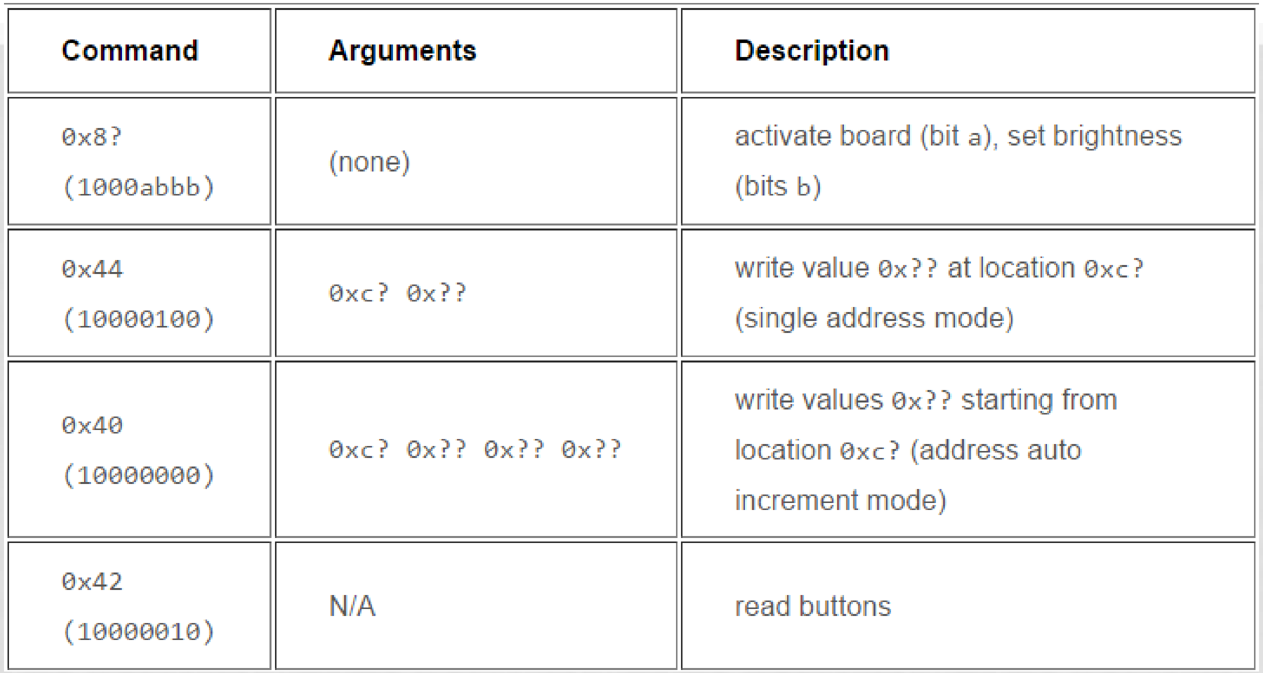


a
f
g
b
e
d
c
dp
A B C D E F G DP
S1 S2 S3 S4 S5 S6 S7 S8
VCC
R1 10K
R2 10K
R3 10K
MCU
DIO
CLK
STB
C1 100P
C2 100P
C3 100P
TM1638
SEG1 SEG2 SEG3 SEG4 SEG5 SEG6 SEG7 SEG8 SEG9 SEG10
GR1 GR2 GR3 GR4 GR5 GR6 GR7 GR8
VDD
GND
VCC
C4 100uF
C5 0.1uF
KS1 KS2 KS3 KS4 KS5 KS6 KS7 KS8
K1 K2 K3
S1 S2 S3 S4 S5 S6 S7 S8
S9 S10 S11 S12 S13 S14 S15 S16
S17 S18 S19 S20 S21 S22 S23 S24

TM1638

- data : first byte = command, following bytes arguments for the selected function
- board has 4 functions:
  - activate/deactivate board and initialize display
  - write a byte at specific address (internal RAM)
  - write bytes starting from specific address (internal RAM)
  - read buttons
- English 'user manual' on Blackboard
- https://blog.3d-logic.com/2015/01/10/
  using-a-tm1638-based-board-with-arduino/

COMMANDS



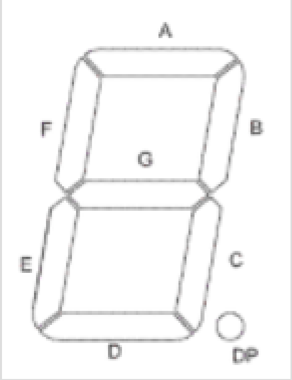
Command	Arguments	Description

`0x8?`
`(1000abbb)`
	(none)	activate board (bit `a`), set brightness (bits `b`)

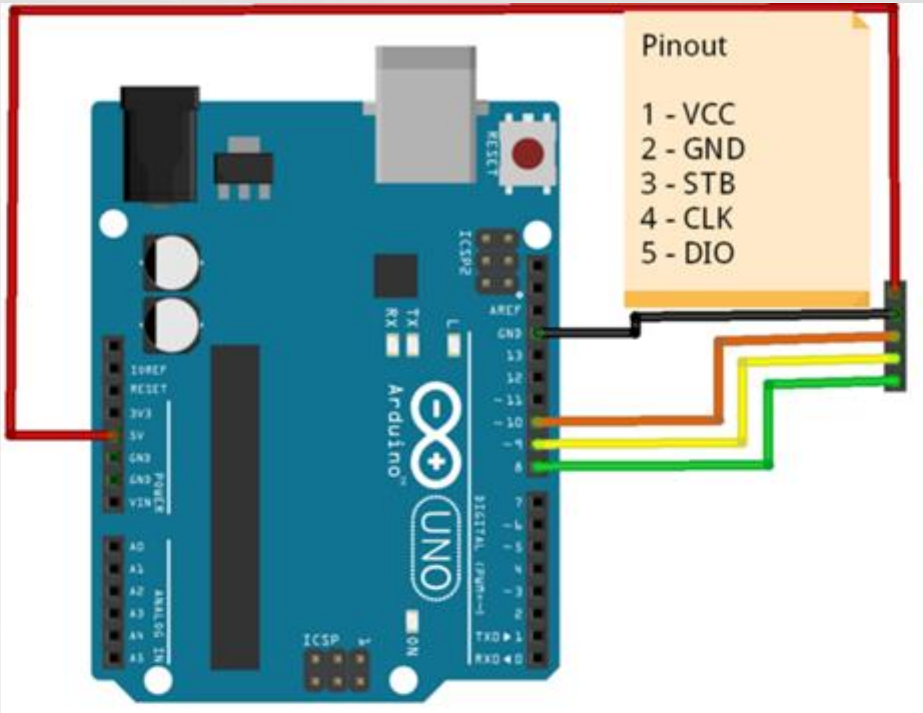
`0x44`
`(10000100)`
	`0xc? 0x??`	write value `0x??` at location `0xc?` (single address mode)

`0x40`
`(10000000)`
	`0xc? 0x?? 0x?? 0x??`	write values `0x??` starting from location `0xc?` (address auto increment mode)

`0x42`
`(10000010)`
	N/A	read buttons



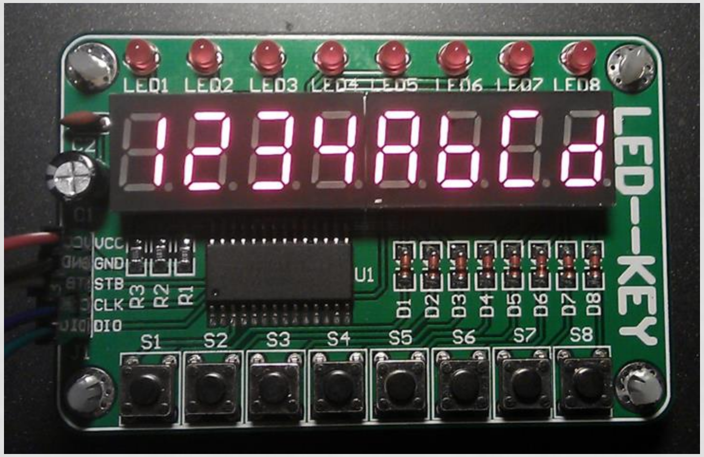
each digit contains 7 segments and a dot, coded as [DP]GFEDCBA

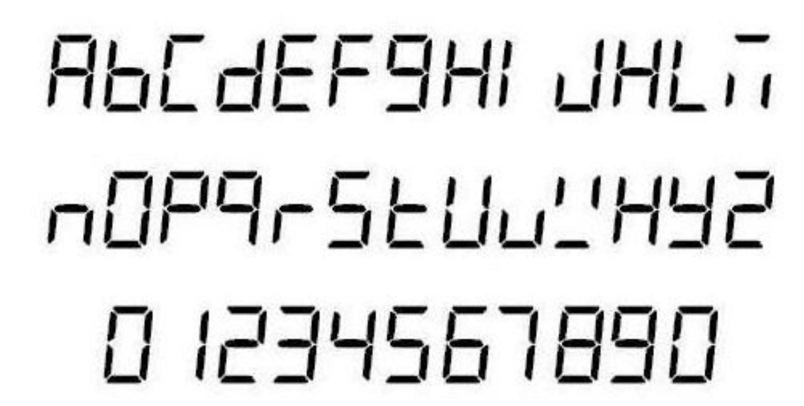


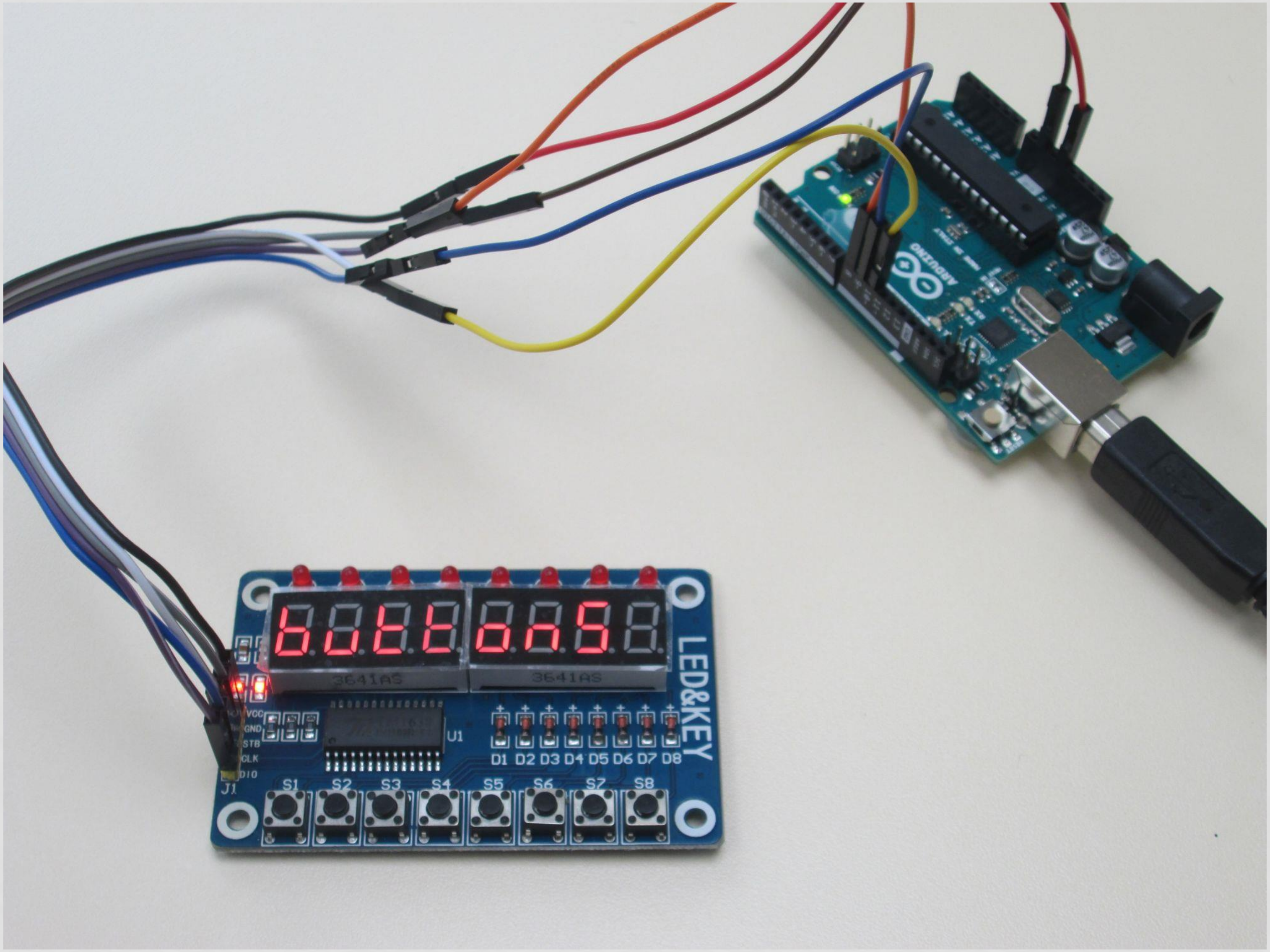
Pinout

1 - VCC
2 - GND
3 - STB
4 - CLK
5 - DIO

Vcc : +5V
GND : ground
DIO : data (=pin 8)
CLK : clock (=pin 9)
STB : strobe (=pin 10)



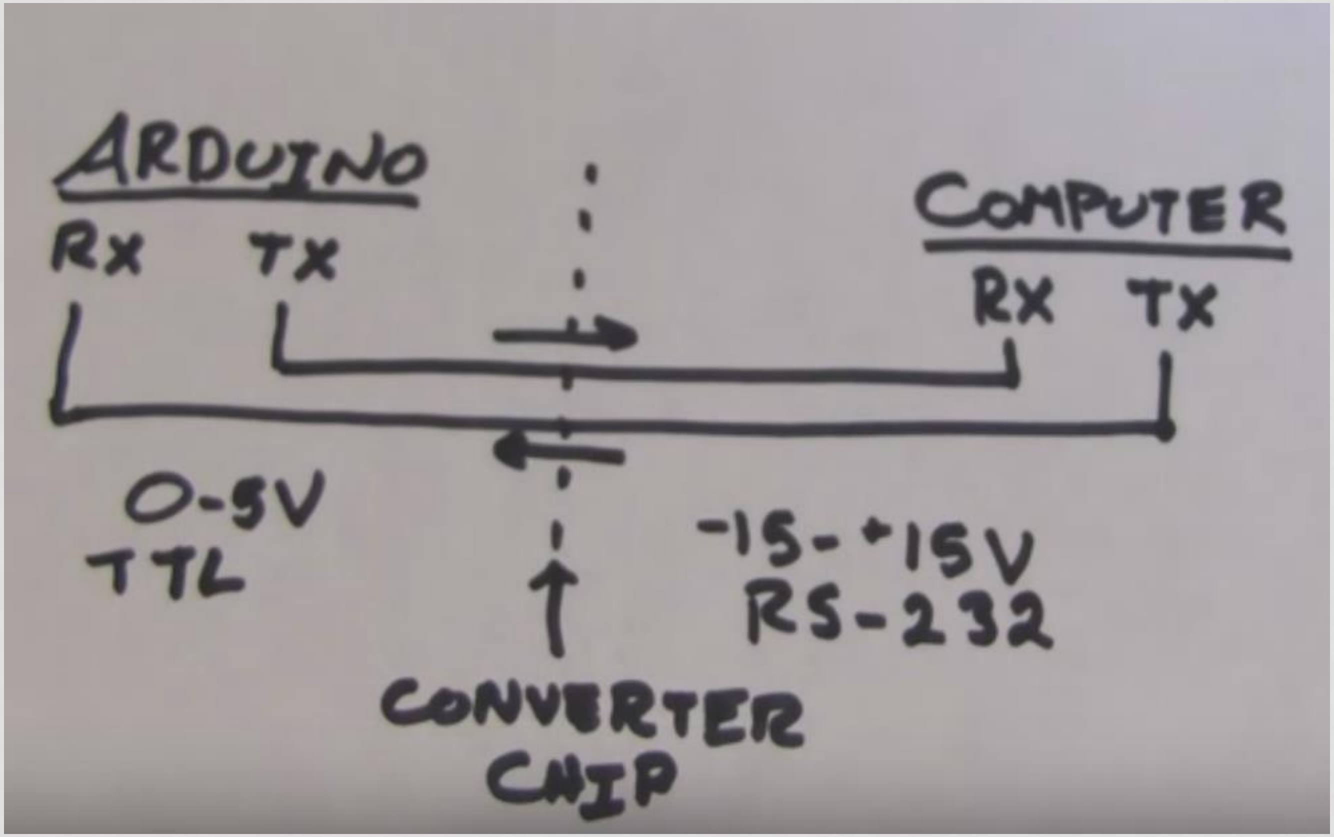



buttons
3641AS
3641AS
LED&KEY
VCC
GND
STB
CLK
DIO
J1
U1
D1 D2 D3 D4 D5 D6 D7 D8
S1 S2 S3 S4 S5 S6 S7 S8
ARDUINO

AGENDA

- interrupts in C
- het volatile keyword
- TM1638 led&key
- **UART**

USB/RS232


ARDUINO
RX
TX
COMPUTER
RX
TX
0-5V
TTL
-15-+15V
RS-232
CONVERTER
CHIP

WAAROM SERIEEL ?

- problemen bij parallelle I/O:
  - veel koperdraad
  - neemt veel ruimte
  - kostbaar
  - snelheid beperkt door beïnvloeding signaal (ruis en kruismodulatie)

RS-232

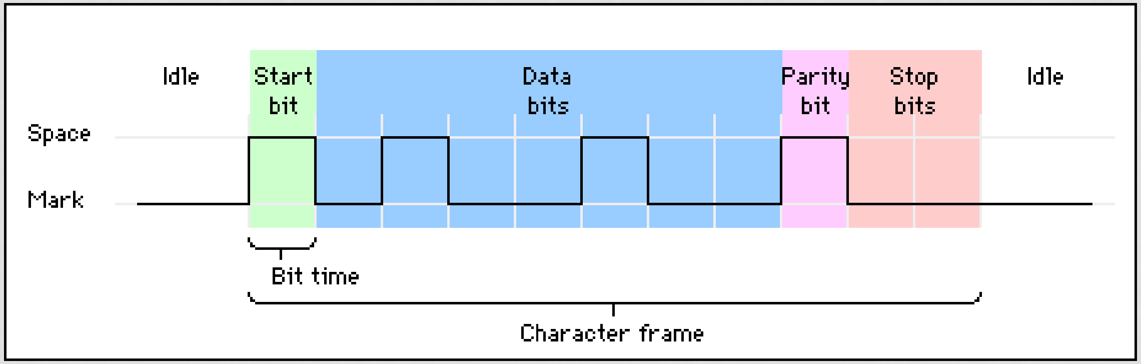
- in embedded wereld gebruikt voor :
  - communicatie met PC/laptop
  - program downloaden naar on-chip flash
- past in de fysieke laag
  - spannings niveaus, connector layout, informatie snelheid
- point-to-point
- asynchroon
  - betekent : ontvanger moet zijn eigen klok synchroniseren
  - hoe kan de ontvanger dit doen?

RS-232

- transmit en receive gescheiden circuits
  - full duplex is mogelijk
- baudrates : 300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 33600, 56000, 115000
- logisch 0 = + 12V en logisch 1 = -12 V
  - ATmega32 levert alleen +/- Vcc, ongeveer 5V; ATMEGA6-U2 zorgt voor conversie 5V - 12V

RS-232

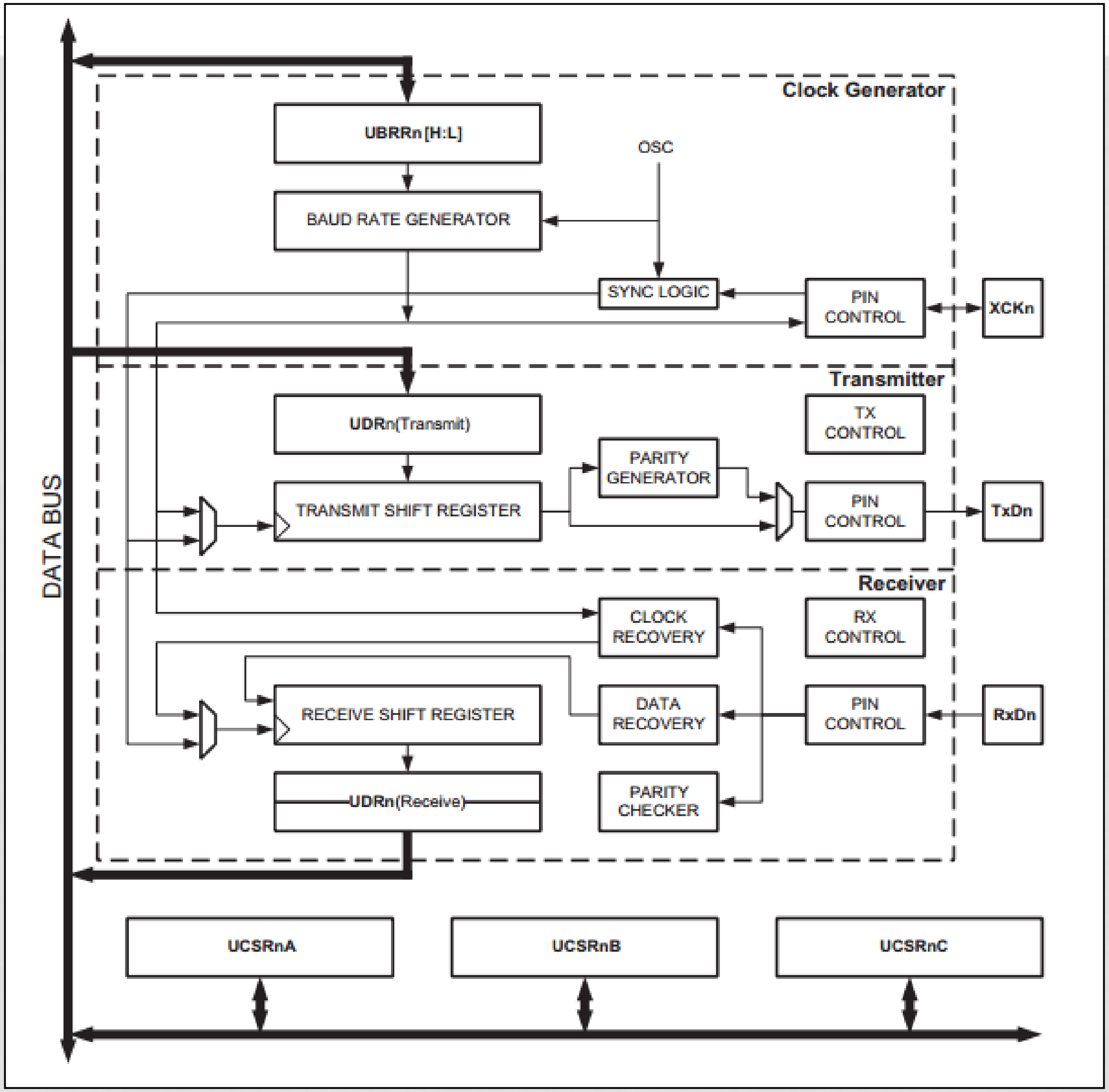
- RS-232 is karakter georiënteerd:
  - start bit
  - data bits (7 of 8) met eventueel parity bit
  - stop bits (1 of 2)
  - wel/geen parity bit



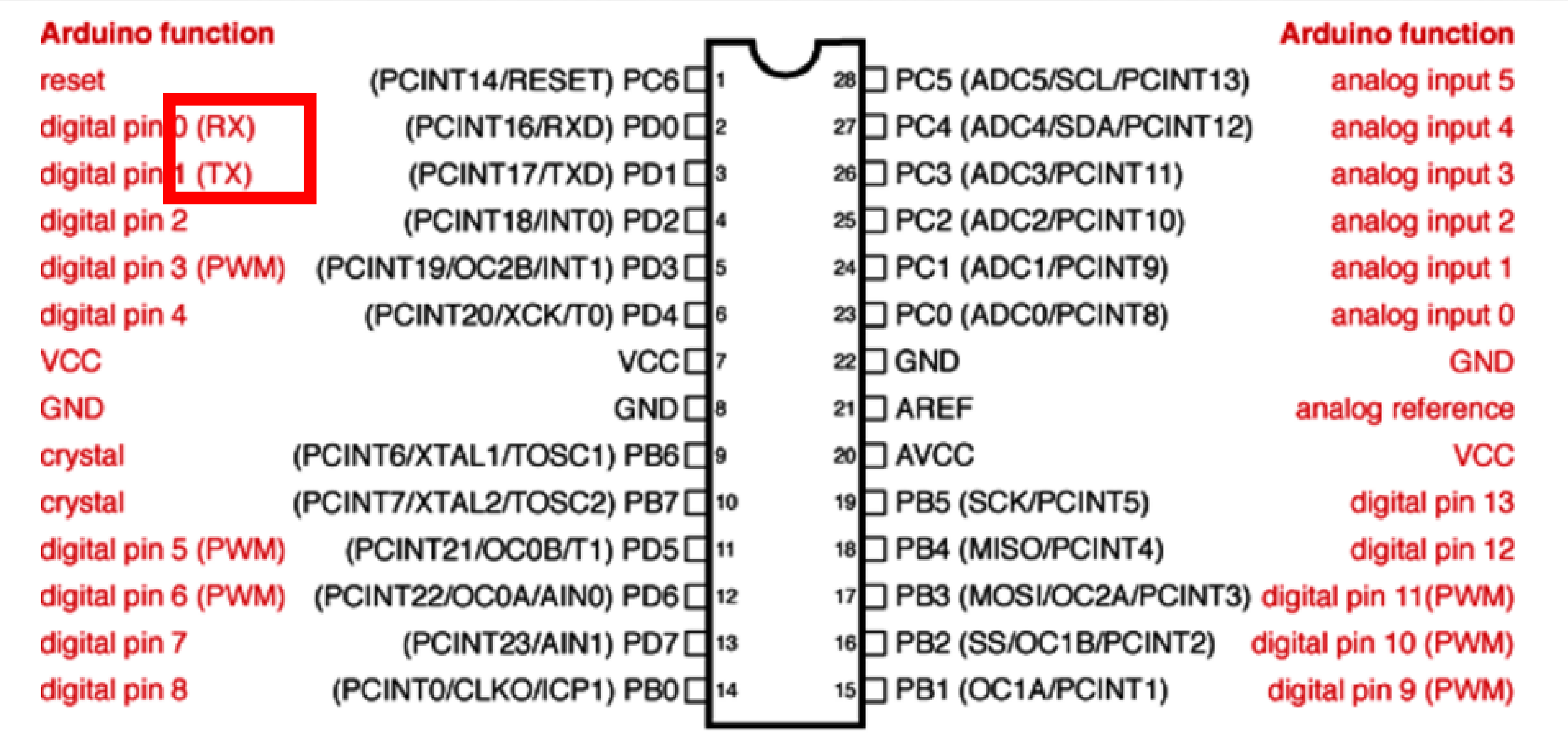
UART REGISTERS

- Universal Asynchronous Receiver Transmitter
  - USART : S = Synchronous
  - implementatie RS232 protocol
- UDR : data register
  - eigenlijk 2 registers : afhankelijk van lezen of schrijven wordt juiste register automatisch gekozen
  - lezen of schrijven o.b.v. polling of interrupt
- UBRR L+H : baud register
- UCSR A, B en C : control en status (frame format)

UART ATMEGA328P



TX & RX



Arduino function					Arduino function
reset	(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)	analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)	analog input 0
VCC	VCC	7	22	GND	GND
GND	GND	8	21	AREF	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)	digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)	digital pin 9 (PWM)

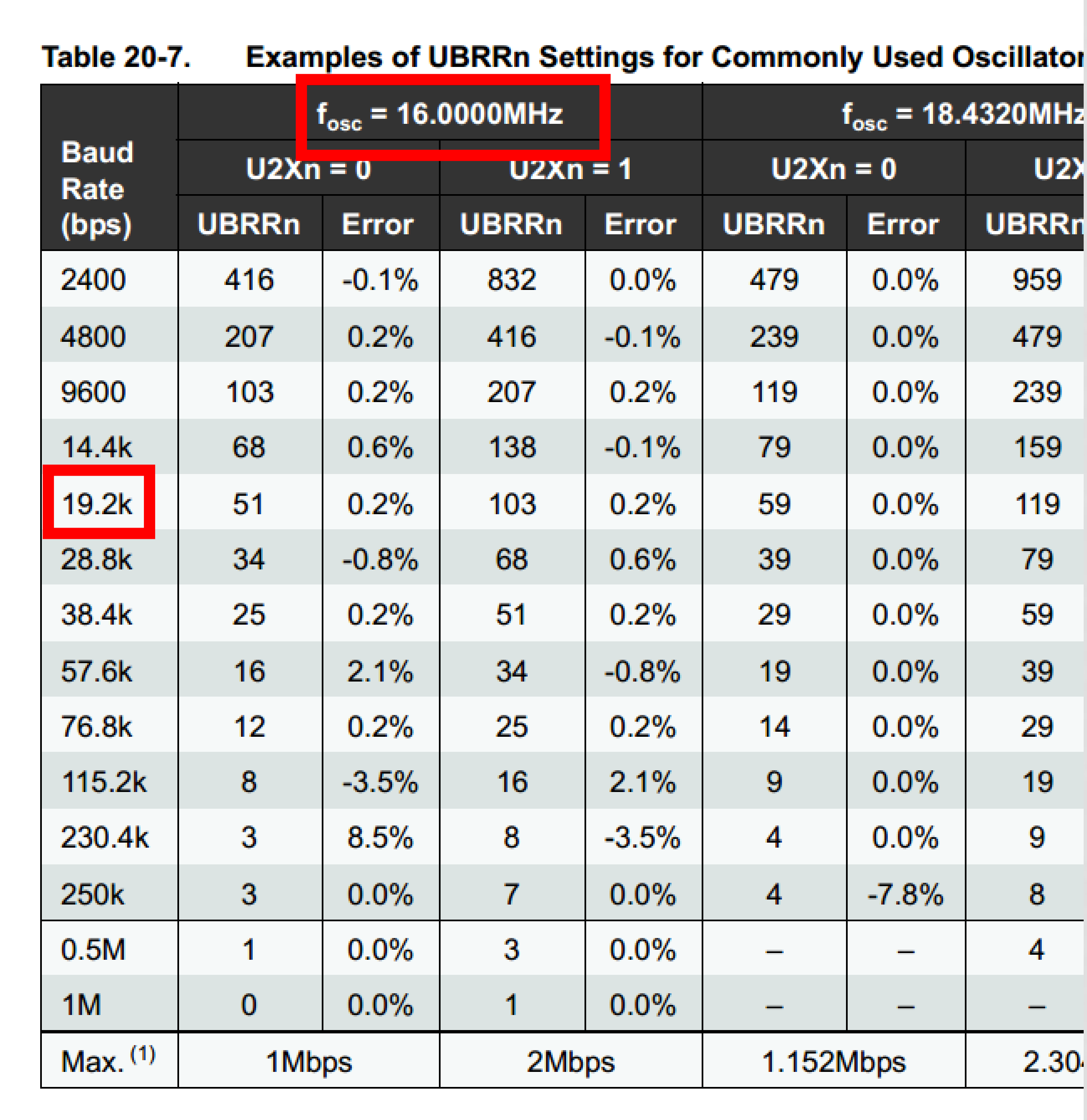


Table 20-7. Examples of UBRRn Settings for Commonly Used Oscillator

Baud Rate (bps)	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz		
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2X
	UBRRn	Error	UBRRn	Error	UBRRn	Error	UBRRn
2400	416	-0.1%	832	0.0%	479	0.0%	959
4800	207	0.2%	416	-0.1%	239	0.0%	479
9600	103	0.2%	207	0.2%	119	0.0%	239
14.4k	68	0.6%	138	-0.1%	79	0.0%	159
19.2k	51	0.2%	103	0.2%	59	0.0%	119
28.8k	34	-0.8%	68	0.6%	39	0.0%	79
38.4k	25	0.2%	51	0.2%	29	0.0%	59
57.6k	16	2.1%	34	-0.8%	19	0.0%	39
76.8k	12	0.2%	25	0.2%	14	0.0%	29
115.2k	8	-3.5%	16	2.1%	9	0.0%	19
230.4k	3	8.5%	8	-3.5%	4	0.0%	9
250k	3	0.0%	7	0.0%	4	-7.8%	8
0.5M	1	0.0%	3	0.0%	–	–	4
1M	0	0.0%	1	0.0%	–	–	–
Max. (1)	1Mbps		2Mbps		1.152Mbps		2.30

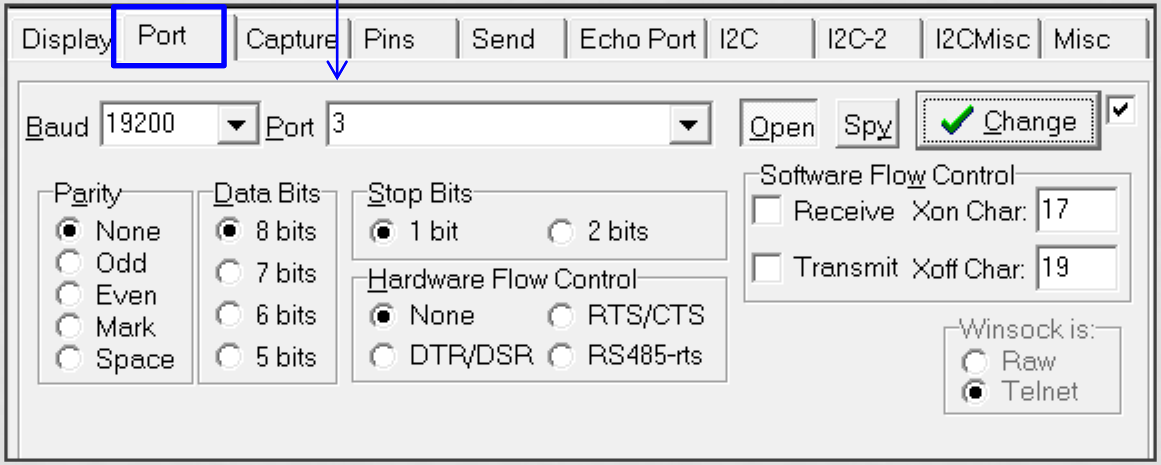
U2Xn : double speed operation

```
#include <avr/io.h>
#include <stdlib.h>
#include <avr/sfr_defs.h>
#define F_CPU 16E6
#include <util/delay.h>

// output on USB = PD1 = board pin 1
// datasheet p.190; F_OSC = 16 MHz & baud rate = 19.200
#define UBBRVAL 51

void uart_init()
{
    // set the baud rate
    UBRR0H = 0;
    UBRR0L = UBBRVAL;
    // disable U2X mode
    UCSR0A = 0;
    // enable transmitter
    UCSR0B = _BV(TXEN0);
    // set frame format : asynchronous, 8 data bits, 1 stop bit, no parity
    UCSR0C = _BV(UCSZ01) | _BV(UCSZ00);
}

void transmit(uint8_t data)
{
    // wait for an empty transmit buffer
    // UDRE is set when the transmit buffer is empty
    loop_until_bit_is_set(UCSR0A, UDRE0);
    // send the data
    UDR0 = data;
}
```

```
int main(void)
{
    uart_init();
    _delay_ms(1000);
    while (1) {
        transmit(0x33); _delay_ms(1000);
        transmit(0x77); _delay_ms(1000);
        transmit(0xbb); _delay_ms(1000);
    }
}
```

RS232-terminal "Realterm"
http://realterm.sourceforge.net/

COM3



Baud 19200 Port 3 ☑

Parity: ● None ○ Odd ○ Even ○ Mark ○ Space

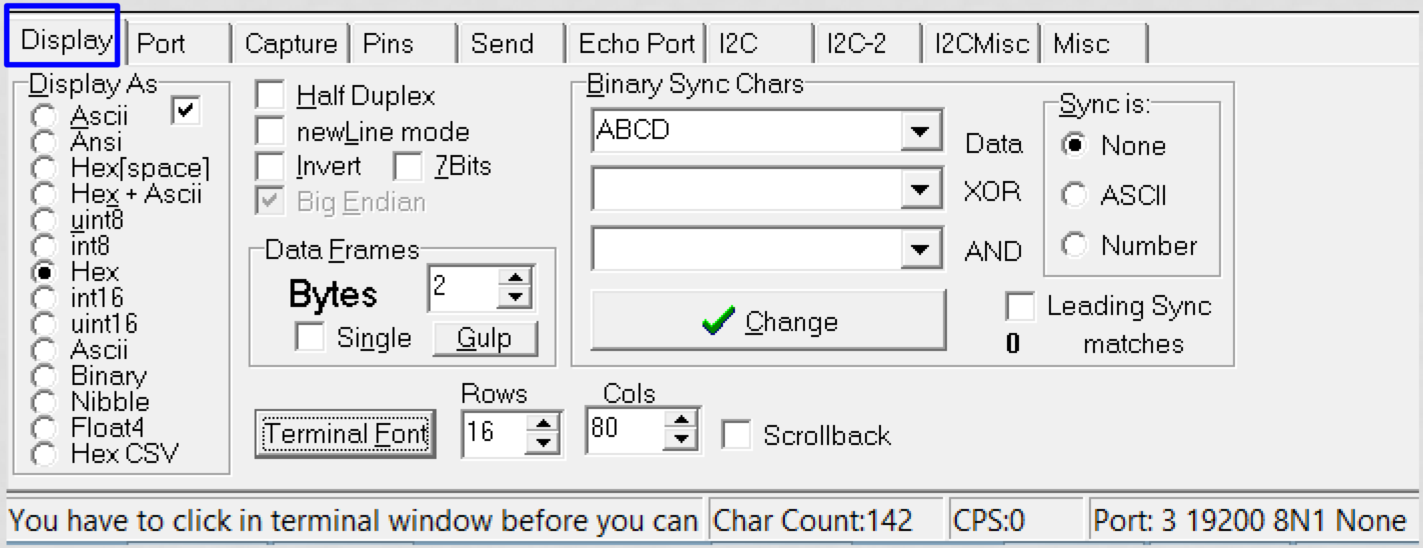
Data Bits: ● 8 bits ○ 7 bits ○ 6 bits ○ 5 bits

Stop Bits: ● 1 bit ○ 2 bits

Hardware Flow Control: ● None ○ RTS/CTS ○ DTR/DSR ○ RS485-rts

Software Flow Control: □ Receive Xon Char: 17 □ Transmit Xoff Char: 19

Winsock is: ○ Raw ● Telnet



Display As ☑
○ Ascii
○ Ansi
○ Hex[space]
○ Hex + Ascii
○ uint8
○ int8
● Hex
○ int16
○ uint16
○ Ascii
○ Binary
○ Nibble
○ Float4
○ Hex CSV

□ Half Duplex
□ newLine mode
□ Invert □ 7Bits
☑ Big Endian

Data Frames: Bytes 2 □ Single

Binary Sync Chars: ABCD Data, XOR, AND

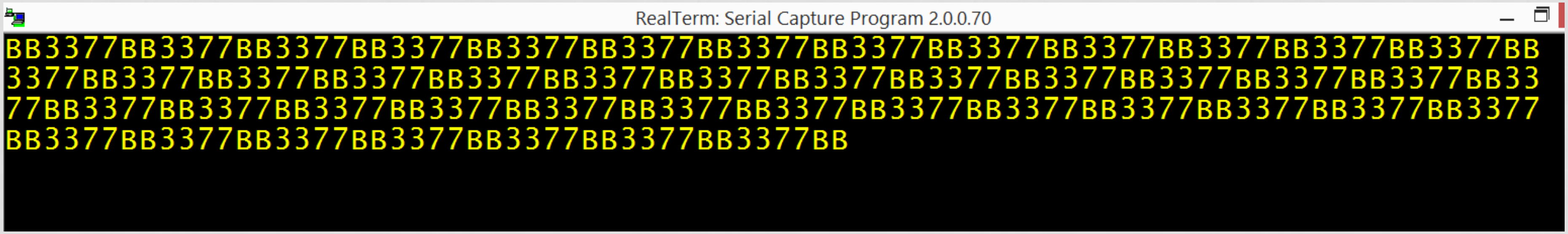
Sync is: ● None ○ ASCII ○ Number

□ Leading Sync
0 matches

Rows 16 Cols 80 □ Scrollback

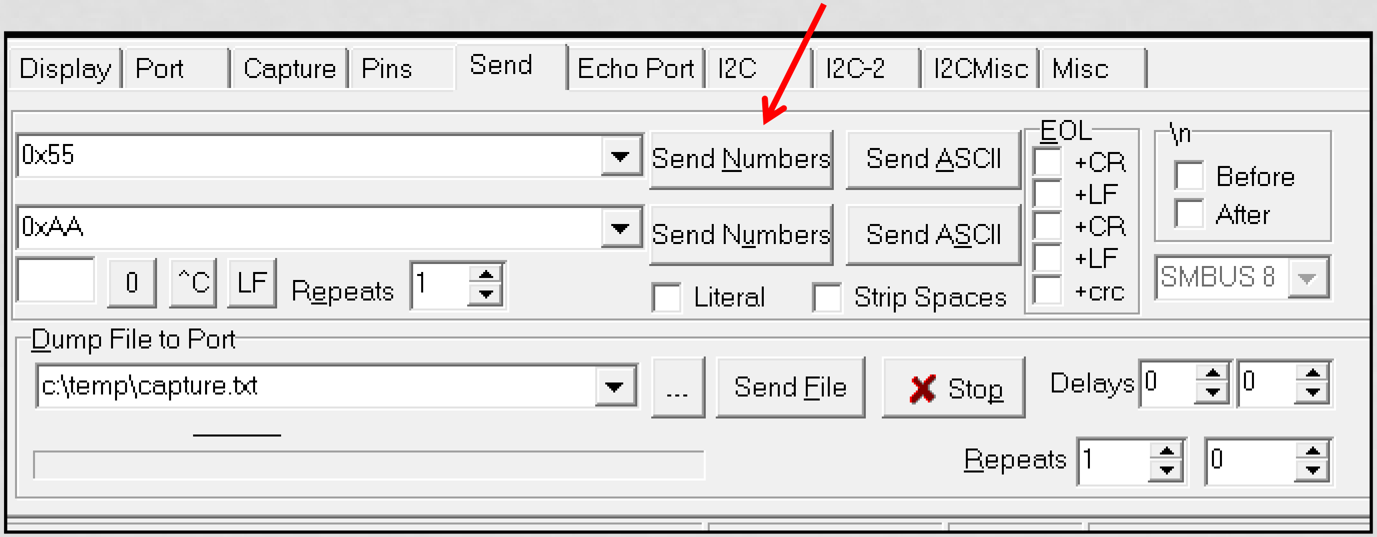
You have to click in terminal window before you can | Char Count:142 | CPS:0 | Port: 3 19200 8N1 None

DISPLAY OUTPUT



```
BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB
3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB33
77BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB3377
BB3377BB3377BB3377BB3377BB3377BB3377BB3377BB
```

You can enter a string of hex or decimal numbers in the Send box e.g. "51 0x31 $32" and press "*Send Numbers*".
(Note that they must be separated by spaces)



zend knop

0x55

0xAA

Repeats 1

Dump File to Port: c:\temp\capture.txt

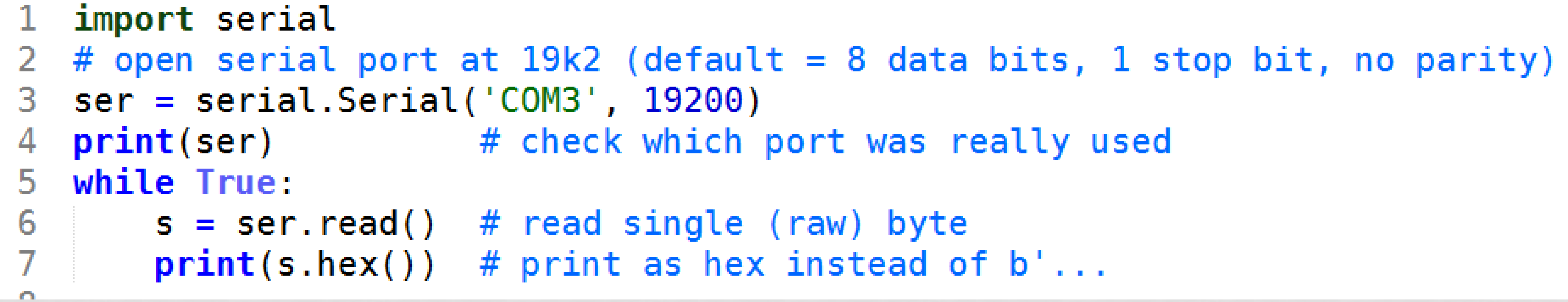
Delays 0 0

Repeats 1 0

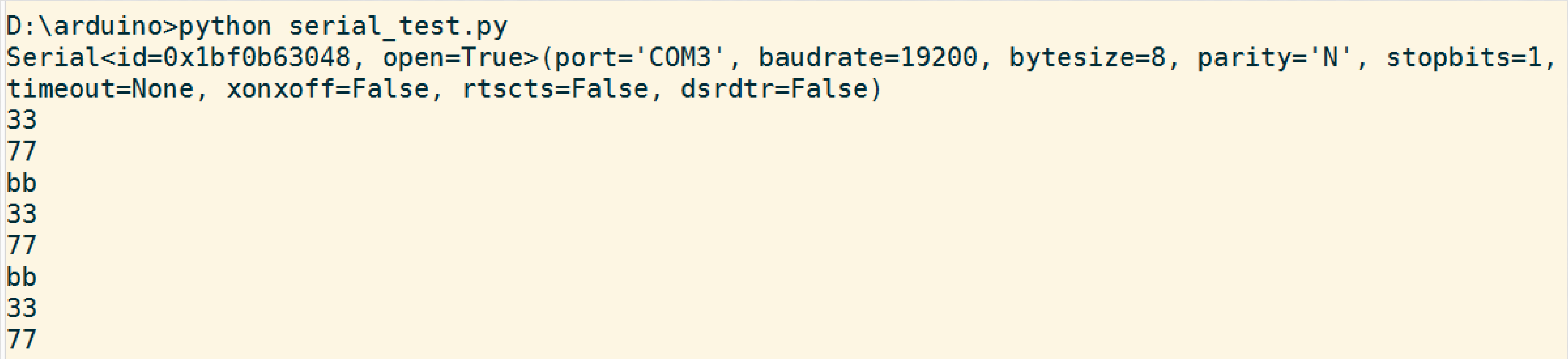
PROBLEMS ?

- port in use : Atmel studio, Realterm, avrdude : only *one* can use the port at any point in time
- Realterm : be sure to set port settings correct
  - frame
  - baud rate

OR WITH PYTHON



```python
import serial
# open serial port at 19k2 (default = 8 data bits, 1 stop bit, no parity)
ser = serial.Serial('COM3', 19200)
print(ser)          # check which port was really used
while True:
    s = ser.read()  # read single (raw) byte
    print(s.hex())  # print as hex instead of b'...
```



```
D:\arduino>python serial_test.py
Serial<id=0x1bf0b63048, open=True>(port='COM3', baudrate=19200, bytesize=8, parity='N', stopbits=1,
timeout=None, xonxoff=False, rtscts=False, dsrdtr=False)
33
77
bb
33
77
bb
33
77
```