2-wire Serial Interface

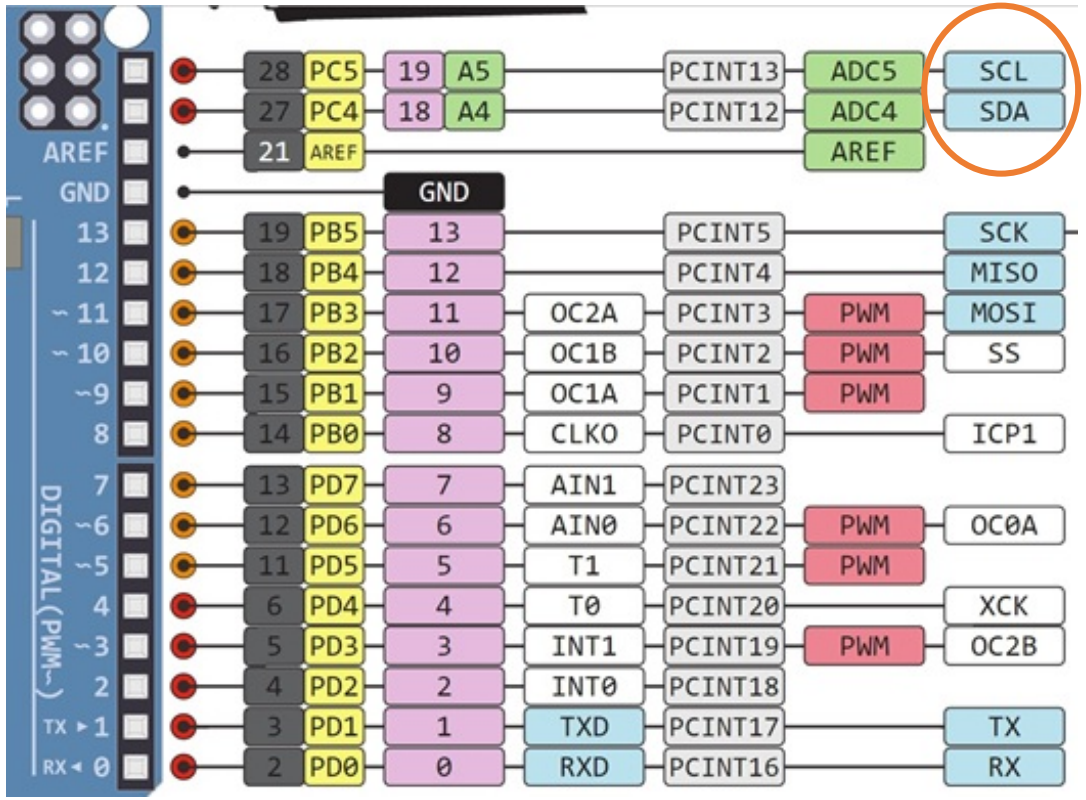
ATmega328P - kompatibilní s Phillips I2C protokolem

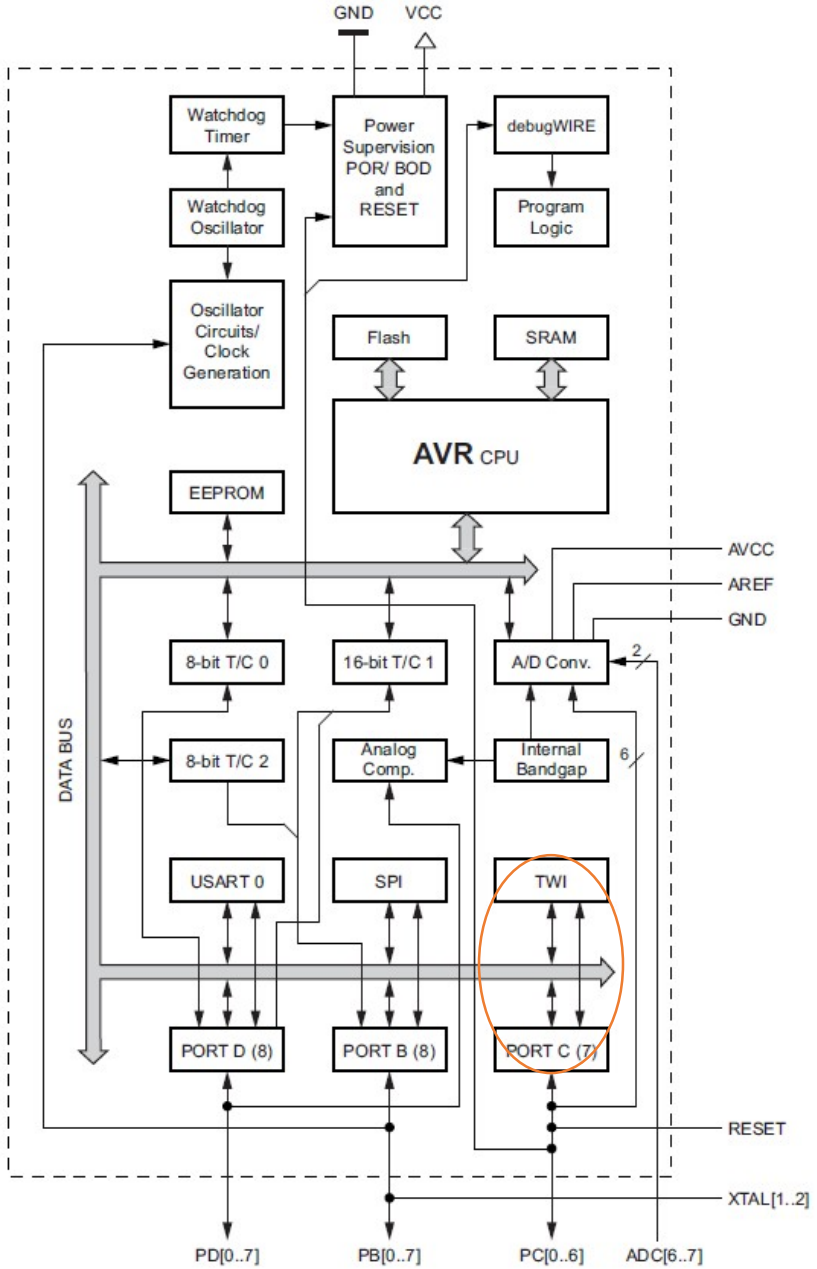
2-wire serial interface - TWI

- Dvouvodičové sériové rozhraní **TWI** je **kompatibilní** s Phillips **I2C.**
- Ideální pro typické aplikace mikrokontrolérů.
- Umožňuje propojit **až 128 různých zařízení**
- Propojení pomocí dvou obousměrných sběrnic - pro **hodiny (SCL)** a pro **data (SDA**).
- **Externí pull-up rezistor** pro každou linku sběrnice (výstup s otevřeným kolektorem).
- **Zařízení připojená ke sběrnici mají** jednotlivé **adresy** a mechanismy pro řízení sběrnice - protokol TWI.
- PRTWI bit v Power Reduction Registru (PRR) musí být vynulován

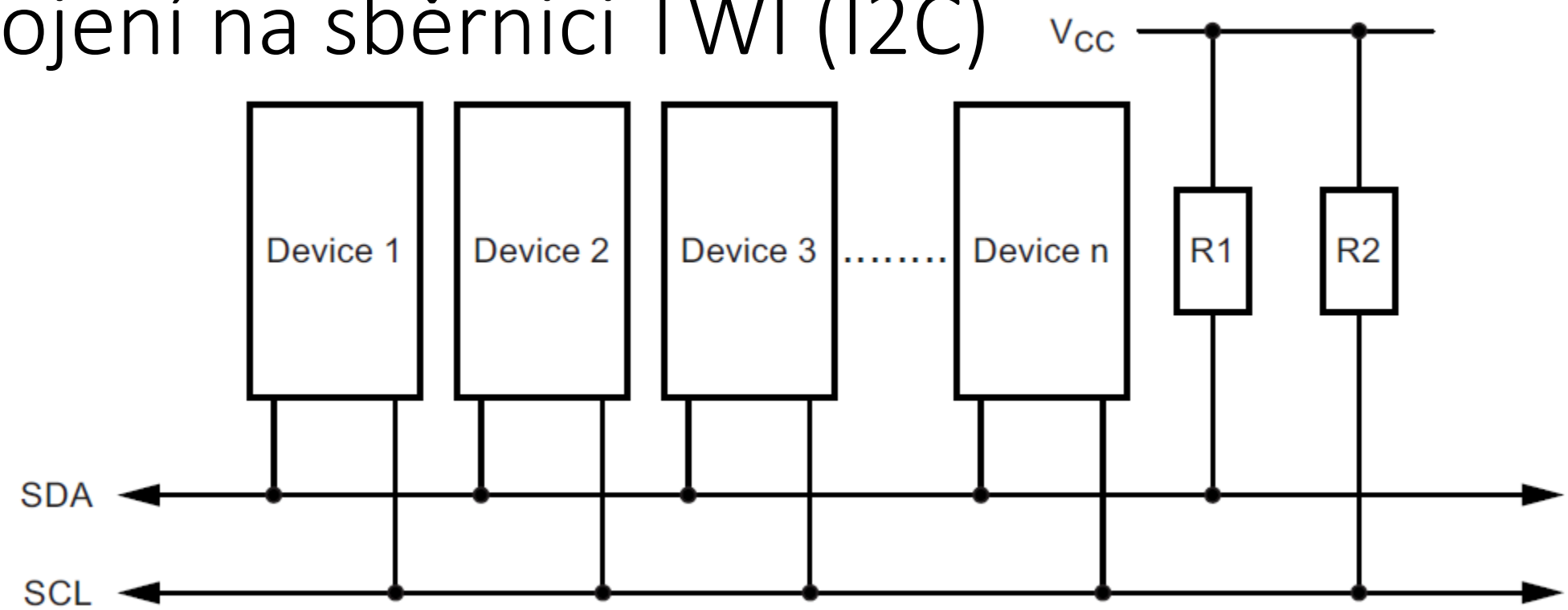
TWI – ATmega328P

- Samostatný hardware pro TWI
  (společné piny portu C - PC4 a PC5 pro analog)





Propojení na sběrnici TWI (I2C)

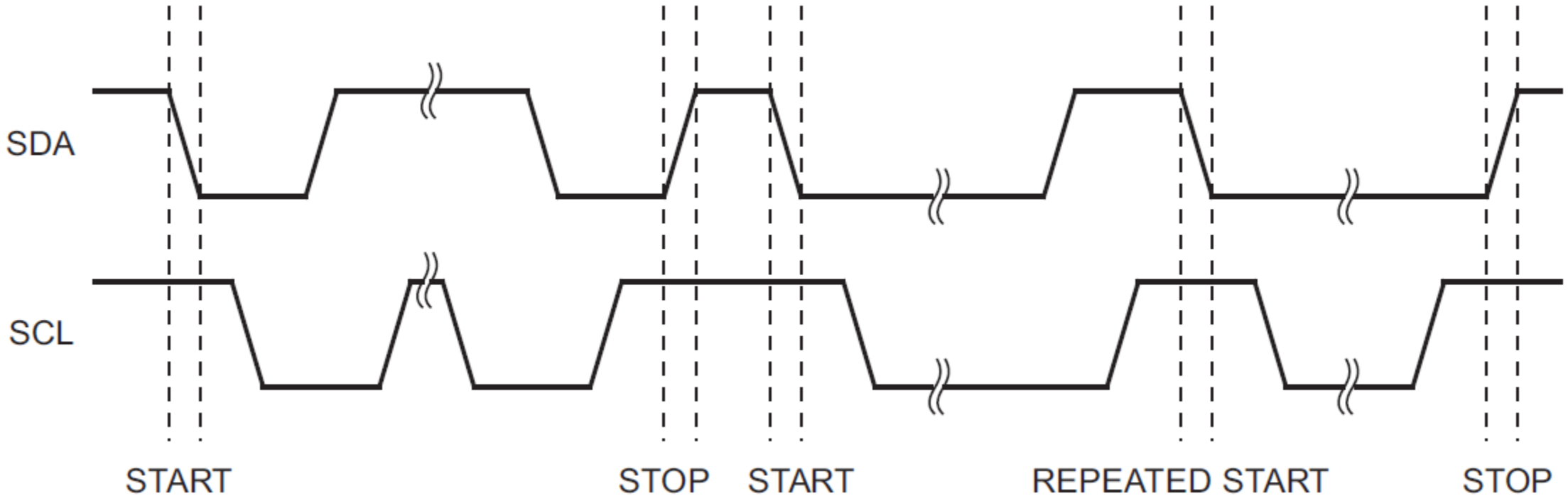


- Budiče sběrnic jsou s otevřeným kolektorem (montážní AND, kde alespoň jedno zařízení vyšle LL – sběrnice je ve stavu LL, aby byla sběrnice ve stavu HL musí všechna zařízení být v odpojeném stavu)
- Parazitní kapacita sběrnice (max. 400 pF)ovlivňuje její max. rychlost (podle specifikace 200 kHz nebo 400 kHz)

Formát a přenos dat TWI

- **Master** - iniciuje a ukončí přenos, generuje hodiny SCL.
- **Slave** – zařízení adresované masterem.
- Vysílač - **Transmitter** umisťuje data na sběrnici.
- Přijímač - **Receiver** čte data ze sběrnice.
- Přenos je zahájen, když master vydá stav **START** na sběrnici a je ukončen, když vydá stav **STOP**.
- Mezi stavem START a STOP je sběrnice považována za zaneprázdněnou.
- OPAKOVANÝ START se používá, když daný master zahájí nový přenos, aniž by se vzdal kontroly nad sběrnicí (vydal stav STOP).
- Stavy **START a STOP** jsou **signalizovány změnou úrovně** signálu **SDA, když** má signál **SCL úroveň HL**.

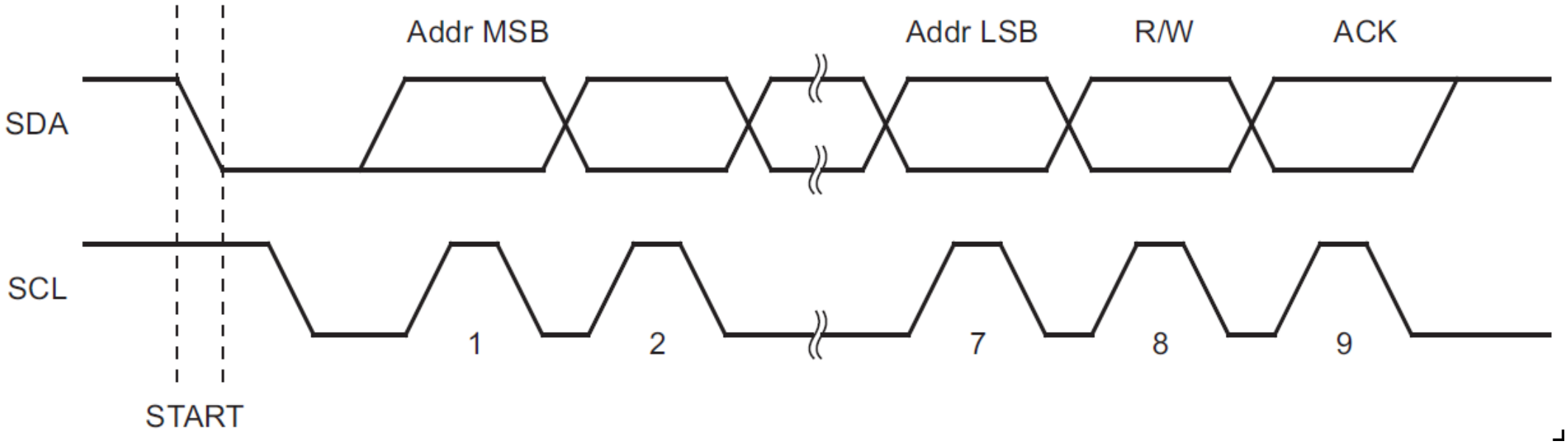
Podmínka START, REPEATED START a STOP



Formát adresového paketu

- Všechny **adresové pakety** přenášené na sběrnici TWI mají délku 9 bitů, skládající se ze **7 adresních bitů**, **jednoho řídicího bitu READ / WRITE** a **potvrzovacího bitu ACK**.
- Adresovaný slave potvrdí příjem paketu v devátém cyklu SCL (ACK) nastavením SDA na HL.
- Adresový paket se skládá ze slave adresy a bitu READ nebo WRITE a nazývá se SLA + R, respektive SLA + W.
- Nejprve se vysílá MSB adresového bajtu.
- Adresa 0000 000 je broadcast - všichni slave vyšlou LL na linku SDA v cyklu ACK. (Používá se pouze pro zápis do slave)
- Adresy 1111 xxx jsou vyhrazeny pro budoucí účely

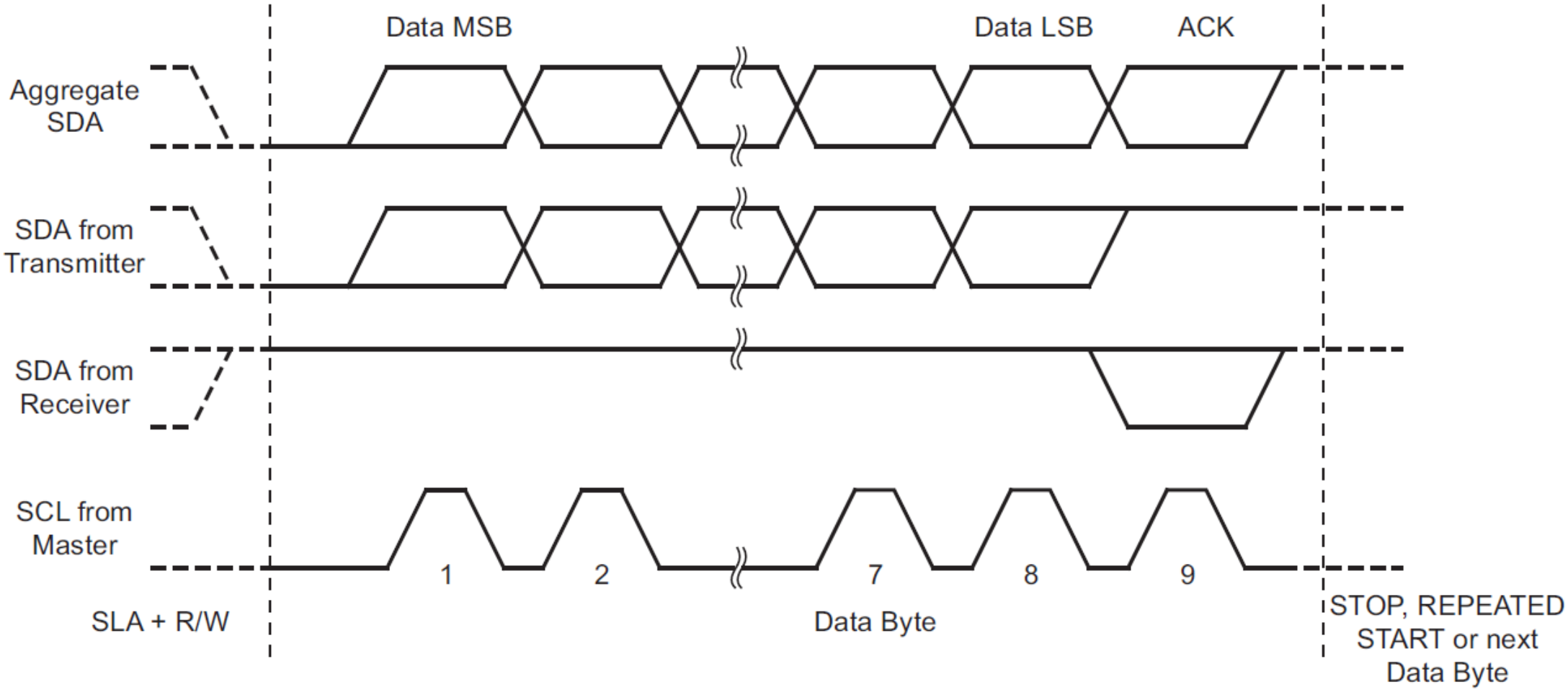
Formát adresového paketu



Formát datového paketu

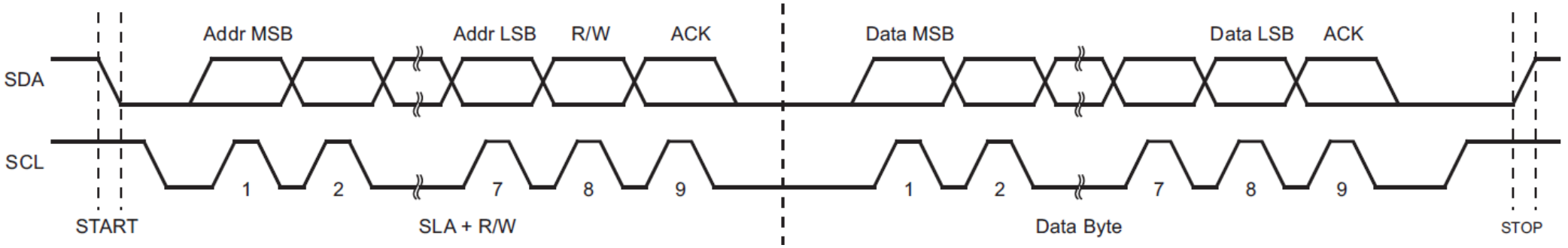
- Všechny datové pakety přenášené na sběrnici TWI jsou dlouhé devět bitů: 1 datový bajt a potvrzení ACK (receiver vystaví LL na SDA) nebo NACK (signalizuje, že přijímač opustí linku nastavením SDA na HL)
- Zasláním NACK přijímač informuje vysílač o příjmu posledního bajtu.
- Nejprve se vysílá MSB datového bajtu.

Formát datového paketu



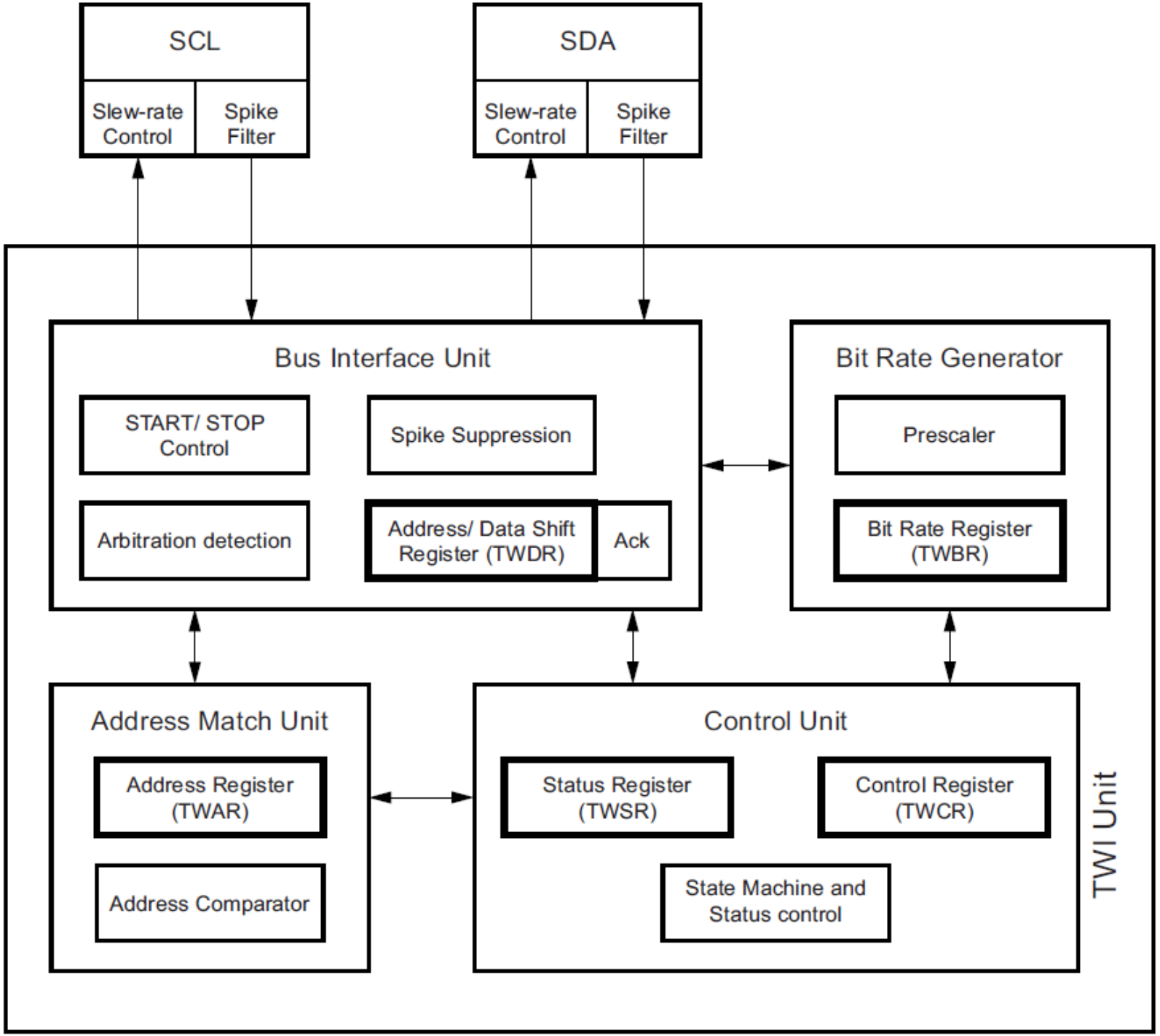
Přenos dat mezi master a adresovaným slave

- Přenos se skládá z podmínky START, SLA + R / W, jednoho nebo více datových paketů (BH1750 pouze jednoho) a podmínky STOP.
- Prázdná zpráva (START – STOP) je nelegální.
- Montážní AND umožňuje prodloužit LL úroveň SCL v případě, že je rychlost hodin nastavená masterem pro slave příliš vysoká nebo pokud slave potřebuje více času na zpracování mezi datovými přenosy. (sníží se rychlost přenosu)



Blokové schéma TWI modulu

- Slew-rate control: řízení směru přenosu
- Spike filter: filtr nežádoucích špiček signálu
- Spike suppression: potlačení nežádoucích špiček signálu
- Arbitration detection: detekce rozhodování (podle protokolu TWI)



Jednotka generátoru přenosové rychlosti

- Řídí periodu SCL při provozu v režimu Master nastavením registru bitové rychlosti TWBR a bity předděliče TWPS1:0 ve stavovém registru TWSR.
- Slave provoz nezávisí na bitové rychlosti nebo nastavení předděliče .
- Taktovací frekvence CPU v zařízení slave musí být alespoň 16x vyšší než frekvence SCL.

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \times (\text{PrescalerValue})}$$

Pro TWPS1= 0 and TWPS0 = 0 (PrescalerValue = 1) je hodnotaTWBR funkcí CPU Clock SCL frekvence:

$$\mathbf{TWBR = ((CPU\ Clock\ frequency / SCL\ frequency) - 16) / 2}$$

TWBR – TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0
(0xB8)	**TWBR7**	**TWBR6**	**TWBR5**	**TWBR4**	**TWBR3**	**TWBR2**	**TWBR1**	**TWBR0**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TWDR – TWI Data Register

Bit	7	6	5	4	3	2	1	0
(0xBB)	**TWD7**	**TWD6**	**TWD5**	**TWD4**	**TWD3**	**TWD2**	**TWD1**	**TWD0**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	1

TWCR – TWI Control Register

- povolení TWI rozhraní
- generování stavu START - zahájení přístupu aplikací na sběrnici
- generování potvrzení přijímačem ACK / NACK
- generování stavu zastavení STOP
- indikace kolize zápisu, pokud dojde k pokusu o zápis dat do TWDR, zatímco registr je nepřístupný.

TWCR – TWI Control Register

- Bit 7 – TWINT: TWI Interrupt Flag **nastavuje hardware když TWI dokončí aktuální úlohu**. Nastavení příznaku TWINT prodlouží úroveň LL signálu SCL. Vymazání TWINT softwarem zahájí provoz TWI. Operace s registry TWAR, TWSR a TWDR musí být před vymazáním tohoto příznaku dokončeny.
- Bit 6 – TWEA: TWI Enable Acknowledge Bit
- Bit 5 – TWSTA: TWI START Condition Bit
- Bit 4 – TWSTO: TWI STOP Condition Bit
- Bit 3 – TWWC: TWI Write Collision Flag
- Bit 2 – TWEN: TWI Enable Bit
- Bit 0 – TWIE: TWI Interrupt Enable

Bit	7	6	5	4	3	2	1	0
(0xBC)	**TWINT**	**TWEA**	**TWSTA**	**TWSTO**	**TWWC**	**TWEN**	**–**	**TWIE**
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W
Initial Value	0	0	0	0	0	0	0	0

TWSR – TWI Status Register

- Bits 7..3 – TWS: TWI Status (podle tabulky stavových kódů)
- Bits 1..0 – TWPS: TWI Prescaler Bits

Bit	7	6	5	4	3	2	1	0
(0xB9)	**TWS7**	**TWS6**	**TWS5**	**TWS4**	**TWS3**	**–**	**TWPS1**	**TWPS0**
Read/Write	R	R	R	R	R	R	R/W	R/W
Initial Value	1	1	1	1	1	0	0	0

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

TWAR – TWI (Slave) Address Register

- Bits 7..1 – TWA: TWI (Slave) Address Register
- Bit 0 – TWGCE: TWI General Call Recognition Enable Bit (adresa 0x00)

Bit	7	6	5	4	3	2	1	0
(0xBA)	**TWA6**	**TWA5**	**TWA4**	**TWA3**	**TWA2**	**TWA1**	**TWA0**	**TWGCE**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	1	1	1	1	1	1	0

TWAMR – TWI (Slave) Address Mask Register

- Každý z bitů v TWAMR může maskovat (deaktivovat) odpovídající adresní bity v registru adres TWI (TWAR).
- Nastavením bitu masky logika shody adresy ignoruje porovnání mezi bitem příchozí adresy a odpovídajícím bitem v TWAR.

Bit	7	6	5	4	3	2	1	0
(0xBD)	TWAM[6:0]							–
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Initial Value	0	0	0	0	0	0	0	0

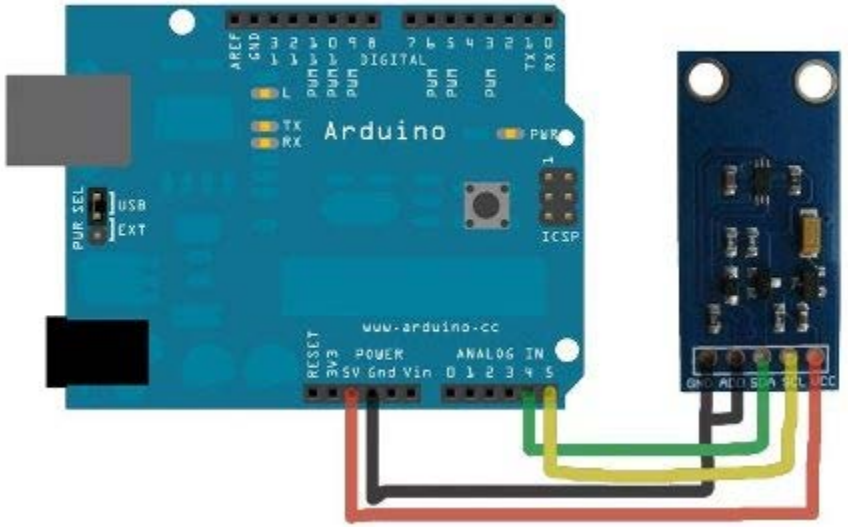
Nastavení rychlosti přenosu I2C rozhraní

```
;setup speed of i2c port:
        ldi     temp, BitRate
        sts     TWBR, temp      ; bitrate I2C
        ldi     temp, 0x00
        sts     TWSR, temp      ; preescaller = 1
```

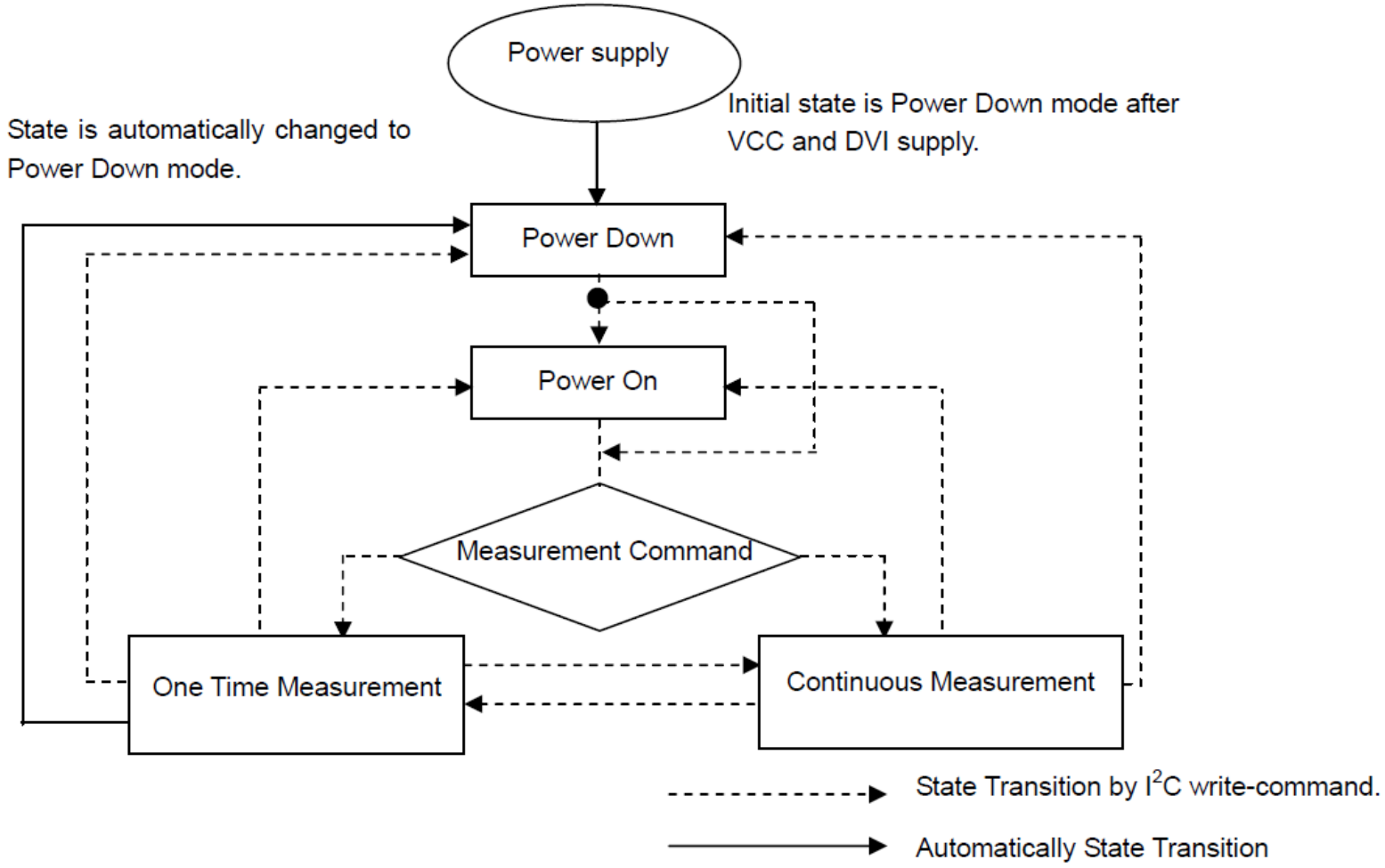
Příklad komunikace I2C se senzorem světla

- senzor světla BH1750
- Adresový pin senzoru volí adresu senzoru LL: 0x23, HL: 0x5C (na obrázku černý signál LL)
- Piny A4 a A5 jsou propojeny s piny SDA, SCL a odpovídají pinům PC4 a PC5 portu C

```
;setup DDR/IO: set pullups, DDRC output
ldi temp, (1<<PC4)|(1<<PC5)
out DDRC,temp
out PORTC,temp
```



Proces měření BH1750

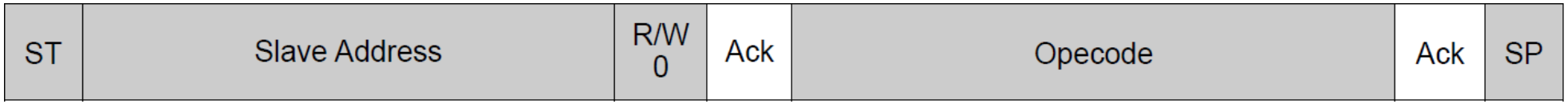


Operační kódy BH1750

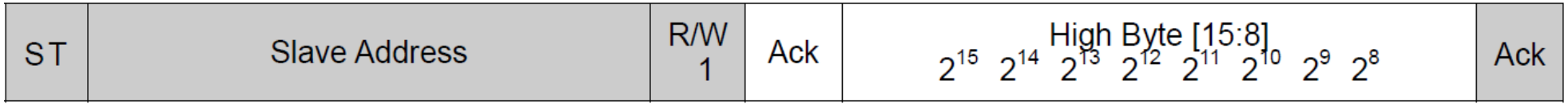
Instruction	Opecode	Comments
Power Down	0000_0000	No active state.
Power On	0000_0001	Waiting for measurement command.
Continuously H-Resolution Mode	0001_0000	Start measurement at 1lx resolution. Measurement Time is typically 120ms.
Continuously L-Resolution Mode	0001_0011	Start measurement at 4lx resolution. Measurement Time is typically 16ms.

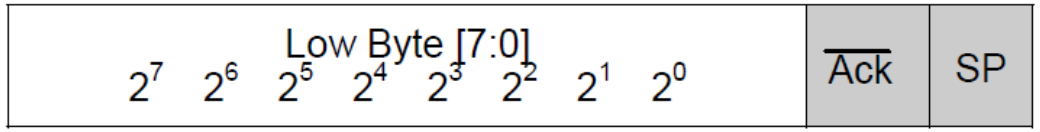
Formát komunikace na Slave BH1750

- BH1750 neakceptuje vícebajtový Opecode bez Stop podmínky
- Write formát:

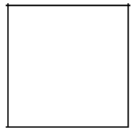


- Read formát:





from Master to Slave



from Slave to Master

Příklad dat od Slave BH1750:

Vyšší a nižší byte: 1000 0011 1001 0000_B = 33680_D , hodnota v luxech: 33680/1,2 = 28066,6 Lx

Definice proměnných a konstant

```
; Defines
.def    temp =r16               ;worker register
.def    data =r17
.def            resultL =r18
.def            resultH =r19

; Equate statements
.equ            F_CPU   = 16000000      ; CPU clk (16 MHz)
.equ            F_SCL   = 100000        ; SCL frequency (100 KHz)
.equ            BitRate = ((F_CPU / F_SCL)-16) / 2 ; TWI Bit Rate Register
.equ            SLA     = 0x5C; Slave Address BH1750 (0x5C ADDR pin = HL)(0x23 ADDR pin = LL)
.equ            W       = 0             ; Write
.equ            R       = 1             ; Read
.equ            START   = 0x08                          ; Start Condition
.equ            PowerOn         = 0x01                  ; Power On - Opcode BH1750
.equ            ContHRMode      = 0x10                  ; Continuously H-Resolution Mode
```

Setup BH1750 PowerOn

```
call    I2CSTART                ;start condition
ldi     data, SLA
lsl     data                    ;shift left LSA - adresa slave
ori     data, W                 ;add W (SLA+W)
call    I2CPUT                  ;transmit SLA-W
ldi     data, PowerOn           ;PowerOn : 0x01
call    I2CPUT                  ;transmit PowerOn
call    I2CSTOP                 ;stop condition
```

Setup BH1750 Continuously H-Resolution

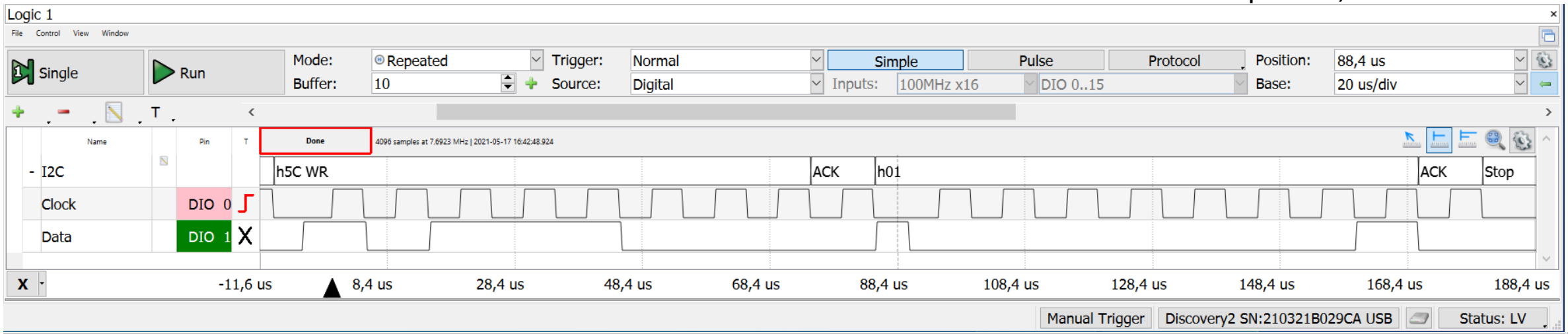
```
call    I2CSTART                ;start condition
ldi     data, SLA
lsl     data                    ;shift left LSA
ori     data, W                 ;add W (SLA+W)
call    I2CPUT                  ;transmit SLA-W
ldi     data, ContHRMode        ;Continuously H-Resolution Mode
call    I2CPUT                  ;transmit Continuously H-Resolution Mode
call    I2CSTOP                 ;stop condition
```

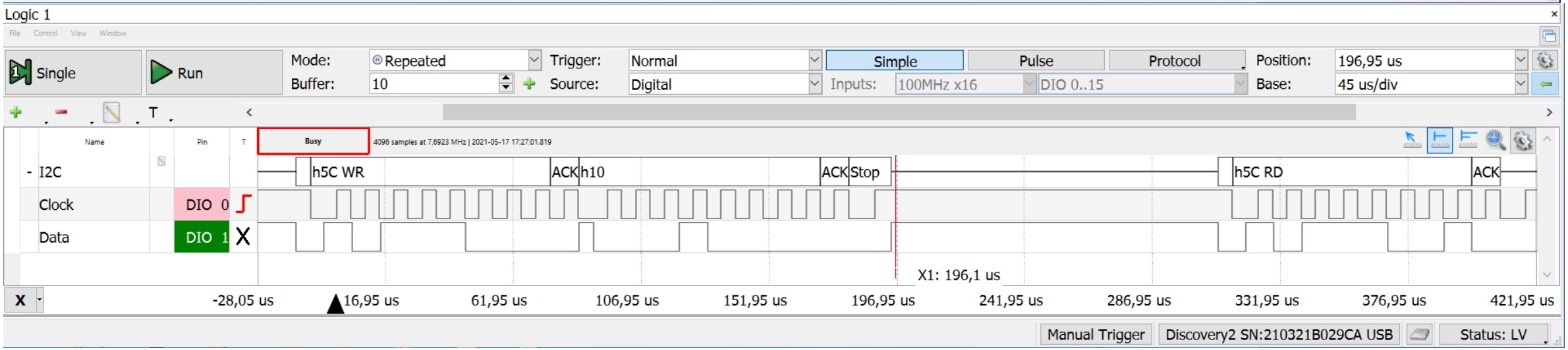
Čtení BH1750 16bit hodnoty světla

```
I2CLOOP:                                ;receive data from Slave BH1750 in loop
        call    I2CSTART                ;start condition
        ldi     data, SLA
        lsl     data
        ori     data, R                 ;SLA+R
        call    I2CPUT                  ;transmit SLA-W
        call    I2CGETACK               ;HI data byte
        mov             resultH, data
        call    I2CGETNACK              ;LO data byte
        mov             resultL, data
        call    I2CSTOP                 ;stop condition
rjmp    I2CLOOP
```

Průběh inicializace BH1750

ADDR pin = L, SLA = h23
ADDR pin = H, SLA = h5C

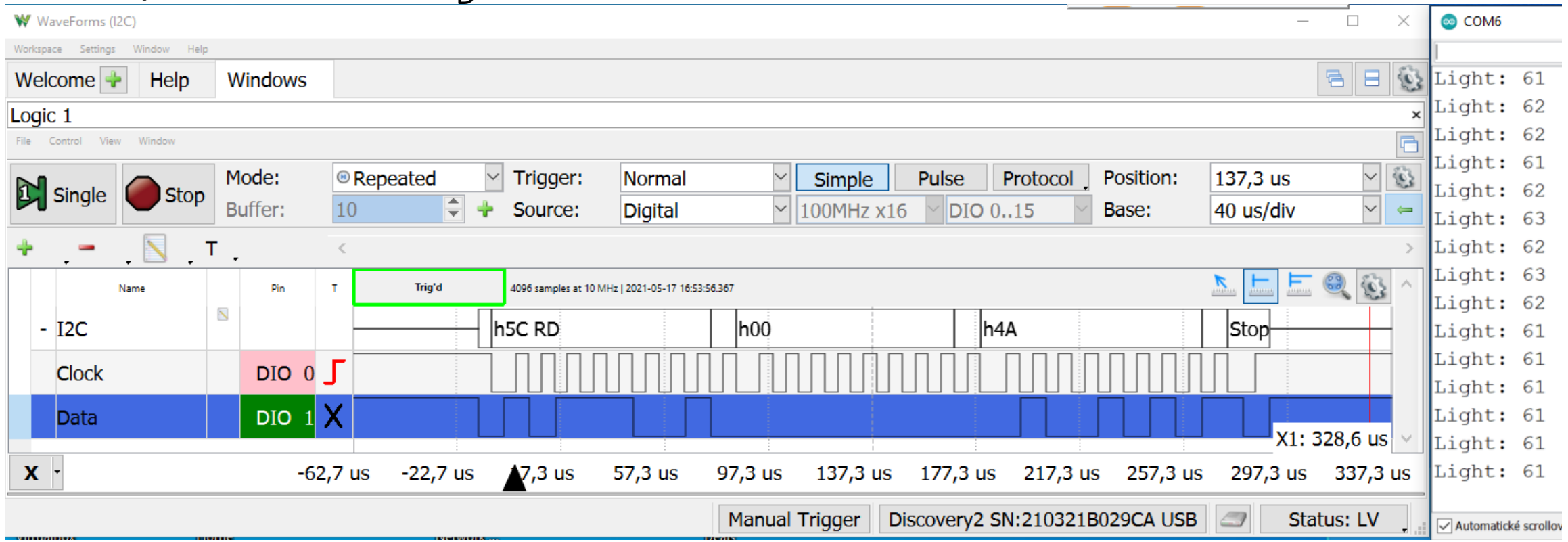




Power On: h01, Continuously H-Resolution Mode: h10

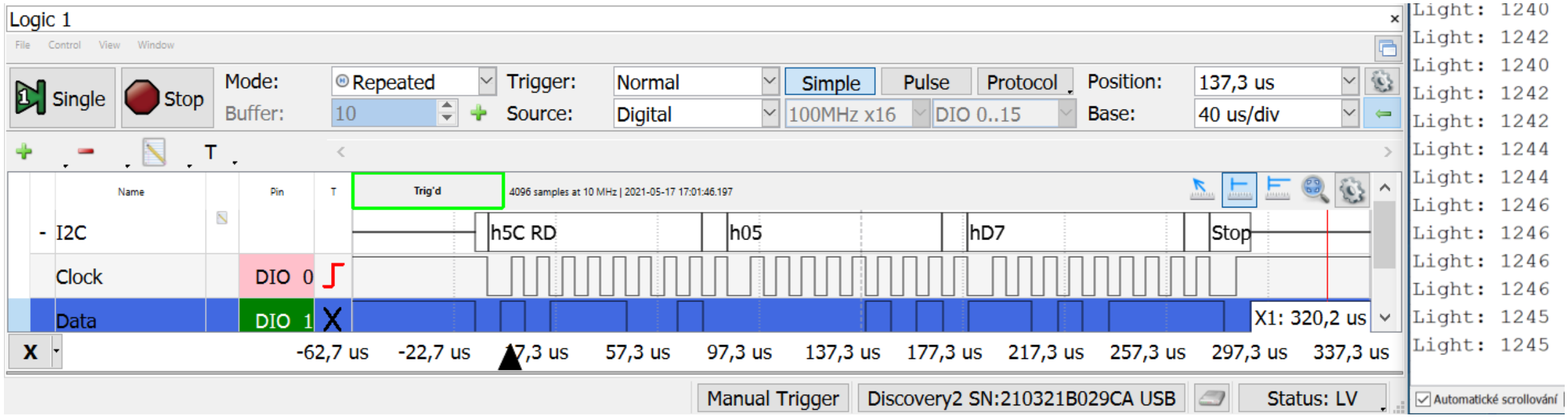
Zaslání dat z BH1750

- Změřená intenzita světla v Lx odpovídá hodnotě 16bit slova v desítkové soustavě /1,2
- Např. 0x004A = 74_D , 74/1,2 = 61,6 Lx



Zaslání dat z BH1750

- Změřená intenzita světla v Lx odpovídá hodnotě 16bit slova v desítkové soustavě /1,2
- Např. 0x05D7 = 1495_D , 1495/1,2 = 1245,8 Lx



Obsluha I2C - START

```
I2CSTART:
        ldi     temp, (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)
        sts     TWCR,temp
        ;wait for start condition to be sent.  when TWINT in TWCR is cleared, it is sent
        WAIT_START:
                lds             temp, TWCR
                sbrs    temp, TWINT
        rjmp    WAIT_START
        ;check TWSR for bus status
        lds             temp, TWSR
        andi    temp, 0b11111000        ;masks last three bits, which are 2=? 1:0prescaler value
        cpi     temp, START
        breq    PC+2
        jmp     errloop
ret
```

Obsluha I2C - PUT

```
I2CPUT:
        ldi     temp, (0<<TWINT) | (1<<TWEN)    ; disable the TW int
        sts     TWCR, temp
        sts     TWDR, data                                  ; write data to
TWDR
        ldi     temp, (1<<TWINT) | (1<<TWEN)    ; enable the TW int, wait for an ack
        sts     TWCR, temp
        ;another wait for the TWINT flag, which lets us know if ACK/NACK is back
        WAIT_DONE:
                lds     temp, TWCR
                sbrs    temp, TWINT
        rjmp    WAIT_DONE
ret
```

Obsluha I2C - GET

```
I2CGETACK:
        ldi     temp, (1<<TWINT) | (1<<TWEN) | (1<<TWEA); enable TWEA, then wait for TWINT
        sts     TWCR, temp
rjmp I2CGET

I2CGETNACK:
        ldi     temp,(1<<TWINT) | (1<<TWEN) | (0<<TWEA); disable TWEA, then wait for TWINT
        sts     TWCR,temp

I2CGET:
        WAIT_FOR_BYTE:
                lds     temp, TWCR
                sbrs    temp, TWINT
        rjmp    WAIT_FOR_BYTE
        lds     data,TWDR
ret
```

Obsluha I2C - STOP

```
I2CSTOP:
        ldi     temp, (1<<TWINT)|(1<<TWSTO)|(1<<TWEN)
        sts     TWCR,temp
        ;check TWCR to see if there is still a transmission- if not, stop bit has been sent
        Check1:
                lds     temp,TWCR
                andi    temp,0b00010000     ; Check to see that no transmission is going on
        brne    Check1
ret
```