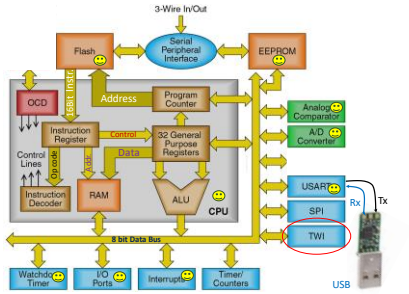
Týždeň	Prednášky	Cvičenia
1.	~~Úvod do predmetu. Číselné sústavy v elektronike, kódovanie čísel. Jazyk C. Úvod do digitálnej elektroniky.~~	~~Inštalácia: - IDE Microchip Studio - driver pre Arduino kit~~
2.	~~Polovodičové pamäte~~	~~Práca s Microchip Studio, definovanie projektu~~
3.	~~ATmega328, pamäť, generovanie hodinových signálov, manažment napájania, reset~~	~~Práca s Microchip Studio preklad, simulácia projektu~~
4.	~~Vstupy-výstupy~~	~~Práca so vstupmi a výstupmi~~
5.	~~Externé a interné prerušenia, Watchdog časovač~~	~~Práca s prerušeniami~~ ~~Výber zadania na zápočet~~
6.	~~Sériové komunikačné prostriedky USART – Tx~~	~~Práca s USART – TX.~~ ~~Práca na zadaní~~
7.	~~Sériové komunikačné prostriedky USART – Rx~~	~~Práca s USART – RX.~~ ~~Práca na zadaní~~
8.	~~TC0 - Normal mode~~	~~Práca s TC0.~~ ~~Práca na zadaní~~
9.	~~TC0 – Fast PWM mode, fázovo korektná a CTC PWM~~	~~Práca s TC0.~~ ~~Práca na zadaní~~
10.	~~AC komparátor, AD prevodník~~	~~Programovanie AD.~~ ~~Práca na zadaní~~
11.	~~EEPROM, FUSE bity~~	~~Programovanie EEPROM.~~ ~~Práca na zadaní~~
12.	I2C zbernica	**Zápočet - Obhajoba zadania (8+8)**
13.	Predtermín - **Obhajoba zadania (8+8)**	**Zápočet - Obhajoba zadania (8+8)**

1

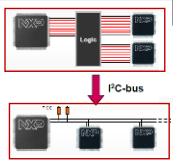
Architektúra mikrokontrolerov AVR



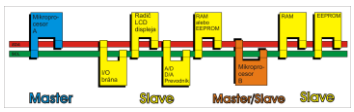
2

I²C (*Inter - Integrated Circuit*) (104-138)

- I²C zbernica je dvojvodičová, stredne rýchla komunikačná zbernica vyvinutá firmou *NXP* (*Philips Semiconductors*) začiatkom 80. rokov. Tento typ zbernice bol navrhnutý na zjednodušenie komunikácie vo vnútri elektronických zariadení a na znižovanie výrobných nákladov. I²C zbernica sa považuje za priemyselný štandard a tento komunikačný protokol používa množstvo elektronických obvodov.
- Zbernicu tvoria dva vodiče:
  - dátový vodič SDA (*Serial Data*)
  - hodinový vodič SCL (*Serial CLock*).
- I²C zariadenia môžu byť buď typy *master* (nadriadený) alebo typu *slave* (podriadený). Komunikáciu na zbernici vždy iniciuje *master*.
- I²C protokol umožňuje pripojiť na jednu zbernicu súčasne viacero *master* zariadení, pričom rozhodovací proces - *arbitráž zbernice* určuje, ktoré *master* zariadenie bude riadiť zbernicu.



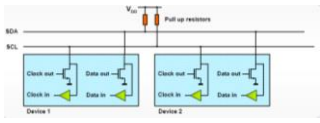
Pojem	Popis
Vysielač	Vysiela dáta na zbernicu.
Prijímač	Prijíma dáta zo zbernice.
Master	Začína a končí prenos, generuje SCL.
Slave	Adresované *master* zariadením.
Multi-*master*	Viac *master* zariadení na jednej I2C zbernici.

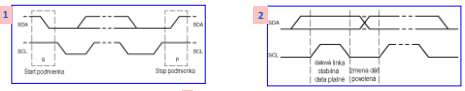


3

I²C (*Inter - Integrated Circuit*)

- Vodiče SDA a SCL sú pripojené k napájaciemu napätiu cez *pull-up* rezistory. Keď je zbernica voľná, oba vodiče sú na logickej úrovni „H". Koncové stupne zariadení pripojených k zbernici sú navrhnuté ako *open-collector* alebo *open-drain*.



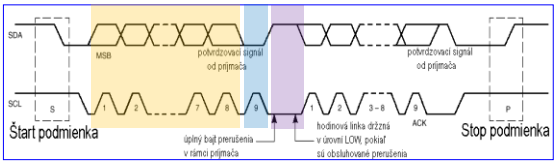


- Komunikáciu na zbernici riadi *master*, ktorý generuje (1) štart a stop podmienku. Po štarte je zbernica považovaná za obsadenú, zatiaľ čo po vyslaní stop podmienky je zbernica opäť voľná.
- Prenos bitu na zbernici (2) sa uskutočňuje tak, že vodič SDA je nastavený na logickú úroveň (*HIGH, LOW*), pričom je na vodiči SCL logická úroveň *HIGH* (log. 1). Keď je na SCL *LOW* (log. 0), môže sa logická úroveň na vodiči SDA zmeniť.

4

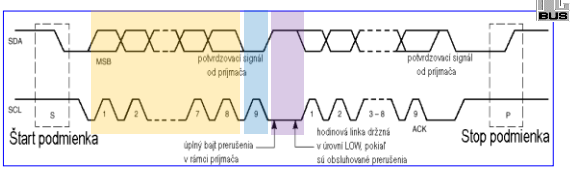
I²C (*Inter - Integrated Circuit*)

- Každý bajt vyslaný na SDA vodič musí **pozostávať z 8 bitov**. Počet bajtov prenesených v rámci jedného prenosu nie je obmedzený. Dáta sú prenášané tak, že najprv sa vysiela najvýznamnejší bit (MSB).
- Za každým bajtom musí nasledovať **bit potvrdenia** (ACK). Potvrdzovací bit umožňuje, aby prijímač signalizoval vysielaču, že bajt bol úspešne prijatý a môže byť odoslaný ďalší bajt.
- Potvrdzovací signál je definovaný nasledovne: vysielač uvoľní vodič SDA počas potvrdzovacieho hodinového impulzu, aby prijímač mohol nastaviť vodič SDA na úroveň *LOW* počas trvania impulzu potvrdenia. Tým je príjem bajtu potvrdený prijímačom.
- Ak SDA vodič zostane na úrovni *HIGH* počas deviateho hodninového impulzu, definuje sa to ako signál nepotvrdenia (NACK).



5

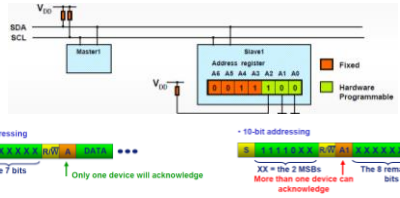
I²C (*Inter - Integrated Circuit*)



- Ak *master* prijme signál nepotvrdenia NACK môže generovať:
  - Buď stop podmienku na prerušenie prenosu,
  - Alebo opakovať štart podmienku na spustenie nového prenosu.
- Situácie, ktoré vedú ku generovaniu NACK:
  - Na zbernici nie je prítomný prijímač s danou adresou, ktorá by odpovedalo potvrdením.
  - Prijímač nie je schopný prijímať ani vysielať, pretože vykonáva úlohy v reálnom čase a nemôže začať komunikáciu.
  - Počas prenosu prijímač nemôže prijímať žiadne ďalšie dátové bajty.
  - Vysielač informuje *master* prijímač, aby ukončil prenos, pretože už nebude vysielať.

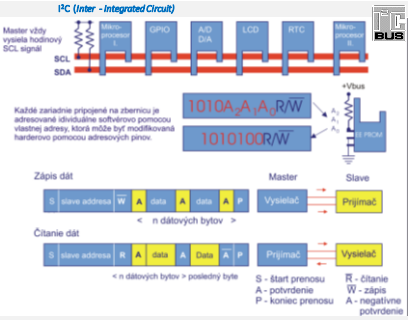
6

- Každé I²C zariadenie je adresované jedinečnou adresou, ktorá je tvorená 7 alebo 10 bitmi (128 alebo 1024 možných adries zariadení). Prideľovanie adries pre výrobcov obvodov je koordinované.
- Jedinečná adresa (7 alebo 10 bitov) môže mať všetky bity naprogramované (nastavené) výrobcom alebo, niektoré bity môžu byť nastavené prostredníctvom vývodov obvodu (hardvérová (užívateľská) adresa). Hardvérové nastavenie bitov adresy umožňuje, aby niekoľko rovnakých zariadení zdieľalo tú istú I2C zbernicu.
- 10-bitový formát adresy využíva 2-bajtovú adresu: 1111 0A9A8R/W + A7A6A5A4A3A2A1A0



7

- *Master* začína prenos na zbernici, vysланím *slave* adresy zariadenia. Všetky *slave* zariadenia adresu prijmú, ale v komunikácií pokračuje len to zariadenie, ktorého adresa bola na zbernicu vysланá.
- *master* je v úlohe vysielača riadi hodinový SCL aj dátový SDY signál.
- *master* je najskôr v úlohe vysielača riadi hodinový aj dátový signál, potom je v úlohe prijímača a ovláda len SCL signál, SDA ovláda *slave* vysielač.



8

I²C Read and Write Operations (2)

- **Combined Write and Read**

< n data bytes > < m data bytes >

S slave address W A data A data A Sr slave address R A data A data A P

"0" ≡ Write — Each byte is acknowledged by the slave device

"1" ≡ Read — Each byte is acknowledged by the master device (except the last one, just before the STOP condition)

- **Combined Read and Write**

< n data bytes > < m data bytes >

S slave address R A data A Sr slave address W A data A data A P

"1" ≡ Read — Each byte is acknowledged by the master device (except the last one, just before the Re-START condition)

"0" ≡ Write — Each byte is acknowledged by the slave device

9

I²C - Protocol

Summary

START	HIGH to LOW transition on SDA while SCL is HIGH
STOP	LOW to HIGH transition on SDA while SCL is HIGH
DATA	8-bit word, MSB first (Address, Control, Data): - Must be stable when SCL is HIGH - Can change only when SCL is LOW - Number of bytes transmitted is unrestricted
ACKNOWLEDGE	- Done on each 9th clock pulse during the HIGH period - The transmitter releases the bus - SDA goes HIGH - The receiver pulls DOWN the bus line - SDA goes LOW
CLOCK	- Generated by the Master(s) - Maximum speed: (100, 400, 1000, 3400 kHz) but NO min - A receiver can hold SCL low when performing another function (transmitter in a Wait state) - A master can slow down the clock for slow devices
ARBITRATION	- Master can start a transfer only if the bus is free - Several masters can start a transfer at the same time - Arbitration is done on SDA line - Master that lost the arbitration must stop sending data

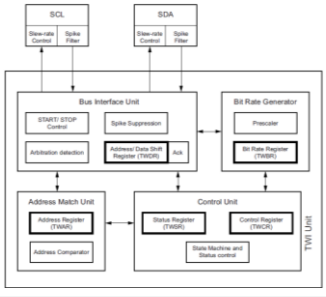
10

ATmega328P obsahuje jednoduché a výkonné synchrónne komunikačné rozhranie s dvoma vodičmi (TWI), čo odpovedá štandardu I²C:

- Podporuje režim *master* aj *slave*,
- Môže pracovať ako vysielač alebo prijímač,
- 7-bitová adresa umožňuje až 128 rôznych *slave* adries,
- Rýchlosť prenosu dát až 400 kHz,
- Programovateľná *slave* adresa,
- Umožňuje prebudenie z režimu *Sleep*,

- TWBR – TWI Bit Rate Register
- TWCR – TWI Control Register
- TWSR – TWI Status Register
- TWDR – TWI Data Register

- TWAR – TWI (Slave) Address Register
- TWAMR – TWI (Slave) Address Mask Register



11

Generátor prenosovej rýchlosti generuje hodinový signál na vodiči SCL, keď je MCU v režime *master*. Frekvencia hodinového signálu SCL sa nastavuje pomocou bitov registra TWBR.TWBR[7:0] a bitov status registra TWSR.TWPS[1:0]. Ak je MCU v režime *slave* frekvencia hodín CPU musí byť aspoň 16-krát vyššia ako frekvencia SCL vodiča.

$$f_{SCL} = \frac{f_{osc}}{16+2(TWBR)\cdot(PrescalerValue)} = \frac{16000000}{16+2\cdot0x48\cdot1} = 100000\ Hz, \quad PrescalerValue = 1$$

```
//------------------------------------------------------------------------
void I2C_Init (void) {
        TWBR = 0x48;                    // Set bit rate register (Baudrate).
        TWDR = 0xFF;                    // Default content = SDA released.
        TWCR = (1<<TWEN);               // Enable TWI-interface and release TWI pins.
}
```

Name: TWCR
Offset: 0xBC
Reset: 0x00
Property:

Bit	7	6	5	4	3	2	1	0
	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN		TWIE
Access	R/W	R/W	R/W	R/W	R	R/W		R/W
Reset	0	0	0	0	0	0		0

Bit 7 – TWINT: TWI Interrupt Flag
Bit 6 – TWEA: TWI Enable Acknowledge
Bit 5 – TWSTA: TWI START Condition
Bit 4 – TWSTO: TWI STOP Condition
Bit 3 – TWWC: TWI Write Collision Flag
Bit 2 – TWEN: TWI Enable
Bit 0 – TWIE: TWI Interrupt Enable

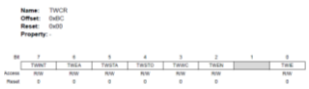
12

ATmega328P – *master send start, stop* - TWI *Two-Wire Interface* (26.)

Start I2C, Stop I2C

```
//------------------------------------------------------------------------------
void starti2c (void) {
        TWCR = (1<<TWEN)|(1<<TWINT)|(1<<TWSTA);
        while (!(TWCR & (1<<TWINT)));
}

//------------------------------------------------------------------------------
void stopi2c (void) {
        TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
}
```



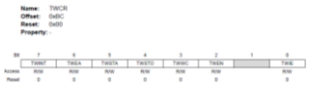
Bit 7 – TWINT: TWI Interrupt Flag
Bit 6 – TWEA: TWI Enable Acknowledge
Bit 5 – TWSTA: TWI START Condition
Bit 4 – TWSTO: TWI STOP Condition
Bit 3 – TWWC: TWI Write Collision Flag
Bit 2 – TWEN: TWI Enable
Bit 0 – TWIE: TWI Interrupt Enable

13

ATmega328P – *master send data, master receive data ACK, master receive data NACK* - TWI *Two-Wire Interface* (26.)

Send I2C, Recieve I2C ack, Receive I2C nack

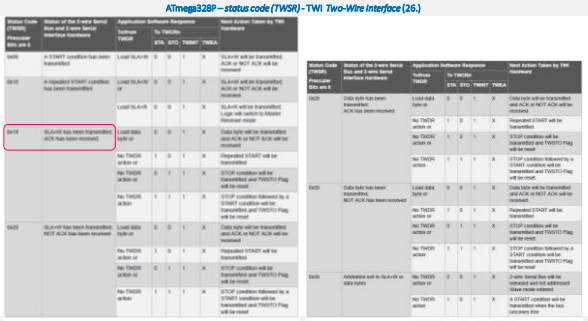
```
//------------------------------------------------------------------------------
void sendtoi2c(char data) {
        TWDR = data;
        TWCR = (1<<TWINT)|(1<<TWEN);
        while (!(TWCR & (1<<TWINT)));
}
//------------------------------------------------------------------------------
char receivei2c_ack(void) {
        TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);
        while (!(TWCR & (1<<TWINT)));
        return (TWDR);
}
//------------------------------------------------------------------------------
char receivei2c_nack(void) {
        TWCR = (1<<TWINT)|(1<<TWEN)|(0<<TWEA);
        while (!(TWCR & (1<<TWINT)));
        return (TWDR);
}
```



Bit 7 – TWINT: TWI Interrupt Flag
Bit 6 – TWEA: TWI Enable Acknowledge
Bit 5 – TWSTA: TWI START Condition
Bit 4 – TWSTO: TWI STOP Condition
Bit 3 – TWWC: TWI Write Collision Flag
Bit 2 – TWEN: TWI Enable
Bit 0 – TWIE: TWI Interrupt Enable

14

ATmega328P – *status code (TWSR)* - TWI *Two-Wire Interface* (26.)



15

ATmega328P – TWI *Two-Wire Interface* (26.)

Test prítomnosti I2C zariadenia na zbernici, pomocou status registra TWSR.

```
unsigned char i2c_Component (unsigned char adr)
{
        unsigned char status;

        starti2c();
        sendtoi2c(adr);
        status = TWI_STATUS;
        stopi2c();
        return (status);
}

int main (void) {

        if (0x18 == i2c_Component (PCF8574))
            USART_Transmit_Text ("PCF8574 OK!\n\r");
        else USART_Transmit_Text ("PCF8574 Error!\n\r");

        while(1) {
        }
}
```


16

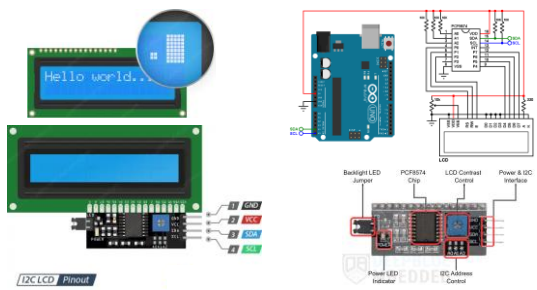
ATmega328P – *I2Cutility.c* – TWI *Two-Wire Interface* (26.)

```
#include <avr/io.h>
#include "I2CUtility.h"

void I2C_Init (void) {
    TWBR = TWI_TWBR;            // Set bit rate register (Baudrate). Defined in header file.
    //TWSR = TWI_TWPS;          // Not used. Driver presumes prescaler to be 00.
    TWDR = 0xFF;                // Default content = SDA released.
    TWCR = (1<<TWEN)|           // Enable TWI-interface and release TWI pins.
           (0<<TWIE)|(0<<TWINT)|    // Disable Interupt.
           (0<<TWEA)|(0<<TWSTA)|(0<<TWSTO)| // No Signal requests.
           (0<<TWWC);
}
void starti2c (void) {
    TWCR = (1<<TWEN)|(1<<TWINT)|(1<<TWSTA);
    while (!(TWCR & (1<<TWINT)));
}
void sendtoi2c(char data) {
    TWDR = data;
    TWCR = (1<<TWINT)|(1<<TWEN);
while (!(TWCR & (1<<TWINT)));
}
void stopi2c (void) {
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
}
char receivei2c_ack(void) {
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);
    while (!(TWCR & (1<<TWINT)));
    return (TWDR);
}
char receivei2c_nack(void) {
    TWCR = (1<<TWINT)|(1<<TWEN)|(0<<TWEA);
    while (!(TWCR & (1<<TWINT)));
    return (TWDR);
}
```

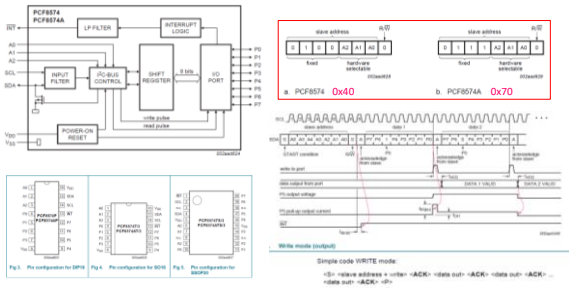

17

Príklad - ovládanie *Dot Matrix LCD* pomocou I2C PCF8574 8 bitového I/O expanderu



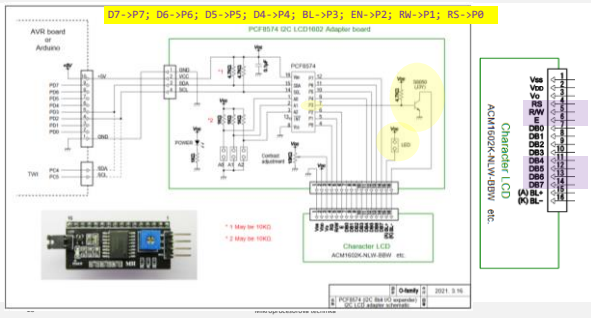
18

PCF8574 – Remote 8-bit I/O expander with interrupt



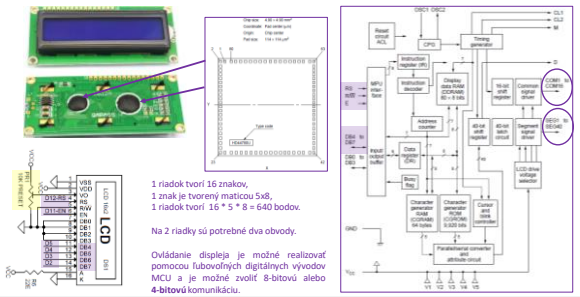
19

Dot Matrix LCD + PCF8574 I/O expander



20

Dot Matrix, LCD Controller/Driver HD44780U, AIP31066L, ST7066



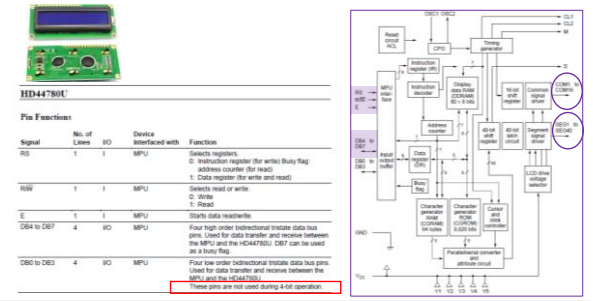
1 riadok tvorí 16 znakov,
1 znak je tvorený maticou 5x8,
1 riadok tvorí 16 * 5 * 8 = 640 bodov.

Na 2 riadky sú potrebné dva obvody.

Ovládanie displeja je možné realizovať pomocou ľubovoľných digitálnych vývodov MCU a je možné zvoliť 8-bitovú alebo **4-bitovú** komunikáciu.

21

Dot Matrix, LCD Controller/Driver HD44780U, AIP31066L, ST7066



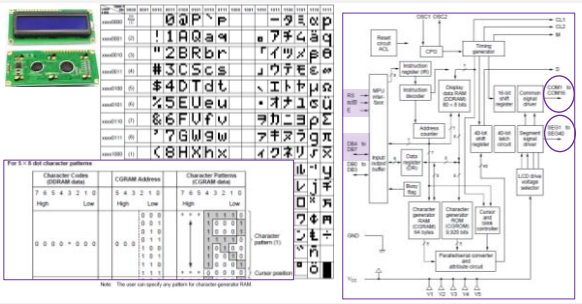
HD44780U

Pin Functions:

Signal	No. of Lines	I/O	Device Interfaced with	Function
RS	1	I	MPU	Selects registers. 0: Instruction register (for write) Busy flag: address counter (for read) 1: Data register (for write and read)
R/W	1	I	MPU	Selects read or write. 0: Write 1: Read
E	1	I	MPU	Starts data read/write.
DB4 to DB7	4	I/O	MPU	Four high order bidirectional tristate data bus pins. Used for data transfer and receive between the MPU and the HD44780U. DB7 can be used as a busy flag.
DB0 to DB3	4	I/O	MPU	Four low order bidirectional tristate data bus pins. Used for data transfer and receive between the MPU and the HD44780U. These pins are not used during 4-bit operation.

22

Dot Matrix, LCD Controller/Driver HD44780U, AIP31066L, ST7066



23

Dot Matrix, LCD Controller/Driver HD44780U, AIP31066L, ST7066

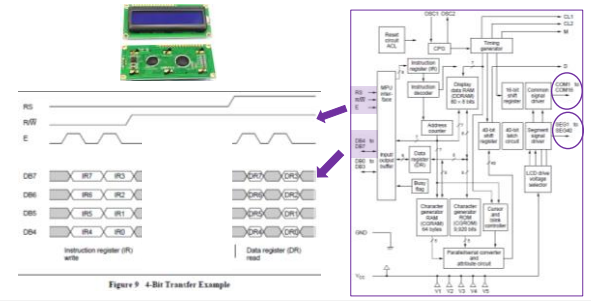
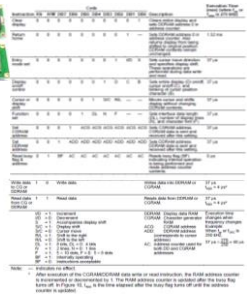
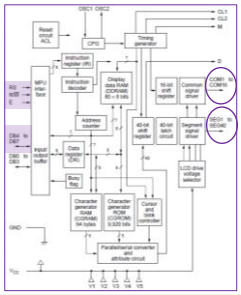


Figure 9 4-Bit Transfer Example

24

Dot Matrix, LCD Controller/Driver HD44780U, AIP31066L, ST7066





25

Dot Matrix LCD + PCF8574 I/O expander

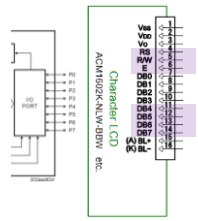
D7->P7; D6->P6; D5->P5; D4->P4; BL->P3; EN->P2; RW->P1; RS->P0

```
void Lcd1602_Init(void) {
        DataToPCF8574(0x00);
        _delay_ms(40);

        DataToPCF8574(0b00110100); // D7=0, D6=0, D5=1, D4=1, BL=0, EN=1, RW=0, RS=0
        _delay_us(5);
        DataToPCF8574(0b00110000); // D7=0, D6=0, D5=1, D4=1, BL=0, EN=0, RW=0, RS=0
        _delay_ms(5);
        DataToPCF8574(0b00110100); // D7=0, D6=0, D5=1, D4=1, BL=0, EN=1, RW=0, RS=0
        _delay_us(5);
        DataToPCF8574(0b00110000); // D7=0, D6=0, D5=1, D4=1, BL=0, EN=0, RW=0, RS=0
        _delay_us(100);
        DataToPCF8574(0b00110100); // D7=0, D6=0, D5=1, D4=1, BL=0, EN=1, RW=0, RS=0
        _delay_us(5);
        DataToPCF8574(0b00110000); // D7=0, D6=0, D5=1, D4=1, BL=0, EN=0, RW=0, RS=0
        _delay_us(100);

        DataToPCF8574(0b00100100);
        _delay_us(5);
        DataToPCF8574(0b00100000);
        _delay_us(50);

        Lcd_Cmd(FUN_SET|0x08);          //DL=0, N=1, F=0
        _delay_us(50);
        Lcd_Cmd(DISP_ON_OFF|0x04);      //D2=display_on, D1=cursor_on, D0=cursor_blink
        _delay_us(50);
        Lcd_Cmd(CLEAR_DISP);
        _delay_us(2000);
        Lcd_Cmd(MODE_SET|0x02);         //I/D=1
        Lcd1602_Out(1,0,"Mikroprocesorova");
        Lcd1602_Out(2,0,"----Technika----");
}
```



26

Dot Matrix LCD + PCF8574 I/O expander

I2C_displej.c
I2C_1602.c
I2CUtility.c
I2CUtility.h
PCF8574.c

27