Vážení zákazníci,

dovolujeme si Vás upozornit, že na tuto ukázku knihy se vztahují autorská práva, tzv. copyright.

To znamená, že ukázka má sloužit výhradně pro osobní potřebu potenciálního kupujícího (aby čtenář viděl, jakým způsobem je titul zpracován a mohl se také podle tohoto, jako jednoho z parametrů, rozhodnout, zda titul koupí či ne).

Z toho vyplývá, že není dovoleno tuto ukázku jakýmkoliv způsobem dále šířit, veřejně či neveřejně např. umisťováním na datová média, na jiné internetové stránky (ani prostřednictvím odkazů) apod.

redakce nakladatelství BEN – technická literatura
redakce@ben.cz

```
  Sleep(1);                  //přesah
  Control.OutPort($00);      //STB=1
end;

//konstruktor výjimky:
constructor TParPortException.Create(const Msg:String);
begin
  inherited;
end;

end.
```

3.4.2 Uživatelský popis

Pro praktické použití je nutno seznámit se s metodami třídy **TParallel**:

- **constructor Create(Number:Integer; Address:Word)** – konstruktor; **Number** udává číslo portu (například pro LPT1 zadáme Number = 1), **Address** je bázová adresa portu (zjistíme například v dialogu dle *obr. 3.3*, výchozí hodnota je obvykle Address = $378). Při neúspěchu se vyvolá výjimka **TParPortException**,
- **destructor Destroy** – destruktor, zajistí zavření portu,
- **procedure SPPWriteByte(AByte:Byte):Boolean** – pošle bajt **AByte** na SPP port.

Dále jsou k dispozici vlastnosti:

- **Data, Status, Control** – jednotlivé porty (tedy porty s bázovými adresami BA, BA+1 a BA+2),
- **Name** – řetězec jméno portu (například LPT1),
- **Addr** – bázová adresa portu BA (například $378).

Použití třídy **TParallel** si ukážeme na zajímavém příkladu řízení LCD displeje.

3.5 LPTLCD – ŘÍZENÍ LCD DISPLEJE PŘES PARALELNÍ PORT

Připojení LCD displeje k paralelnímu portu názorně ukazuje, že pro některé případy jsou možnosti portu pracujícího s 8 výstupními linkami naprosto postačující.

3.5.1 Stručný popis ovládání LCD displeje

Všechny řádkové displeje jsou řízeny obvodem **HD44780** od firmy Hitachi (případně jeho ekvivalentem). Úplný popis v angličtině najdete na doprovodném CD-ROM

v adresáři **DATASHEET**. V následujícím textu je uveden velmi stručný popis (podrobnější informace lze nalézt například v [3] nebo v [21]).

Displej je schopen přijímat data po čtyřech nebo osmi datových linkách. V případě 4bitové komunikace musí přenos probíhat nadvakrát (nejdříve se přenese horní a potom dolní polovina bajtu). Pro případ 8bitové komunikace se použijí linky **DB0 až DB7**. Pro 4bitovou komunikaci je nutno vývody **DB0 až DB3** přivést na **GND**, poloviny bajtu se připojují na vývody **DB4 až DB7**.

Pro řízení komunikace jsou důležité linky **RS** (výběr přenosu dat nebo řídicího příkazu), **R/$\overline{W}$** (čtení nebo zápis) a **E** (povolovací vstup).

Ve většině případů není požadováno zpětné čtení údajů z displeje. Proto je vývod **R/$\overline{W}$** velmi často připojen na **GND**. Pak je tedy možno pouze zapisovat, touto „operací" se uspoří jeden ovládací vývod. Další úspory vzniknou při použití 4bitové komunikace. Celkově je pak třeba pro řízení pouze 6 vývodů.

Zápis dat resp. příkazu začíná nastavením signálu **RS** podle toho, zda se zapisují data nebo příkaz. Vývod **E** se připojí do log. 1 a vystaví se horní 4 bity na vývody **DB4 až DB7**. Poté se vývod **E** uvede do log. 0 (potvrzení zápisu). Operace se opakuje pro zápis dolních 4 bitů.

Tab. 3.2 Vývody displeje

Číslo vývodu	Signál	Funkce
1	GND	Zem (0 V)
2	U_{CC}	Napájecí napětí (4,75 až 5,25 V)
3	U_o	Nastavení kontrastu displeje
4	RS	Příkaz (0), data (1)
5	R/$\overline{W}$	Čtení (1), zápis (0) dat nebo příkazu
6	E	Vstup povolení
7	DB0	Data/příkaz (dolní bit)
8	DB1	Data/příkaz
9	DB2	Data/příkaz
10	DB3	Data/příkaz
11	DB4	Data/příkaz
12	DB5	Data/příkaz
13	DB6	Data/příkaz
14	DB7	Data/příkaz (horní bit)
15	A	Kladný pól podsvěcovací LED
16	K	Záporný pól podsvěcovací LED

V níže realizovaném přípravku **LPTLCD** byl použit LCD displej typu **EL1602A-FL-YBW**, který disponuje podsvícením (nemusíme jej však použít). Tento displej je nabízen na stránkách **shop.hw.cz** za velmi příznivou cenu okolo 220 Kč.

Pro elementární komunikaci s displejem je ještě uveden seznam příkazů ve formě *tab. 3.3*.

Tab. 3.3 Řídicí příkazy displeje

Příkaz/data	RS	Data								Čas provedení
		7	6	5	4	3	2	1	0	
Vymaž displej	0	0	0	0	0	0	0	0	1	1,64 ms
Návrat na začátek	0	0	0	0	0	0	0	1	X	1,64 ms
Volba režimu	0	0	0	0	0	0	1	I/$\overline{D}$	S	40 μs
Zapni/vypni displej	0	0	0	0	0	1	D	C	B	40 μs
Posun zobrazení/kurzoru	0	0	0	0	1	S/$\overline{C}$	R/$\overline{L}$	X	X	40 μs
Nastavení komunikace	0	0	0	1	DL	N	0	X	X	40 μs
Nastavení adresy CG RAM	0	0	1	adresa CG RAM						40 μs
Nastavení adresy DD RAM	1	1	adresa DD RAM							40 μs
Zápis dat do CG/DD RAM	1	data pro CG/DD RAM								40 μs

X libovolná hodnota (0 nebo 1),

I/$\overline{D}$ inkrementace (1), dekrementace (0),

S režim displeje (S = 0 – normální práce, S = 1 – kombinovaný posun displej, jsou-li data zapsána),

D displej zapnut (1), vypnut (0),

C zobrazování kurzoru zapnuto (1), vypnuto (0),

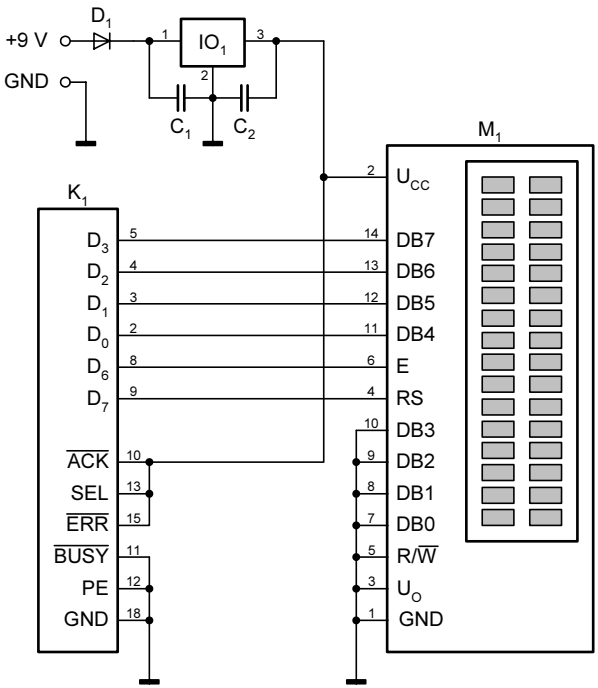
B blikání kurzoru zapnuto (1), vypnuto (0),

S/$\overline{C}$ posun displeje jsou-li data zapsána (1), posun kurzoru jsou-li data zapsána (0),

R/$\overline{L}$ posun doprava (1), doleva (0),

DL 8bitová komunikace (1), 4bitová komunikace (0),

N dva řádky (1), jeden řádek (0).



*Obr. 3.4 Schéma zapojení přípravku **LPTLCD***

3.5.2 Schéma zapojení

Schéma zapojení vychází z výše uvedené úvahy připojení vývodů displeje a je uvedeno na *obr. 3.4*.

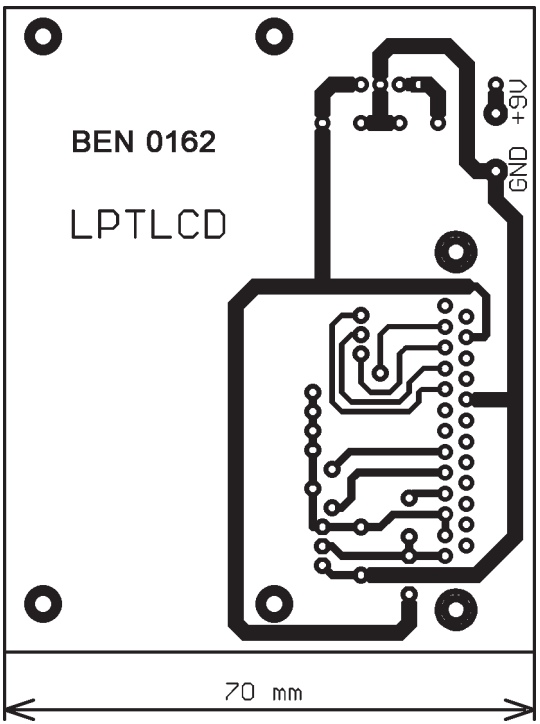
Vývody **DB0 až DB3** jsou tedy společně s **R/$\overline{\mathbf{W}}$** a $\mathbf{U_0}$ (zajistí maximální kontrast) připojeny na **GND**. Vývody **DB4 až DB7** odpovídají linkám $\mathbf{D_0}$ až $\mathbf{D_3}$ paralelního portu. Vývody **E** a **RS** jsou řízeny linkami $\mathbf{D_6}$ a $\mathbf{D_7}$. Linky $\mathbf{D_4}$ a $\mathbf{D_5}$ nejsou používány (mohly by se použít pro řízení podsvícení nebo pro jiné účely).

Napájení je získáno z vnějšího zdroje, přepólování brání dioda D_1. Stabilizátor IO_1 pak vytvoří napětí 5 V. Podsvícení displeje není zapojeno, je však možno jej zapojit.

Všimněte si vzájemného propojování linek **$\overline{\mathbf{ACK}}$**, **SEL** a **ERR** na log. 1 a vývodů **$\overline{\mathbf{BUSY}}$**, **PE** na log. 0. Tímto spojením je zajištěna úspěšná komunikace dle *obr. 3.1*.

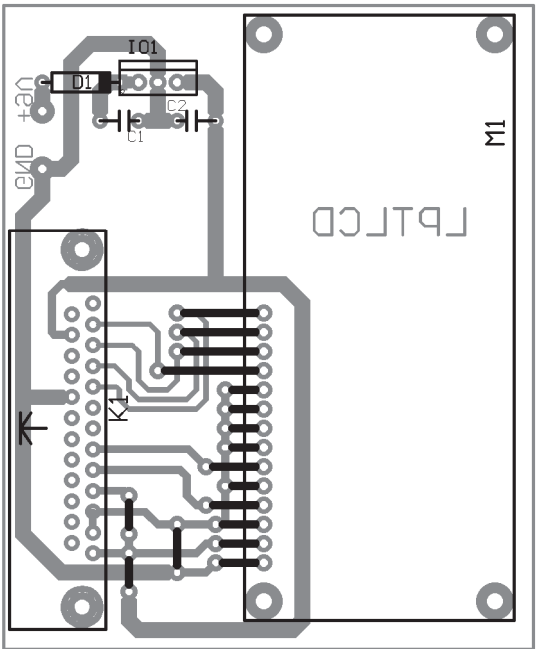
3.5.3 Plošný spoj

V amatérských podmínkách lze LCD displej připojit ke konektoru **CAN 25 V 90** přímo pomocí plochého kabelu a přidat další součástky. Reprodukovatelnější výsledek však poskytuje umístění součástek na desku plošných spojů. Výkres je uveden na *obr. 3.5*.



*Obr. 3.5 Výkres desky plošných spojů přípravku **LPTLCD** (BEN 0162)*

Vzhledem k tomu, že displej je umístěn nad deskou, musí být jeho vývody připojeny krátkými drátky. Situace je zřejmá z *obr. 3.6*.



Obr. 3.6 Osazovací plánek

Rozpis součástek pro LPTLCD (přibližná cena 270 Kč):

C_1, C_2	CK 100N/63V	2 ks
D_1	1N4148	1 ks
IO_1	7805	1 ks
K_1	CAN 25 V 90	1 ks
M_1	EL1602A-FL-YBW	1 ks

3.5.4 Testovací aplikace

Pro řízení displeje jsem se rozhodl vytvořit speciální třídu **TLCDInterface**. Důvod je prostý, tímto způsobem umožním čtenářům snažší použití ovládacích rutin pro nové účely. Pro vlastní realizaci se využívá kód obsažený ve třídě **TParallel**.

Implementace je vcelku jednoduchá. Konstruktor třídy **TLCDInterface** podle parametrů **Number** a **Address** pozná číslo portu a bázovou adresu a sestaví instanci třídy **TParallel** tak, aby se používal žádaný paralelní port. Navíc vyšle inicializační sekvenci a zajistí smazání displeje a rozblikání kurzoru.

Metoda **SendData** posílá 8bitová data určená parametrem **AByte** po 4 bitech na dvě volání metody **SendSubData**. Vzhledem k tomu, že se nejdříve posílá

horní polovina dat a linky **DB4 až DB7** jsou připojeny na vývody portu D_0 **až** D_3, musí se údaj **AByte** při prvním voláním metody **SendSubData** posunout o 4 bity vpravo (operátorem **shr**).

Podobně posílání příkazů probíhá metodou **SendCommand** na dvě volání metody **SendSubCommand**. Metody **SendSubCommand** a **SendSubData** se liší pouze logickou úrovní linky **RS**.

LCDINTERFACE.PAS:

```
unit LCDInterface;

interface

uses
 Windows, SysUtils, Parallel;

//třída TLCDInterface:
type
   TLCDInterface = class(TObject)
   private
      Port:TParallel;
      procedure SendSubCommand(AByte:Byte);
      procedure SendSubData(AByte:Byte);
   public
      constructor Create(Number:Integer;Address:Word);
      destructor Destroy; override;
      procedure SendData(AByte:Byte);
      procedure SendCommand(AByte:Byte);
end;

//výjimka:
type
   TLCDException = class(Exception)
   public
      constructor Create(const Msg:String);
end;

implementation

//konstruktor,
//Number určí číslo paralelního portu:
constructor TLCDInterface.Create(
   Number:Integer;Address:Word);
begin
   inherited Create;
   try
```

```
      Port:=TParallel.Create(Number,Address);
      //inicializační sekvence:
      SendSubCommand($03);
      Sleep(4);
      SendSubCommand($03);
      SendSubCommand($03);
      SendSubCommand($02);
      SendCommand($28);
      SendCommand($01);
      SendCommand($08);
      SendCommand($0C);
      SendCommand($06);
      SendCommand($0F);
    except
      raise
    end;
end;

//destruktor (odevzdá port):
destructor TLCDInterface.Destroy;
begin
  Port.Free;
  inherited Destroy;
end;

//pošle data na displej:
procedure TLCDInterface.SendData(AByte:Byte);
begin
  //pošle horní 4 bity:
  SendSubData(AByte shr 4);
  //pošle dolní 4 bity:
  SendSubData(AByte);
end;

//pošle data na linky DB4 až DB7:
procedure TLCDInterface.SendSubData(AByte:Byte);
var  PByte:Byte;
     b1,b2:Boolean;
begin
  //vem spodní 4 bity:
  PByte:=AByte and $0F;
  //RS=1, E=1 a data:
```

```
  b1:=Port.WriteByte(PByte or $C0);
  Sleep(1);
  //RS=1, E=0 a data:
  b2:=Port.WriteByte(PByte or $80);
  Sleep(1);
  //test chyby komunikace:
  if (not b1) or (not b2) then
    raise TLCDException.Create(
      'Přípravek není připojen');
end;

//pošle příkaz na displej:
procedure TLCDInterface.SendCommand(AByte:Byte);
begin
  //pošle horní 4 bity:
  SendSubCommand(AByte shr 4);
  //pošle dolní 4 bity:
  SendSubCommand(AByte);
end;

//pošle příkaz na linky DB4 až DB7:
procedure TLCDInterface.SendSubCommand(AByte:Byte);
var PByte:Byte;
    b1,b2:Boolean;
begin
  //vem spodní 4 bity:
  PByte:=AByte and $0F;
  //RS=0, E=1 a data:
  b1:=Port.WriteByte(PByte or $40);
  Sleep(1);
  //RS=0, E=0 a data:
  b2:=Port.WriteByte(PByte);
  Sleep(1);
  //test chyby komunikace:
  if (not b1) or (not b2) then
    raise TLCDException.Create(
      'Přípravek není připojen');
end;

//konstruktor výjimky:
constructor TLCDException.Create(const Msg:String);
begin
```

```
  inherited;
end;

end.
```

Vlastní příklad naleznete na doprovodném CD-ROM v adresáři **PROGRAMY\KAP_03\LPTLCD**. Hlavní formulář aplikace je v souboru **LCDUNIT.PAS**.

Vzhledem k tomu, že většina kódu je zapsána již ve třídách **TLCDInterface** a **TParallel**, je uvedený ovládací program velmi jednoduchý.

Formulář obsahuje editační pole s názvem **Text** a tlačítko s názvem **Vymaz**, viz *obr. 3.7*. Dále jsou připojeny události **OnCreate** a **OnDestroy** formuláře, **OnKeyDown** editačního pole a **OnClick** tlačítka.

Obr. 3.7 Testovací aplikace

***POZNÁMKA**: Nezapomeňte, že do projektu musí být připojena jednotka **LCDInterface**, která obsahuje zdrojový text třídy **TLCDInterface**.*

LCDUNIT.PAS:

```
unit LCDUnit;

interface

uses
  LCDInterface,
  Windows, Messages, SysUtils, Variants, Classes,
  Graphics, Controls, Forms, Dialogs, StdCtrls;

//formulář:
type
  TLPTForm = class(TForm)
    Vymaz: TButton;
    Text: TEdit;
    Popisek: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
```

```
    procedure TextKeyDown(Sender: TObject;
      var Key: Word;Shift: TShiftState);
    procedure VymazClick(Sender: TObject);
  private
    LCD:TLCDInterface;
  end;

//instance:
var
  LPTForm: TLPTForm;

implementation

{$R *.dfm}

//OnCreate formuláře:
procedure TLPTForm.FormCreate(Sender: TObject);
begin
  try
    //zkus vytvořit instanci pro LPT1:
    LCD:=TLCDInterface.Create(1,$378);
  except
    //reakce na případnou chybu:
    on E: Exception do begin
      Application.MessageBox(
       PChar(E.Message),
       'LPTLCD',
       MB_ICONHAND);
      Application.Terminate;
    end;
  end;
end;

//OnDestroy formuláře:
procedure TLPTForm.FormDestroy(Sender: TObject);
begin
  //uvolni instanci:
  LCD.Free;
end;

//OnKeyDown pro editační pole Text:
procedure TLPTForm.TextKeyDown(
  Sender: TObject; var Key: Word;
  Shift: TShiftState);
```

```
var i:Integer;
begin
   //akce pro stisk klávesy ENTER:
   if(Key=VK_RETURN)then
      //pošli znaky na displej:
      for i:=1 to Length(Text.Text) do
         LCD.SendData(ord(Text.Text[i]));
end;

//OnClick pro tlačítko Vymaz:
procedure TLPTForm.VymazClick(Sender: TObject);
begin
   //smaž displej:
   LCD.SendCommand($01);
end;

end.
```

Při startu aplikace se pomocí událost **OnCreate** formuláře vytvoří instance třídy **TLCDInterface** a odkaz je uložen do atributu pojmenovaného **LCD**. Komunikace je napevno stanovena pro port LPT1 (počítače obvykle nemají více paralelních portů).

Je jasné, že tato operace nemusí být vždy bezproblémová, proto je volání konstruktoru třídy **TLCDInterface** zapsáno do chráněného bloku.

Například se může stát, že port LPT1 není právě k dispozici (co když právě probíhá tisk?). Třída **TParallel** v takovém případě zajistí vytvoření výjimky **TParPortException** s odpovídajícím chybovým hlášením. Situace je zřejmá z *obr. 3.8*.

Obr. 3.8 Stav, když není port k dispozici

Pokud nenastala chyba, pokračuje aplikace normálně ve svém běhu:

- Pokud uživatel zapíše text do editačního pole a stiskne ENTER, přečte se řetězec textu obsažený v editačním poli a pomocí metody **SendData** odešle na displej,
- Po stisku tlačítka Vymaž se pošle příkaz $01 metodou **SendCommand**, který způsobí vymazání displeje (viz *tab. 3.3*).