Kapitola 1

CVIČENÍ ARP - Úvod

První cvičení je zkrácené. Spouštění překladače, mapování cest. Připomenutí základních logických, aritmetických a bitových operací v jazyce C.

Příklad přístupu k technickým prostředkům počítače v jazyce C - videopaměť počítače PC. Další podrobnosti jsou uvedeny v příkladu zdrojového kódu.

Kapitola 2

LED diody na LPT

2.1 Zapojení LED na LPT

Na paralelní port je možno zapojit přímo LED diody s nízkým příkonem $1 \div 2$ mA. Dle schématu na obrázku 2.1 je vidět, že LED diody se aktivují vždy v úrovní logické 1. Aby proud protékající jednotlivými diodami nepřekročil přípustnou mez, je každá dioda zapojena v sérii s ochranným odporem. Jednotlivé vývody D0-D7 odpovídají i v uvedeném pořadí osmi datovým bitům datového portu LPT.

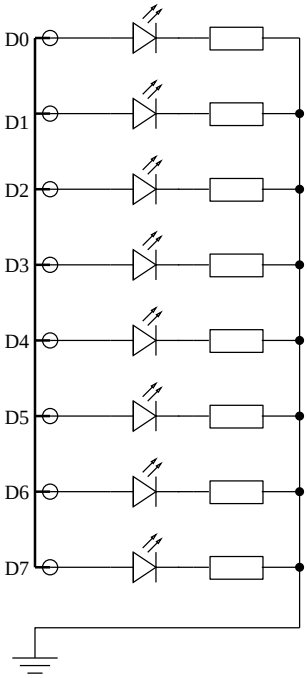
2.2 Řízení jasu

Princip

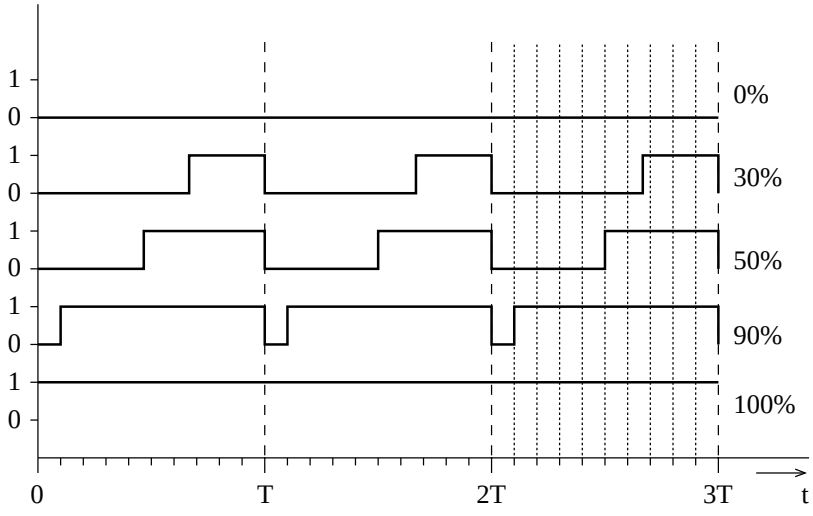
Úkolem je řízení jasu diod změnou střídy (duty cycle) - častěji je ale používán pojem pulsně šířková modulace (PWM - pulse wide modulation). Jde o měnící se poměr délky signálu a mezery při konstantní časové periodě. Vyjadřujeme jej v procentech. Na obrázku 2.2 je příklad několika takových signálů s odlišnou střídou.

Úkoly

- Rozsvítit každou diodu zadaným jasem, jednotlivé hodnoty stačí zadat do pole.
- Diody postupně rozsvítit a postupně zhasnout. Postupně se rozumí nejprve pomalu rozsvítit první, pak postupně druhou a první zůstává svítit, atd. A pak postupně pozhasínat.



Obrázek 2.1: Zapojení LED diod na LPT



Obrázek 2.2: Časový průběh signálů s různou střídou

- 4 a 4 diovy se rozsvěcují a zhasínají současně. Buď prokládaně, nebo po čtveřicích.

Postup

- Rozsvítit vybranou LED diodu.
- Rozblikat vybranou LED diodu se střídou 50%.
- Najít periodu $T = 1/f$, tak aby nebylo blikání viditelné.
- Řídit jas jedné LED.
- Řídit jas dvou LED současně. Nebo zajistit postupné rozsvícení. Časovou periodu rozdělit po jedné ms a hledat místa, kdy se mění stavy LED diod.
- Dokončit zadání.

2.3 Řízení frekvence blikání

Printcip

Úkolem je blikání diod s určitou frekvencí. Pozor! Obvyklá chyba při programování: $T = 1/f$ je časová perioda, ale ta se musí rozdělit se střídou 50% na „svití/nesvítí“. Bude-li zvolená perioda použita pro svícení i zhasnutí, výsledná frekvence bude poloviční!

Úkoly

Podobné jako při řízení jasu.

Postup

Hledat společný násobek všech potřebných frekvencí nelze.

Jsou tedy dvě možnosti: můžeme mít tolik čítačů, jako je LED diod a po každé ms se každý čítač inkrementuje a kontroluje na přetečení.

Nebo můžeme mít jeden „nekonečný“ čítač inkrementovaný opět po každé ms a pak celočíselným dělením tohoto čítače periodou pro každou LED dostaneme zbytky po dělení. Tento zbytek (modulo) bude pro každou LED nula nebo různý od nuly. Při nule nastane změna stavu příslušné LED (ideální je operace XOR, ale někdo raději provede OR/AND s vhodnou bitovou maskou).

První řešení umožňuje řídit i fázi, druhé je jednodušší. Způsobí ale drobnou asynchronnost při přetečení čítače, která je ale očima nepozorovatelná.

Vybranou metodu řešení pak můžeme aplikovat v následujících krocích:

- Rozblikat jednu LED se zadanou frekvencí.
- Najít frekvenci, při která není blikání vidět.
- Rozblikat dvě diody, každou jinou frekvencí (ne celistvý násobek).
- Dokončit zadání.

Kapitola 3

Nepotvrzovaná komunikace s periériemi

3.1 Propojení počítačů přes LPT

Schéma zapojení je na obrázku 3.1. Paralelní port je v standardním režimu vybaven pouze pěti vstupními piny. Jejich přesný význam je možno dohledat v technické dokumentaci, např. SYSMAN, TECHHELP a pod.

Z pohledu komunikace jde ale o pět vstupních (přijímacích) datových bitů. Pro předávání (vysílání) informace se používá pět nejnižších datových bitů datového portu LPT rozhraní.

Při programování je ale důležitá jen čtveřice funkcí pro ovládání komunikace: $set\_data(int), int\ get\_data(), set\_ctrl(int), int\ get\_ctrl()$. Jde vždy o dvojici funkcí pro ovládání čtyř datových bitů a řídícího signálu.

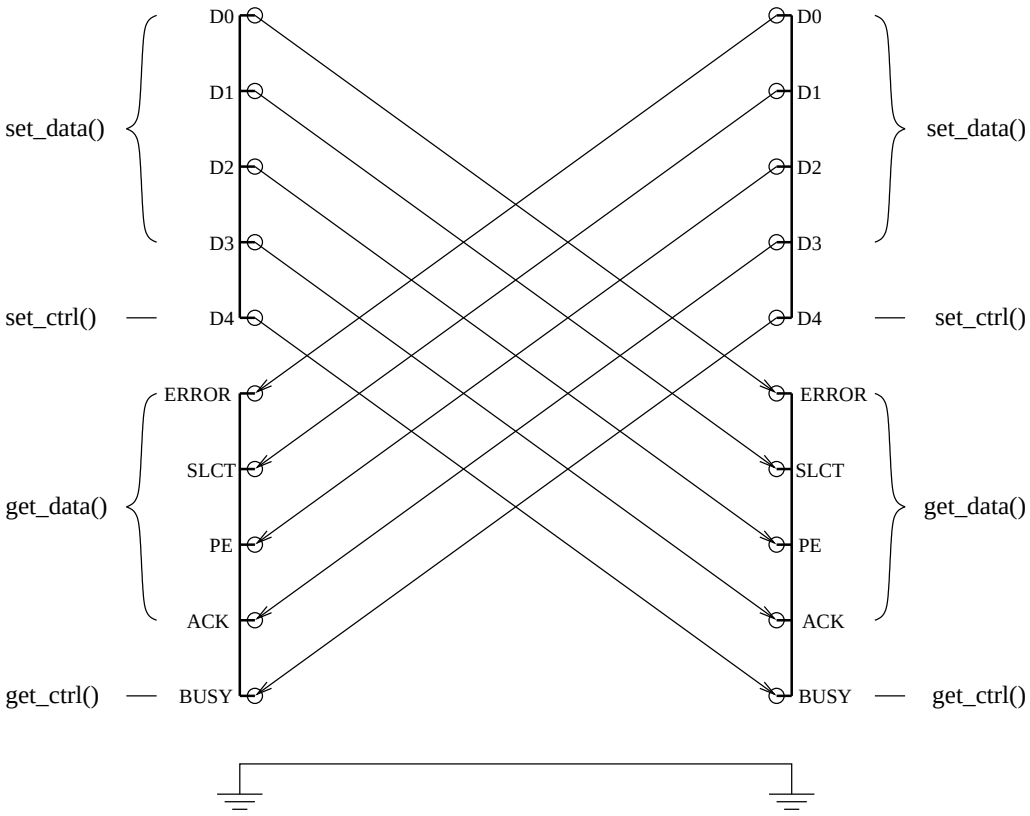
Pozor! Už ze zapojení na obrázku 3.1 by mělo být patrné, že vysílač použije funkce set_xxx, aby předal data přijímači. Ten si je převezme funkcemi get_xxx.

Zapojení pětice paraleních vodičů je obousměrné a proto spolu mohou dva počítače komunikovat v obousměrném (full duplex) režimu.

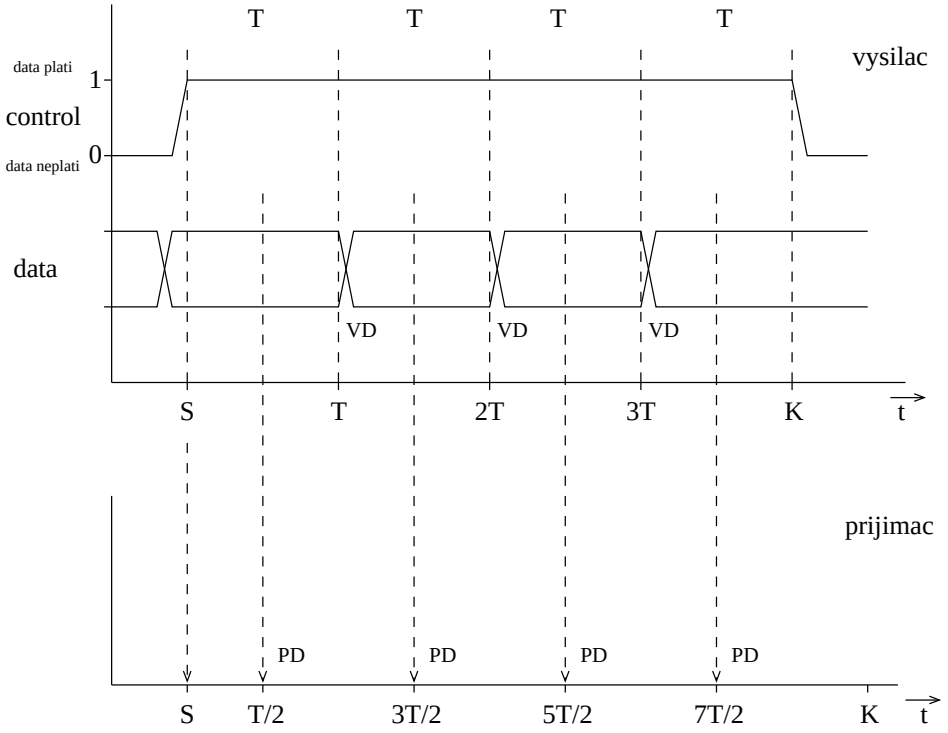
3.2 Arytmický (nesprávně asynchronní) přenos

Popis komunikace

Typickým příkladem takovéto komunikace je sériová linka. Stejný princip komunikace si můžeme modelovat i přes LPT a proto si jej popíšeme:



Obrázek 3.1: Propojení dvou počítačů přes LPT



Obrázek 3.2: Časové schéma arytmického přenosu

- Přenos může začít kdykoliv, proto se někdy říká asynchronní komunikace.
- Přijímač i vysílač má svůj vlastní zdroj hodin. Zdroje nejsou po dobu přenosu synchronizovány.
- Zdroje hodinového signálu jsou synchronizovány jen na počátku přenosu a po určitém čase již nelze oba zdroje považovat za synchronizované (každý zdroj času pracuje jen s určitou přesností) a docházelo by ke ztrátě dat.
- Začátek a konec vysílání je dán hodnotou řídícího signálu. Začátek přenosu určuje vysílač, dále pracují oba samostatně až do konce přenosu.
- Začátek vysílání - náběžná hrana řídícího signálu - slouží pro přijímač k synchronizaci.
- Vysílač mění data vždy po uplynutí časové periody T.
- Příjímač může převzít data bezpečně jen uprostřed časové periody vysílače. Počká tedy půl časové periody a převezme data.
- Dále přijímač počká celou periodu T, a pokud $conrol == 1$, přečte další data. Tento bod se znovu opakuje.

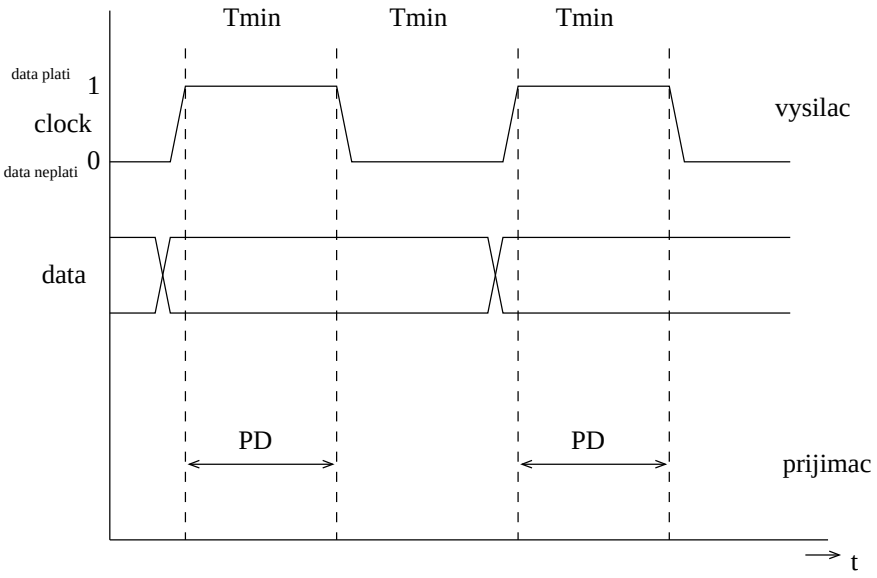
Časový průběh arytmického přenosu je na obrázku 3.2. Jednotlivé zkratky v obrázku mají následující význam:

- VD vysílač vymění (zapíše) data
- PD přijímač přijme (přečte) data
- S náběžnou hranou řídícího signálu se synchronizuje vysílač s přijímačem
- K konec přenosu
- T časová perioda, kterou musí být schopen příjímač půlit

3.3 Synchronní přenos (s vlastními hodinami)

Popis komunikace

Obdoba I^2C komunikace.



Obrázek 3.3: Časové schéma přenosu s hodinovým signálem

- Rychlost přenosu je dána hodinovým signálem. Ten nemusí být pravidelný.
- Data na lince musí být platná nejpozdější v okamžiku náběžné hrany hodin. Vysílač tedy musí data zapsat dříve, než změní *clock* z 0 na 1.
- Po dobu trvání $clock = 1$ se data na lince nesmí změnit.
- Vysílač nesmí zkrátit čas mezi náběžnými a sestupnými hranami signálu *clock* pod povolený limit $Tmin$.
- Přijímač čeká na náběžnou hranu *clock*. Neprodleně přebírá data a čeká na sestupnou hranu *clock*. Tím je přenos jednoho datového údaje ukončen.

Průběh komunikace v čase je zakreslen v obrázku 3.3. Zkratky použité v obrázku mají následující význam:

- PD čas po který smí přijímač přijmou (přečíst) data
- Tmin minimální časová perioda, kterou musí být schopen příjímač bezpečně rozeznat

3.4 Nepotvrzovaný přenos dat přes LPT

Princip

Klasický paralelní port umožňuje komunikovat pěti bity z vysílače na přijímač. Datová komunikace tak bude 4 bity a řídící signál. Pro přenos jednoho byte je nutno použít vždy dva cykly arytmického nebo synchronního přenosu. Jen se autoři musí dohodnout, zda byte pošlou „hlavou“, nebo „nohama“ napřed.

Úkoly

Napište CHAT pro dva počítače přes LPT. Přenos textu může být po řádcích, nebo po jednotlivých znacích.

Postup

- Obě strany se musí dohodnout, kdo je vysílač a kdo přijímač.
- Dle zadané metody zrealizovat přenos jednoho půl-bytu (nibble) - vytvořit samostatnou funkci.
- Zrealizovat přenos jednoho bytu - opět samostatná funkce!
- Přenos znaků zadávaných na klávesnici vysílače na přijímač a jejich zobrazování.
- Spojení dvou programů do jednoho - výsledkem by měl být CHAT pracující v obousměrném (full duplex) režimu.
- Zdatnější programátoři mohou i rozdělit obrazovku na dvě části.

Kapitola 4

Potvrzovaná komunikace - HandShake

HandShake, korespondenční režim, přenos s potvrzováním.

Při komunikaci mezi dvěma PC přes LPT zůstává pořád pět vodičů tam a pět zpět. Pro zpětné potvrzení stačí jeden bit, ale je tím znemožněna full-duplexní potvrzovaná komunikace. Vysílač používá patý bit pro řízení komunikace a přijímač opět pátým bitem odpovídá (v obrázcích je označeno jako Req a Ans). 4 datové bity slouží pro jednosměrnou komunikaci.

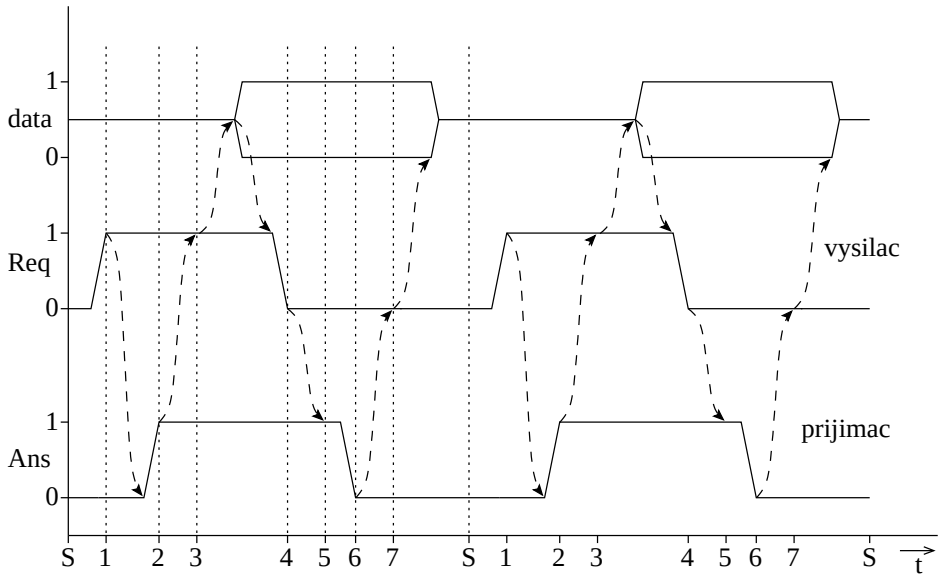
Celá komunikace je dána jako přesně definovaná posloupnost událostí. Nikde nejsou definovány žádné časové údaje, zpoždění nebo frekvence. Pomalejší z obou účastníků určuje rychlost přenosu, rychlejší se přizpůsobí.

4.1 Plný (dvojitý) HandShake

Popis komunikace

Úplný handshake je na obrázku 4.1 a je vidět chování vysílače a přijímače. Celý postup komunikace lze rozdělit do osmi kroků, během kterých se přenese jeden datový údaj, v našem případě 4 bity.

Úplný režim je výhodný pro bezpečnější režim přenosu dat. Je-li datová linka obousměrná, je tento režim nezbytný při vzájemné komunikaci, aby se nemohlo stát, že by vysílač i přijímač provedli zápis dat (viz bod 1. a 2.).



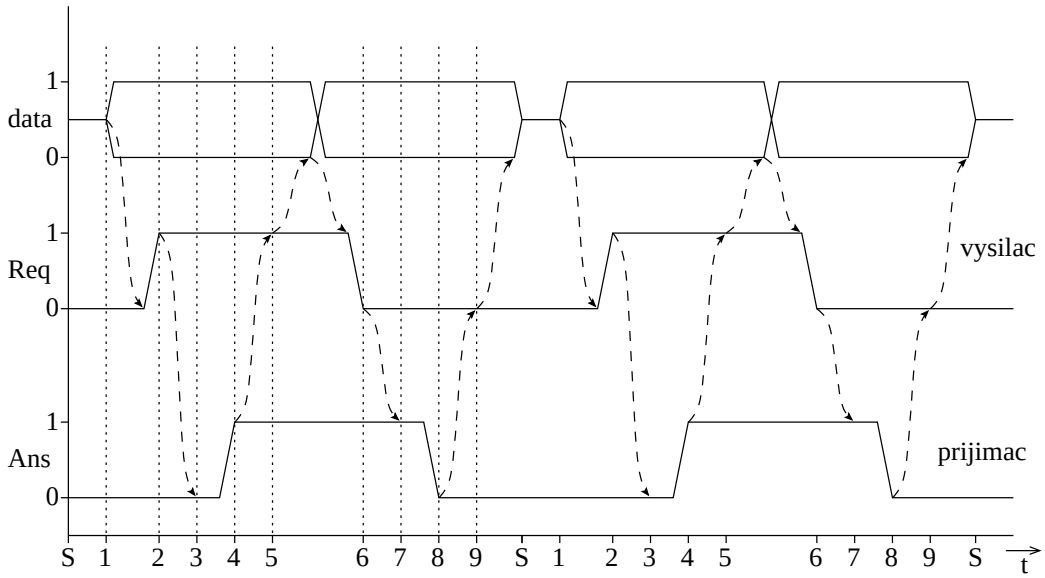
Obrázek 4.1: Časový průběh úplného potvrzovaného přenosu

	Vysílač	Přijímač
S.	Výchozí stav (Ans=Req=0)	
1.	Můžu předat data? (Req=1)	Čeká na dotaz pro zahájení přenosu
2.	Čeká na povolení zápisu dat	Potvrdí povolení zápisu dat (Ans=1)
3.	Zapíše data	
4.	Potvrdí platnost dat (Req=0)	Čeká na signál, že data platí
5.	Čeká na potvrzení převzetí dat	Data platí, probíhá převzetí dat
6.		Data byla převzata (Ans=0)
7.	Zneplatní data	
S.	Výchozí stav (Ans=Req=0)	

4.2 Zjednodušený (zrychlený) HandShake

Popis komunikace

Pokud požadujeme vyšší rychlost a máme jistotu, že datová linka je jednosměrná, můžeme handshake zjednodušit na potvrzování změnou.



Obrázek 4.2: Časový průběh zjednodušeného potvrzovaného přenosu

	Vysílač	Přijímač
S.	Výchozí stav (Ans=Req=0)	
1.	Zapíše data	Čeká na informaci o datech
2.	Data připravena (Req=1)	
3.	Čeká na potvrzení převzetí dat	Přebírá data
4.		Potvrdí převzetí (Ans=1)
5.	Zapíše další data	Čeká na informaci o datech
6.	Data připravena (Req=0)	
7.	Čeká na potvrzení převzetí dat	Přebírá data
8.		Potvrdí převzetí (Ans=0)
9.	Zneplatní data	
S.	Výchozí stav (Ans=Req=0)	

Úkoly

1. Změřte přenosovou rychlost mezi dvěma počítači.
2. Přeneste soubor z počítače na počítač.

Postup řešení

- Uveďte svůj počítač na začátku i na konci programu do výchozího stavu.
- Počkejte, až bude ve výchozím stavu i připojený počítač.

- Určete si, kdo je master (řídící,vysílací) a kdo slave (podřízený, přijímací).
- Napište podprogram, který provede jeden cyklus HandShake, bez přenosu dat (je výhodné použít tuto funkci na začátku programu pro synchronizaci).
- Napište podprogram, který přenese jeden byte.
- Napište podprogram, který přenese blok dat zadané velikosti (první 4 přenesené byty budou představovat velikost bloku)
- Použijte navrženou funkci pro realizaci vašeho úkolu.
- Spojte vaše programy do jednoho a na počátku si zvolte M/S.

Kapitola 5

Dekódování signálu dálkového IR ovladače

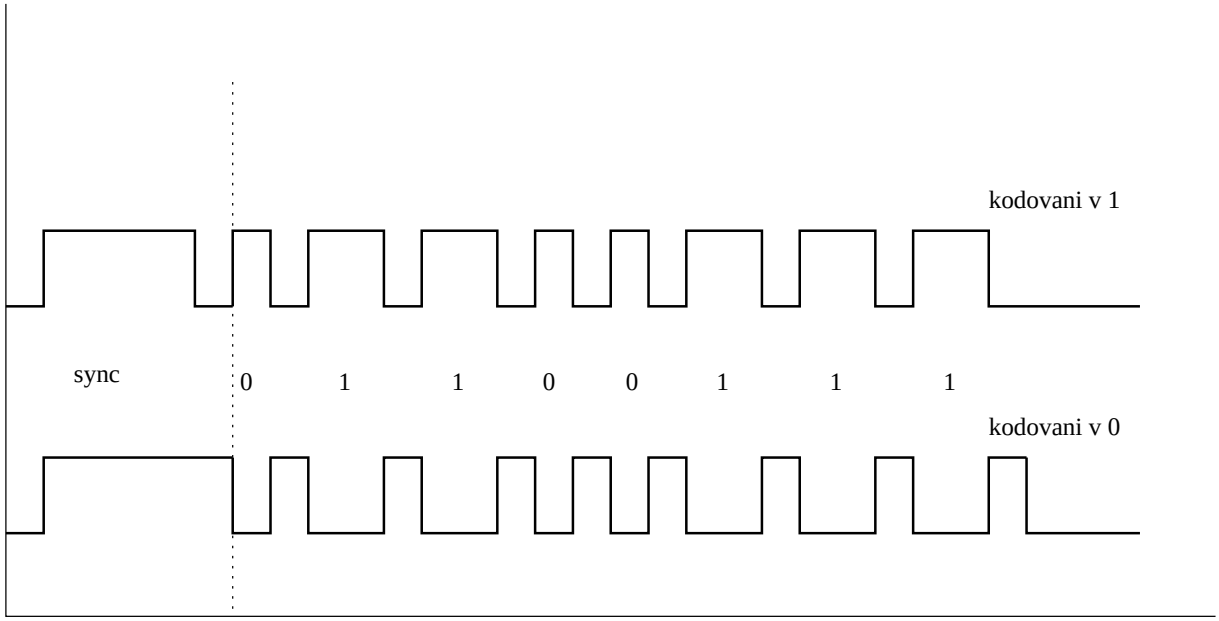
Princip

Dálkové ovladače pro spotřební elektroniku nejsou standardizovány, a proto nelze stanovit jednoznačný postup a algoritmus dekódování jejich signálu. Můžeme pouze shrnout základní charakteristiky, které pomohou při realizaci výsledného řešení:

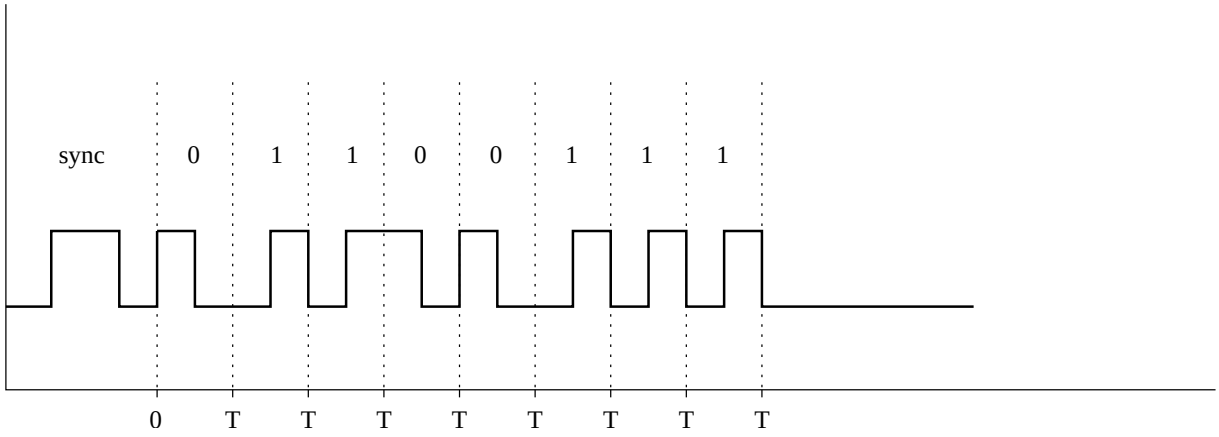
- Bitové informace 0/1 jsou kódovány rozdílnou délkou signálu, a to buď v úrovni 0 nebo 1. Tyto signály nesoucí informace, jsou odděleny oddělovači konstatní délky (v opačné logické úrovni). Příklady kódování osmibitového signálu "01100111" jsou na obrázku 5.1.
- Na počátku každého datového signálu je úvodní synchronizační signál. Ten může být složen z jednoho až dvou pulzů významně delších, než jsou následující datové signály. Některé ovladače ovšem úvodní synchronizaci nevysílají.
- Délka dat je různá. Typicky 8, 12, 16, 24 a 32 bitů.

Některé ovladače nevyužívají kódování délkou signálu, ale kódování změnou, tzv. Manchester. V takovém případě je signál vysílán s konstatní časovou periodou T a datová 0 se kóduje jako sestupná hrana a datová 1 jako hrana náběžná. Tato změna se sleduje vždy uprosřed časové periody. Na přelomu časové periody je přechod mezi úrovní 0/1 pouze v případě potřeby. Příklad zakódování stejné posloupnosti bitů jako v přechozím případě, je na obrázku 5.2.

Takovéto kódování umožňuje vysílat všechny kombinace signálů s konstantní délkou, ale vyžaduje synchronizaci přijímače.



Obrázek 5.1: Příklad kódování IR signálu v úrovni 0 a 1



Obrázek 5.2: Příklad kódování IR signálu změnou (Manchester)

Postup

Příklady programu pro toto cvičení obsahují funkci, jejíž návratová hodnota je počet mikrosekund od zapnutí počítače. Jde o 32 bitovou bezznaménkovou hodnotu.

Dále pak obsahuje funkci, která vrací hodnotu 0/1, podle toho, zda přijímač zaregistroval IR signál, či nikoliv.

Doporučený postup řešení.

- Nejprve napište program, který bude detekovat, zda dochází ke změnám signálu, pokud dálkovým ovladačem vysíláte libovolný signál.
- Napište program, který zaznamená čas vždy, když zaregistruje změnu signálu. Zaznamenejte 80-160 změn. Rozdíl času mezi dvěma změnami je délka signálu 0 nebo 1. POZOR! Během měření neprovádět žádné výpisy! Hodnoty ukládat do pole.
- Nakreslete si průběh signálu alespoň dvou tlačítek. Zkontrolujte, zda pro dané tlačítko máte vždy stejný výsledek.
- Rozhodněte se podle výsledného průběhu, o jaký způsob kódování jde.
- Dekódujte signál jako posloupnost 0 a 1.
- Vytvořte z dekódované posloupnosti 0 a 1 n-bitové číslo.
- Dekódujte alespoň 3 tlačítka. Program musí běžet v "nekonečném" cyklu.

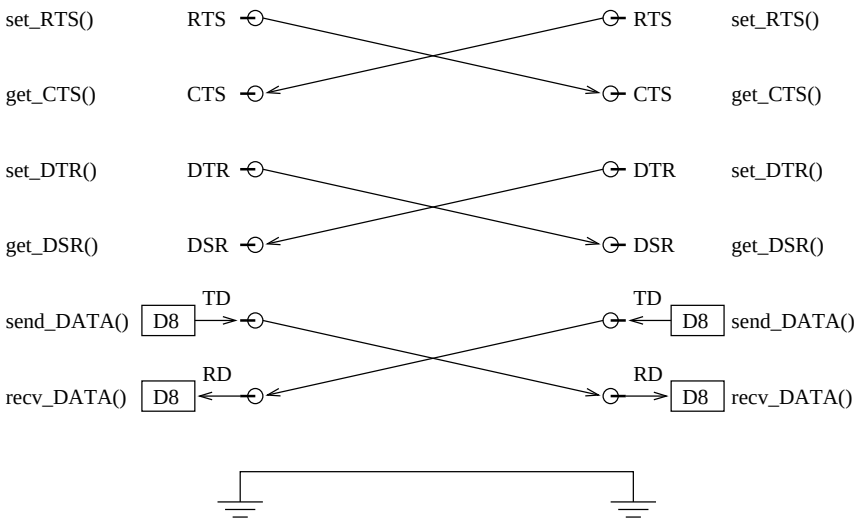
Kapitola 6

Sériový port - RS232

6.1 Popis rozhraní

Sériový port zůstává dlouhodobě nepostradatelným komunikačním rozhraním pro svou spolehlivost, jednoduchost a dobrou standardizaci. V počítačích PC je rozhraní RS232 realizováno obvodem USART (Universal Synchronous-Asynchronous Receiver/Transmitter) 16550. Podrobnější technickou specifikaci lze najít v mnoha technických manuálech, společně se zapojením pinů nejčastěji používaných konektorů.

Typické zapojení dvou sériovych portů pro vzájemnou komunikaci je na obrázku 6.1.



Obrázek 6.1: Zapojení sériových rozhraní

Význam jednotlivých zkratek uvedených v obrázku 6.1 je následující:

- **RTS** - (Request To Send) výzva k vysílání
- **CTS** - (Clear To Send) pohotovost k vysílání
- **DTR** - (Data Terminal Ready) pohotovost koncového zařízení
- **DSR** - (Data Set Ready) pohotovost ukončujícího zařízení
- **TD** - (Transmit Data) vysílaná data
- **RD** - (Receive Data) přijímaná data

V obrázku 6.1 jsou také vidět posuvné datové registry označené jako **D8**. Každé sériové rozhraní musí mít dva takovéto registry, aby mohlo probíhat odesílání i příjem dat současně.

Na obrázku 6.1 jsou také přiřazeny jednotlivým signálům funkce pro programování komunikace - `set_RTS()`, `set_DTR()`, `get_CTS()`, `get_DSR()`, `send_DATA()` a `recv_DATA()`. Význam funkcí je zřejmý. Uvedených 6 funkcí musíme ještě rozšířit o dvě funkce:

- `ready_SEND()` - je posuvný registr vysílače prázdný?
- `ready_RECV()` - jsou v datovém registru přijímače data?

Tyto dvě funkce mají zásadní význam pro korektní řízení komunikace. Rychlost sériového rozhraní je v porovnání s rychlostí práce počítače velmi malá, musíme proto VŽDY před čtením či zápisem z/do datového registru přijímače/vysílače zkontrolovat, zda je registr připraven. V případě vysílače musíme mít vždy jistotu, že dříve zadaná data byla odvysílána a registr je prázdný. U přijímače nemůžeme data číst dříve, než jsou přijaty všechny bity.

Poslední funkcí je `set_SERIAL(rychlost, data, stopb, parita)`. Na obou stranách komunikační linky musí být oba sériové porty nastaveny shodně, aby mohla komunikace probíhat. Bude-li se nastavení lišit, nebude komunikace probíhat vůbec, nebo budou přijatá data nesmyslná. Na straně vysílače se ovšem chyba nijak neprojeví. Parametry sériové linky jsou následující:

- **rychlost** - rychlost [baud/s] v rozsahu $1200 \div 115200$.
- **data** - počet datových bitů $5 \div 8$.
- **stopb** - počet stop bitů.
- **parita** - sudá, lichá, nebo žádná.

Předdefinované konstanty pro jednotlivé parametry jsou v přiloženém programu v jazyce C.

6.2 Úkol

Naprogramujte jednoduchý CHAT pro dva počítače propojené sériovou linkou. Data neodesílejte po znacích, ale po celých řádcích. Aby se na obrazovce „nepromíchávaly“ řádky textu odesílané a přijímané, použijte řídící signály RTS/CTS.

6.3 Postup řešení

- Nastavte na obou komunikujících stranách sériovou linku stejnými parametry.
- Odešlete znak a zkontrolujte, zda jej druhá strana korektně přijala.
- Odešlete více znaků za sebou a ověřte, zda byly všchny korektně přijaty.
- Čtěte znaky z klávesnice a odesílejte je. Zároveň čtěte a zobrazujte znaky přijaté po lince.
- Načtěte celý řádek znaků a teprve po stisku klávesy *Enter* je odešlete.
- Před odesláním řádku kontrolujte stav signálu CTS, zda je druhá strana připravena data přijímat.
- Svým signálem RTS po dobu psaní řádku textu nepovolte druhé straně odesílání.

Kapitola 7

Sběrnice I^2C

7.1 Popis sběrnice

Sběrnice I^2C byla navržena firmou Philips Semiconductors pro komunikaci jednočipových mikropočítčů s dalšími číslicovými obvody.

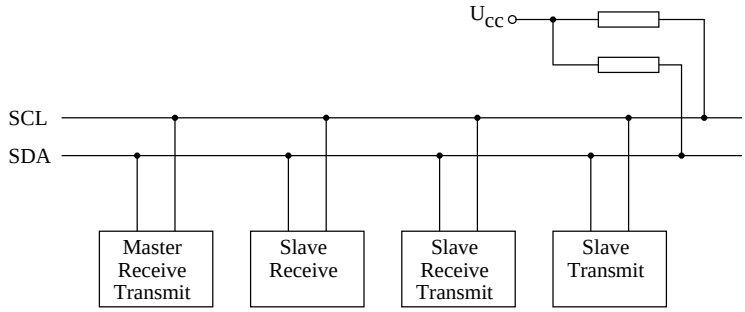
I^2C je obousměrná dvouvodičová sběrnice. Někdy bývá ale nesprávně označována jako sériová linka. Zkratka I^2C označuje *Inter Integrated Circuit Bus*. Český překlad by mohl být *meziobvodová sběrnice*.

Základní charakteristika

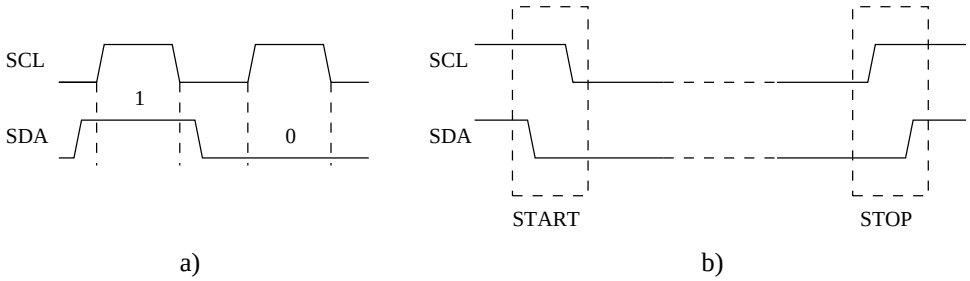
Sběrnice se skládá ze dvou linek. **SDA** slouží pro obousměrný přenos dat a **SCL** linka slouží jako hodinový signál. Obě linky pracují v napěťových logických úrovních shodně s technologií TTL.

V klidové stavu jsou obě linky v úrovni log. 1. Tento stav je udržovány dvěma zdvihacími (pull-up) rezistory. Výstupy číslicových obvodů jsou řešeny jako výstup typu *otevřený kolektor*. Tím je zaručena obousměrnost linky.

Základní uspořádání systému můžeme vidět na obrázku 7.1. Na sběrnici může být připojeno více zařízení. Obvykle jedno zařízení označované jako *Master* řídí vysílání a příjem zpráv. Další zařízení, označovaná jako *Slave*, po výzvě nadřízeným obvodem data přebírají (receive), nebo odesílají (transmit). Na sběrnici smí být i více zařízení typy *Master*, ale taková zapojení již nepředstavují jednoduché systémy vhodné pro výuku.



Obrázek 7.1: Uspořádání typického systému se sběrnicí I^2C



Obrázek 7.2: Průběhy signálů na sběrnici I^2C

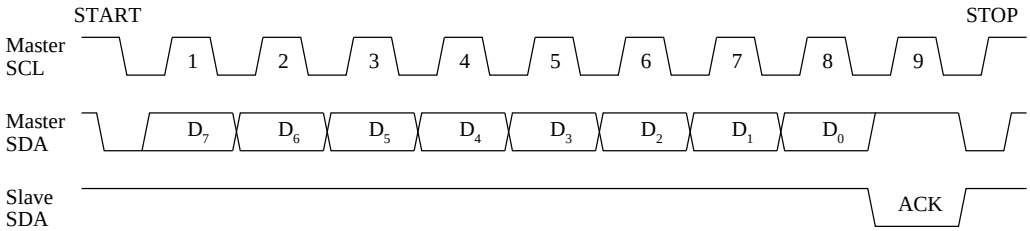
Přenos bitů

Přenos bitů probíhá na sběrnici synchronně. Pojem synchronní komunikace byl probírán již v jedné z předchozích kapitol. V praxi to znamená, že jeden bit je přenesen vždy v jednom hodinovém cyklu. Jak vypadá přenos hodnoty 1 a 0 můžeme vidět na obrázku 7.2a. Z obrázku je patrné, že datové signály se mění vždy jen v době, kdy je signál SCL v úrovni log. 0. Pokud dochází ke změně signálu SDA v době, kdy ke úroveň signálu SCL v log. 1, jde o signály řídící.

Řídící signály START a STOP

Vzhledem k počtu vodičů sběrnice I^2C, můžeme definovat jen dva řídící signály. Těmi jsou signály START (S) a STOP (P). Jak vypadá průběh těchto signálů můžeme vidět na obrázku 7.2b. Slovní popis je následující: z klidového stavu, kdy signál SDA i SCL jsou v log. 1 musíme přejít do komunikačního režimu signálem START. Tím rozumíme přechod signálu SDA z log. 1 na log. 0 při neměnné hodnotě signálu SCL v log. 1. Pak teprve musí následovat i přechod signálu SCL z log. 1 na log. 0.

Chceme-li ukončit komunikaci, musíme uvést obě linky SDA i SCL do



Obrázek 7.3: Odeslání jednoho bajtu i s potvrzením na sběrnici I^2C

klidového stavu. Učiníme tak signálem STOP, který představuje nejprve nastavení SCL na log. 1 a poté nastavení i SDA na log. 1.

Komunikace po bajtech

Při přenosu dat mezi signály START a STOP není množství přenesených dat nijak omezeno. Data se ovšem nepřenáší po jednotlivých bitech, ale po celých bajtech. Průběh celého přenosu je vidět na obrázku 7.3. Z průběhu signálů a jejich popisu je patrné, že významově nejvyšší bit se přenáší jako první.

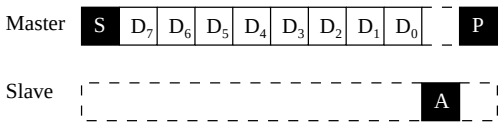
Povrzování

Na obrázku 7.3 je zobrazen nejen přenos jednoho bajtu, ale i další důležitá součást přenosu informace. Tou je potvrzování. Každý obvod musí příjem bajtu povrdit bitem ACK, což je log. 0 linky SDA v následujícím hodinovém cyklu po odvysílání bajtu. Potvrzení může být i NACK, tedy negovaný ACK, což je hodnota log. 1 linky SDA (master tak naznačuje, že ukončuje přenos dat).

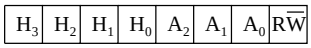
Upozornění: všimněte si na obrázku 7.3 chování vysílače (Master) v devátém hodinovém cyklu, kdy se očekává signál ACK od přijímače (Slave). Vysílač nastaví SDA na log. 1, aby jeho výstup neovlivňoval linku SDA (viz výše informace o výstupu typu otevřený kolektor). **Master musí nastavit SDA na log. 1 před příjmem každého bitu od podřízeného obvodu.**

Zjednodušené zobrazení

Pro další popis komunikace budeme používat zjednodušené zobrazení, jako na obrázku 7.4.



Obrázek 7.4: Zjednodušené zobrazení průběhu komunikace na lince I^2C



Obrázek 7.5: Formát adresy obvodu na I^2C

Adresování

Na sběrnici I^2C může být připojeno více obvodů současně, proto musí mít každý takto připojený obvod svou jednoznačnou adresu, aby jej bylo možno „oslovit“.

Každá adresa má osm bitů, které jsou rozděleny podle obrázku 7.5. Bity označené $H_0 \div H_3$ jsou dány výrobcem a nelezneme je vždy v technickém manuálu daného obvodu. Bity $A_0 \div A_2$ si při zapojení obvodu nastaví uživatel. Poslední bit adresy je $R/\overline{W}$. Tímto bitem říkáme, zda následující data budeme z pohledu řídícího obvodu do příslušného obvodu zapisovat ($R/\overline{W} = 0$), nebo budeme data číst ($R/\overline{W} = 1$).

7.2 Digitální teploměr Dallas DS1621

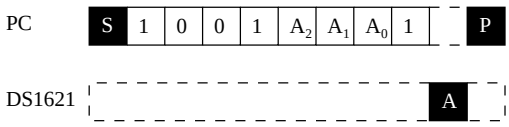
Nedílnou a nezbytnou součástí následujícího textu je i technický manuál výrobce **DS1621.pdf**.

Obvod DS1621 je digitální teploměr, který může pracovat jako teploměr pro spojité měření teploty, nebo provádět měření teploty na požádání, nebo pracovat jako termostat s nastavitelnou hysterezí.

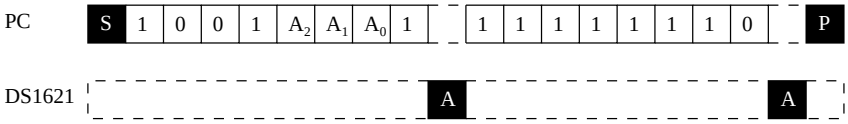
Pro výukové účely budeme preferovat použití teploměru pro spojité měření teploty.

Formát teploty

Na straně 4 manuálu DS1621 je uveden digitální formát naměřené teploty. Jde o devítibitové znaménkové číslo, kde nejnižší bit představuje rozlišení půl stupně. Protože devíti bity je překročen formát jednoho bajtu, je naměřená teplota předávána ve dvou bajtech za sebou.



Obrázek 7.6: Příklad „zaadresování“ I^2C obvodu



Obrázek 7.7: Zaslání příkazu *Start Convert* (EEh) obvodu DS1621

Adresa

Na straně 8 manuálu DS1621 je uvedena adresa přidělená výrobcem pro nevyšší 4 bity adresy obvodu. Je to hodnota **1001**. Zbylé tři bity adresy se nastavují instalovanými přepínači přímo u teploměru. Požadovaná hodnota bude zadána přímo na cvičení.

Příkazy

Na straně 10 manuálu DS1621 jsou uvedeny všechny příkazy, kterými se teploměr ovládá. Pro účely cvičení budou potřebné 4 základní.

Příkaz **ACh** (Access Config) slouží pro nastavení chování teploměru. Formát konfiguračního slova je na straně 5 manuálu DS1621. Pokud má obvod pracovat jako teploměr, je nejdůležitější bit označený jako *1SHOT*, kterým se určuje, zda teploměr bude měřit teplotu spojitě, nebo na požádání.

Příkaz **EEH** (Start Convert) spustí převod teploty.

Príkaz **22h** (Stop Convert) zastaví převod teploty.

Příkaz **AA** (Read Temperature) slouží pro čtění teploty. Obdrží-li obvod tento příkaz, vyšle následně 2 bajty s naměřenou teplotou.

Příklad komunikace

Nejjednodušším příkladem komunikace je ověření („zaadresování“), zda daný obvod má skutečně přidělenou adresu dle dokumentace a uživatelského nastavení. Povinností obvodu, který přijme jeden bajt, je příjem potvrdit signálem ACK. Pokud má náš obvod DS1621 adresu $A_0 \div A_2$, pak musí na zaslání bajtu dle obrázku 7.6 odpovědět. Tím máme jistotu, že obvod komunikuje.

Chceme-li předat obvodu nějaký příkaz, například požadavek na spuštění převodu teploty, zašleme příkaz **EEh** na požadovanou adresu. Postup je na obrázku 7.7.

Další přesný popis všech komunikačních možností s obvodem DS1621 naleznete v dokumentaci DS1621 na straně 9.

Programátorské rozhraní

Pro komunikaci s obvody pro I^2C je na PC vhodné rozhraní LPT. Z datových pinů lze některé obvody přímo napájet a není tak třeba externí zdroj a navíc má několik obousměrných vývodů, které jsou vhodné pro simulaci SDA a SCL signálů. Následující skupina funkcí je již implementována:

void LPT_Start() - inicializace rozhraní PC, nutno zavolat na začátku každého programu jako první funkci.

void I2C_SDA(*int value*) - nastavení hodnoty SDA na 0 nebo 1.

int I2C_SDA() - přečtení stavu linky SDA.

void I2C_SCL(*int value*) - nastavení SCL na 0 nebo 1.

void I2C_Init() - inicializace, signály SDA a SCL jsou nastaveny na 1.

void I2C_Start() - vygenerování sekvence START.

void I2C_Stop() - vygenerování sekvence STOP.

void I2C_Ack() - odeslání ACK.

void I2C_NAck() - odeslání NACK.

int I2C_getAck() - přečtení ACK.

int I2C_Vystup() - odeslání jednoho bajtu a převzetí ACK.

int I2C_Vstup() - NUTNO IMPLEMENTOVAT.

Úkoly na cvičení

- Prostudování dokumentace a připravených programů je součástí přípravy před cvičením.
- Prvním úkolem na cvičení bude nastavit obvodu DS1621 požadovanou tří-bitovou adresu.
- Programem ověřit, zda je obvod na požadované adrese. Ten musí odpovědět signálem ACK.
- Implementovat funkci pro příjem dat ze sběrnice.
- Nastavit teploměr pro spojitý převod teploty.

- Spustit převod teploty.
- Číst aktuální teplotu.

UPOZORNĚNÍ: jak již bylo uvedeno v předchozím textu, nezapoměňte při implementaci příjmu dat ze sběrnice na fakt, že před každým čtením dat z linky SDA musíte uvést tuto linku do klidového stavu log. 1!