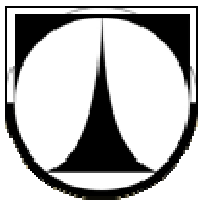
TECHNICKÁ UNIVERZITA V LIBERCI
TECHNICAL UNIVERSITY OF LIBEREC



FAKULTA STROJNÍ

KATEDRA APLIKOVANÉ KYBERNETIKY

FACULTY OF MECHANICAL ENGINEERING
DEPARTMENT OF APPLIED CYBERNETICS

APLIKACE PRO TERMINÁL SBĚRU DAT Z VYROBNÍHO ZAŘÍZENÍ
APPLICATION TO PRODUCTION EQUIPMENT DATA COLLECTION TERMINAL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Denys Bilousov**
AUTHOR

VEDOUCÍ PRÁCE **Ing. Michal MOUČKA, Ph.D.**
SUPERVISOR

KONZULTANT **Ing. Miroslav SMEJKAL**

LIBEREC 2011

ZADÁNÍ

ABSTRAKT

Obsahem této práce je návrh a realizace programového vybavení pro terminál sběru dat z vyrobního zařízení. Terminál je schopný provádět sběr dat o stavu vyrobního zařízení, vypočítávat čas prostoje strojů, zaznamenat čas příchodu a odchodu pracovníka, obsluhující stroj, vyměňovat si údaje s PC.

KLÍČOVÁ SLOVA

Terminál, RFID, LCD, senzorový panel, DS1337, I2C, SRAM, USART, AD-převodník, RS232, RS422, Atmega1280.

ABSTRACT

The contents of this thesis is to design and implementation a software for the terminal which collect data from production equipment. The terminal can collect status data of production equipment, to count equipment downtime, record the time of employee arrival and departure, data exchange with PC.

KEYWORDS

Data terminal, RFID, LCD, touchpad, DS1337, I2C, SRAM, USART, AD converter, RS232, RS422, Atmega1280.

PROHLÁŠENÍ O VYUŽITÍ DIPLOMOVÉ PRÁCE

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č.121/2000 o právu autorském, zejména § 60 (školní dílo) a § 35 (o nevýdělečném) užití díla k vnitřní potřebě školy).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé práce a prohlašuji, že **souhlasím** s případným užitím mé práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

V Liberci dne…….. …….

(podpis autora)

MÍSTOPŘÍSEŽNÉ PROHLÁŠENÍ

Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury, a na zakladě konzultací s vedoucím diplomové práce a konzultantem.

V Liberci dne…….. …….

(podpis autora)

OBSAH

PŘÍLOHY NA CD:

Příloha 1 – Zdrojový text v jazyce C.

Příloha 2 – Aplikace.

1 ÚVOD

Dnešní etapa rozvoje výroby s velmi vysokou rychlostí rozvoje vědy a techniky je charakterizována aktivně probíhajícím postupem informatizace. Hodnota informace prudce vzrostla. Informace jsou údaje o skutečných datech a souhrn znalostí o souvislostech mezi nimi, jež jsou předmětem uchování, zpracování a přenosu.

Informatizace výroby je proces integrace software do výrobního procesu a vytvoření výrobní evidence. Čím větší je podnik, obsah vykonávaných prací, počet evidovaného zařízení, tím vyšší je potřeba v informatizaci. S nárůstem obsahu informace vzniká před společností otázka o nedostatečné kvalitě dat, které pracovníci evidují pomocí pomocných prostředků, např. pomocí tabulkového procesoru MS-EXCEL. Takto nashromážděné informace jsou velmi nízké kvality a je nemožné je slučovat a efektivně reprezentovat. Současné chápání informačního systému předpokládá využití PC jako základního technického prostředku pro zpracování informace.

Cílem práce je vytvořit programové vybavení pro terminál sběru dat z výrobních zařízení pro řešení problému nedostatku dat nezbytných pro optimalizaci výroby. Tato práce je vypracována na základě zadání firmy Juta a.s., která se zabývá textilní výrobou. Vedení společnosti se snaží sestavit celkovou situaci výroby. Proto třeba realizovat sběr dat o prostojích zařízení a o jejich příčině. Sběr dat je možné provádět pomoci tužky a papíru, a pak je vynášet do tabulek v počítači. Ale v společnosti, která má více než 10 závodů, takový sběr dat je pomalý a obtížný, protože je dat mnoho. Jejich zpracování musí být rychlé. Pro řešení tohoto problému byl ve firmě navržen terminál, pro sběr dat ze zařízení a jejich následnou archivaci v databázi na PC.

Vstupními daty pro terminál jsou stav zařízení v pracovní době, čas příchodu a odchodu pracovníka obsluhující stroj. Výstupem jsou pakety dat pro uložení v tabulkách databáze na PC.

Terminál byl vyvinut na základě mikrokontroléru Atmega1280. To je spolehlivý integrovaný obvod s rozhraním USART pro realizaci čtení RFID-značek a vysílání dat do PC pomoci sběrnice RS422, AD-převodník, rozhraní I2C pro komunikaci s modulem hodin reálného času. Atmega1280 taky má dostatečný počet vstupů pro připojení vnější paměti RAM a pro připojení ke stroji. Výhodou obvodu je, že je možné pro něj na internetu bezplatně stáhnout programové prostředí CodeVisionAVR a AVRstudio.

2 TERMINÁL

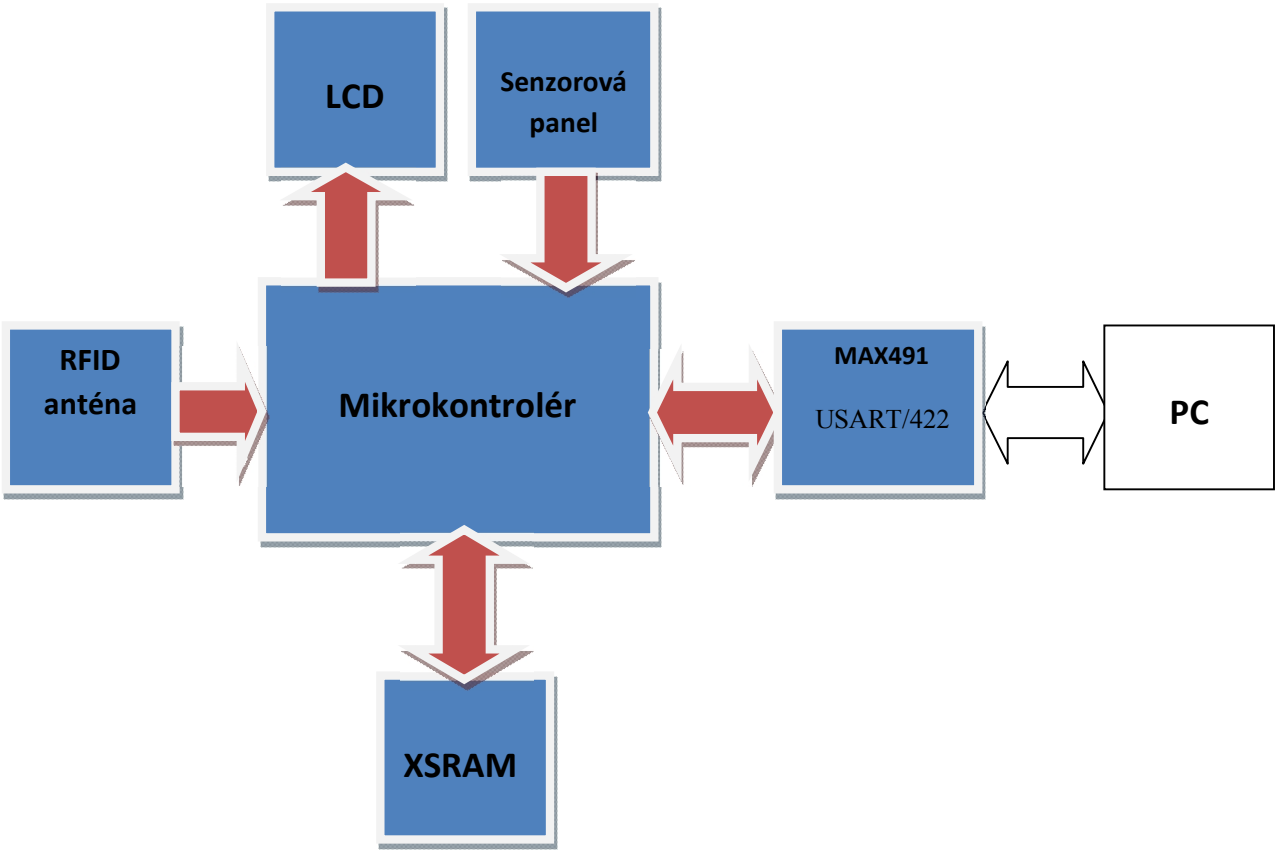
Obr.2.1: Vnější vzhled terminálu

Terminál musí být umístěn na pracovišti s pohodlným přístupem pracovníků. Jeden terminál obsluhuje 4 stroje. Data z terminálu se vysílají do PC přes rozhraní RS422, proto je potřeba využití převodníku RS422/232. Vstupní data pro terminál jsou identifikační číslo pracovníka, čas příchodu a odchodu pracovníka z pracoviště, stav výrobního zařízení v pracovní době. Výstupní – připravené pakety dat pro uložení v tabulkách databáze.

2.1 Struktura terminálu

Terminál se skládá z několika základních částí:

1. Mikrokontrolér
2. LCD
3. Senzorový panel
4. RFID-anténa
5. Obvod hodin reálného času
6. Externí paměť”
7. Obvod MAX491
8. Jako pomocný prvek pro komunikaci terminálu s PC je použit převodník RS422/232.



Obr.2.2: Blokové schéma terminálu

2.2 RFID-anténa

RFID (angl. ***R**adio **F**requency **ID**entification*, identifikace na rádiové frekvenci) — metoda automatické identifikace objektů, kterou se prostřednictvím rádiového signálu čtou nebo zapisují data, chránící se v takzvaných transpondérech nebo RFID-značkách.

Jakýkoliv RFID-systém se skládá ze senzorového zařízení (čítač, reader nebo dotazovač) a transpondéru (RFID-značka, občas se taky používá název RFID-tag).

Většina RFID-značek se skládá ze dvou jednotek. První — integrovaný obvod (IO) pro uchování a zpracování informace, modelování a demodelování radiofrekvenčního (RF) signálu a některých jiných funkcí. Druhá — anténa pro příjem a vysílání signálu.

Podle typu zdroje napájení RFID-značky třídí :

- Pasivní
- Aktivní
- Polopasivní

Pasivní RFID-značky nemají vestavěný zdroj energie. Elektrický proud, indukovaný v anténě elektromagnetickým signálem od čítače, zabezpečuje dostatečný výkon pro fungování křemíkového CMOS-čipu, rozmístěného ve značce, a vysílání odvetného signálu.

Aktivní RFID-značky mají vlastní zdroj napájení a nezávisí na energii čítače, v důsledku čeho se čtou ve vzdálenosti, mají větší velikosti a mohou mít doplňkovou elektroniku. Nicméně jsou takové značky nejdražší a baterie mají omezenou dobu životnosti.

Polopasivní RFID-značky, tak zvané poloaktivní, jsou velmi podobné pasivním značkám, jsou ale vybaveny baterií, jež zajišťuje napájení čipu. Přitom vzdálenost účinku těchto značek závisí pouze na citlivosti přijímače čítače a proto můžou fungovat na velké vzdálenosti a s lepšími charakteristikami.

V tomto projektu se používají pasivní RFID-značky a anténa CORE-12 s provozní frekvencí 125kHz. Každá značka obsahuje v sobě individuální kód, který se skládá z 10 ASCII znaků. Anténa se připojuje do mikrokontroléru do vstupu RXD univerzálního asynchronního přijímače/vysílače USART1.

Formát dat vypadá takto:

STX (02h)	DATA (10 ASCII)	CHECK SUM (2 ASCII)	CR	LF	ETX (03h)

Kde: STK – symbol začátku textu

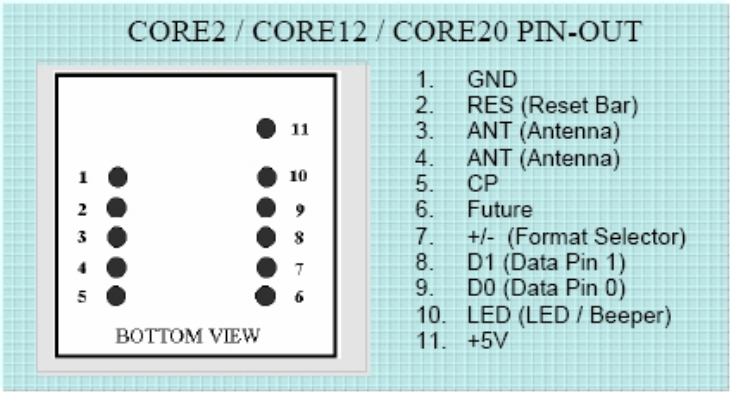
DATA – 10 ASCII-znaků

CHECK SUM – kontrolní součet (2 ASCII-znaků)

CR – návrat

LF – řádkování

ETX – konec textu



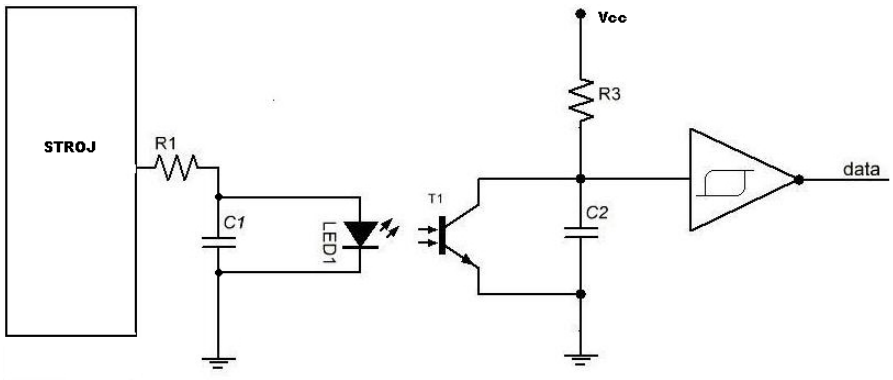
Obr.2.3: Pohled zdola CORE-2

Tab.2.1: Hlavní parametry antény CORE-12

Parametry	**CORE-12RW**
Čtecí vzdálenost	12+cm (Unique format)
Rozměry	26mm x 25mm x 7mm
Frekvence	125kHz
Karty formátu	Temec Q5555
Kódování	Manchester modulus 64
Příkon	5VDC , 30mA nominal
Output proud	-
Rozsah napětí	+4.6V ÷ +5.4V
Cívka	-

2.3 Sběr dat ze strojů

Každý terminál může číst data maximálně z 4 strojů. Do pinů 0,1,2 a 3 portu K přicházejí data o činnosti zařízení ve tvaru diskrétních stavů: výšková fronta - stroj funguje, dolní fronta - zastavený. Pro vytváření pravoúhlého signálu se na vstupu kontroléru používá klopný obvod, a pro galvanické oddělení – optron. Na obr.2.4 je uvedená schéma zapojení stroje do jednoho ze vstupů mikrokontroléru.



Obr. 2.4: Sběr dat ze stroje

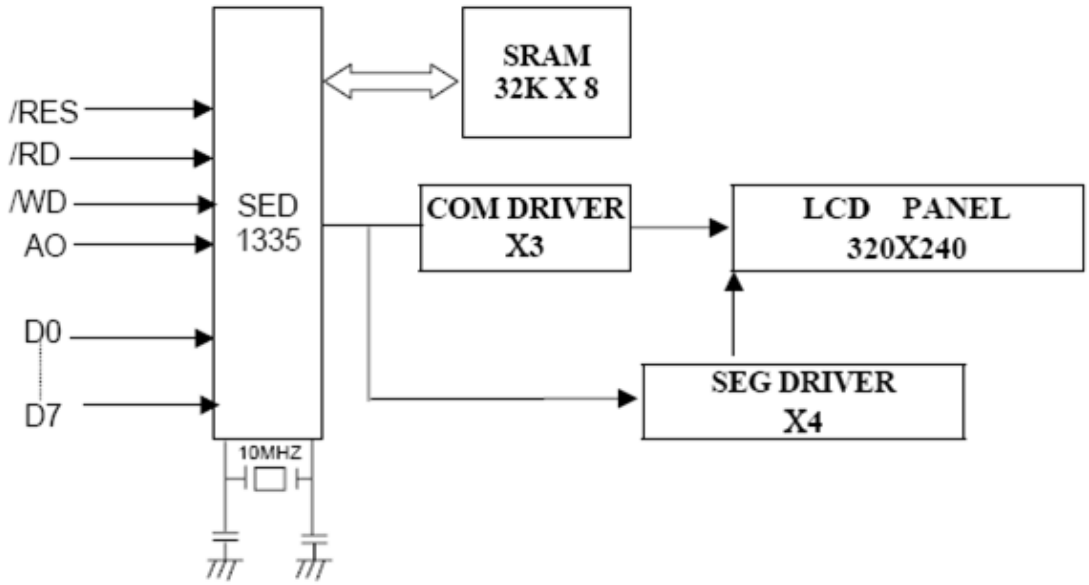
2.4 LCD

Digitální displej je formátu 320x240 elementů (1/4 VGA). Díky malým rozměrům, malé hmotnosti a tenkému profilu, nízké spotřebě a velkému objemu zobrazených grafických a textových informací našli tyto displeje široké použití v mobilních zařízení sběru a zpracování informací s autonomním napájením, v měřicích zařízení, v lékařské technice a jiných malorozměrných výpočtových zařízeních.

V projektu je použito LCD-displej GM322402 se vestavěným kontrolérem SED1335.

Základní parametry displeje:

Formát displeje:	320 pixelů (W) x 240(H) pixelů
Velikost pixelů	0.33(W) x 0.33(H) mm
Oblast zobrazení:	122(W) x 92(H) mm
Vnější velikost:	167.1(W) x 109.0(H) x 12.0(T) mm Max



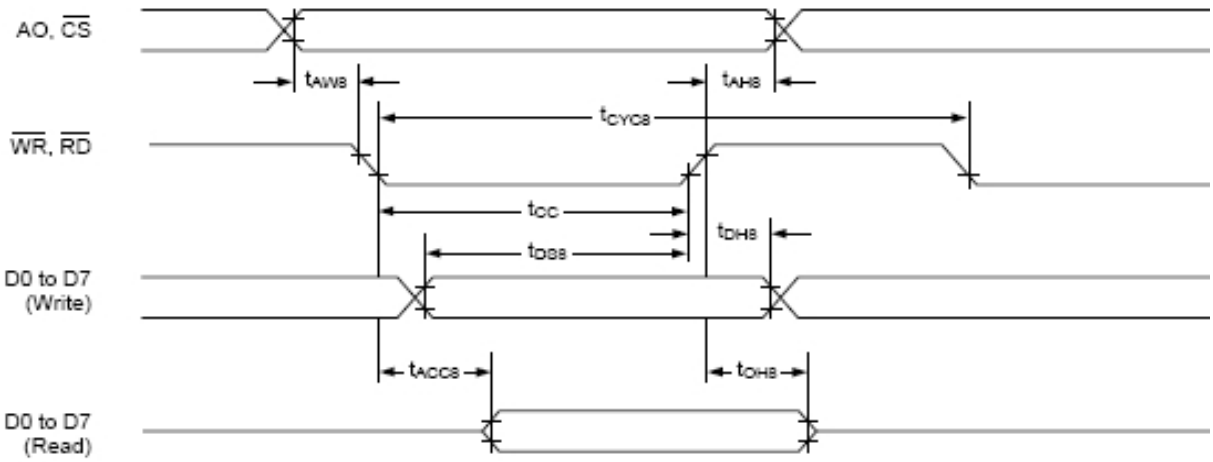
Obr.2.5: Blokové schéma LCD-displeje

Ovládání displejem je prováděno kontrolérem SED1335 prostřednictvím signálů, uvedených v tabule 7.1.

Tab.2.2: Signály ovládání

Číslo kontaktu	**signál**	**Úroveň H(vysoký)/L (nízký)**	**Popis**
1	/RESET	H/L	Nulování(Reset Signal)
2	/RD	H/L	80 řada: signál čtení (Read Signal). 68 řada: signál povolení (Enable Signal)(E)
3	/WR	H/L	80 řada: signál záznamu (Write Signal) 68 řada: signál čtení/zápisu (R/W Signal)
4	/CS	H/L	Signál výběru kristalu (Chip Select Signal)
5	A0	H/L	Výběr typu přenosu – data/příkaz (Data Type Selection)
6 ~ 13	DB0~DB7	H/L	Vysílaná data 8 bit (Data Input 8 bits)
14	VCC	-	Napětí napájení (+3–5 B) (Power Supply for Logic)
15	VSS	-	«Země» (Ground, 0 B)
16	VCTL	-	Hladina kontrastu (Contrast Adjustment Input)
17	EL_ON	H/L	Osvětlení zapnutí./vypnutí. Zapnutí – H; Vypnutí – L (EL On/Off Signal; H: EL On L: EL Off)
18	/DISPOFF	H/L	Vypnutí displeje (Display Off Function)

V displeji jsou realizovány dva typy vnějšího signálního rozhraní pro řízení displejem — od mikroprocesoru řad 8080 a 6800. Předvolený je signální rozhraní 8080, který budeme nadále používat. Časové diagramy předávání příkazů a dat pro rozhraní 8080 jsou uvedeny na obr. 2.6.



Obr.2.6: Časové diagramy

V tab.2.3 jsou uvedeny úrovně signálů pro zabezpečení režimů záznamu a čtení parametrů a dat, a v tabulce 2.4 — časové parametry signálů.

Tab.2.3: Úrovně signálů.

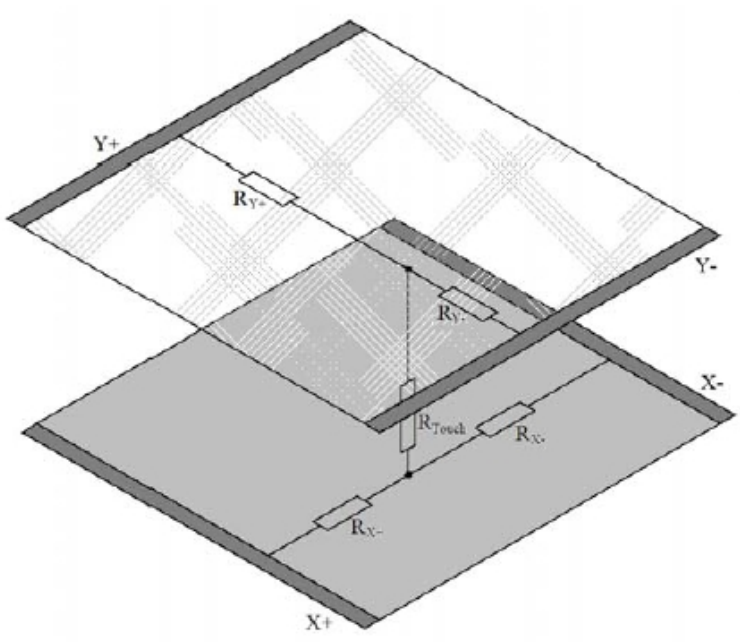
A0	/RD	/WR	**Popis**
0	0	1	čtení flagu statusu
1	0	1	čtení dat z displeje a adresy kurzoru
0	1	0	záznam dat a parametrů
1	1	0	záznam příkazů

Tab.2.4: Časové parametry signálů

Parametr	Podmínka	Označení	Min	Max	Jednotka měření	Poznámka
Address Hold Time	CL=100 pF VDD=2.7~4.5	tAH8	10		ns	A0,/CS
Address Setup Time		tAW8	0		ns	
System Cycle Time		tCYC	Note		ns	/WR,/RD
Strobe Pulse Width		tOC	150		ns	
Data Setup Time		tDS8	120		ns	DB0~DB7
Data Hold Time		tDH8	5		ns	
Access Time		tACC8	-	80	ns	
Output Disable Time		tOH8	10	55	ns	

2.5 Senzorový panel

Pro výběr jednotlivých položek menu na displeji používáme senzorový panel AVR341. AVR341 – Odporový senzorový panel, který se skládá ze skleněného panelu a ohebné plastové membrány. Na panel a na membránu je naneseno odporové pokrytí. Prostor mezi sklem a membránou je zaplněn mikroizolátory, jež jsou rovnoměrně rozloženy po aktivní oblasti panelu a spolehlivě izolují. Když se na panel tlačí, panel a membrána se spojí, kontrolér pomocí analogovo-digitálního převodníku registruje změnu odporu, převádí změnu odporu a konvertuje ji do souřadnic dotyku (X a Y).



Obr.2.7: Princip činnosti 4-vodičového odporového senzorového panelu

Algoritmus čtení:

1. Na horní elektrodu je připojeno napětí +5V, dolní elektroda je uzemněna. Levý s pravým se spojují nakrátko a se kontroluje napětí, které na nich je. Toto napětí odpovídá Y-souřadnice panelu.
2. Analogicky se do levého a pravého elektrodu podává +5V a zem, z horního a dolního se odečítá X-souřadnice.

V sestavě Atmega1280 je 10-bitový AD-převodník postupného přiblížení. Základní parametry tohoto AD-převodníku:

- absolutní chyba ±2 LSB;
- integrační nelinearita ±0.5 LSB;
- rychlost až 15 tis. snímání/s.

Jako zdroj referenčního napětí se pro AD-převodník může používat jak napětí napájení mikrokontroléru, tak i vnitřní nebo vnější zdroj referenčního napětí.

Během činnosti může AD-převodník fungovat ve dvou režimech:

- režim jednotlivého převádění, kdy se spuštění každého převádění iniciuje uživatelem;
- režim nepřetržitého převádění, kdy se spuštění převádění provádí nepřetržitě po určitý časový interval.

2.6 Hodiny reálného času

2.6.1 Obvod DS1337

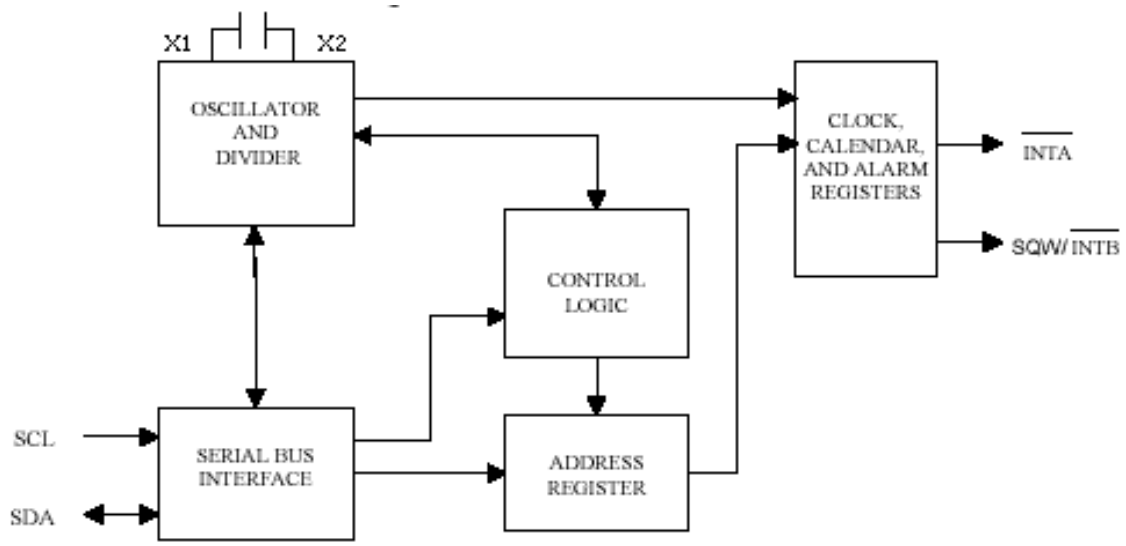
DS1337 – integrovaný obvod hodin reálného času nejmenších rozměrů s 2-drátovým rozhraním a dvěma signály varování denně.

Odlišnosti:

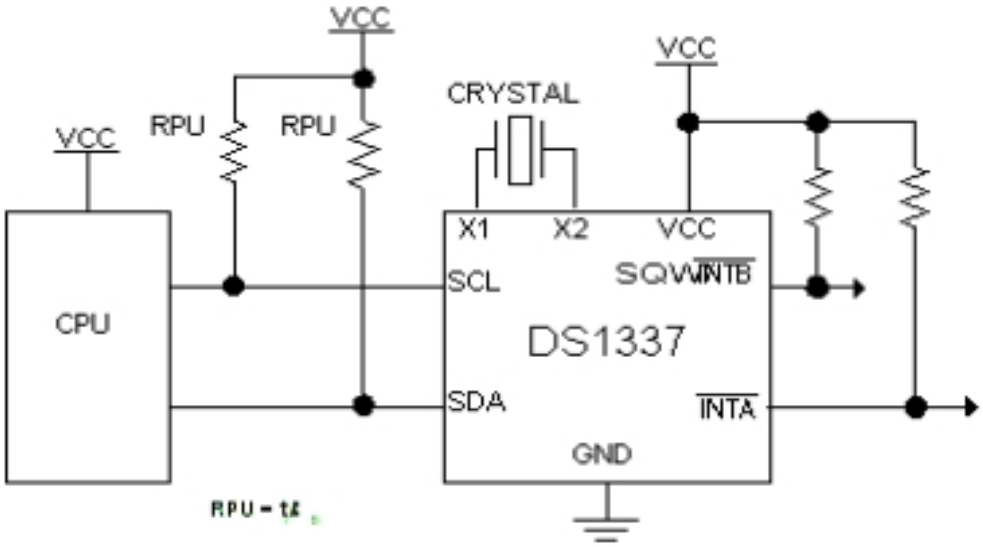
- Čtení sekund, minut, hodin, dní týdne, čísel měsíce, měsíců a let do roku 2100
- Dvoudrátový sériový interface
- Dva signály varování denně
- Flag zastavení generátoru
- Programovatelný výstup pravoúhlého signálu
- Při podávání napájení, výchozí, 32 kHz
- 8-pinový kryt DIP, SO nebo SOP

ADDRESS	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	FUNCTION	RANGE
00H	0	10 SECONDS			SECONDS				Seconds	00-59
01H	0	10 MINUTES			MINUTES				Minutes	00-59
02H	0	12/24	AM/PM 10HR	10HR	HOUR				Hours	1-12 +AM/PM 00-23
03H	0	0	0	0	0	DAY			Day	1-7
04H	0	0	10 DATE		DATE				Date	00-31
05H	CENTURY	0	0	10 MO	MONTH				Month/ Century	01-12+ Century
06H	10 YEAR				YEAR				Year	00-99
07H	A1M1	10 SECONDS			SECONDS				Alarm 1 Seconds	00-59
08H	A1M2	10 MINUTES			MINUTES				Alarm 1 Minutes	00-59
09H	A1M3	12/24	AM/PM 10HR	10HR	HOUR				Alarm 1 Hours	1-12+ AM/PM 00-23
0AH	A1M4	DY/DT	10 DATE		DAY DATE				Alarm 1 Day Alarm 1 Date	1-7 1-31
0BH	A2M2	10 MINUTES			MINUTES				Alarm 2 Minutes	00-59
0CH	A2M3	12/24	AM/PM 10HR	10HR	HOUR				Alarm 2 Hours	1-12+ AM/PM 00-23
0DH	A2M4	DY/DT	10 DATE		DAY DATE				Alarm 2 Day Alarm 2 Date	1-7 1-31
0EH	$\overline{EOSC}$	0	0	RS2	RS1	INTCN	A2IE	A1IE	Control	
0FH	OSF	0	0	0	0	0	A2F	A1F	Status	

Obr.2.8: Mapa mikroobvodu DS1337



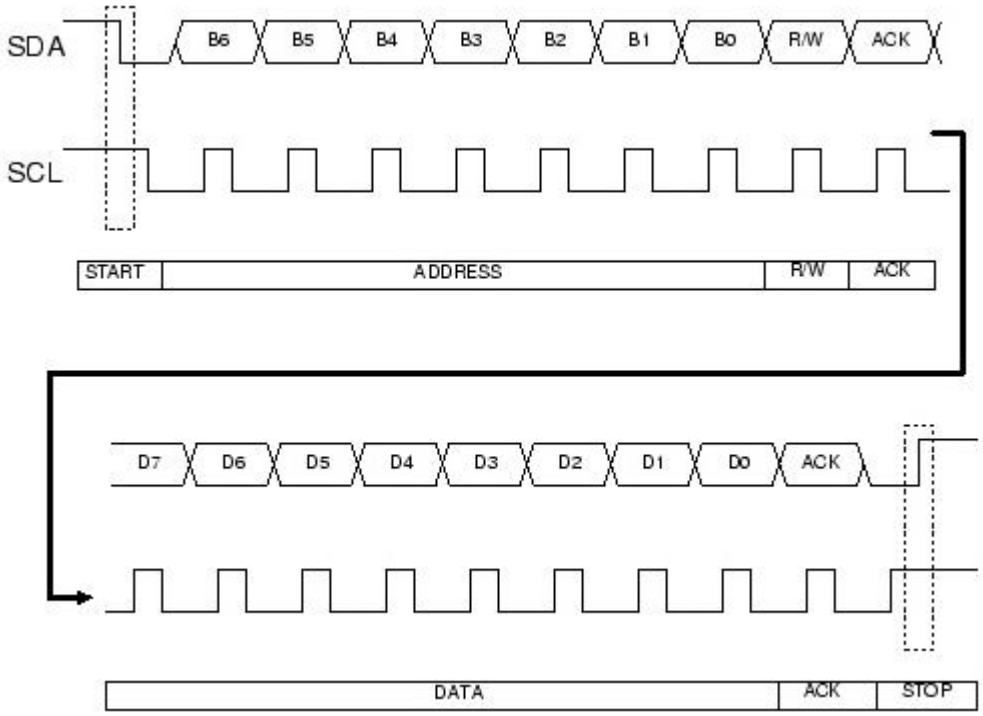
Obr.2.9: Strukturní schéma DS1337



Obr.2.10: Schéma zapojení DS1337 do CPU

2.6.2 Rozhrání I2C

Přenos dat mezi mikrokontrolérem a hodinami probíhá v synchronním režimu pomocí rozhraní I2C. Informační linie dat SDA je obousměrná a synchronizační signály SCL posílá hnací zařízení (master). Rychlost přenosu dat rozhraní až 400 kbit/s. Na obr.2.11 je uvedené schéma protokolu výměny na linii.



Obr. 2.11: Časový diagram protokolu výměny dat.

START se uskutečňuje při nízké frontě SDA v okamžiku kdy je SCL vysoký, bity dat se synchronizují vysokým frontem SCL. Po nulovém bitě dat přijímač formuje nízký stav na linii SDA odpovídající signál ASK. Signál STOP se formuje vysokým frontem SDA kdy SCL je vysoký.

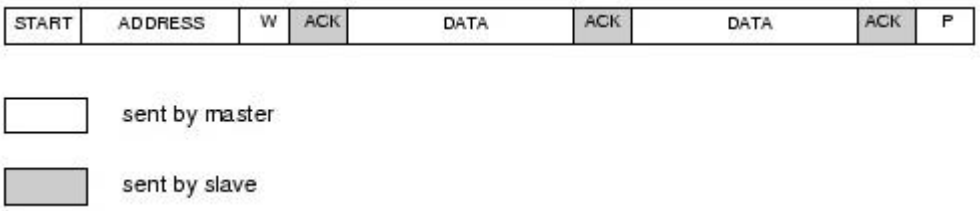
Každé zařízení má svou adresu a se určuje prvním bajtem.

7b, 6b, 5b, 4b - typ zařízení;

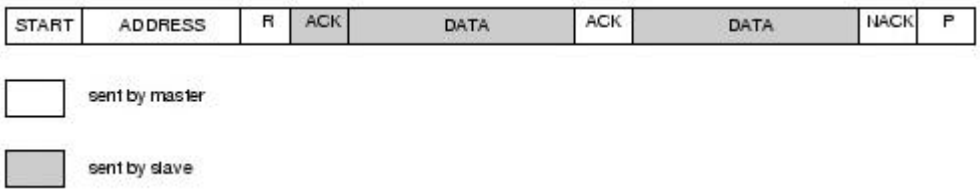
3b, 2b, 1b - číslo zařízení daného typu;

0b = 0 - zápis;

0b = 1 - čtení.



Obr. 2.12: Příklad záznamu dat rozhraním.



Obr. 2.13: Čtení dat z běžné adresy.

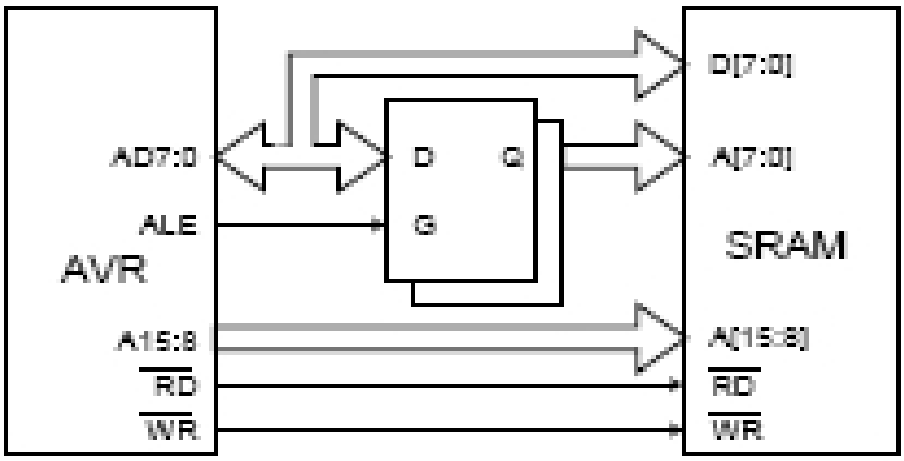
Na obr.2.12 je uvedený režim záznamu dat rozhraním, na obr.2.13 - čtení dat z běžné adresy.

2.7 Externí Paměť

Charakteristiky rozhraní externí paměti umožňují použití nejen pro připojení externí statické RAM nebo flash-paměti, ale i jako rozhraní s vnějšími periferními zařízení (např.: LCD-displeje, AD-převodníky a DA-převodníky). Jeho základními odlišnosti jsou:

- Možnost zadávání čtyř různých stavů očekávání podle délky trvání, včetně stavu bez očekávání.
- Možnost ustavení různých stavů očekávání pro odlišné sektory vnější paměti (velikost sektoru se konfiguruje).

- Možnost volby počtu zapojených bitů v starším adresním bajtu.
- Zařízení uchování stavu sběrnice pro minimalizaci spotřeby proudu .



Obr. 2.14: Schéma připojení SRAM do CPU

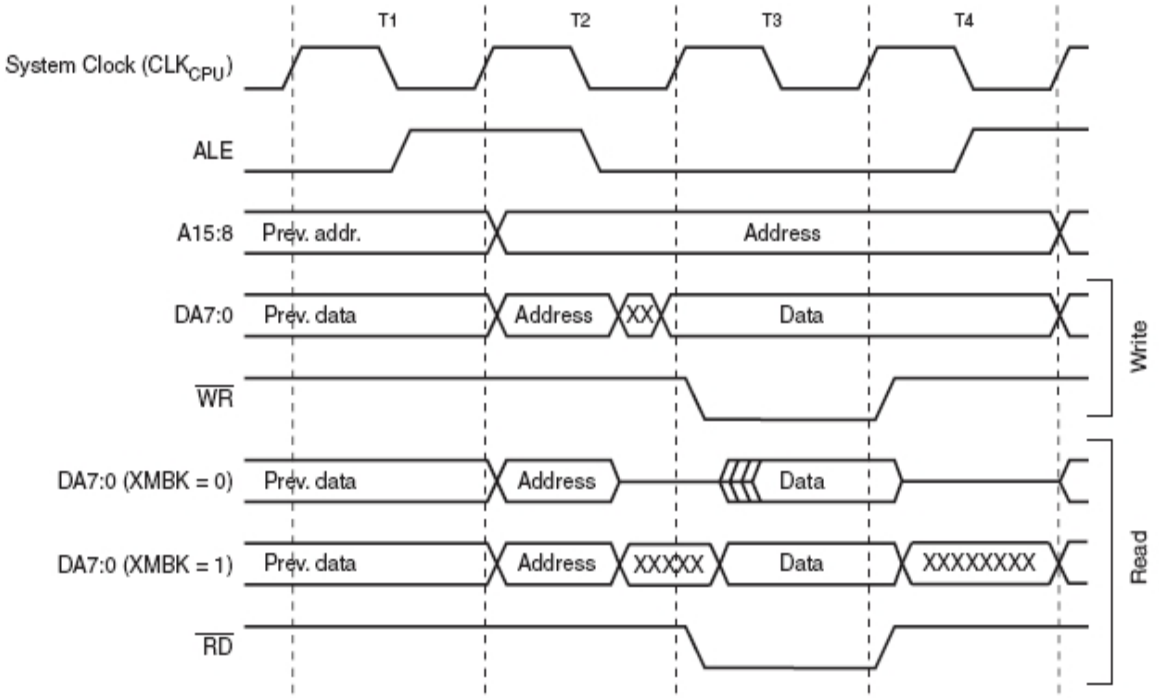
Pokud to vnější paměť (XMEM) dovolí, stává se dostupným adresový prostor mimo vnitřní statický RAM přes předurčené výstupy pro tuto funkci.

Rozhraní se skládá z:

- AD7:0: Multiplexorní mladší sběrnice adresy/sběrnice dat.
- A15:8: Starší sběrnice adresy (s konfigurovaným počtem bitů).
- ALE: Brána adresy vnější paměti.
- RD: Brána čtení z vnější paměti.
- WR: Brána záznamu do vnější paměti.

Bity řízení rozhraním vnější paměti jsou umístěny ve třech registrech: registr řízení mikrokontrolérem – **MCUCR**, registr A řízení vnější pamětí – **XMCRA** a registr B řízení vnější pamětí – **XMCRB**.

Po povolení práce rozhraní XMEM změní nastavení registrů směru dat portů, linii nichž jsou předurčeny pro vyplnění funkcí rozhraní XMEM.



Obr. 2.15: Časový diagram

Komunikace počítače s terminálem nemůže existovat stále, proto je potřeba zápisu do paměti terminálu dat o stavu zařízení. Ale vnitřní paměti 4kB často není dost. Proto je lepší používat dodatečnou paměť. To bude levnější než použití mikrokontroléru s větší vnitřní SRAM. Rozhraní pro připojení XSRAM se vytváří tak, že při správném připojení a správném nastavení registrů práce s vnější SRAM probíhá stejně jako s vnitřní, při čem, kontrolér sám hardwarově zajišťuje zápis a čtení podle časového diagramu (obr.2.15).

2.8 Schéma terminálu

3 MIKROKONTROLÉR ATMEGA1280

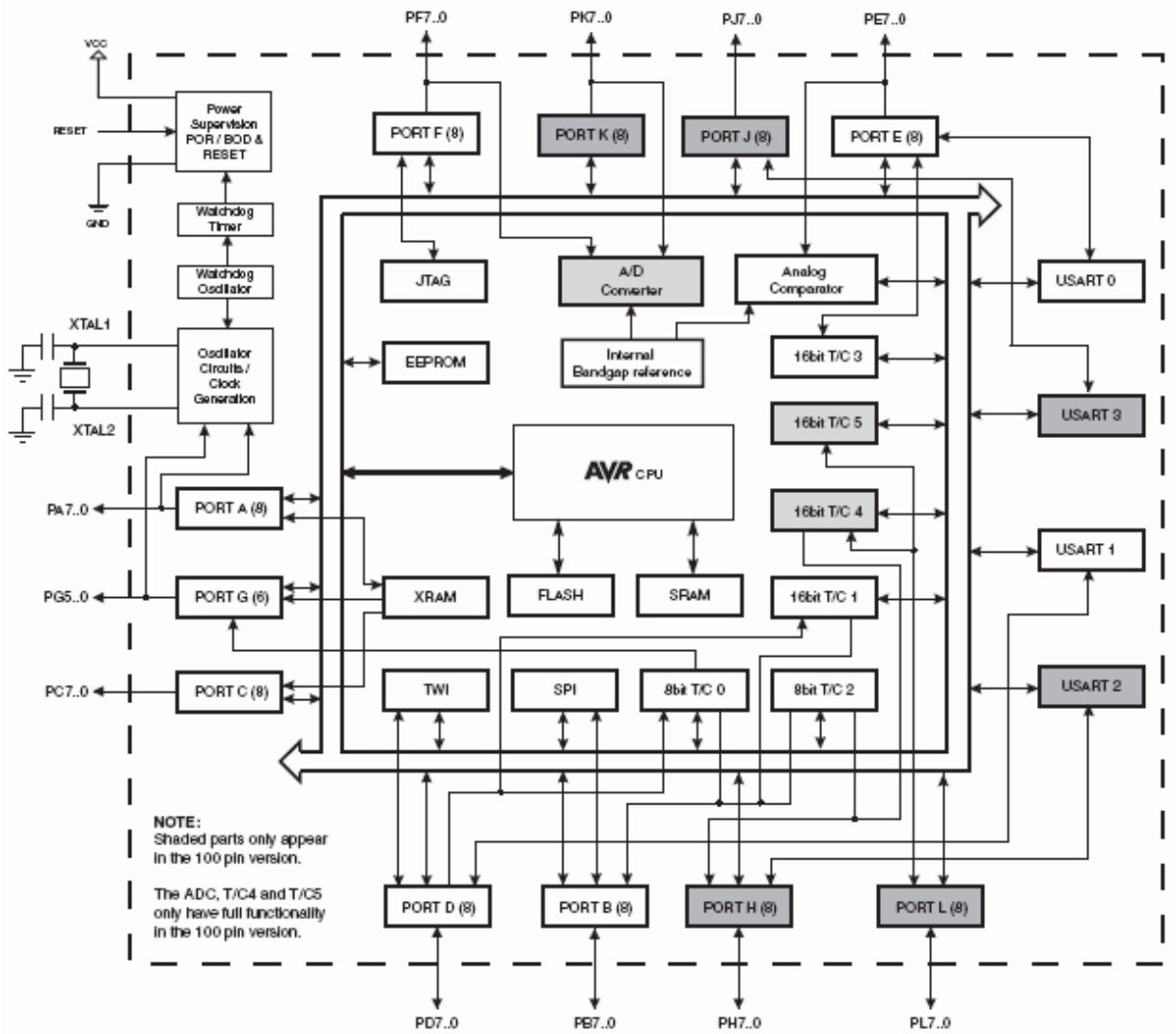
3.1 Popis mikrokontroléru

Činnost celého terminálu ovládá mikrokontrolér Atmega1280. To je malovýkonový 8-bitový CMOS mikrokontrolér, zhotovený na základě AVR-jádra s RISC-architekturou. Díky vyplnění téměř celé instrukce za jeden strojový cyklus dosahuje ATmega1280 výkonu 1 mil. operací za sekundu při taktovací frekvenci 1 MHz.

AVR jádro obsahuje rozsáhlou sadu instrukcí s 32 pracovními registry všeobecného určení. Všech 32 registrů je bezprostředně zapojeno do ALU, což umožňuje ukazovat dva registry v jedné instrukci a vykonat jí za jeden cyklus. Uvedená architektura má větší účinnost kódu a desetinásobně větší výkonnost v porovnání s CISC mikrokontroléry.

ATmega1280 obsahuje následující uzly: 128 kB vnitroobvodo-programovatelnou flash-paměť s možností čtení během zápisu, 4 kB EEPROM, 8 kB SRAM, 54 linii vstup-výstup, 32 pracovních registrů všeobecného účelu, šest pružných časovačů-čítačů s režimy porovnání a PWM(pulzní šířková modulace), 4 univerzální synchronní a asynchronních sériové přijímače-vysílače, 2-vodičová sériová rozhraní s přenosem po bajtech, 16-kanálový 10-bitový A/D převodník s diferenčním vstupním stupněm a programovatelným zesílením, programovatelný hlídací časovač s vnitřním generátorem, sériový port SPI, JTAG rozhraní pro scanování adresového prostoru, ladění v reálném čase a programování a také šest programově nastavitelných postupů řízení spotřeby energie. Režim běhu naprázdno (Idle) zastavuje CPU, ale nechává v chodu SRAM, časovače-čítače, port SPI a systém přerušení. Režim snížené spotřeby (Power-down) zachovává obsah registrů, ale zastavuje generátor, vypíná všechny vestavěné funkce do vzniku následujícího požadavku přerušení nebo přístrojového nulování. V úsporném režimu (Power-save) asynchronní časovač pokračuje v činnosti, umožňující uživateli jeho použití, a ostatní zařízení jsou vypnutá. V režimu snížení šumů A/D převodníků (ADC Noise Reduction) se zastaví CPU a všechny moduly vstupu-výstupu s výjimkou asynchronního časovače a A/D převodníku, tímto se minimalizuje vliv digitálního šumu na výsledek přetvoření. V pohotovostním režimu (Standby) je generátor s křemenným oscilátorem v činnosti a ostatní časti jsou vypnuty. Daný režim umožňuje provést rychlé spuštění v kombinaci s malou spotřebou. V rozšířeném pohotovostním režimu (Extended Standby) jsou hlavní generátor i asynchronní časovač v činnosti.

Mikrokontrolér se vyrábí podle zpracované Atmel technologie EEPROM paměti vysokého obsahu. Vestavěná ISP flash-paměť se může vnitřně přeprogramovat přes sériový interface SPI, obyčejným programátorem energeticky nezávislé paměti nebo spuštěným programem v sektoru výchozího zavedení AVR jádra. Program v sektoru výchozího zavedení může pro zápis programu použít jakékoliv rozhraní. Program se v sektoru výchozího zavedení provádí za obnovení aplikace flash-paměti. Přitom je zabezpečena skutečná možnost čtení v čase zápisu. Kombinací 8-bitového RISC CPU s vnitřně samoprogramovatelnou flash-pamětí na jednom krystalu je ATmega1280 výkonným mikrokontrolér zajišťující vysokou universálnost za nízkou cenu. Proto je ideální pro použití při navrhování vestavěných systémů řízení.



Obr.3.1: Blokové schéma Atmega1280

3.2 Programování

3.2.1 Vývojové prostředí

Pro tuto diplomovou práci bylo pro programování mikrokontroléru Atmega1280 použito vývojové prostředí CodeVisionAVR. CodeVisionAVR je vývojové integrované prostředí (IDE – Integrated Development Environment), vyvinuté pro řadu AVR-mikropočítačů firmy Atmel, které má kompilátor jazyka C a automatický generátor programů (CodeWizardAVR).

Software je 32-bitová aplikace, jež funguje s operačním systémem Windows 95, 98, NT4, 2000 i XP.

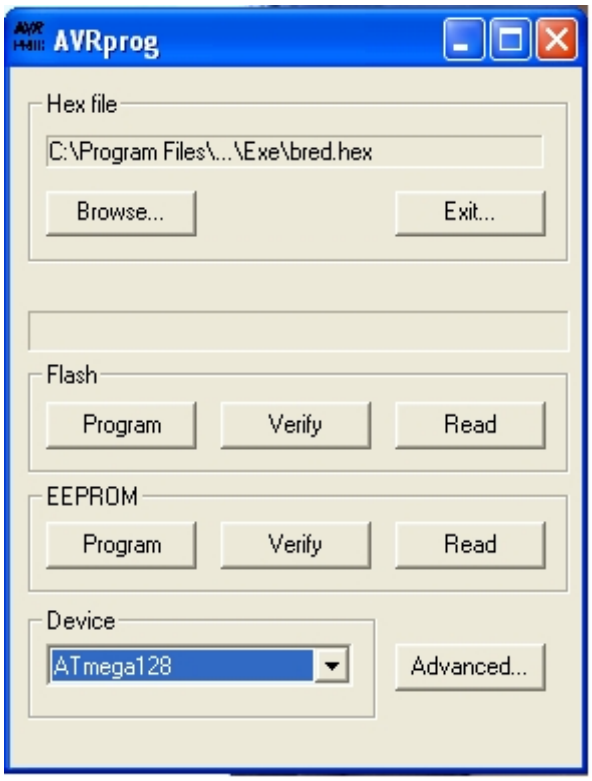
IDE je programově vybaven jako vestavěný vnitřní programátor čipů AVR, který umožňuje automaticky předávat úspěšně zkompilované programy do mikrokontroléru. Programové vybavení programátoru může pracovat spolu s Atmel STK500/AVRISP/AVRProg, Kanda Systems STK200+/300, Dontronics DT006, Vogel

Elektronik VTEC-ISP, Futurlec JRAVR a deskou zpracovatele MicroTronics ATCPU/Mega2000.

Pro ladění navržených systémů, které používají sériové spojení, má IDE vestavěný terminál.

CodeVisionAVR obsahuje automatický generátor – **CodeWizardAVR**, který umožňuje napsat za několik minut úplný kód, nutný pro vyplnění následující funkce:

- nastavení přístupu do vnější paměti,
- identifikace zdrojů poklesu čipu,
- inicializace portu vstupu/výstupu,
- inicializace vnějších přerušení,
- inicializace časovačů/čítačů,
- inicializace hlídacího časovače,
- inicializace USART a přerušení, které ovládají bufferem sériové vazby,
- inicializace analogového komparátoru,
- inicializace AD-převodníku,
- inicializace rozhraní SPI,
- inicializace sběrnice I2C,
- inicializace sběrnice 1-Wire,
- inicializace LCD.



Obr.3.2: AVRProg

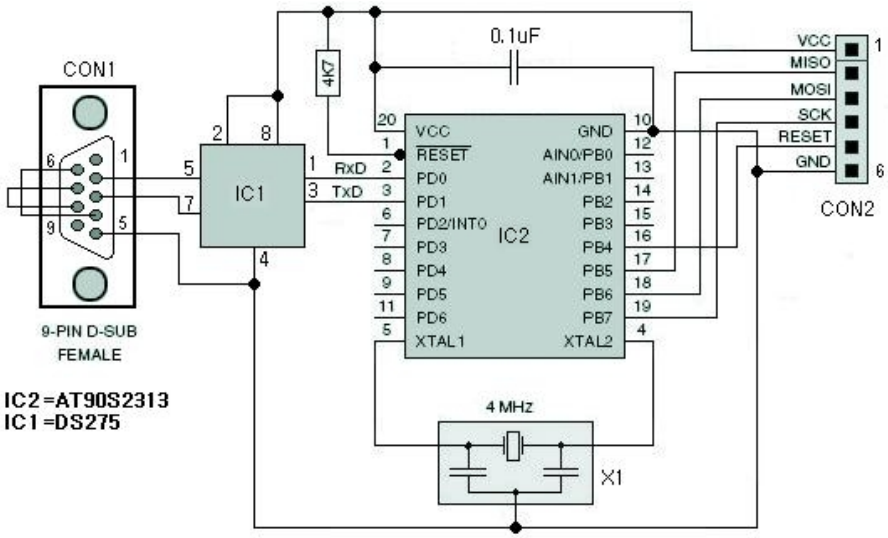
Do paketu Atmel AVR Tools patří program AVRProg, který je používán pro zápis programu v mikrokontroléru. S její pomocí možná naprogramovat mikrokontrolér i bez spuštění programu AVR Studio. Musíme jenom ukázat místo umístění překládaného souboru, paměť, do které se program zapisuje, a stisknout tlačítko "Program". Pro mě to je velmi pohodlné, protože pro napsání programu používám CodeVisionAVR.

3.2.2 Programátor

V projektu byl pro programování mikrokontroléru použit poměrně laciný programátor AVR ISP. AVR ISP - je vnitřní systémový programátor pro AVR flesh-mikrokontroléry Atmel. AVR ISP je kompaktní a spolehlivý prostředek programování a připojuje se do účelového zařízení přes 6- nebo 8-svorkový konektor. Používá integrované prostředí projektování AVR Studio, které může fungovat s Windows nebo DOS s rozhraním příkazového řádku. AVR ISP je napájen od účelového zařízení, doplňkový zdroj napájení tedy není nutný.

Obr.3.3 Zapojení programátoru

Rozhraní programování AVR ISP je integrován v AVRStudiu, kde jsou volby programování flash-paměti, EEPROM, bit stínění a konfigurace. Mohou se používat individuálně a s použitím posloupnosti automatického programování.



Obr.3.4 Schéma programátoru AVR ISP

4 NÁVRH PROGRAMOVÉHO VYBAVENÍ

Prvním druhem vstupních dat pro terminál je stav zařízení v pracovní době. Totiž výšková fronta znamená, že stroj funguje, a dolní fronta - stroj zastaven. Pro určení stavu zařízení lze pohodlně používat přerušení PC_INT2, protože se vyvolá jakákoliv změna stavu na jakémkoliv pinu portu K, do kterého jsou připojeny stroje. Proto, v programu je použita funkce zpracovávání přerušení.

Druhým typem vstupních dat je identifikační číslo zaměstnance, který obsluhuje daný stroj. Pro identifikace pracovníka jsou používány RFID-značky, kód z kterého je možné číst pomocí RFID-antény CORE-12. Anténa je připojena přes rozhraní USART mikrokontroléru, v našem případě USART1.

Třetím typem vstupních dat je čas. A to je:

- čas příchodu pracovníka;
- čas odchodu pracovníka;
- čas zastavení zařízení;
- čas zapnutí zařízení;

Pro určení času v terminálu se používají hodiny reálného času DS1337. Inicializace hodin a čtení času prochází přes rozhraní I2C.

Výsledkem práce terminálu mají být hotové pakety dat, které budeme posílat pomocí sběrnice RS422. Pakety dat rozdělíme na 3 skupiny:

1. Paket má obsahovat číslo stroje, čas zastavení, čas zapnutí a příčinu zastavení.
2. Paket má obsahovat identifikační číslo pracovníka a čas příchodu.
3. Paket má obsahovat identifikační číslo pracovníka a čas odchodu.

Pro posílání paketů dat používáme rozhraní USART0 a obvod MAX491 pro převádění USART/RS422. Pro vysílání paketů dat je potřeba sestavení protokolu výměny dat mezi PC a terminálem. Pro komunikaci člověka s terminálem se používá LCD-displej a senzorový panel.

S ohledem nato můžeme navrhnout základní algoritmus práce terminálu:

pracovník přistoupí k terminálu a přiloží svůj individuální klíč. Program musí přečíst identifikační číslo a zkontrolovat existenci tohoto čísla v seznamu. Když není pracovník v databázi, LCD ukazuje odpovídající zprávu. Když je pracovník v seznamu, data o čase spuštění zařízení se odešle do databáze na PC. Od tohoto momentu se každých 5 sekund terminál musí kontrolovat stav zařízení. Jestliže je čas zastavení zařízení více něž 5 minut, terminál fixuje data o zastavení v paměti. Po spuštění zařízení terminál pošle zprávu o spuštění a zastavení zařízení do PC. Na konci pracovní doby pracovník opět přiloží klíč, informace o odhlášení se odešle do PC. V případě absence komunikace

terminálu s PC, se data, které se mají posílat do PC, musí byt zapsána do SRAM. Příkazem vedoucího se všechna data z paměti přenesou do PC.

5 REALIZACE

5.1 Struktura programu

Při vlastní realizaci byl algoritmus programu rozdělen do několika bloků:

1. Inicializace proměnných.
2. Inicializace rozhraní.
3. Inicializace přerušení a časovače.
4. Inicializace hodin reálného času a LCD.
5. Základní cyklus, z kterého se budou vyvolávat funkce zpracovávání přerušení, funkcí posílání dat do PC, funkcí čtení času z hodin reálného času a funkce čtení dat z registru AD-převodníku.

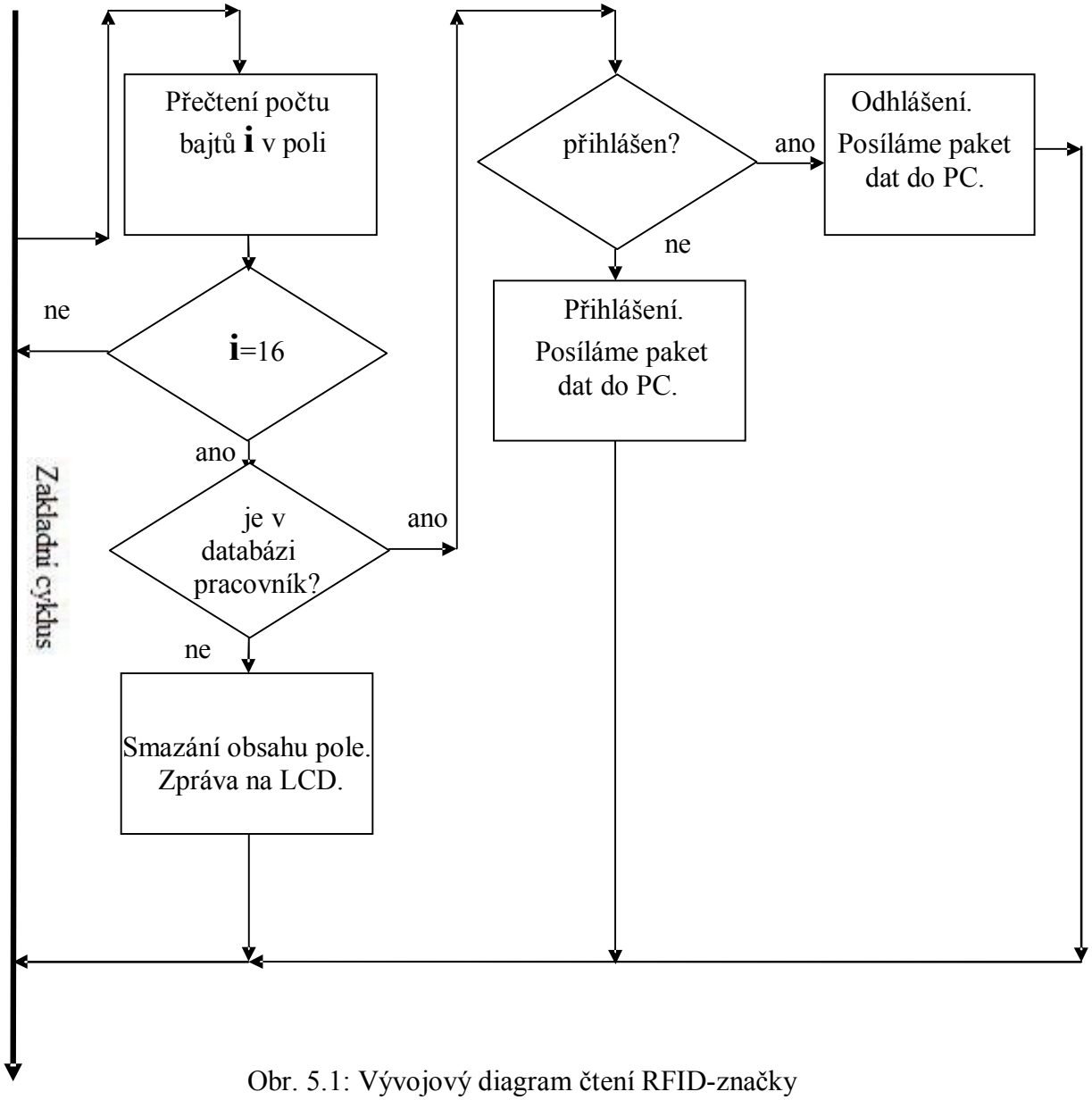
5.2 Čtení RFID-značek

ASCII-znaky se z antény do mikrokontroléru vysílají přes rozhraní USART1 s rychlostí 9600 baud. Před použitím USART1 pro synchronizace s CORE-12 musíme nastavit rychlost čtení. Proto máme zapsat nutné parametry v registrech nastavení USART1. Pro inicializace rozhraní USART1 používáme funkce:

```
void Init_UART1(void) {
    UCSR1A=0x00; // Komunikační parametry: 8 Data, 1 Stop, No Parity
    UCSR1B=0x90; // USART1 Receiver: On
    UCSR1C=0x06; // USART1 Transmitter: Off
    UBRR1H=0x00; // USART1 Mode: Asynchronous
    UBRR1L=0x5F; // USART1 Baud Rate: 9600
    }
```

UCSR1A - registr A řízení a stavu USART1
UCSR1B - registr B řízení a stavu USART1
UCSR1C - registr C řízení a stavu USART1
UBRR1H - registr rychlosti přenášení USART1, starší bajt
UBRR1L - registr rychlosti přenášení USART1, mladší bajt

Čtení každého bajtu probíhá v přerušení a zapisuje se do pole dočasného uložení znaků. RFID-značka obsahuje 16 bajtů. Abychom určili existence 16 bajtů v poli, každý cyklus v něm kontroluje počet bajtů. Jestli je 16, přečteme z něho všechny bajty a smažeme obsah pole. Proto můžeme znázornit čtení RFID-značky ve vývojovém diagramu:



Obr. 5.1: Vývojový diagram čtení RFID-značky

5.3 Sběr dat ze strojů

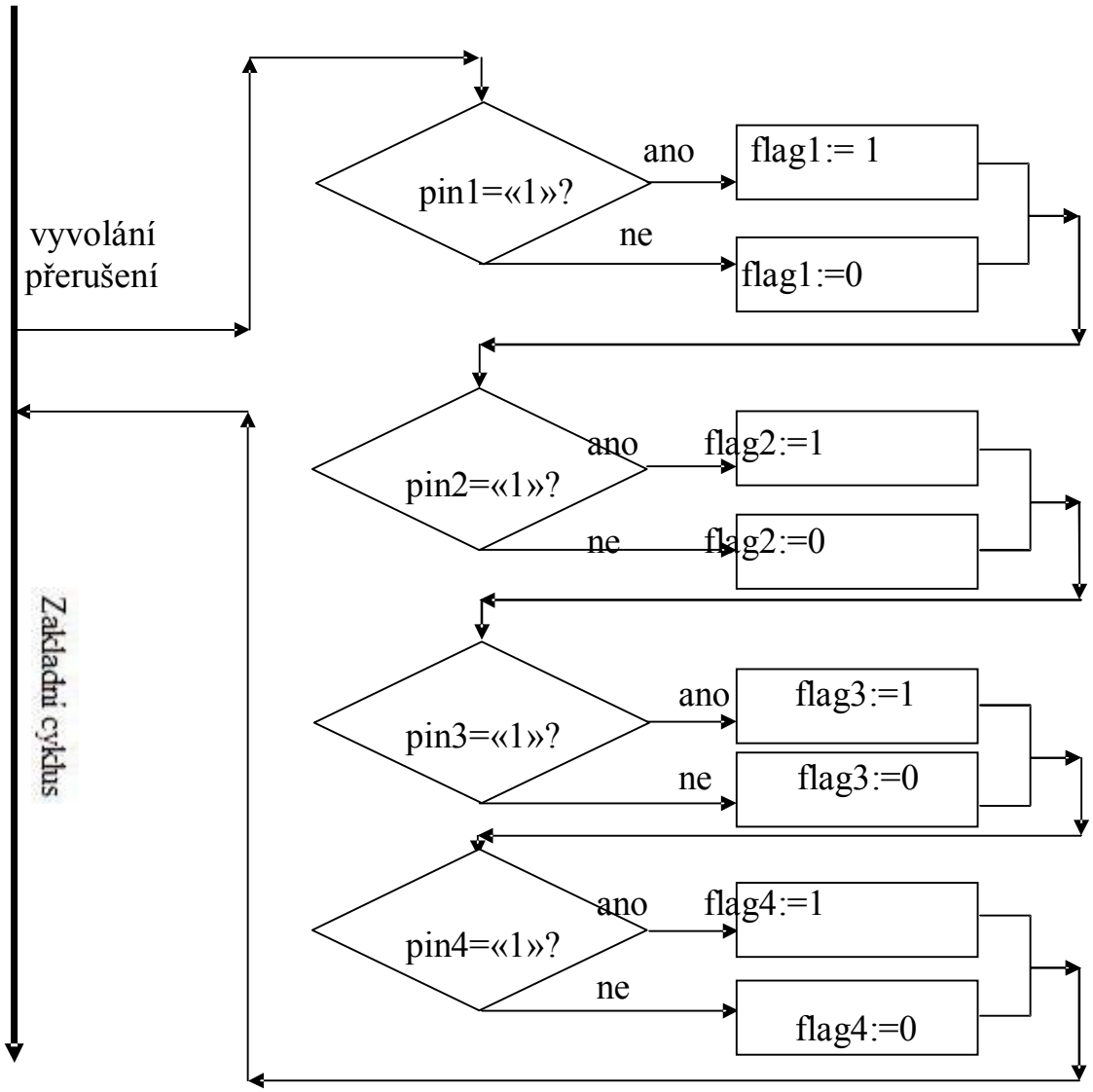
Pro inicializaci vnějších přerušení je nutno naladit registry vnějších přerušení PCMSK0, PCMSK1, PCMSK2 i PCICR. Pro inicializaci používáme funkce:

```
void Init_int16_19(void){
      PCMSK0=0x00;
      PCMSK1=0x00;
      PCMSK2|=(1<<PCINT16)|(1<<PCINT17)|
                  (1<<PCINT18)|(1<<PCINT19);
      PCICR = ( 1<<PCIE2);
      }
```

PCMSK0 – registr masky 0. přerušení podle změny stavů výstupů (přerušení PCINT7:0, nepoužíváme);
PCMSK1 – registr masky 1. přerušení podle změny stavů výstupů (přerušení PCINT15:8, nepoužíváme);
PCMSK2 – registr masky 1. přerušení podle změny stavů výstupů (přerušení PCINT23:16, používáme);
PCICR – registr řízení přerušením podle změny stavů výstupů.

Přerušení PC_INT2 se vyvolávají při jakékoliv změně stavu na jednom z pinů portu K. Proto pro stanovení stavu vstupu, je potřeba ve funkci zpracování přerušení určit pin, na kterém je změna stavu a zapsat původní stav do "flagu".

Proces zpracování přerušení je možné znázornit ve formě vývojového diagramu:



Obr. 5.2: Vývojový diagram zpracování přerušení

Stav flagů se obnovuje při jakékoliv změně stavu pinů portu K. Pro čtení stejných časových intervalů použijeme 8-bitový časovač T/C0. Časovač nastavím tak, aby se jeho přerušení vyvolávaly každých 5 sekund. Ve funkci zpracování přerušení časovače se bude kontrolovat stav každého flagu. Všechny přerušení budeme počítat. Jestli po zapnutí stroje počet vyvolaných přerušení časovače se stavem flagu «0» je více než 60 jeden za druhým, to znamená, že stroj stojí více než 5 minut. V tomto případě je třeba určit příčinu zastavení. Proto na LCD zobrazíme zprávu pro pracovníka, aby zvolil v menu příčinu zastavení. Po zvolení příčiny kontrolér pošle na PC paket dat o příčině a čase prostoje.

5.4 Zobrazení textu na LCD

Pro realizaci signálů řízení LCD používáme makry:

```
#define A0_0  PORTD &= 0xEF
#define A0_1  PORTD |= 0x10
#define CS_0  PORTD &= 0xDF
#define CS_1  PORTD |= 0x20
#define WR_0  PORTD &= 0xBF
#define WR_1  PORTD |= 0x40
#define RD_0  PORTD &= 0x7F
#define RD_1  PORTD |= 0x80
#define RES_0  PORTB &= 0x7F
#define RES_1  PORTB |= 0x80
#define DISPOFF_0 PORTB &= 0xef
#define DISPOFF_1 PORTB |= 0x10
```

Pro ovládání displejem používáme funkce podle časových diagram:

```
void PoslatPrikaz(unsigned char cmd){          //funkce záznamu příkazu
                                               //do displeje
        DDRL = 0xff;
        delay_us(1);
        PORTL = cmd;
        delay_us(1);
        CS_0;
        WR_0;
        delay_us(1);
        CS_1;
        WR_1;
        delay_us(1);
        }
```

```
void PoslatData(unsigned char data){                    // funkce záznamu dat
                                                        // do displeje
        DDRL = 0xff;
        delay_us(1);
        PORTL = data;
        CS_0;
        A0_0;
        WR_0;
        delay_us(1);
        CS_1;
        A0_1;
        WR_1; delay_us(1);
        }
```

Pro zobrazení textu na LCD jsou použity vhodné posloupnosti funkcí, které byli sestaveny s ohledem na časové diagramy, například:

```
PoslatPrikaz(SetCurAddr);           //příkaz umístění kurzoru do buňky
      PoslatData(0x15);             //číslo buňky, mladší bajt
      PoslatData(0x00);             //číslo buňky, starší bajt
PoslatPrikaz(MemWrite);             //příkaz zápisu dat do paměti LCD
      PoslatData('A');              //data
      PoslatData('h');      //se zápisem každého znaku se kurzor posune o jednu buňku
      PoslatData('o');
      PoslatData('j');
```

5.5 Sběr dat ze senzorového panelu

Algoritmus zpracování zavedený pracovníkem dat realizujeme jednoduše. Když po zastavení více než na 5 minut stroj se rozeběhne, kontrolér vypíše zprávu na LCD aby pracovník zvolil příčinu zastavení. Mohou být dvě příčiny - oprava zařízení, nebo výměna látky. Po vypsání zprávy, kontrolér cyklicky kontroluje stav registru AD-převodníku. Jestli pracovník stisknul senzorový panel v zóně jedné z tlačítek, náš program ukončí kontrolování hodnoty napětí v registru, zformuje paket dat a pošle ho na počítač.

Pro inicializace AD-převodníku používáme funkce:

```
void Init_ADC(void){
        DIDR0=0xFF;       //zákaz digitálních vstupů
        DIDR2=0xFF;
        ADMUX=0x60;       //výběr referenčního napětí a vstupního kanálu
        ADCSRA=0xA7;      //povolení AD-převodníku, zadání frekvence převodu
        ADCSRB&=0xF8;     //výběr vstupního kanálu
        }
```

DIDR0 – registr 0 vypnutí digitálních vstupů.
DIDR2 – registr 2 vypnutí digitálních vstupů.

ADMUX – registr řízení multiplexorem AD-převodníku.
ADCSRA – registr A řízení a stavu AD-převodníku.
ADCSRB – registr B řízení a stavu AD-převodníku.

Hodnotu napětí na vstupu AD-převodníku určujeme pomocí funkce **read_adc**() z knihovny CodeVisionAVR, která vrátí hodnotu uloženou v registru ADCH.

5.6 Čtení času z obvodu DS1337

Čas a datum v hodinách DS1337 jsou uloženy ve formátu BCD, proto pro inicializace hodin je nutné převádět čas z binárně-decimálního formátu do BCD-formátu, a po čtení naopak – z BCD-formátu do binárně-decimálního. Proto používám funkce konverze formátů.

Z binárně-decimálního do BCD:

```
unsigned char bin2bcd(unsigned char num) {
        return ((num/10 * 16) + (num % 10));}
```

Z BCD do binárně-decimálního:

```
unsigned char bcd2bin(unsigned char num){
        return ((num/16 * 10) + (num % 16));}
```

Před použitím rozhraní I2C je třeba ho nastavit. CodeVisionAVR zjednodušuje práce s rozhraním, protože ve své knihovně má hotové funkce inicializace I2C (i2c_init()), funkce nastavení rozhraní v polohu START (i2c_start()) a funkce nastavení v polohu STOP (i2c_stop()) . Po inicializace rozhraní musíme inicializovat hodiny DS1337. Proto skládáme funkce DS1337_init();

Pro inicializace hodin nutno nastavit rozhraní v polohu START stanovením SDA v «0» a SCL v «1». Pak je třeba přes rozhraní poslat adresu hodin a sdělit hodinám, že budeme zapisovat. Adresa hodin DS1337 – 11010000. Jestli je nutno zapisovat čas, nejmladší bajt musí být «0». Potom umístím kurzor do registru, z kterého začneme zápis, v našem případě to je registr sekund, má číslo 0x00. Z tohoto okamžiku můžeme posílat hodinám čas, při čem se kurzorem budeme pohybovat do dalších registrů. Po probrání 14 registrů kurzorem, nastane v registru kontroly, a pak v registru statusu. Po inicializaci je třeba nastavit rozhraní v polohu STOP, stanovením SDA a SCL v «1».

```
void  DS1337_init(void) {

        SREG &= ~(1<<7);          //zákaz všech přerušení
        i2c_start();              //START
        delay_ms(1);
        i2c_write(D0);            //adresa DS1337
        i2c_write(0x00);          //začínáme z registru sekund
        i2c_write(0x00);          //sekundy
        i2c_write(0x00);          //minuty
        i2c_write(0x40);          //hodiny
```

```
        i2c_write(0x00);            //den
        i2c_write(0x00);            //číslo
        i2c_write(0x00);            //měsíc
        i2c_write(0x00);            //rok
        i2c_write(0x00);            //alarm1 sekundy
        i2c_write(0x00);            //alarm1 minuty
        i2c_write(0x00);            //alarm1 hodiny
        i2c_write(0x00);            //alarm1 den/číslo
        i2c_write(0x00);            //alarm2 minuty
        i2c_write(0x00);            //alarm2 hodiny
        i2c_write(0x00);            //alarm2 den/číslo
        i2c_write(0x0e);            //control registr
        i2c_write(0x0f);            //status registr
        i2c_stop();                 //STOP
        delay_ms(100);
        #asm("sei")

}
```

5.7 Výměna dat

Pro výměnu dat mezi PC a terminálem stačí použít 6 druhů paketů a 5 pomocných příkazů proměnlivé délky. Pakety dat se začínají identifikátorem akce, obsahuje posílaná data a kontrolní součet. Příkazy obsahují pouze identifikátor akce.

Pakety dat:

Záznam identifikačního čisla zaměstnance do paměti EEPROM terminálu:

„M“	ID (10 ASCII-znaků)	Kontrolní součet

Tento paket požíváme jenom pro záznam identifikačního čísla pracovníka, který bude mít přístup k práci se strojem, do paměti EEPROM. Identifikační čísla pracovníka zapisujeme do energeticky nezávislé paměti, protože po ztrátě napětí, data nebudou ztracena.

Příčina zástavení stroje:

“P”	Číslo stroje (1 - 4)	Čas zastavení (3 bajty)	Čas spuštění (3 bajty)	Příčina	Kontrolní součet

Tento paket bude poslán terminálem do PC. Obsahuje čas zastavení, čas spuštění, příčinu zastavení a číslo stroje, ke kterému patří tato informace.

Stav všech strojů:

“I”	Stroj 1	Stroj 2	Stroj3	Stroj 4	Čas (3 bajty)	Kontrolní součet

Přihlášení:

“H”	ID (10 ASCII-znaků)	Čas (3 bajty)	Kontrolní součet

Obsahuje identifikační číslo pracovníka a čas přihlášení.

Odhlášení:

“O”	ID (10 ASCII-znaků)	Čas (3 bajty)	Kontrolní součet

Obsahuje identifikační číslo pracovníka a čas odhlášení.

Zobrazení textu na displeji terminálu:

“D”	Číslo řádku	Číslo sloupce	Text (max.20 znaků)	Kontrolní součet

Tento paket se bude možná používat, jestli vedoucí bude chtít zobrazit nějakou informace na displeji, například ”Pozor, stroj je rozbitý”. Obsahuje číslo řádku a sloupce, z kterého začneme vypisovat text a samotný text.

Příkazy, které posílá PC na terminál:

Odesílání stavu zařízení do PC:

“R”	Kontrolní součet

Tento příkaz používáme jestli chceme kdykoliv přečíst stav všech zařízení.

Smazání EEPROM:

“E”	Kontrolní součet

Tento příkaz slouží pro úplné smazání paměti EEPROM.

Zahájení komunikace PC s terminálem:

“Z”	Kontrolní součet

Tento příkaz potřebujeme, aby terminál věděl, že komunikace s PC je zahájená, proto data nemůžeme zapisovat do vnější paměti, ale posílat na PC.

Ukončení komunikace PC s terminálem:

“K”	Kontrolní součet

Jestliže terminál dostal tento příkaz, znamená to že není komunikace s PC, proto všechna data se budou zapisovat do vnější paměti.

Přemístit všechny data SRAM na PC:

“A”	Kontrolní součet

Tento příkaz pro terminál znamená, že všechny data z vnější paměti musí poslat na PC.

Pro výměnu dat s PC používáme rozhrání USART0. Pro jeho práci je vyhrazeno 5 registrů:

UCSR0A - registr A řízení a stavu USART0
UCSR0B - registr B řízení a stavu USART0
UCSR0C - registr C řízení a stavu USART0
UBRR0H - registr rychlosti přenosu USART0, starší bajt
UBRR0L - registr rychlosti přenosu USART0, mladší bajt

Pro inicializace USART0 použijeme funkce:

```
void Init_UART0(void){
    UCSR0A=0x00;    // Komunikační parametry: 8 Data, 1 Stop, No Parity
    UCSR0B=0x98;    // USART0 Receiver: On
    UCSR0C=0x06;    // USART0 Transmitter: On
    UBRR0H=0x00;    // USART0 Mode: Asynchronous
    UBRR0L=0x5F;    // USART0 Baud Rate: 9600
    }
```

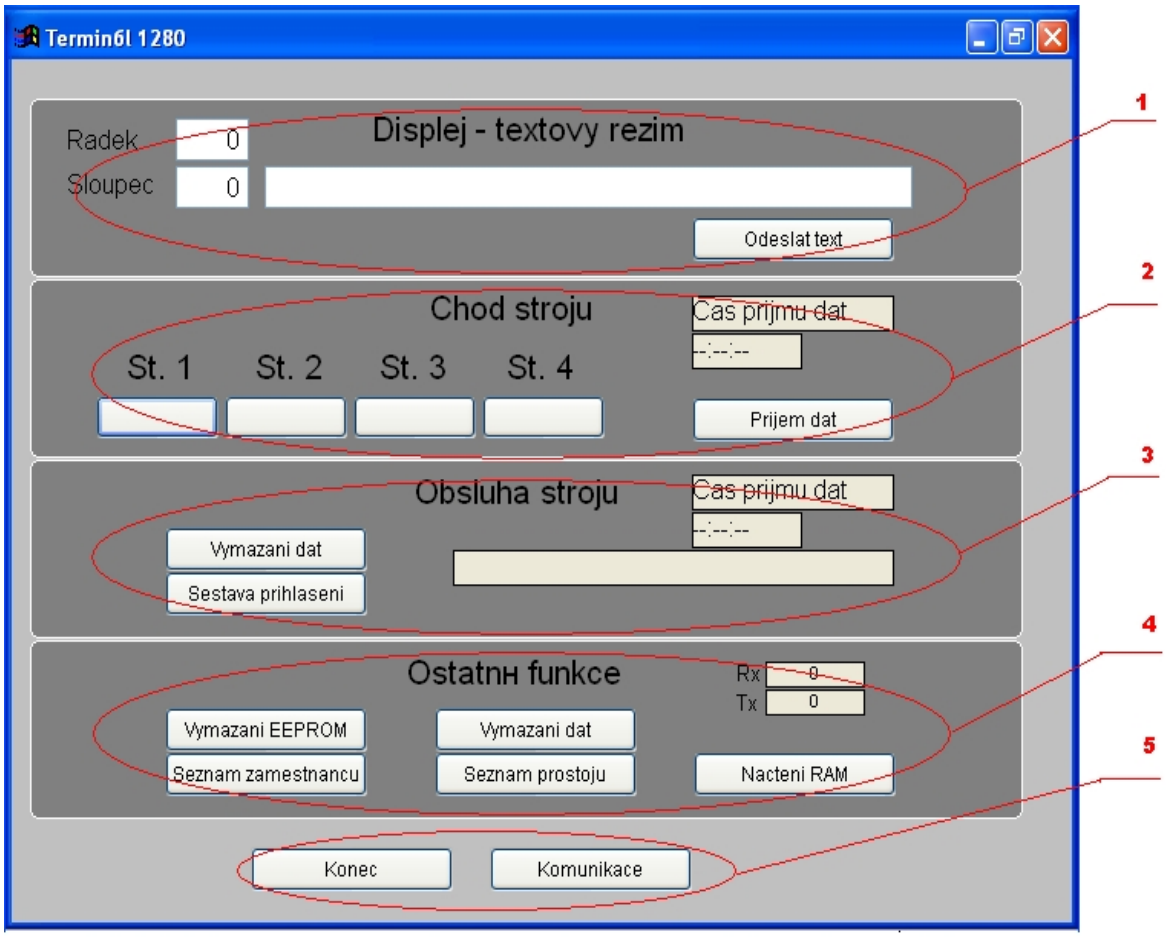
Příjem bajtů dat probíhá v přerušení USART0_RXC, proto pro zpracování přerušení použijeme funkce z knihovny CodeVisionAVR:

```
interrupt [USART0_RXC] void usart0_rx_isr(void){
    char status,data;
    status=UCSR0A;
    data=UDR0;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0){
    rx_buffer0[rx_wr_index0]=data;
    if (++rx_wr_index0 == RX_BUFFER_SIZE0) rx_wr_index0=0;
    if (++rx_counter0 == RX_BUFFER_SIZE0){
            rx_counter0=0;
            rx_buffer_overflow0=1;
}}}
```

V okamžik, kdy je potřeba dostat nejstarší bajt, používáme funkci **getchar**(), a pro odeslání řádku používáme funkci **printf()** z knihovny CodeVisionAVR **stdio.h.**

6 APLIKACE

Pro demonstrace práce terminálu používáme jednoduchý databázový systém, zpracovaný ve Visual Fox Pro.



Obr. 6.1: Databáze

Popis menu:

1 – je určen pro výpis textu na displeji terminálu. Po uvedení čísla řádku displeje, sloupce displeje a uvedení nutného textu je třeba stisknout tlačítko “Odeslat text”. Výsledkem toho je zasílán do terminálu paketu, který obsahuje text a místo zobrazení textu na LCD.

2 – tato sekce menu slouží pro kontrolu stavu činnosti zařízení v libovolný čas. Jestli stiskneme “Příjem dat”, do terminálu se odešle dotaz stavu všech strojů. Terminál v odpovědi pošle paket dat se stavem zařízení.

3 – v této oblasti menu se ukazuje informace o přihlášení a odhlášení zaměstnanců. Po stisknutí tlačítka “Sestava přihlášení” uvidíme celou historii příchodu a odchodu zaměstnanců z pracoviště a po stisknutí “Vymazání dat” vymažeme v databáze celou sestavu přihlášení.

4 – 4.sekce menu se používá pro záznam do databáze zaměstnanců, které obsluhují zařízení a pro prohlédnutí historie ztrátové doby strojů. Jestli že stiskneme tlačítko

”Seznam zaměstnanců”, otevře se menu pro záznamu zaměstnanců do seznamu, který se uchovává v terminálu v paměti EEPROM. Stisknutím tlačítka “Vymazání EEPROM” pošleme do terminálu příkaz o vymazání celé EEPROM. Po stisknutí “Seznam prostojů” uvidíme tabulku s historií všech prostojů, a stisknutím “Vymazání dat” smažeme obsah této tabulky. Stisknutím “Načtení RAM” do terminálu pošleme dotaz na všechny data, které jsou uloženy v paměti RAM, to jsou všechna prostoje, která se nepodařilo poslat do PC z důvodu absence komunikace. V odpovědi terminál zašle pakety, které obsahují čísla stroje, čas a příčiny prostoje.

5 – je určen pro odesílání příkazů o počátku komunikace a o ukončení komunikace. Toto je nutno proto, aby kontrolér mohl zvolit mezi záznamem dat do paměti a vysíláním do PC. Výsledkem stisknutí tlačítek “Konec” a “Komunikace” je poslána terminálu informace o tom, že je komunikace s terminálem nebo ne.

7 ZÁVĚR

V této práce bylo navrženo a zrealizováno programové vybavení pro terminál sběru dat z výrobního zařízení, který se používá v závodě Juta a.s. Pro programování byl použit jazyk C.

Daný terminál provádí tyto funkce:

- čtení 10-místného kódu z RFID-značek;
- čtení času ze schématu hodin reálného času;
- sběr dat z výrobního zařízení;
- zobrazení textu na LCD;
- čtení dat ze senzorového panelu;
- výměna dat s PC.

Pro realizaci čtení identifikačního čísla z RFID-značek byla použita funkce zpracování přerušení USART z knihovny CodeVisionAVR a cyklickou kontrolu přítomnosti ASCII-znaků v bufferu dočasného uchování identifikačního kódu.

Práce s obvodem hodin reálného času je realizována pomocí funkcí pro rozhrání I2C, které patří do knihovny funkci CodeVisionAVR. Tyto funkce byly použity pro sestavení funkcí inicializace hodin reálného času a čtení času.

Pro sběr dat ze zařízením byla sestavena funkce zpracování přerušení, která je vyvolávána při jakékoliv změně stavů vstupů.

Pro vypsání textu na LCD byla sestavena funkce inicializace displeje a funkce vypsání znaků.

Pro čtení dat ze senzorového panelu byl sestaven algoritmus čtení dat ze senzorového (dotykového) panelu.

Pro výměnu dat mezi terminálem a PC byl navržen speciální protokol.

Zdrojový text programu pro terminál v jazyce C a vytvořená aplikace jsou uloženy na přiloženém CD.

POUŽITÁ LITERATURA

[1] HEROUT, P.: Učebnice jazyka C. České Budějovice, KOPP 1994
[2] HEROUT, P.: Učebnice jazyka C – 2.díl. České Budějovice, KOPP 2006
[3] MANN, B.: C pro mikrokontroléry. Praha, BEN 2003
[4] БАРАНОВ, В.: Применение микроконтроллеров AVR: схемы, алгоритмы, программы. Москва, Додека-XXI 2004
[5] ЕВСТИФЕЕВ, А.: Микроконтроллеры AVR семейств Tiny и Mega фирмы Atmel. Москва, Додека-XXI 2005
[6] ЛЕБЕДЕВ, М.: CodeVisionAVR: пособие для начинающих. Москва, Додека-XXI 2008
[7] ШПАК, У.: Программрование на языке C для AVR и PIC микроконтроллеров. Киев, МК-Пресс 2006

INTERNETOVÉ ADRESY

[7] http://ru.wikipedia.org
[8] http://www.gaw.ru – popis mikrokontroléru Atmega1280