PRAXE

č. 3	STŘEDNÍ PRŮMYSLOVÁ ŠKOLA CHOMUTOV	David Herko
21. 1. 2025	ZPRACOVÁNÍ DCF ČASOVÉ INFORMACE	V4

Zadání

S využitím vývojového kitu AVR a přijímače DCF časové informace (buď v reálné nebo virtualizované podobě) sestavte v jazyku C program implementující čtení a dekódování časové informace. Tu zobrazte do terminálu připojeného na sériové lince procesoru. Pro její obsluhu použijte knihovnu, kterou máte k dispozici v rámci ukázkových kódů. Výsledné řešení musí obsahovat následující funkcionalitu:

- Spolehlivá synchronizace a vyhodnocení jednotlivých časových značek (bitů) z přijímače

- Uložení přijaté informace do vhodně zvolené datové struktury umožňující její následné vyhodnocení

- Dekódování přijaté informace do čitelné podoby, detekce chyb a částečné vyhodnocení nekompletní informace

- Zobrazení dekódované informace, příp. její části, do terminálu připojeného sériovou linkou.

Teorie

DCF77 je signál vysílaný z Frankfurtu, který udává současný datum a čas. Používá se v hodinách, budíkách a jiných zařízeních, které vyžadují práci s přesným časem. V úloze se nepoužívá reálný přijímač, nýbrž simulátor nahraný do ATMega128 procesoru.

Signál se skládá z 59 bitů, kdy je jeden bit vysílán po dobu jedné vteřiny. V oné jedné vteřině je vysílána nula nebo jednička, nás zajímá časový úsek vysílané nuly. Pokud je tento časový úsek v rozmezí 50 – 120ms, výsledný bit se rovná nule; pokud je v rozmezí 140 – 240ms, jedná se o jedničku.

Výsledných 59 bitů se seskládá a dostaneme časovou informaci. 20. bit je log 0 – start signálu. 21. – 27. bit je minuta, 28. bit je parita minuty. 29. – 34. bit je hodina, 35. jest parita hodin. 36. – 41. bity udávají den, 42. - 44. den v týdnu, 45. – 49. jsou měsíc, 50. - 58. je rok. A konečně 59. bit je parita kalendářních dat.

Popis řešení

V hlavním voidu jsem vytvořil jsem smyčku, která se opakuje každou milisekundu. V té se získá hodnota na pinu D, ke kterému je připojen výstup ze simulátoru. Pak program kontroluje, jak dlouho jsou na pinu nuly, podle čehož určí, jestli je přijatá informace jedna či nula. Pokud za sebou následuje dvacet nul, program pozná začátek vysílání (takhle to fungovat nemá, ale já to lépe neudělal) a začne následující přijaté hodnoty ukládat do 59 místného pole. Když se pole naplní všemi hodnotami, na sériovou linku se vypíší bity vysílání.

Rozbor proměnných a metod

main.c

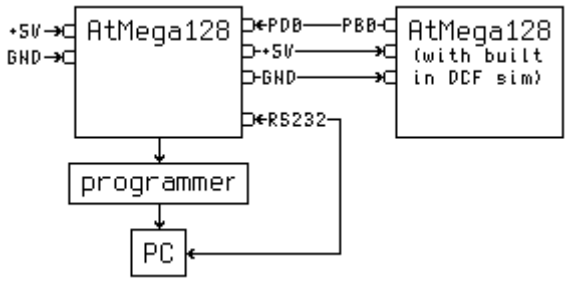
`void setup(void)`	Nastavení CPU portů
`int getAt(int input, int pos)`	Získá bit na pozici pos v čísle input
`void parseData(int dcfData[59])`	Parsování DCF signálu, v mém řešení pouze vypíše bity
`int main(void)`	Vstupní část programu

int main(void)

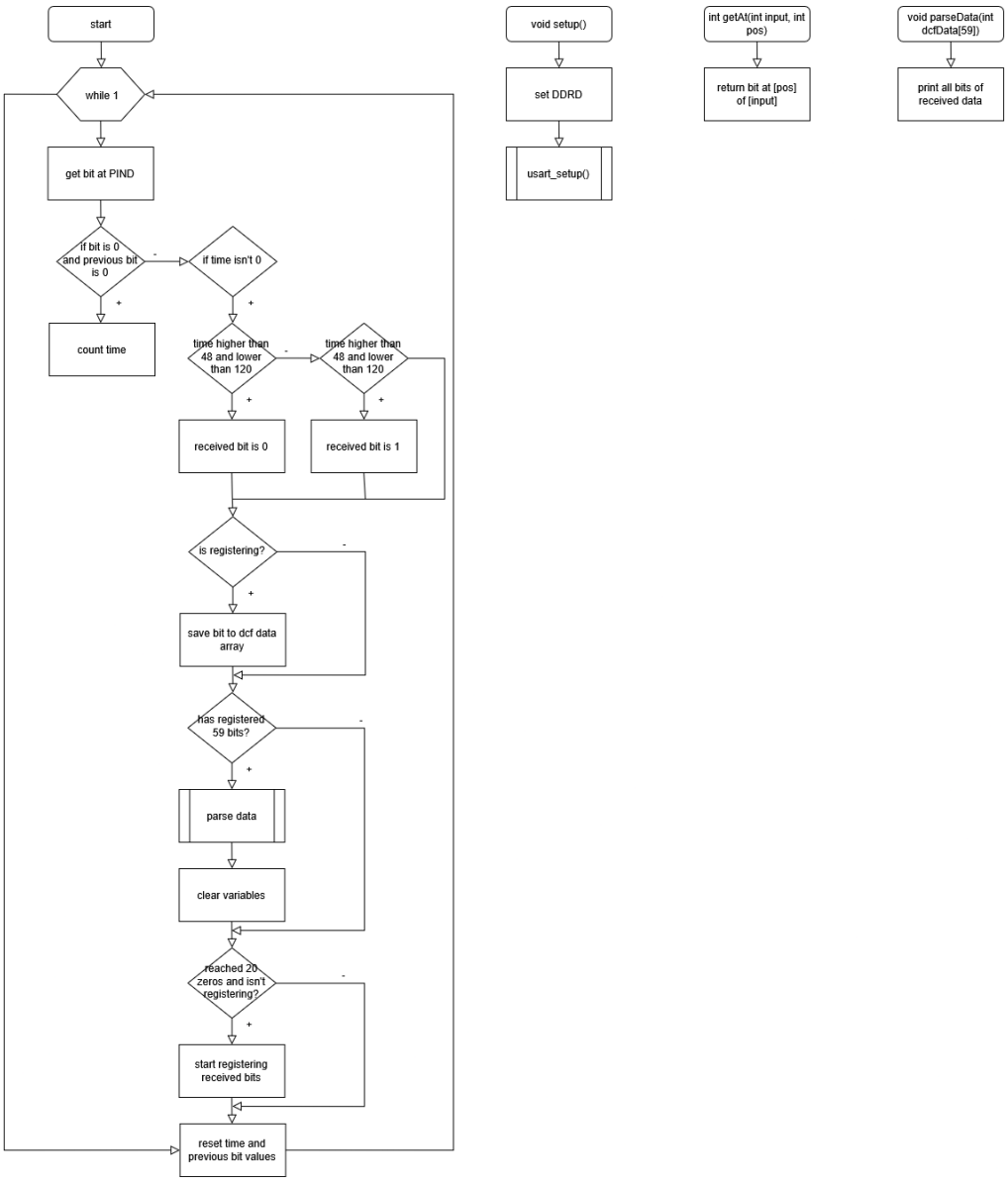
`int dcfData[59]`	Ukládání bitů DCF signálu

`int registerPos = 21`	Pozice začátku samotných DCF dat
`int prevBit = -1`	Ukládání předchozího přijmutého bitu
`int zeroCount = 0`	Počítání přijatých nul
`int isRegistering = 0`	Boolean, jestli má program ukládat přijaté bity
`int time = 0`	Ukládání času bitu

Schéma zapojení



Vývojový diagram



Komentovaný výpis kódu

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <string.h>
#include "usart.h"

//configure CPU and devices
void setup(void) {
   DDRD = 0b00000000;

   usart_setup();
}

//gets bit at (pos) in (input)
int getAt(int input, int pos) {
   return (input >> pos) & 1;
}

//parsing dcf data
void parseData(int dcfData[59]) {
   printf("\n\r");

   for(int i = 0; i < 59; i++) {
      printf("%i", dcfData[i]);
   }

   printf("\n\r");
}

int main(void) {
   setup();

   int dcfData[59];
   int registerPos = 21;

   int prevBit = -1; //saving previous bit for checking

   int zeroCount = 0; //count how much 0s are after eachother to get start
   int isRegistering = 0; //once 20 zeros are reached, start registering

   int time = 0; //time in ms for how long the 0 has been there

   while(1) {
      //gets the bit which changes
      int x = getAt(PIND, 0);

      //count zeros
      if(x == 0 && prevBit == 0) {
         time++;
      } else {
         if(time != 0) {
            int bit = -1; //the bit in the dcf signal
            printf("%i ", time);

            if(time >= 48 && time <= 120) { //the bit is 0
               bit = 0;
```

```
            zeroCount++;
         } else if(time >= 138 && time <= 240) { //the bit is 1
            bit = 1;
            zeroCount = 0;
         }

         //if we're registering write bit to array
         if(isRegistering == 1) {
            dcfData[registerPos] = bit;
            registerPos++;
         }

         //parse data if sequence finished
         if(registerPos == 59) {
            parseData(dcfData);
            zeroCount = 0;
            isRegistering = 0;
            registerPos = 21;
         }

         //start registering if 20 zeros hit
         if(isRegistering == 0 && zeroCount == 20) {
            printf("r ");
            isRegistering = 1;
         }

         //reset values
         time = 0;
         prevBit = -1;
      }
   }
   //set previous bit
   prevBit = x;

   _delay_ms(1);
}
}
```

Odpovědi na otázky

1. Pokuste se navrhnout jak byste řešili požadavek na přesný čas v případě nepravidelně zapínaného zařízení pracujícího v izolovaném režimu bez sítě. Čím musí být takové zařízení vybaveno?

> Nejspíš by šel použít modul reálného času (RTC module). Je to zařízení které poskytuje aktuální čas, i když není nějakou dobu napájeno ze sítě.[1]

2. Množství počítačových systémů používá pro interní reprezentaci času tzv. unix timestamp. Uveďte princip uložení času v tomto formátu a naznačte, jak byste realizovali výpočet počtu dnů mezi dvěma uloženými časy.

> UNIX timestamp se ukládá jako počet sekund od 1.1.1970. Půlnoc tohoto dne má timestamp `0`, dvanáct hodin tohoto dne má timestamp `43200`. Devátého června 2025, devět hodin, čtrnáct minut a šest sekund je `1749460446`.[2] Rozdíl dnů vypočítám převodem na vteřiny → minuty → hodiny → dny.

3. Historie se stále opakuje. Tzv. Y2k bug naštěstí žádný počítačový Armagedon nepřinesl. Pro systémy používající unix timestamp může být podobnou komplikací 19.leden 2038. Pokuste se naznačit proč.

> Protože 32 bitové systémy ukládají UNIXový čas jako integer, dojde k overflow hodnoty a z maximální intové hodnoty se stane minimální. Počítač si tedy bude myslet, že není 19. ledna 2038 3:14:07, ale 13. ledna 1901 20:45:52.[3]

Závěr

Opět úloha psaná v Céčku, která mi nesedla. Dostal jsem se do fáze, kdy se mi na sériovou linku vypisují bity z DCF signálu, a to je vše - převádění do čitelného času jsem nestihl. Navíc můj program hledá začátek signálu pomocí dvaceti za sebou jdoucích nul, což je špatně. Místo toho mám čekat na dlouhý signál značící příjem dat.

1 https://botland.cz/536-rtc-moduly

2 https://www.epochconverter.com/

3 https://en.wikipedia.org/wiki/Year_2038_problem