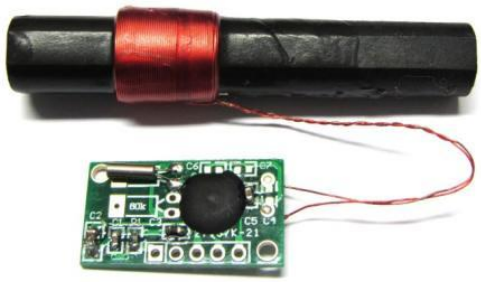
Projekt na Systemy Wbudowane

Opis zestawu:

Projekt opierał się będzie o płytkę testową atmega8 kupiona w firmie nestor-electronic. Do płytki dołączony został moduł odbiornika DCF77:



Płytka zawiera zegar czasu rzeczywistego PCF8583, podczas komunikacji mikrokontrolera z układem czasem występowały zakłócenia. Powodem zakłóceń okazał się brak wymaganych oporników podciągających na magistrali I2C, które według dokumentacji układu są wymagane do poprawnej komunikacji. Zostały wlutowane oporniki 10K w miejsce pozostawione na płytce do wlutowania (w dokumentacji nie było podane dokładnej wartości jednakże w Internecie występują różne wartości tych oporników ostatecznie wybrane zastały 10K). Po ich wlutowaniu komunikacja z mikrokontrolera z układem była pozbawiona zakłóceń.

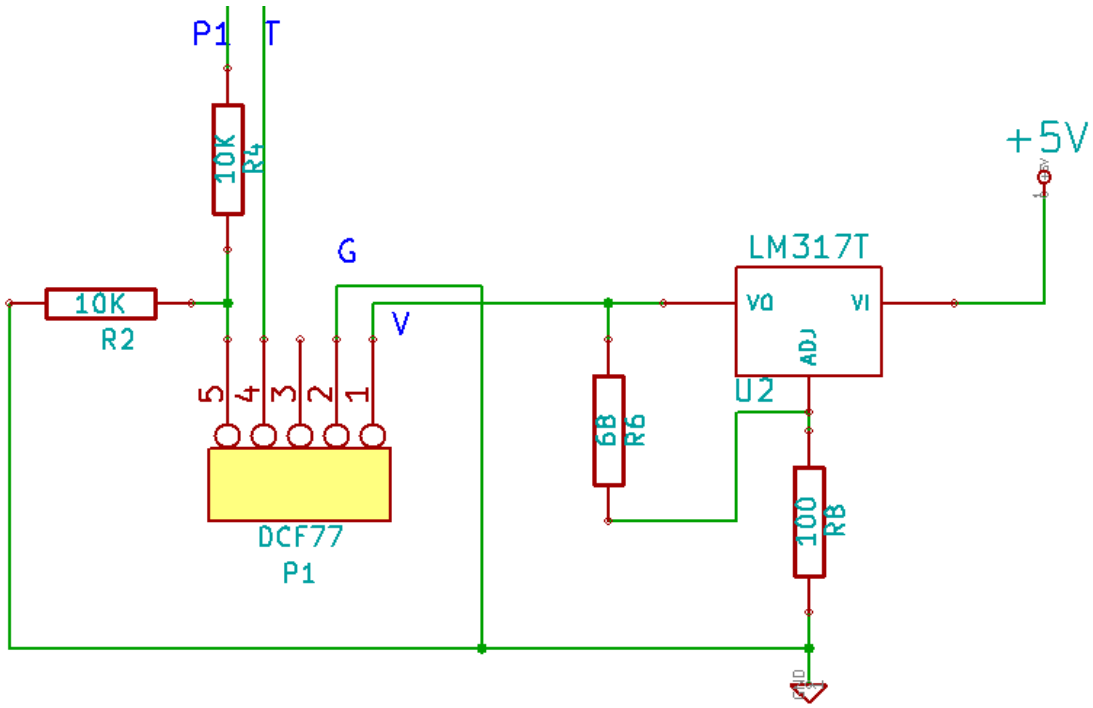
Ponieważ do dekodowania sygnału najlepiej jest posłużyć się obsługując przerwanie z układu DCF77 dlatego należało podłączyć ten sygnał do wejścia INT0 lub INT1 mikrokontrolera. Niestety oba wejścia były już zajęte które zostały podłączone przez producenta płytki do wyświetlacza. W takim wypadku należało przeciąć ścieżkę np. INT1. I zastąpić to połączenie do wyświetlacza innym wolnym pinem.

Ponieważ zasilanie DCF77 powinno wynosić do 3.6V do płytki został dolutowany stabilizator który na wyjściu podaje 3V.

Włączenie odbiornika następuje podaniem stanu niskiego na wejście P1, stan wysoki oznacza wyłączenie odbiornika. Aby zapewnić kompatybilność napięć mikrokontroler 5v odbiornik 3V. Na wyjściu PD3 atmegi został dołożony dzielnik napięcia przez 2, czyli odbiornik DCF77 na wejściu **P1** otrzyma napięcie wynoszące około 2.5v które jest bezpieczne dla układu.

Wyjście **T** od odbiornika jest podłączone do wejścia INT1 mikrokontrolera.

Wejście **G** oznacza masę układu, **V** to źródło napięcia 3V.



Jak widać na schemacie do stabilizatora LM317T dołączone są dwa oporniki dzięki którym można sterować napięciem wyjściowym. Dobór oporników można obliczyć używając kalkulatora ze strony: http://www.electronics-lab.com/articles/LM317/

O	Góra
O	OK
O	Dół

Po włączeniu urządzenia na wyświetlaczu pojawi się godzina oraz data która pobierana jest z układu RTC za pomocą magistrali i2c.

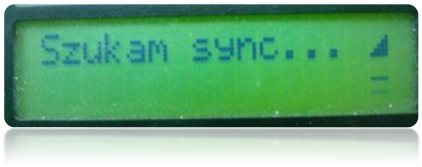
Obsługa menu:

Pozycje głównego menu zmienia się przyciskami Góra/Dół:

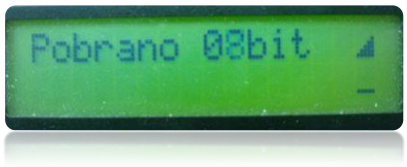
1) Godzina i data

2) Ustaw z DCF
   - Po naciśnięciu przycisku OK, nastąpi włączenie modułu odbiornika DCF, rozpocznie się

   poszukiwanie bitu synchronizacji:



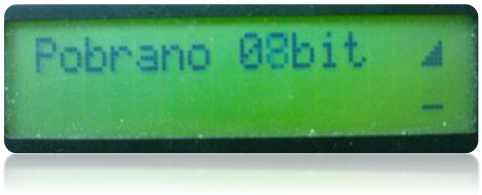
   - Gdy zostanie odnaleziony bit synchronizacji, nastąpi pobieranie kolejnych bitów co sekundę:



   - Jeżeli zostanie pobranych 58 bitów oraz gdy funkcja dekodująca sprawdzi czy sygnał nie zawiera błędów to czas wraz z datą zostanie zapisany do układu RTC, po zapisaniu zostanie wyświetlony ekran 1
   - Jeżeli podczas pobierania danych zostanie wykryty błąd sygnału(wskaźnik zasięgu się zmieni), to pobieranie czasu zaczyna się od nowa czyli najpierw szukanie bitu synchronizacji i następnie pobieranie kolejnych bitów.
   - Aby przerwać synchronizacje i powrócić do głównego menu należy nacisnąć przycisk OK.
3) Porównaj z DCF

   - Po naciśnięciu przycisku OK, nastąpi włączenie modułu odbiornika DCF, rozpocznie się poszukiwanie bitu synchronizacji:

   - Gdy zostanie odnaleziony bit synchronizacji, nastąpi pobieranie kolejnych bitów co sekundę:



- Jeżeli podczas pobierania danych zostanie wykryty błąd sygnału(wskaźnik zasięgu się zmieni), to pobieranie czasu zaczyna się od nowa czyli najpierw szukanie bitu synchronizacji i następnie pobieranie kolejnych bitów.
- Jeżeli zostanie pobrane 58 bitów i pobrane bity będą prawidłowe zostanie wyświetlony czas pobrany z DCF (d) i RTC (r) po sekundach po kropce znajdują się dodatkowo milisekundy:

- Po chwili porównanie zacznie się od nowa czyli od pobrania sygnału DCF77
- Aby przerwać porównywanie i powrócić do głównego menu należy nacisnąć przycisk OK.

4) Ustaw ręcznie..

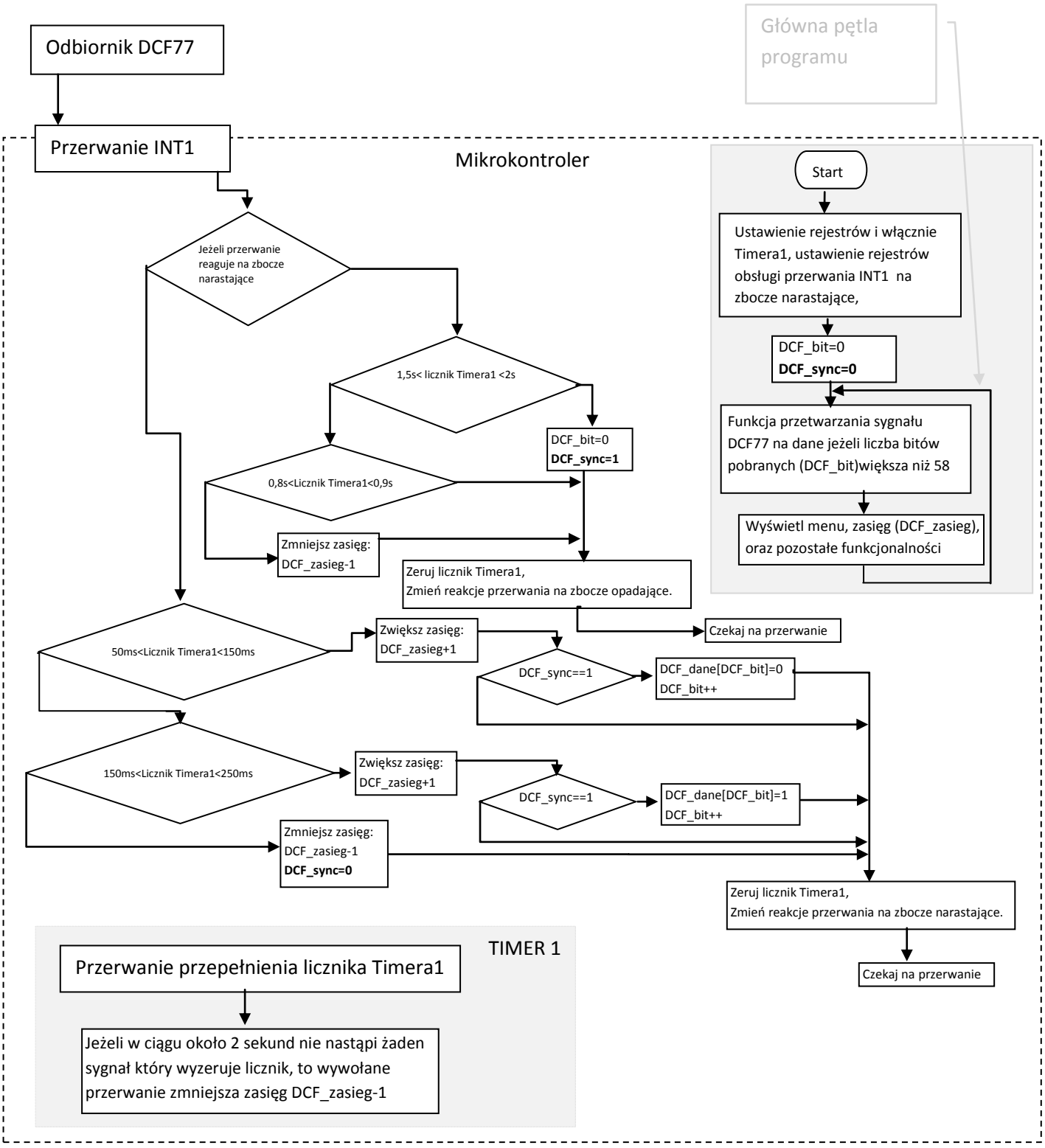
- Po naciśnięciu przycisku OK pokaże się godzina i data którą można edytować wybraną część godziny lub daty, no naciśnięciu przycisku OK następuje edycja kolejnej cześć składowej, gdy wybrana jest ostatnia część daty czyli rok naciśnięcie przycisku OK spowoduje pojawienie się:

- Czyli po naciśnięciu przycisku OK wybrana godzina i data zostanie zapisana, po czym pojawi się godzina i data (ekran 1), jeżeli zostanie naciśniety przycisk Góra lub Dół, ustawienia zostaną anulowane.

W celu ułatwienia ustawienia odbiornika DCF77 w odpowiednim miejscu z prawej strony dla menu 2 i 3 pojawia sie zasięg , oraz aktualny stan bitu danych.

Aby móc poprawnie odebrać sygnał DCF widoczny zasięg musi być cały czas pełny. Jeżeli podczas odbierania sygnału zasięg się pogorszy lub zostaną wykryte zakłócenia, sygnał nie zostanie odebrany.

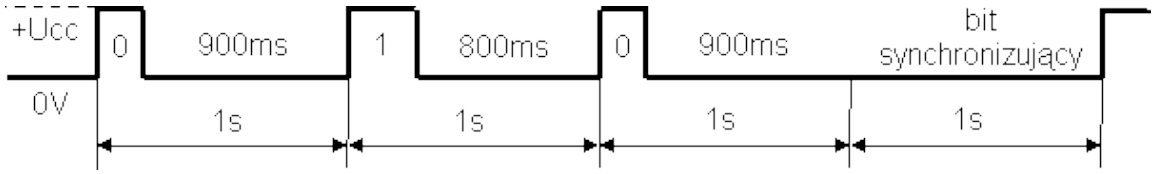
Algorytm odbierania sygnału DCF77 oraz badanie zasięgu.



Algorytm dekodowania sygnału w postaci listy kroków:

1. Przerwanie INT1 ustawione na zbocze narastające.
2. Ustawienie flagi DCF_sync=false – która oznacza ze sygnał nie został jeszcze zsynchronizowany.
3. Uruchomione przerwanie INT1 od zbocza narastającego sygnału DCF77.
    a. Wyłączenie TIMER'a. Sprawdzenie czasu TIMERA czy jego wartość mieści się w przedziale od 1,5s do 2s jeżeli tak to znaczy ze znaleziono bit 0 (bit synchronizacji), ustawiona zostaje flaga DCF_sunc=true oraz zmiennej DCF_bit=0;
    b. następnie zostaje uruchomiony TIMER z początkową wartością licznika 0 oraz następuje zmiana rejestrów przerwania INT1 tak aby przerwanie reagowało na zbocze opadające.
4. Uruchomione przerwanie INT1 od zbocza opadającego sygnału DCF77.
    a. Wyłączenie TIMER'a. Zmiana rejestrów przerwania INT1 aby reagował na zbocze narastające.
    b. Jeżeli flaga DCF_sync jest ustawiona na wartość true, to można przejść do kroku 5, jeżeli nie to należy włączyć TIMER z wartością licznika równą 0, i oczekiwać na kolejne przerwanie.
5. Należy sprawdzić czas z TIMER'a jeżeli mieści się pomiędzy 150ms a 250ms oznacza to że bit jest równy 0, jeżeli czas z TIMER'a mieści siew przedziale od 250ms do 350ms oznacza bit o wartości 1.
    a. Wartości bitów zapisywane są do globalnej tablicy DCF_data[] o indenie DCF_bit który jest zwiększany.
    b. Jeżeli DCF_bit jest większy od 57, to uruchamiana jest funkcja DCF_decode(), która dekoduje pobrany sygnał.
    c. Jeżeli czas z TIMER'a nie mieści się w określonych przedziałach czasu oznaczać to może zakłócenia, flaga DCF_syn ustawiona jest na false i następuje oczekiwanie na ponowny bit synchronizacji.

Przebieg sygnału:



Dekodowanie sygnału:

Sygnał DCF77 składa się z 59 bitów które są odczytywane co sekundę. Poniższa tabela ilustruje znaczenie kolejnych bitów.

Bit (sekunda)	Znaczenie
0	początek transmisji, zawsze 0
1-14	informacje pogodowe (od listopada 2006)
15	antena normalna – 0, antena pomocnicza – 1
16	normalnie – 0, zapowiedź zmiany czasu (przez godzinę przed zmianą) – 1
17-18	od najbardziej znaczącego bitu (18,17) – czas zimowy 10, czas letni 01
19	normalnie – 0, zapowiedź dodatkowej sekundy – 1
20	start informacji czasowej, zawsze 1
21-24	(w kolejności bity 24,23,22,21) jednostki minut w BCD
25-27	(w kolejności bity 27,26,25) dziesiątki minut w BCD
28	bit parzystości dla bitów 21-27
29-32	(w kolejności bity 32,31,30,29) jednostki godzin w BCD
33-34	(w kolejności bity 34,33) dziesiątki godzin w BCD
35	bit parzystości dla bitów 29-34
36-39	(w kolejności bity 39,38,37,36) jednostki dni miesiąca w BCD
40-41	(w kolejności bity 41,40) dziesiątki dni miesiąca w BCD
42-44	(w kolejności bity 44,43,42) dni tygodnia w BCD (1 = poniedziałek; 7 = niedziela)
45-48	(w kolejności bity 48,47,46,45) jednostki miesiąca w BCD
49	dziesiątki miesiąca w BCD
50-53	(w kolejności bity 53,52,51,50) jednostki lat w BCD
54-57	(w kolejności bity 57,56,55,54) dziesiątki lat w BCD
58	bit parzystości dla bitów 36-57
(59)	brak impulsu – zapowiedź następnej ramki

Opis kodu programu

Program został napisany w języku c, skompilowany za pomocą kompilatora gcc poprzez środowisko AVR studio 4, program został wgrany do mikrokontrolera za pomocą narzędzia AVR Burn-O-Mat poprzez programator dołączony do płytki testowej.

W celu obsługi wyświetlacza oraz zegara PCF8583, pobrane zostały odpowiednie biblioteki, które są ogólno dostępne w Internecie:
- Pliki do obsługi wyświetlacza: HD44780.c, HD44780.h,
- Pliki do obsługi RTC: pcf8583.c, tools.c , pcf8583.h, tools.h

W celu podziału programu na mniejsze części stworzona została biblioteka DCF.h, DCF.c która obsługuje wyłącznie odbiór i dekodowanie sygnału DCF77. Wszystkie biblioteki znajdują się w katalogu lib.

Opis biblioteki DCF.h

W bibliotece znajdują się definicje takie jak:

```
#define TIMER1_ON (TIMSK |= _BV(TOIE1))
#define TIMER1_OFF (TIMSK &= ~_BV(TOIE1))
```

- ✓ Służą odpowiednio do włączenia i wyłączenia timera1

```
#define DCF_INT INT1
#define DCF_INT_vector INT1_vect
```

- ✓ Wybór przerwania oraz wektora przerwania

```
#define DCF_INT_ON GIFR|=_BV(INTF1)
#define DCF_INT_OFF GIFR&=~_BV(INTF1)
```

- ✓ Włączenie i wyłączenie obsługi przerwania od wejścia INT1

```
#define DCF_INT_RISING MCUCR =_BV(ISC11) | _BV(ISC10)
#define DCF_INT_FALLING MCUCR=_BV(ISC11)
```

- ✓ Ustawienia przerwania aby reagował na zbocze narastające lub opadające

```
#define DCF_IF_RISING ((MCUCR & _BV(ISC11)) && (MCUCR & _BV(ISC10)))
#define DCF_IF_FALLING ((MCUCR & _BV(ISC11)) && (~MCUCR & _BV(ISC10)))
```

- ✓ Definicja sprawdzająca czy przerwanie reaguje na zbocze narastające czy opadające

```
#define DCF_P1_DIR      DDRB
#define DCF_P1_PORT     PORTB
#define DCF_P1          _BV(PB0)
```

- ✓ Definicja rejestru kierunku portu, portu, oraz konkretnego wyjścia portu, służy do późniejszego inicjalizowania kierunku portu który służy do włączania lub wyłączania odbiornika DCF77.

```
#define DCF_ON          DCF_P1_PORT &=~ DCF_P1
#define DCF_OFF         DCF_P1_PORT |= DCF_P1
```

- ✓ Włączanie(pin podaje logiczne 0) lub wyłączenie (pin podaje logiczną 1) odbiornika DCF77

```
typedef struct DCF_dt { uint8_t min;
                        uint8_t h;
                        uint8_t month;
                        uint8_t day;
                        uint16_t year;
} DCF_datetime;
```

- ✓ Definicja struktury danych zawierająca: minuty, godziny oraz datę

Zadeklarowane zmienne:

```
volatile uint8_t DCF_sync;
```

- ✓ Zawiera informacje czy odnaleziony został bit synchronizacji (0 -fałsz lub 1 -prawda)

```
volatile uint8_t DCF_zasieg;
```

- ✓ Zawiera informacje na temat zasięgu wartość od 1 do 4

```
volatile uint8_t DCF_bit;
```

- ✓ Zawiera informacje który bit będzie aktualnie pobierany od 0 do 59

```
volatile uint8_t DCF_dane[60];
```

- ✓ Tablica zawierająca pobrane dane

Metody biblioteki:

```
void DCF_Initalize();
```

- ➢ Służy do inicjalizowania zmiennych oraz ustawienia odpowiednich rejestrów do obsługi przerwania i timera

```
uint8_t DCF_decode(DCF_datetime *DCF_dt);
```

- ➢ Dekoduje pobrany sygnał DCF77, na postać wcześniej zadeklarowanej struktury **DCF_dt**

```
ISR(TIMER1_OVF_vect);
```

- ➢ Obsługa przerwania od przepełnienia licznika Timera1

```
ISR (INT1_vect);
```

- ➢ Obsługa przerwania sygnału od DCF77

Opis biblioteki DCF.c

Funkcja przygotowująca mikrokontroler do odczytu sygnału DCF77

```
void DCF_Initalize()  {

        DCF_P1_DIR |= DCF_P1;
```

- ➢ konfiguracja kierunku portu służącego do włączenia lub wyłączenia odbiornika

```
        TCCR1B = _BV(CS12);         //preskaler 256 TIMER1
        TCNT1 = 0;                  //wyzerowanie licznika
        TIFR |= _BV(TOV1) ;
```

- ➢ konfiguracja TIMER1 w taki sposób aby licznik timera mógł pomieścić czas 2 sekund ze względu na bit synchronizacji który należy przyjąć iż mieści się w przedziale od 1,5s do 2s
  Ponieważ mikrokontroler pracuje z częstotliwością 8MHz a Licznik Timera 1 jest 16bitowy to preskaler został obliczony w następujący sposób:
  2s / 65535(max wartość 16b) = 0,00003052s jest to wartość co jaki czas musi się zwiększać licznik aby mógł pomieścić czas 2s
  8MHz = 1/8000000s – czas jednego taktu mikrokontrolera
  0,00003052s / 1/8000000s = 244 – co 244 takty mikrokontrolera powinien być zwiększany licznik Timera. Najbliższy możliwy preskaler wynosi **256**

```
        GICR|=_BV(DCF_INT);
        DCF_INT_RISING; //zbocze narastajace -szukanie bitu synchronizacji
        DCF_INT_ON;
```

- ➢ konfiguracja przerwania od DCF77 (wykorzystywane wcześniejsze definicje DCF_*), z ustawieniem reakcji na zbocze narastające, włączenie obsługi przerwania

```
        DCF_sync=0;
        DCF_bit=0;
        DCF_zasieg=1;
        sei();
```

- ➢ Ustawienia początkowe zmiennych, brak synchronizacji, kolejny pobierany bit = 0, ustawienie zasięgu na minimum, oraz globalne włączenie obsługi przerwań się()

```
}
```

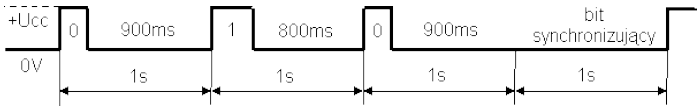
Obsługa przerwania od przepełnienia licznika Timera 1

```
ISR(TIMER1_OVF_vect)
{
        if (DCF_zasieg>1)
                DCF_zasieg--;
}
```

- Przerwanie przepełnienia generowane jest podczas gdy na wejściu przerwania INT1 nie będzie żadnego sygnału przez okres większy niż 2 sekundy, wtedy zasięg zostaje stopniowo zmniejszany.

Obsługa przerwania od zmiany sygnału DCF77

Przed przystąpieniem do analizy funkcji obsługi przerwania, należy obliczyć jaka wartość licznika Timera 1 reprezentuje czas trwania ciągłego sygnału logicznego.
Prawidłowy sygnał DCF77:



Mikrokontroler pracuje z częstotliwością 8MHz czas jednego taktu wynosi **1/8000000 s**, preskaler Timera 1 wynosi 256.
Czyli czas zmiana wartości licznika Timera wynosi: **1/8000000 * 256 = 0,000032s**
Użyte w funkcji czasy prezentuje poniższa tabela która uwzględnia tolerancje(górną i dolną granicę czasu):

znaczenie	Czas typowy	Czas dolna granica	Czas górna granica	Czas dolna granica / 0,000032s	Czas górna granica / 0,000032s
Bit o wartości 0	100ms	50ms	150ms	1562	4687
Bit o wartości 1	200ms	150ms	250ms	4688	7812
Czas bitu synchronizacji	1,8s – 1,9s	1,5s	2s	46875	15625
Stan logiczny 0 poza bitem synchronizacji	800ms – 900ms	750ms	950ms	23437	62500

Poniższa funkcja jest implementacją powyższego schematu blokowego

```
ISR (DCF_INT_vector)
{
      if (DCF_IF_RISING) //jeżeli reakcja na zbocze narastające rancza że poprzednim poziom logiczny to 0
      {
            if ((TCNT1 > 46875) && (TCNT1 < 62500)) //szukaj bitu synchronizacji czas od 1,5s do 2s
            {
                  DCF_sync=1; // ustaw bit synchronizacji
                  DCF_bit=0; // kolejny bit zapisany do tablicy o indexie 0
            }
            else if (!((TCNT1 > 23437) && (TCNT1 < 29687))) // jeżeli sygnał 0 nie mieści sie w normach
                                                                  (750ms - 950ms) to zmniejsz zasięg
            {
                  if (DCF_zasieg>1)
                        DCF_zasieg--; // zmniejsz zasięg
            }
            TCNT1=0; // wyzereuj licznik Timera 1
            DCF_INT_FALLING; // ustawienie reakcji przerwania na zbocze opadające
      }
      else
      {
            if ((TCNT1 > 1562) && (TCNT1 < 4687)) // sprawdź czy czas mieści się w przedziale 50ms - 150ms
            {
                  if (DCF_zasieg<4)
                        DCF_zasieg++; //zwiesz zasięg

                  if (DCF_sync==1)// jeżeli odnaleziono bit synchronizacji to pobierz bit
                  {
                        DCF_dane[DCF_bit]=0; // zapisz bit
                        DCF_bit++; // zwiększ index
                  }
            }
            else if ((TCNT1 > 4688) && (TCNT1 < 7812)) // jeżeli czas 150ms - 200ms to oznacza bit 1
            {
                  if (DCF_zasieg<4)
                        DCF_zasieg++; // zwiększ zasięg

                  if (DCF_sync==1)// jeżeli odnaleziono bit synchronizacji to pobierz bit
                  {
                        DCF_dane[DCF_bit]=1;// zapisz bit do tablicy o określonym indexie DCF_bit
                        DCF_bit++; //zwiększ index
                  }
            }
            else // jeżeli inny czas to błąd sygnału
            {
                  DCF_sync=0; //usuń flagę synchronizacji
                  if (DCF_zasieg>1)
                        DCF_zasieg--; // zmniejsz zasięg
            }
      TCNT1=0; // wyzeruj licznik Timera 1
      DCF_INT_RISING; // ustaw reakcje przerwania na zbocze narastające
}}
```

Funkcja dekodująca pobrany sygnał:

W początkowej fazie funkcja sprawdza kilka kryteriów, sumy kontrolne dopiero po spełnieniu warunków dekoduje sygnał.

```
uint8_t DCF_decode(DCF_datetime *DCF_dt)
{

      if (DCF_bit<59)
            return 0; //jeżeli za mało danych to zakończ
```

✓ Kryterium: potrzebne jest 59 bitów aby zdekodować wszystkie dane

```
      if ((DCF_dane[0]!=0)||(DCF_dane[20]!=1))
            return 0; //jeżeli bity niepoprawne to zakończ
```

✓ Kryterium: bit 0 zawsze równy 0, bit 20 zawsze równy 1

```
      uint8_t DCF_parzystosc_m=0;
      for (int a=21; a<29; a++) //sumowanie bitów
            DCF_parzystosc_m+=DCF_dane[a];
```

➢ Sumowanie bitów 21 – 28 (BCD(minut) + bit parzystości)

```
      if ((DCF_parzystosc_m%2)!=0)
            return 0; // jeżeli suma bitów jest nieparzysta to zakończ
```

✓ Kryterium: suma bitów 21 – 28 musi być parzysta

```
      uint8_t DCF_parzystosc_h=0;
      for (int a=29; a<36; a++)
            DCF_parzystosc_h+=DCF_dane[a];
```

➢ Sumowanie bitów 29 – 34 (BCD(godzin) + bit parzystości)

```
      if ((DCF_parzystosc_h%2)!=0)
            return 0; // jeżeli suma bitów jest nieparzysta to zakończ
```

✓ Kryterium: suma bitów 21 – 28 musi być parzysta

```
      uint8_t DCF_parzystosc_data=0;
      for (int a=36; a<59; a++)
            DCF_parzystosc_data+=DCF_dane[a];
```

➢ Sumowanie bitów 36 - 58 (data + bit parzystości)

```
      if ((DCF_parzystosc_data%2)!=0)
            return 0; // jeżeli suma bitów jest nieparzysta to zakończ
```

✓ Kryterium: suma bitów 36 - 58 musi być parzysta

➢ Po sprawdzeniu powyższych kryteriów można przejść do właściwej części dekodowania sygału

➢ **dekodowanie minut**

```
      DCF_dt->min    = DCF_dane[21]
                     + 2*DCF_dane[22]
                     + 4*DCF_dane[23]
                     + 8*DCF_dane[24]
                     + 10*DCF_dane[25]
                     + 20*DCF_dane[26]
                     + 40*DCF_dane[27];
```

21-24	(w kolejności bity 24,23,22,21) jednostki minut w BCD
25-27	(w kolejności bity 27,26,25) dziesiątki minut w BCD
28	bit parzystości dla bitów 21-27

➢ **dekodowanie godzin**

```
      DCF_dt->h = DCF_dane[29]
                     + 2*DCF_dane[30]
                     + 4*DCF_dane[31]
                     + 8*DCF_dane[32]
                     + 10*DCF_dane[33]
                     + 20*DCF_dane[34];
```

29-32	(w kolejności bity 32,31,30,29) jednostki godzin w BCD
33-34	(w kolejności bity 34,33) dziesiątki godzin w BCD
35	bit parzystości dla bitów 29-34

➢ **dekodowanie dnia miesiąca**

```
      DCF_dt->day    = DCF_dane[36]
                     + 2*DCF_dane[37]
                     + 4*DCF_dane[38]
                     + 8*DCF_dane[39]
                     + 10*DCF_dane[40]
                     + 20*DCF_dane[41];
```

36-39	(w kolejności bity 39,38,37,36) jednostki dni miesiąca w BCD
40-41	(w kolejności bity 41,40) dziesiątki dni miesiąca w BCD

➢ **dekodowanie miesiąca**

```
      DCF_dt->month  = DCF_dane[45]
                     + 2*DCF_dane[46]
                     + 4*DCF_dane[47]
                     + 8*DCF_dane[48]
                     + 10*DCF_dane[49];
```

45-48	(w kolejności bity 48,47,46,45) jednostki miesiąca w BCD
49	dziesiątki miesiąca w BCD

- **dekodowanie roku**

```
DCF_dt->year = DCF_dane[50]
               + 2 * DCF_dane[51]
               + 4 * DCF_dane[52]
               + 8 * DCF_dane[53]
               + 10 * DCF_dane[54]
               + 20 * DCF_dane[55]
               + 40 * DCF_dane[56]
               + 80 * DCF_dane[57]
               + 2000;
```

50-53	(w kolejności bity 53,52,51,50) jednostki lat w BCD
54-57	(w kolejności bity 57,56,55,54) dziesiątki lat w BCD

```
if ((DCF_dt->h>24)||(DCF_dt->min>60)||(DCF_dt->month>12)||(DCF_dt->day>31))
     return 0;
```

- Kryterium: godzina nie większa niż 24, minuty nie większe niż 60

```
return 1;
```

- zakończenie sukcesem

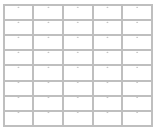
```
}
```

Opis głównego programu

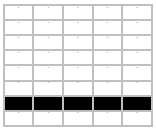
```
#define KEY_PIN          PINB
#define KEY_DDR          DDRB
#define KEY_PORT         PORTB
#define KEY_UP           PB5
#define KEY_DOWN         PB4
#define KEY_OK           PB3
```

- Definicja klawiszy, rejestr odczytu stanów pinów, kierunek portu, rejestr zapisu stanów poszczególnych pinów, oraz definicja 3 pinów

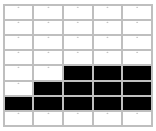
```
char zasieg[4][8]={     {0,0,0,0,0,0,0,0},
```



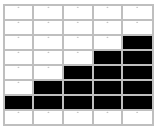
```
                        {0,0,0,0,0,0,0x1F,0},
```



```
                        {0,0,0,0,0x07,0x0F,0x1F,0},
```



```
                        {0,0,0x01,0x03,0x07,0x0F,0x1F,0}
```



```
                    };
```

- Definicja 4 znaków służących do prezentacji zasięgu

```
DCF_datetime DCF_dekode;
```

- Zmienna w której zapisywane są zdekodowane dane
  Wywołana funkcja **DCF_decode**(&DCF_dekode) jeżeli zwróci wartość różną od 0 to sygnał poprawnie zdekodowany i zdekodowane dane zostały zapisane do zmiennej DCF_dekode.

```
pcf8583_ctl data_save;
pcf8583_ctl data_read;
```

- Pomocnicze zmienne służące do zapisu i odczytu danych z układu RTC

Główna funkcja programu

Zajmuje się obsługą menu, oraz zapisem i odczytem danych z i do RTC, oraz pobiera czas z odbiornika DCF77 za pomocą wcześniej opisanej biblioteki.

```
int main()
{
        pcf8583_ctl data;
```

- zmienna przechowująca aktualnie pobrany czas

```
        twi_init();
```

- inicjalizacji magistrali i2c do komunikacji z RTC

```
        KEY_DDR&=~(_BV(KEY_UP)|_BV(KEY_OK)|_BV(KEY_DOWN));
        KEY_PORT|=_BV(KEY_UP)|_BV(KEY_OK)|_BV(KEY_DOWN); //pull-up
```

- Ustawienie pinów klawiszy jako wejście, oraz dodatkowo ustawiony Pull-Up, czyli wewnętrzne podciągniecie wejścia do plusa, ponieważ piny mikrokontrolera podłączone do przycisków podczas stanu nieprzewodzenia, mogą generować zakłócenia ponieważ poszczególne piny wiszą w powietrzu, ponieważ na płytce nie ma zewnętrznych oporników podciągających.

```
        DDRD&=~_BV(PD3);
```

- Pin PD3 jest to wejście przerwania od DCF77, dlatego należy ustawić kierunek pinu na wejście

```
        DCF_Initalize();
```

- Inicjalizacja składników biblioteki DCF77, czyli ustawienia rejestrów przerwania, konfiguracja Timera1

```
        LCD_Initalize();
        LCD_Def_Char(zasieg[0], 1);
        LCD_Def_Char(zasieg[1], 2);
        LCD_Def_Char(zasieg[2], 3);
        LCD_Def_Char(zasieg[3], 4);
```

- Inicjalizacja wyświetlacza oraz zdefiniowanie 4 znaków zasięgu które dostępne będą w wyświetlaczu pod kodem 1..4

```
        char buff[20];
```

- Bufor na tekst który zostanie wyświetlony na wyświetlaczu

```
        uint8_t menu=0;
        uint8_t podmenu=0;
```

- Zmienne służące do obsługi menu: menu [0..3] – aktualna pozycja menu
  podmenu – służy do obsługi wewnątrz wybranej pozycji menu

```
        uint16_t licznik=0;
        uint8_t show=0;
```

- Zmienne służące do obsługi migotania aktualnie edytowanej pozycji (składowej godziny lub daty) w menu (ustaw ręcznie)

```
while(1) {
```

- Główna nieskończona pętla (zawiera dwie części pierwsza to obsługa przycisków i menu druga to obsługa wyświetlacza)

- Sprawdzanie stanu przycisków (KEY_UP, KEY_OK, KEY_DOWN) i odpowiednie przestawienia zmiennych, po naciśnięciu dowolnego przycisku ustawione jest opóźnienie _delay_ms(150) które alei minuje zjawisko drgania styków:

- Jeżeli aktywne menu = 3 oznacza to menu w którym ustawiana jest godzina i data ręcznie, przy czym zmienna podmenu 1..8 wskazuje aktualną edytowaną składakową np. godzina, minuta…

```
...
if (bit_is_clear(KEY_PIN, KEY_UP)) {
     if (menu==3)
     {
          switch (podmenu) // zwiększanie poszczególnych wartości
          {
               case 1:if (data.hr<23) data.hr++; break; // godziny max 23
               case 2:if (data.min<59) data.min++; break; // minuty max 59
               case 3:if (data.sec<59) data.sec++; break; //sekundy max 59
               case 4:if (data.days<31) data.days++; break; // dzień max 31
               case 5:if (data.month<12) data.month++; break; // miesiąc max 12
               case 6:if ((data.year/100)<99) data.year+=100; break; // część roku liczba setek
                                                                     max 99
               case 7:if ((data.year/100)<99) data.year++; break; //cześć roku dziesiątki
                                                                     jedności
               case 8: LCD_Clear(); podmenu=0;
                       _delay_ms(1000);
                       break; //przejście do potwierdzenia zapisu godziny
          }
          _delay_ms(100);
          show=1;
     }
...
}
```

- Zmienna show odpowiada za mruganie aktualnie zmienianą częścią godziny lub daty, dlatego podczas zmiany wartości show=1, dzięki czemu przy zmianie wartości nie zdarzy się sytuacja, iż edytowana składowa będzie ukryta

```
// sprawdź czy wciśnięty przycisk KEY_OK
else if (bit_is_clear(KEY_PIN, KEY_OK))
{
     ...
          _delay_ms(150);
     ...
}
else if (bit_is_clear(KEY_PIN, KEY_DOWN))
{
     ...
     if (menu==3) // ustawianie czasi i daty
     {
          switch (podmenu) // zwiekszanie poszczególnych wartosci
          {
               case 1:if (data.hr>0) data.hr--; break; // godziny min 0
               case 2:if (data.min>0) data.min--; break; // minuty min 0
               case 3:if (data.sec>0) data.sec--; break; //sekundy min 0
               case 4:if (data.days>1) data.days--; break; // dzien min 1
               case 5:if (data.month>1) data.month--; break; // miesiac min 1
               case 6:if (data.year>=2100) data.year-=100; break; // czesc roku liczba setek
                                                                     min 20
               case 7:if (data.year>2012) data.year--; break; //czesc roku dziesiateki
                                                                     jednosci min 2012
               case 8:LCD_Clear(); podmenu=0;
                               _delay_ms(1000)
                                break; //przejście do potwierdzenia zapisu godziny
          }

          ...
          _delay_ms(100);
          show=1;
     }

}
```

- Część druga czyli prezentacja danych na wyświetlaczu, 4 główne pozycje menu (0..3)
  0: prezentacja godziny i daty
  1: Ustaw z DCF
  2: porównaj z DCF
  3: Ustaw ręcznie

- Poniżej znajduje się obsługa wyświetlania, pobrania czasu z DCF77 i zapisu do RTC, dla menu 1 (Ustaw z DCF)

```
DCF_ON;
LCD_GoTo(15,0);
sprintf(buff, "%c", DCF_zasieg);
LCD_WriteText(buff);
LCD_GoTo(15,1);
if (bit_is_set(PIND, PD3))
     LCD_WriteText("-");
else
     LCD_WriteText("_");
```

- Następuje włączenie odbiornika DCF77 za pomocą zdefiniowanej na początku definicji DCF_ON, zostaje wyświetlony zasięg w postaci wcześniej zdefiniowanych znaków dla wyświetlacza o indexie 1..4, sprintf(buff, "%c", DCF_zasieg) oraz w następnej linii pokazuje się aktualny stan sygnału z odbiornika 1 „-”, 0 „_”

```
if (DCF_sync==0)
{
     LCD_GoTo(0,0);
     LCD_WriteText("Szukam sync... ");
}
```

- Jeżeli nie został odnaleziony bit synchronizacji to wyświetl napis „Szukam sync...”

```
else
{
     LCD_GoTo(0,0);
     sprintf(buff, "Pobrano %02dbit  ", DCF_bit);
     LCD_WriteText(buff);
}
```

- Jeżeli bit synchronizacji został znaleziony to wyświetl ilość pobranych bitów

```
if (DCF_decode(&DCF_dekode)!=0)
{
```

- Funkcja DCF_decode w przypadku pobrania 58 bitów i po sprawdzeniu poprawności zwróci wartość = 1 zapisze zdekodowane dane do struktury DCF_dekode

```
     data_save.h_sec=0;
     data_save.sec=0;
     data_save.min=DCF_dekode.min;
     data_save.hr=DCF_dekode.h;
     data_save.days=DCF_dekode.day;
     data_save.month=DCF_dekode.month;
     data_save.year=DCF_dekode.year;

     write_pcf8583(&data_save);
```

- Przepisanie danych z DCF_dekode do zmiennej data_save, oraz zapisanie czasu i daty do układu PCF8583

```
     LCD_GoTo(2,1);
     sprintf(buff, "%02d:%02d:%02d ", DCF_dekode.h, DCF_dekode.min, 0);
     LCD_WriteText(buff);

     _delay_ms(4000);
     LCD_Clear();
     podmenu=0;
     menu=0;
```

- Zostanie wyświetlony pobrany czas i po chwili na wyświetlaczu pojawi się godzina i data (ekran 1)

```
}

break;
```

- Działanie funkcjonalności „porównaj z DCF”, różni się od powyższego listingu użyciem funkcji read_pcf8583(&data_read); czyli pobraniem czasu z RTC zamiast write_pcf8583(&data_save);

- Po czym następuje wyświetlenie czasu z DCF oraz z RTC wraz z milisekundami, po chwili następuje ponowne pobranie wartości z DCF i kolejne porównanie

Cały schemat układu: