Zegar ClockRDS

W prezentowanym zegarze zastosowano rzadko spotykany sposób na uzyskanie automatycznej synchronizacji czasu. Wykorzystanie systemu RDS okazało się idealnym rozwiązaniem do takiego urządzenia, a zastosowanie nowoczesnego wyświetlacza OLED i niewielkie wymiary płytki, tylko uzupełniły listę jego zalet.

W praktyce każdego elektronika amatora i profesjonalisty są takie projekty, których samodzielne wykonanie to tak zwane „must have". Jednym z takich przykładów jest projekt zegara cyfrowego, który w takiej czy innej formie przewijał się również pośród moich projektów, jednak głównie jako dodatkowa funkcjonalność urządzenia docelowego. Tym razem postanowiłem zaprojektować i wykonać urządzenie dedykowane, którego jedyną funkcjonalnością będzie precyzyjne odmierzanie czasu. Nie chciałem jednak, by był to projekt trywialny, w związku z czym postanowiłem, że zegar ten, oprócz typowej funkcjonalności, będzie dysponował możliwością automatycznej synchronizacji czasu z jakimś bardzo dokładnym wzorcem czasu. Co więcej, założyłem, że projektowane urządzenie wyposażone będzie w nowoczesny i efektowny, graficzny wyświetlacz OLED.

Zgodnie z powyższymi założeniami rozpocząłem prace projektowe, których pierwszym krokiem było określenie medium transmisyjnego, przy udziale którego zegar ten będzie miadał możliwość automatycznej synchronizacji czasu. Na pierwszy ogień poszedł bardzo popularny wzorzec czasu oznaczany skrótem DCF-77, który jest niczym innym, jak specjalnym sygnałem czasu nadawanym na falach długich na częstotliwości 77,5 kHz. Zaopatrzony w gotowy moduł DCF-77 wyposażony w antenę ferrytową, jak i megaoptymizm, rozpocząłem testy praktyczne. Niestety dość szybko okazało się, że złapanie niezakłóconego sygnału wzorca czasu nie jest w praktyce takie łatwe, gdyż zależy w dużym stopniu od warunków środowiskowych, pory dnia, czy ustawienia anteny modułu, co znalazło potwierdzenie na różnych forach poświęconych elektronice. Niestety, w moim wypadku było jeszcze gorzej, gdyż w zasadzie ani razu nie udało mi się odebrać kompletnej ramki danych, co spowodowało, że porzuciłem pierwotny pomysł wykorzystania sygnału DCF-77 jako wzorca czasu.

Dodatkowe materiały do pobrania ze strony www.media.avt.pl

W ofercie AVT* AVT-5677

Podstawowe parametry:
- automatyczna synchronizacja czasu sygnałem RDS,
- prezentacja czasu i daty na wyświetlaczu OLED,
- obsługa za pomocą jednego przycisku,
- zasilanie napięciem 4,5…6 V

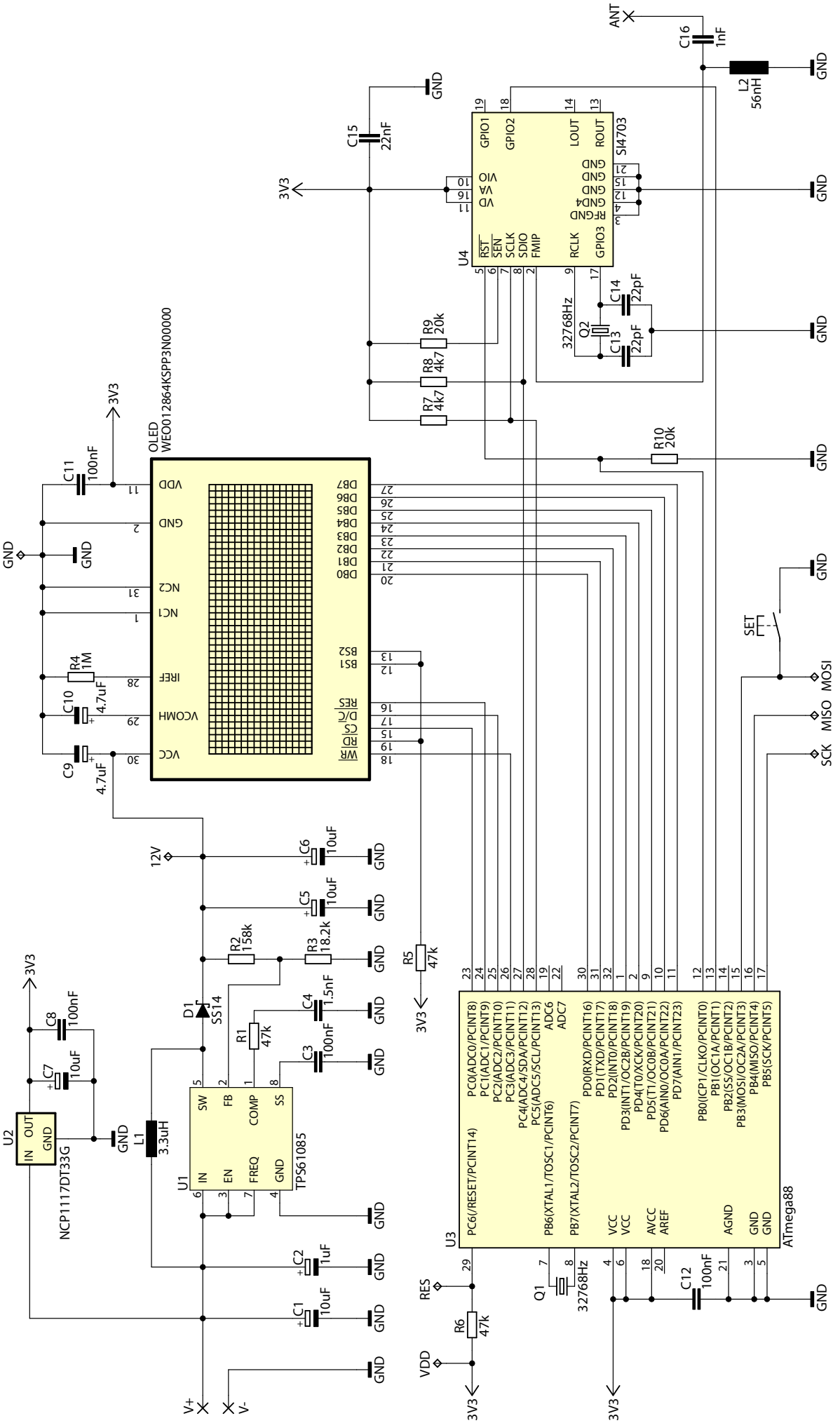
Projekty pokrewne na www.media.avt.pl:

AVT-5640	Rozbudowany zegar (EP 7/2018)
AVT-5522	Zegar ustawiany za pomocą GPS (EP 9/2015)
AVT-3132	Prosty zegar LED (EdW 7/2015)
AVT-1832	Zegar z budzikiem (EP 10/2014)
AVT-5377	Mega stoper – wielofunkcyjny licznik, nie tylko czasu (EP 12/2012)
AVT-513	Zegar ze stuletnim kalendarzem i termometrem (EP 10-11/2011)
AVT-5281	„Inteligentny" zegar z wyświetlaczem LED (EP 3/2011)
AVT-5273	Zegar cyfrowy z analogowym sekundnikiem (EP 1/2011)
AVT-2632	Gigantyczny zegar (EdW 5/2002)
AVT-5022	Programowany zegar z DCF77 (EP 6-7/2001)
AVT-5002	Zegar cyfrowy z wyświetlaczem analogowym (EP 3/2001)
AVT-2017	DCF Clock (EP 7/1994)

*** Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!**
Podstawową wersją zestawu jest wersja [**B**] nazywana potocznie KIT-em (z ang. zestaw). Zestaw w wersji [**B**] zawiera elementy elektroniczne (w tym [**UK**] – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu.
Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:
- wersja [**C**] – zmontowany, uruchomiony i przetestowany zestaw [**B**] (elementy wlutowane w płytkę PCB)
- wersja [**A**] – płytka drukowana bez elementów i dokumentacji

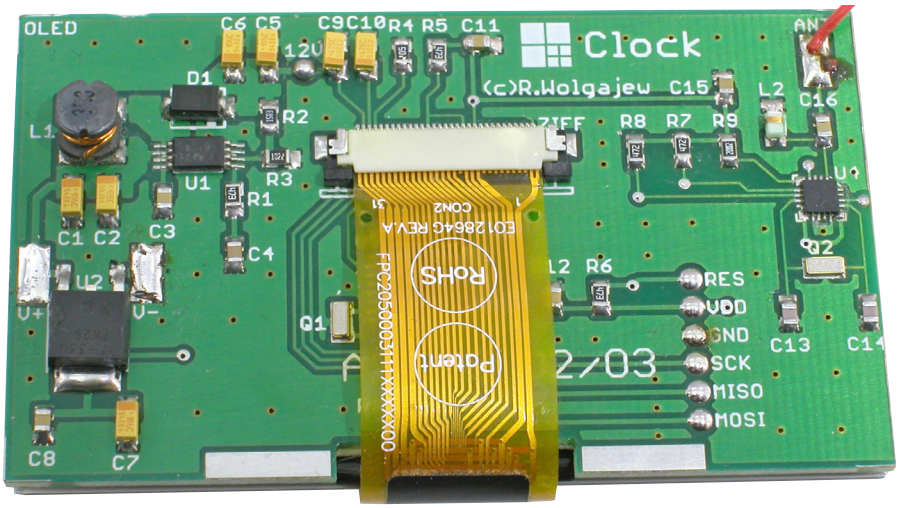
Kity w których występuje układ scalony wymagający zaprogramowania, mają następujące dodatkowe wersje:
- wersja [**A+**] – płytka drukowana [**A**] + zaprogramowany układ [**UK**] i dokumentacja
- wersja [**UK**] – zaprogramowany układ

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! *http://sklep.avt.pl*. W przypadku braku dostępności na *http://sklep.avt.pl*, osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: kity@avt.pl.



Rysunek 1. Schemat ideowy urządzenia ClockRDS.

Ustawienia Fusebitów (ważniejszych):
CKSEL3...0: 0010
SUT1...0: 10
CKDIV8: 0
CKOUT: 1
DWEN: 1



W tym momencie nie pozostało nic innego, jak wznowić poszukiwania. Szybko okazało się, że w praktyce wykorzystuje się dwa inne media transmisyjne dysponujące dokładnym wzorcem czasu. Mam tu na myśli moduł GSM, czy moduł nawigacyjny GPS. Rozwiązania te nie są niestety pozbawione wad, które dyskwalifikują je w tak prostym zastosowaniu, jak synchronizacja czasu. Pierwszy z nich, czyli moduł GSM, jest rozwiązaniem o całkiem sporym zapotrzebowaniu na energię a poza tym wymaga zastosowania aktywnej karty SIM. Drugi, czyli moduł nawigacyjny GPS, wymaga dość dobrej widoczności satelitów, przez co może być bezużyteczny wewnątrz budynków. Już zupełnie inną sprawą jest fakt, że stosowanie tego typu zaawansowanych rozwiązań w postaci wspomnianych modułów wyłącznie do synchronizacji czasu wydaje się przysłowiowym strzelaniem do muchy z armaty. Nie bez znaczenia jest też koszt implementacji tychże rozwiązań.

Cóż więc począć w tak patowej sytuacji? Tutaj z pomocą przychodzi mi doświadczenie zdobyte w implementacji kilku innych rozwiązań, które w swojej budowie wykorzystywały dość rewolucyjny układ scalony odbiornika FM pod postacią elementu o nazwie Si4703 produkowanego przez firmę Silicon Labs, specjalistę w dziedzinie tego typu rozwiązań. Ale zaraz, zaraz. Co wspólnego ma odbiornik FM z możliwością synchronizacji wzorca czasu? Otóż układ, o którym mowa, dysponuje możliwością odbierania komunikatów systemu RDS, wśród których znajduje się komunikat odpowiedzialny za wzorzec czasu, więc idealnie wpasowuje się w założenia projektowe. Co nie bez znaczenia, aplikacja tegoż peryferium ogranicza się do zaledwie kilku elementów zewnętrznych. Wynika to z faktu, że w budowie układu Si4703 wykorzystano zaawansowany, cyfrowy tor przetwarzania sygnału przy użyciu wbudowanych przetworników ADC oraz DAC oraz procesora DSP zajmującego się dekodowaniem i obróbką sygnałów w.cz. jak i audio. Zresztą wystarczy wspomnieć, że element Si4703 jest przykładem jednego z wielu produktów rodziny Si47xx tejże firmy (dostępnych jest ponad 25 rodzajów układów), w zakresie której mieszczą się zarówno odbiorniki, jak i nadajniki we wszystkich dostępnych pasmach radiowych zróżnicowane pod względem zintegrowanej funkcjonalności.

Muszę jednak zaznaczyć, że nie będę w tym miejscu przytaczał szczegółowych informacji dotyczących implementacji tego arcyciekawego peryferium, gdyż po pierwsze, dostarczana dokumentacja producenta jest bardzo dobrze napisana, po drugie zaś, wszelkie szczegóły konfiguracji i obsługi układu zostały bardzo dokładnie opisane w ramach mojego projektu o nazwie „pocketRadio", który został opublikowany na łamach naszego miesięcznika w numerach 06.2013 i 07.2013. W dalszym opisie przedstawię jedynie nowe, jeszcze nieopisywane funkcjonalności.

Wykaz elementów:

Rezystory: (obudowy SMD 0805)
RR1, R5, R6: 47 kΩ
R2: 158 kΩ 1%
R3: 18,2 kΩ 1%
R4: 1 MΩ
R7, R8: 4,7 kΩ
R9, R10: 20 kΩ

Kondensatory:
C1, C5…C7: tantalowy 10 µF/16 V SMD
C2: tantalowy 1 µF/16 V SMD
C3, C8, C11, C12: ceramiczny 100 nF SMD0805
C4: ceramiczny X7R 1,5 nF SMD0805
C9, C10: tantalowy 4,7 µF/16 V SMD
C13, C14: ceramiczny 22 pF SMD0805
C15: ceramiczny 22 nF SMD0805
C16: ceramiczny 1 nF SMD0805

Półprzewodniki:
U1: TPS61085 TSSOP8
U2: NCP1117DT33G TO252
U3: ATmega88 TQFP32
U4: Si4703 QFN20
D1: SS14
OLED: wyświetlacz OLED Winstar WEO012864KSPP3N00000 (128×64 px)

Inne:
Q1, Q2: rezonator kwarcowy 32768 Hz SMD 3,2×1,5×0,9 mm
L1: dławik mocy 3,3 µH typu DLG-0504-3R3
L2: dławik SMD 56 nH SMD1206
ZIF: złącze typu ZIF (0,5 mm, 31-pin, górny kontakt)

Budowa

Schemat naszego urządzenia pokazano na **rysunku 1**.

Jak widać, zaprojektowano dość prosty system mikroprocesorowy zbudowany z wykorzystaniem popularnego mikrokontrolera firmy Atmel typu ATmega88 taktowanego wewnętrznym, wysokostabilnym oscylatorem o częstotliwości 1 MHz, którego zadaniem jest obsługa układu Si4703 za pomocą wbudowanego w mikrokontroler interfejsu TWI oraz, jak już wspomniano wcześniej, obsługa i dekodowanie wiadomości systemu RDS z wykorzystaniem przerwania zewnętrznego Pin Change Interrupt 0 (pin PCINT1 mikrokontrolera). Dodatkowo, w ramach programu obsługi urządzenia, zaimplementowano software'owy zegar czasu rzeczywistego RTC wykorzystujący do swojego działania układ czasowo-licznikowy Timer2 wbudowany w mikrokontroler pracujący w trybie asynchronicznym i taktowany przy udziale zewnętrznego rezonatora kwarcowego o częstotliwości 32768 Hz. Co więcej, mikrokontroler realizuję również obsługę interfejsu użytkownika złożonego z jednego klawisza funkcyjnego „SET" umożliwiającego manualne ustawienie zegara i obsługę doskonałej jakości graficznego wyświetlacza OLED o przekątnej 2,7", grubości zaledwie 2 mm (!) i organizacji 128×64 piksele wyposażonego w sterownik ekranu SSD1309. Jako że element ten potrzebuje do swojego działania napięcia 12 V, w tym celu zastosowano nowoczesną przetwornicę typu step-up o oznaczeniu TPS61085 produkcji firmy Texas Instruments

Listing 1. Procedura obsługi przerwania odpowiedzialna za odbiór i dekodowanie wiadomości RDS typu 4A

```
//Definicja typu odpowiedzialnego za obsługę zegara RDS
typedef struct
{
        volatile uint8_t FlagCT, WeekDay, Year, Month, Day, Hour, Minute;  //WeekDay: 0: Poniedziałek, 1: Wtorek...
} clockRDSType;

//Deklaracja tablicy przechowującej wartości rejestrów sterujących układu Si4703
uint16_t Registers[16];

//Zmienna strukturalna przechowująca czas i datę RDS
clockRDSType ClockRDS;


ISR(PCINT0_vect)
{
        uint32_t MJD;  //Modified Julian Day
        uint32_t y1, m1, x1, k;  //Składniki pośrednie przy obliczaniu daty z MJD

        //Reagujemy tylko na zbocze opadające przerwania Pin Change Interrupt 0
        if(!(RADIO_IRQ_PIN & (1<<IRQ_NR)))
        {
                //Odczyt bieżących wartości wyłącznie rejestrów 0x0A...0x0F układu Si4703 (rejestry o adresach 0x0C...0x0F to rejestry
                //RDSA...RDSD)
                TWI_Start();
                TWI_WriteByte(SI4703RD_ADDR);
                for(uint8_t idx = 0x0A ; idx<0x10; idx++) Registers[idx] = TWI_ReadInteger((idx !=0x0F)? ACK: NACK);
                TWI_Stop();

                //Jeśli odebrano informację typu CT (Clock Time and Date only)
                if((Registers[RDSB_REG]>>11) == RDS_GROUP_TYPE_4A)
                {
                        ClockRDS.Minute = (Registers[RDSD_REG]>>6) & 0b111111;
                        ClockRDS.Hour = (Registers[RDSD_REG]>>12);
                        ClockRDS.Hour |= ((Registers[RDSC_REG] & 0x01) <<4);
                        //Korygujemy obliczoną wartość godzin o offset czasu lokalnego (RDSD.5=0 -> offset+,RDSD.5=1 -> offset-)
                        //Offset podany jest jako wielokrotność wartości pół godziny
                        if(Registers[RDSD_REG] & 0b100000) ClockRDS.Hour -=((Registers[RDSD_REG] & 0b11111)>>1);
                        else ClockRDS.Hour += ((Registers[RDSD_REG] & 0b11111)>>1);

                        //Wyłuskujemy MJD i przy jej użyciu wyznaczamy całą datę
                        MJD = ((Registers[RDSB_REG] & 0b11) << 15) | (Registers[RDSC_REG] >> 1);

                        y1 = (MJD * 100 - 1507820) / 36525;
                        x1 = (y1 * 36525) / 100;
                        m1 = ((MJD * 10 - 149561 - x1 * 10) * 1000) / 306001;
                        k = (m1 == 14 || m1 == 15);

                        ClockRDS.Day = MJD - 14956 - x1 - ((m1 * 306001)/ 10000);  //1...dayLimit
                        ClockRDS.Month = m1 - 1 - k *12;  //1...12
                        ClockRDS.Year = y1 + k - 100;  //0...99
                        ClockRDS.WeekDay = ((MJD+2) % 7) +1;  //1...7

                        //Jeśli odebrane dane są poprawne to ustawiamy flagę otrzymania nowej wiadomości RDS typu CT
                        if((ClockRDS.Minute<60)&&(ClockRDS.Hour<24)&&(ClockRDS.Day<32) &&
                        (ClockRDS.Month<13)&&(ClockRDS.Year<100)&&(ClockRDS.WeekDay<8)) ClockRDS.FlagCT = 1;
                }
        }
}
```

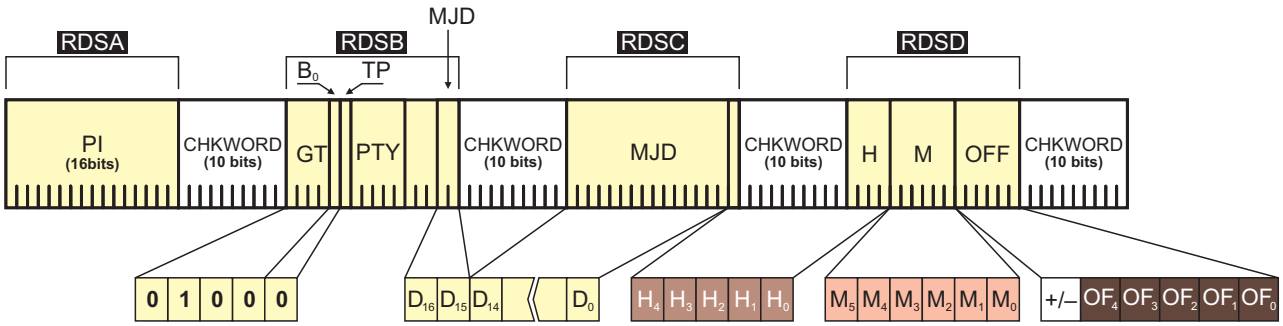
Listing 2. Funkcja odpowiedzialna za inicjalizację i uruchomienie zegara RTC

```
void megaRTCinit(void)
{
        ASSR |= (1<<AS2);  //Timer2 taktowany za pomocą kwarcu 32768Hz podłączonego do wypr. TOSC1/TOSC2
        TCCR2B = (1<<CS22)|(1<<CS20);  //Preskaler = 128, przerwanie przepełnienia co 1s (32768/128/256)
        while(ASSR & ((1<<TCN2UB)|(1<<OCR2AUB)|(1<<OCR2BUB)|(1<<TCR2AUB)|(1<<TCR2BUB)));  //Oczekiwanie na aktualizację rejestrów
        TIFR2 |= ((1<<OCF2B)|(1<<OCF2A)|(1<<TOV2));  //Skasowanie ewentualnych flag przerwań (przez wpisanie jedynek do bitów flag)
        TIMSK2 |= (1<<TOIE2);  //Uruchomienie przerwania od przepełnienia licznika TCNT2
}
```

w swej podstawowej aplikacji pracy. Przetwornica ta odznacza się doskonałymi parametrami elektrycznymi, z których najważniejsze to wysoka sprawność dochodząca do 93%, szeroki zakres napięcia wejściowego (2,3 do 6 V), opcja miękkiego startu, szereg wbudowanych zabezpieczeń (termiczne, od niskiego napięcia wejściowego) oraz duża częstotliwość przełączania 1,2 MHz, dzięki czemu można stosować elementy o mniejszych wymiarach (mniejsza indukcyjność i pojemność). Dodatkowo, układ TPS61085 wyposażono w wejście EN (Enable), za pomocą którego możemy wyłączyć przetwornicę, jeśli zajdzie taka potrzeba (np. wygaszenie wyświetlacza) i tym samym ograniczyć pobór prądu ze źródła napięcia zasilającego. Warto również podkreślić, że nie bez powodu wybrano ten rodzaj wyświetlacza, wszak jego właściwości użytkowe w porównaniu z typowym elementem LCD są nie do przecenienia. Co więcej, cena tego modułu zbliżona jest do ceny analogicznego elementu wykonanego w technologii LCD, co nie pozostawia nam właściwie wyboru. W związku z tym, że wyświetlacz z tego rodzaju sterownikiem znalazł zastosowanie w moim wcześniejszym projekcie o nazwie „Bezprzewodowy system pomiaru temperatury Tmaster/Tslave", opublikowanym na łamach naszego miesięcznika w numerach 08.2018 i 09.2018, z góry uprzedzam, że nie będę powielał w tym miejscu szczegółów implementacyjnych związanych z obsługą tegoż peryferium.

Rozwiązania programowe

Przejdźmy zatem do tych rozwiązań programowych, które nie znalazły zastosowania w poprzednich moich projektach, o których wspomniałem powyżej. Zgodnie z tym, co napisałem wcześniej, układ Si4703 dostarcza kompletne informacje systemu RDS, w ramach których przesyłany jest również wzorzec czasu (jeśli dana rozgłośnia radiowa nadaje taki komunikat). Warto przypomnieć, że dane w systemie RDS zorganizowane są w grupy składające się z czterech bloków danych (każdy po 26 bitów informacji). Każdy blok zawiera 16-bitowe słowo przenoszące właściwą informację oraz 10 bitów danych korekcyjnych przeznaczonych zarówno do sprzętowej korekcji błędów, jak i synchronizacji bloków oraz grup danych. Dwa pierwsze bloki (Blok1…Blok2) zawierają podstawowe struktury danych charakterystyczne dla konkretnej stacji radiowej oraz bieżącej informacji, która jest przez nią transmitowana, i identyfikator grupy danych (bity A3…A0 oraz B0), który determinuje zawartość pozostałych bloków danych (Blok3…Blok4). Wspomniany identyfikator przewiduje 32 typy grup wiadomości (a dokładnie 16 typów, każdy w wersji A oraz B), które determinują znaczenie informacji zawartej w blokach danych Blok2 (ostatnie 5 bitów bloku) oraz Blok3…Blok4.



Rysunek 2. Struktura danych dla grup wiadomości typu 4A

My zajmiemy się tutaj wyłącznie typem danych oznaczanym symbolem 4A, czyli wzorcem czasu, którego strukturę pokazano na **rysunku 2** (zaznaczono odpowiednie rejestry układu Si4703 tj. RDSA…RDSD).

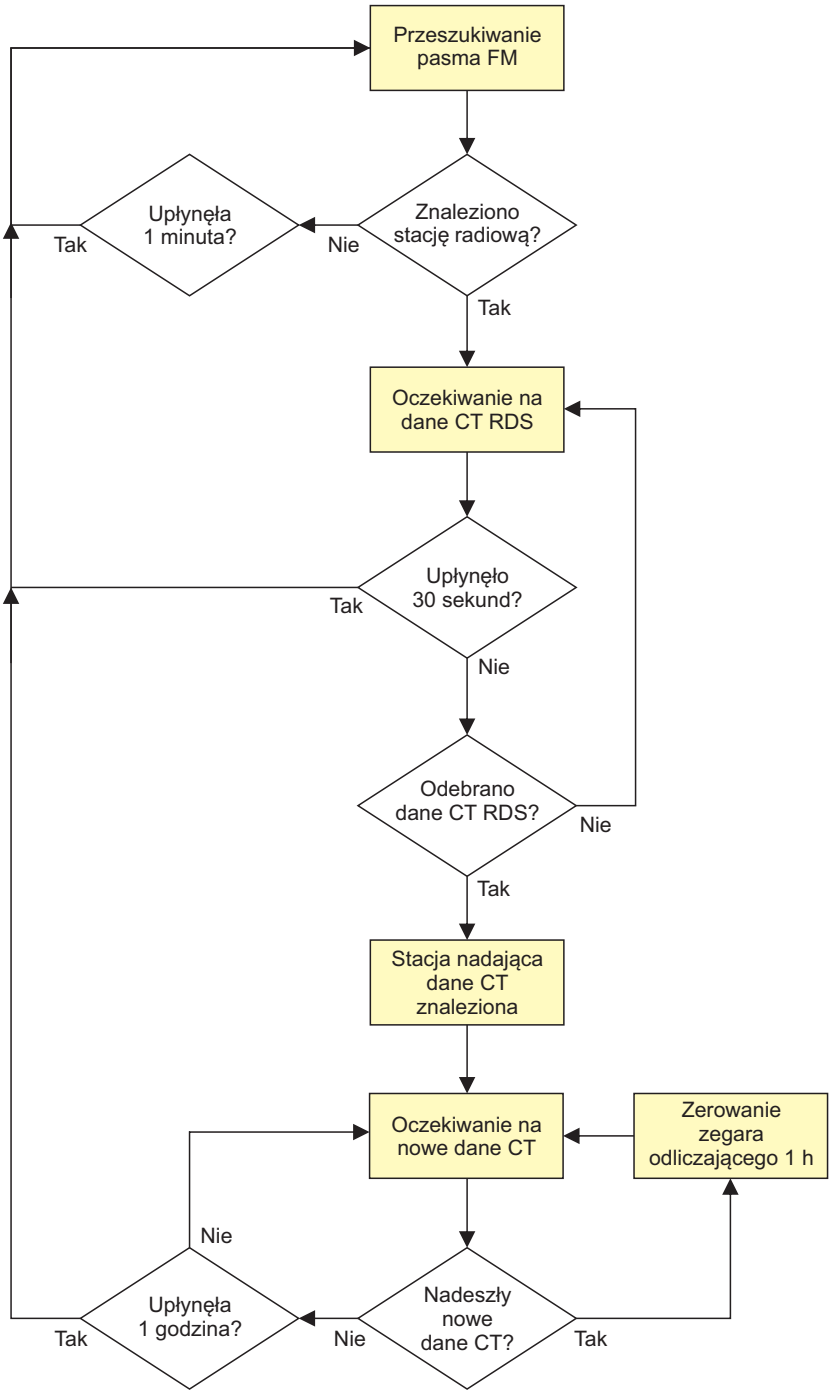
Przeznaczeniem tego typu grupy danych jest przesyłanie informacji o czasie, która może być wykorzystana przez radioodbiornik do synchronizacji wbudowanego zegara czasu rzeczywistego, co wykorzystywane jest w niniejszym projekcie. Struktura tego typu danych zawiera następujące elementy:

- MJD jest 17-bitowym słowem reprezentującym datę zapisaną w konwencji MJD (Modified Julian Date),
- H jest wartością godzin czasu UTC w zapisie 24-godzinnym,
- M jest wartością minut czasu UTC,
- OFF jest offsetem godzin czasu lokalnego w stosunku do wysłanego znacznika czasu podanym jako wielokrotność 30 minut. Znacznik ± określa znak offsetu (0 oznacza offset dodatni a 1 oznacza offset ujemny).

Nie będę w tym miejscu wdawał się w szczegóły implementacyjne systemu RDS, gdyż, jak napisałem, zostały poruszone w ramach innych projektów, lecz przedstawię funkcję obsługi przerwania *Pin Change Interrupt 0* odpowiedzialną za odebranie i rozkodowanie wiadomości przenoszącej wartość wzorca czasu. Ciało tejże funkcji (wraz z deklaracją nowych typów danych) pokazano na **listingu 1**.

Wiemy już, jak zdekodować dane dotyczące wzorca czasu przenoszone w ramach wiadomości systemu RDS, jednak nasz zegar działa niezależnie od tego mechanizmu, gdyż korzysta z implementacji zegara czasu rzeczywistego zrealizowanego z wykorzystaniem układu czasowo-licznikowego Timer2 wbudowanego w mikrokontroler a „zasilanego" sygnałem oscylatora 32768 Hz. Pora na przedstawienie kilku kluczowych funkcji obsługujących tenże mechanizm. Pierwsza z nich, pokazana na **listingu 2**, to funkcja odpowiedzialna za inicjalizację Timera2.

Kolejna z funkcji, niezbędna przy realizacji zaawansowanego zegara RTC, to funkcja, która na podstawie argumentów wywołania (rok i miesiąc) ustala liczbę dni miesiąca, uwzględniając fakt, czy dany rok jest rokiem przestępnym, czy też nie. Ciało tejże funkcji pokazano na **listingu 3**. Funkcja *getMonthDaysLimit()* korzysta ze zmiennej *monthsDays[]* umieszczonej w pamięci programu, której definicja przedstawia się następująco:



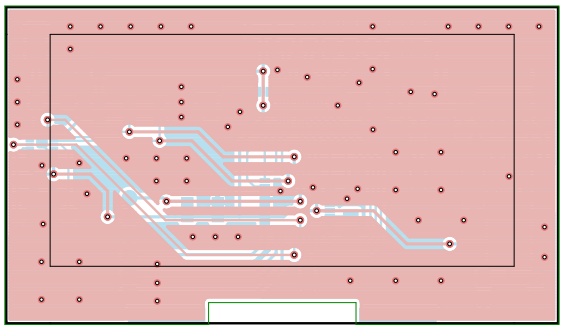
Rysunek 3. Szczegółowy diagram obrazujący algorytm wyszukiwania danych wzorca czasu

Listing 3. Funkcja odpowiedzialna za ustalenie liczby dni miesiąca

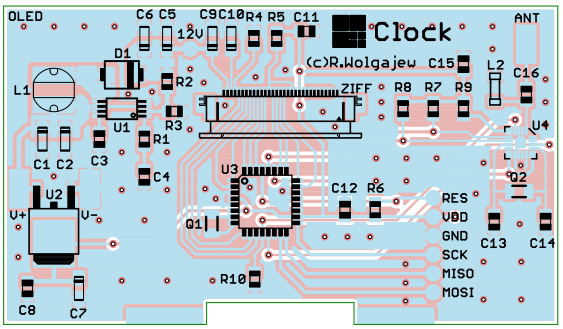
```
inline uint8_t getMonthDaysLimit(uint8_t Year, uint8_t Month)  //Year: 0...99, Month: 1..12
{
          //Ustalamy maksymalną liczbę dni dla miesiąca będącego argumentem funkcji
           register uint8_t dayLimit = pgm_read_byte(&monthsDays[Month-1]);
          //Dla Lutego wyznaczamy liczbę dni w zależności od tego czy rok jest przestępny czy też nie
          if(Month == 2)
          {
                    if(((Year+2000)%4 == 0 && (Year+2000)%100 != 0) || (Year+2000)%400 == 0) dayLimit = 29; else dayLimit = 28;
                    //Rok przestępny lub nie
          }
          return dayLimit;
}
```

Listing 4. Funkcja odpowiedzialna za realizację programowego zegara czasu rzeczywistego (RTC)

```
//Przerwanie przepełnienia licznika wywoływane 1 raz na sekundę (Timer2 taktowany kwarcem zegarkowym 32768Hz)
//Hours: 0...23, Minutes: 0...59, Seconds: 0...59, Year: 0...99, Month: 1...12, Day: 1...dayLimit, Weekday: 1...7

ISR(TIMER2_OVF_vect)
{
          uint8_t Flag = ClockRTC.Flag|CLOCK_RTC_SECOND;  //Optymalizacja volatile

          if(++ClockRTC.Second == 60)
          {
                    ClockRTC.Second = 0; Flag |= CLOCK_RTC_MINUTE;
                    timer1mIncr++;  //Obsługa timera niezwiązanego z RTC

                    if(++ClockRTC.Minute == 60)
                    {
                              ClockRTC.Minute = 0; Flag |= CLOCK_RTC_HOUR;
                              if(++ClockRTC.Hour == 24)
                              {
                                        ClockRTC.Hour = 0; Flag |= CLOCK_RTC_DAY|CLOCK_RTC_WEEKDAY;
                                        if(++ClockRTC.WeekDay == 8) ClockRTC.WeekDay = 1;  //Dzień tygodnia

                                        //Teraz ustalimy maksymalną liczbę dni dla bieżącego miesiąca
                                        if(++ClockRTC.Day > getMonthDaysLimit(ClockRTC.Year, ClockRTC.Month))  //To był ostatni dzień
                                        //bieżącego miesiąca
                                        {
                                                  ClockRTC.Day = 1; Flag |= CLOCK_RTC_MONTH;
                                                  if(++ClockRTC.Month == 13)
                                                  {
                                                            ClockRTC.Month = 1; Flag |= CLOCK_RTC_YEAR;
                                                            if(++ClockRTC.Year == 100) ClockRTC.Year = 0;
                                                  }
                                        }
                              }
                    }
          }

          ClockRTC.Flag = Flag;  //Optymalizacja volatile
}
```



Rysunek 4. Schemat obwodu drukowanego, strona TOP (skala 1:1)



Rysunek 5. Schemat obwodu drukowanego, strona BOTTOM (skala 1:1)

const uint8_t monthsDays[12] PROGMEM = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

I na koniec funkcja przerwania od przepełnienia zawartości licznika Timer2, która wywoływana co 1 s realizuje całą, założoną funkcjonalność programowego zegara RTC. Ciało tejże funkcji pokazano na **listingu 4**.

Schemat działania

Pora przyjrzeć się schematowi działania zegara, a jest on dość prosty. Tuż po włączeniu zasilania i inicjalizacji wszystkich peryferiów uruchamiany jest zegar RTC odmierzający czas. Następnie przeszukiwane jest pasmo radiowe w poszukiwaniu aktywnej stacji FM. Niepowodzenie tego procesu skutkuje zawieszeniem dalszego przeszukiwania pasma radiowego na czas 1 minuty. Powodzenie powoduje z kolei oczekiwanie urządzenia przez czas 30 sekund na dane systemu RDS grupy 4A (Clock & Time). Jeśli w odmierzanych 30 sekundach zostaną odebrane ważne dane RDS grupy 4A zegar nasz synchronizowany jest wzorcem czasu systemu RDS po czym przechodzi w tryb odmierzania czasu 1 godziny, podczas którego musi nastąpić kolejne odebranie danych wzorca czasu. Jeśli wspomniany czas 30 sekund upłynie i nie zostaną odebrane ważne dane RDS grupy 4A algorytm przechodzi w tryb wyszukiwania nowej stacji radiowej. Tak samo stanie się w przypadku, gdy od ostatniego odbioru danych wzorca czasu upłynie więcej niż 1 godzina. Szczegółowy diagram obrazujący algorytm wyszukiwania danych wzorca czasu naszego zegara pokazano na **rysunku 3**.

Obsługa urządzenia

Kilka słów uwagi należy się również manualnej obsłudze zegara za pomocą opcjonalnego przycisku SET (normalnie niemontowany na obwodzie drukowanym). Długie przyciśnięcie tegoż przycisku powoduje wejście w tryb edycji czasu i daty, co sygnalizowane jest podświetleniem wybranego pola poddawanego właśnie edycji. Dalej, w trybie edycji, każdorazowe długie przyciśnięcie przycisku powoduje przejście do kolejnej pozycji, zaś krótkie przyciśnięcie regulację wybranej wartości (w górę, aż do przepełnienia). Co ważne, w czasie manualnej edycji czasu wstrzymane jest odbieranie danych RDS by nie zakłócać bieżącego procesu edycji. Na **fotografii 1** pokazano graficzny interfejs użytkownika urządzenia ClockRDS. Jak widać, oprócz standardowych danych dotyczących

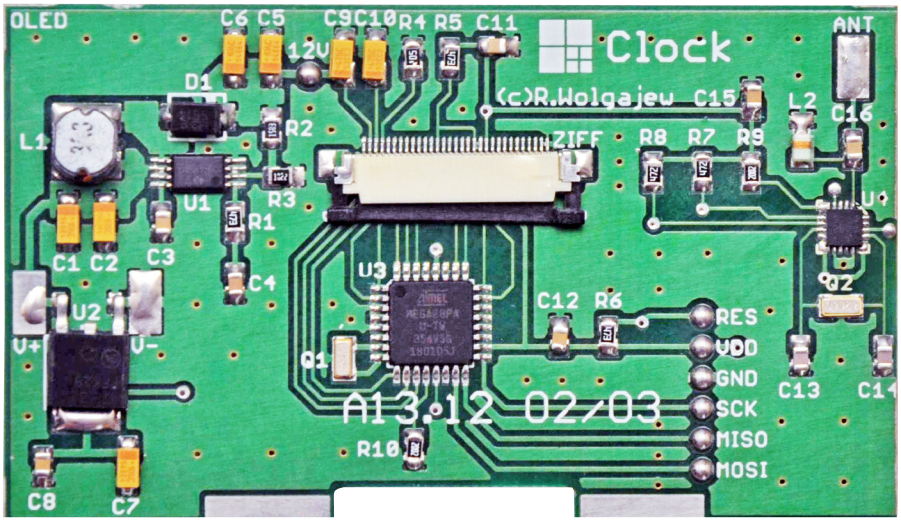
kompletnej daty i godziny pokazywana jest również siła sygnału odbieranej stacji FM (w formie bargrafu) oraz wskaźnik odebrania wiadomości RDS zawierającej wzorzec czasu (ikonka zegarka pokazywana przez 15 s po każdym poprawnym odbiorze wiadomości RDS zawierającej wzorzec czasu).

Montaż

Zaprojektowano bardzo zwarty obwód drukowany (wielkości zastosowanego wyświetlacza OLED) zbudowany wyłącznie z elementów SMD umieszczonych po stronie BOTTOM. Schemat płytki pokazany jest na **rysunkach 4 i 5**.

Z uwagi na fakt, iż moduł wyświetlacza OLED podłączony jest do płytki naszego urządzenia z użyciem gniazda ZIF o bardzo gęstym rastrze (0,5 mm), montaż rozpoczynamy właśnie od przylutowania tegoż gniazda. Najprostszym sposobem montażu elementów o tak dużym zagęszczeniu wyprowadzeń, niewymagającym jednocześnie posiadania specjalistycznego sprzętu, jest użycie typowej stacji lutowniczej, dobrej jakości cyny z odpowiednią ilością topnika oraz dość cienkiej plecionki rozlutowniczej, która umożliwi usunięcie nadmiaru cyny spomiędzy wyprowadzeń złącza. Należy przy tym uważać, by nie uszkodzić termicznie tegoż elementu. Jakość tak wykonanego połączenia sprawdzamy pod lupą, korzystając z najprostszego miernika pozwalającego sprawdzić ciągłość połączeń. Wspomniana kontrola będzie znacznie łatwiejsza, jeśli zmontowaną płytkę przemyjemy alkoholem izopropylowym w celu wypłukania nadmiaru kalafonii lutowniczej.

Niestety dalej zaczynają się przysłowiowe schody, gdyż czas przylutować nasze zgrabne radyjko FM pod postacią układu scalonego Si4703. W tym przypadku nie obędziemy się bez stacji lutowniczej na gorące powietrze (tzw. Hot Air), gdyż niewielka obudowa tego układu o wymiarach 3×3 mm integruje w sobie 20 wyprowadzeń umieszczonych pod spodem i na obrysie obudowy. Po wykonaniu tego delikatnego zadania, które skądinąd wymaga sporego doświadczenia, przechodzimy do przylutowania mikrokontrolera i przetwornicy napięcia TPS61085. Następnie lutujemy pozostałe półprzewodniki a na samym końcu elementy bierne.



Fotografia 2. Zmontowany obwód drukowany, widziany od strony BOTTOM.

Jako antenę przylutowujemy kawałek miedzianego przewodu do pola lutowniczego oznaczonego symbolem ANT. Opcjonalny przycisk SET przylutowujemy pomiędzy wyprowadzenia pola lutowniczego MOSI oraz masę zasilania. Na samym końcu podłączamy wyświetlacz OLED do złącza ZIFF po stronie BOTTOM, zaś sam element przyklejamy po stronie TOP (w miejscu wyznaczonym obrysem), korzystając z dwustronnej taśmy klejącej. Źródło napięcia zasilania (o napięciu z zakresu od 4,5 do 6 V) podłączamy kawałkiem przewodu miedzianego korzystając z pól lutowniczych oznaczonych V+ i V–.

Poprawnie zmontowany układ powinien działać tuż po włączeniu zasilania. Dla porządku sprawdzić można napięcie przetwornicy step-up (punkt lutowniczy 12 V), które powinno oscylować około wartości 12 V. Warto również zauważyć, iż obwód drukowany naszego urządzenia wyposażony został w dwa duże pola lutownicze umieszczone przy jego dolnej krawędzi i niepokryte maską lutowniczą, do których możemy przylutować niewielki kawałek laminatu pokrytego miedzią (pod kątem około 75 stopni), tworząc w ten sposób podstawę, na której postawić możemy nasze urządzenie. Zmontowany obwód drukowany urządzenia ClockRDS widziany od strony BOTTOM pokazano na **fotografii 2.**

Robert Wołgajew, EP