# Mikroprocesorowy włącznik czasowy


Proponowany układ powstał w odpowiedzi na rzeczywistą potrzebę, z jaką się spotkałem. Do pomieszczenia, z którego korzysta się tylko w czasie dnia, został kupiony grzejnik elektryczny. Grzejnik taki powinien być włączony w ciągu dnia, a przez wzgląd na oszczędność energii, wyłączony w nocy. Niedopuszczalne było ręczne dokonywanie włączenia grzejnika ze względu na panujące po całej nocy zimno. Pomieszczenie powinno zostać nagrzane, zanim ktoś się obudzi. Kolejnym założeniem była możliwość ręcznego włączania i wyłączania piecyka. Może ktoś powie, że na rynku są już dostępne odpowiednie rozwiązania. Owszem, ale do rzadkości należą układy umożliwiające ręczną kontrolę nad urządzeniem, zwykle też obsługa takich urządzeń pozostawia wiele do życzenia, zwłaszcza w rozwiązaniach z mechanicznym zegarem. Przedstawiony problem dał też świetną możliwość do przetestowania darmowego kompilatora C dla procesorów AVR – AVR-GCC.

Układ realizujący przedstawione zadania został wykonany i przetestowany przez okres całej zimy. Muszę się jednak przyznać, że nie pracował przez cały ten czas bez awarii. Procesor zawiesił się raz – gdy pracował cały dzień przy obniżonym napięciu (podtrzymaniu bateryjnym) i był niesiony na 10-stopniowym mrozie. Była to jednak sytuacja ekstremalna, która nie powinna zdarzyć się w czasie normalnej eksploatacji. Ponadto, gdy układ został już „włożony do szafy", program został troszkę zoptymalizowany, co umożliwiło dodanie obsługi watchdoga. Powinno to całkowicie wyeliminować jakiekolwiek problemy z niestabilnością programu.

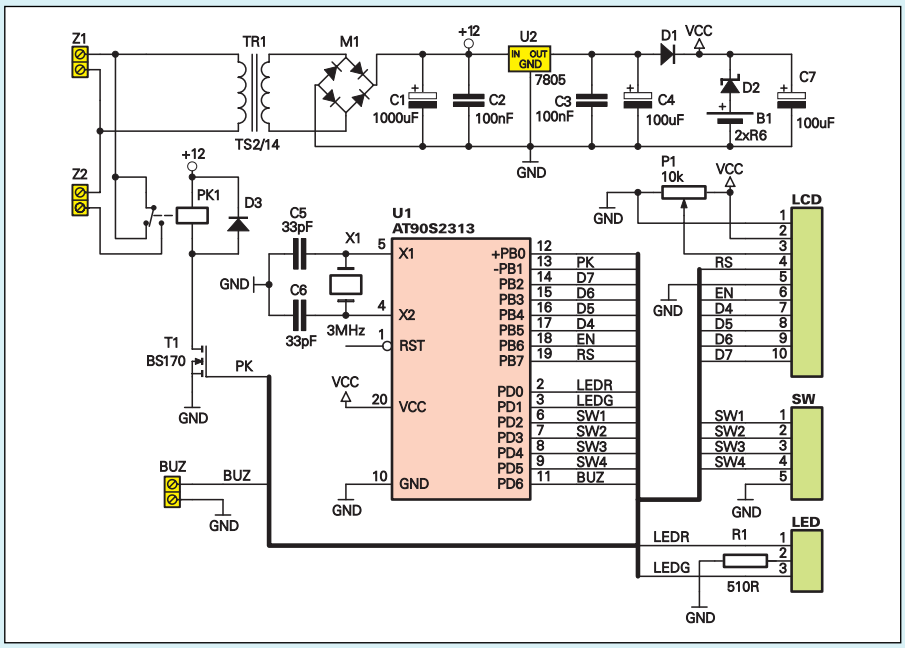
## Opis układu

Schemat ideowy układu przedstawia **rysunek 1.** W obwodzie zasilacza został wykorzystany stabilizator scalony 7805. Znajduje się tutaj także obwód zasilania rezerwowego wykorzystujący dwa popularne „paluszki". Dioda D1 zapobiega przepływowi prądu przez stabilizator w czasie, gdy brak napięcia sieci. Ze względu na znikomy pobór prądu przez układ nie jest konieczne stosowanie w tym miejscu diody prostowniczej – może to być zwykła dioda krzemowa. Dioda D2 z kolei zapobiega ładowaniu dużym prądem ogniw w czasie normalnej pracy układu. Powinna być to dioda o minimalnym spadku napięcia. Producent gwarantuje poprawną pracę układu U1 dla minimalnego napięcia 2,7V. Stosując dwie baterie R6, jesteśmy już niebezpiecznie blisko tej granicy. W modelu zastosowano diodę Schottky'ego o spadku napięcia na poziomie 200mV. Należy nadmienić tutaj, że zasilanie rezerwowe ma za zadanie zapobiec utracie nastaw przez układ w momencie, gdy układ jest przenoszony lub chwilowo brak prądu. **Jeśli planowany jest okres dłuższego wyłączenia sterownika, baterie należy na ten czas usunąć.**

W założeniu obciążenie będzie duże, ale rzadko przełączane. W związku z tym jako element wykonawczy został wybrany przekaźnik. Tranzystor T1, zabezpieczony przed przepięciami diodą D3, steruje bezpośrednio jego cewką. Złącze BUZ pierwotnie było przeznaczone do podłączenia „piszczka". Jednak w trakcie pracy nad modelem z jego zastosowania zrezygnowano. Złącze pozostało do ewentualnej przyszłej rozbudowy. Do złącza LCD podłączymy wyświetlacz alfanumeryczny pracujący z interfejsem 4-bitowym. Złącze SW służy do podpięcia przycisków sterujących, a LED – diody dwukolorowej wskazującej aktualny stan sterownika.

W centrum układu znajduje się, dobrze znany stałym Czytelnikom, mikrokontroler AT90S2313. Jego „serduszko" bije 3 miliony razy na sekundę za sprawą wewnętrznego układu generatora kwarcowego stabilizowanego przez elementy X1, C5 i C6.

**Rys. 1 Schemat ideowy**



Wiemy już, jakie jest zadanie poszczególnych elementów sprzętowych. Czas zobaczyć, o czym „myśli" procesor w trakcie swojej pracy.

Oprogramowanie zostało napisane w języku C, sam proces pisania i uruchamiania był wspomagany przez darmowe środowisko programistyczne udostępnione przez firmę ATMEL, jakim jest AVRStudio. Kompilatorem wykonawczym był AVR-GCC w wersji 20030115 (rok, miesiąc, dzień aktualizacji). Osoby zainteresowane tymi narzędziami odsyłam do numerów archiwalnych *Elektroniki Praktycznej*.

Kod źródłowy został podzielony na cztery pliki:

*lcd.h i lcd.c* – zawierają deklaracje i definicje procedur obsługi wyświetlacza alfanumerycznego z wbudowanym sterownikiem kompatybilnym z HD44780; obsługa odbywa się przez interfejs 4-bitowy.

*settings.h* – znajdują się tutaj przyporządkowania funkcji poszczególnym wyprowadzeniom mikroprocesora.

*timer.c* – jest to główny plik programu.

Dwa pierwsze pliki są o tyle ciekawe, że zawarte w nich procedury można wykorzystać we własnych programach. Zostały one napisane w taki sposób, że bez zmian możliwe jest dowolne podłączenie wyprowadzeń wyświetlacza do procesora, o ile wszystkie one należą do jednego portu. Dostosowanie plików do własnych wymagań wymaga tylko odpowiedniego zdefiniowania kilku stałych opisanych w nagłówku pliku źródłowego.

Procedury, na które chciałbym zwrócić uwagę, są widoczne na **listingu 1.** Pierwsza z nich wyświetla w kodzie dziesiętnym liczbę podaną jako argument w kodzie binarnym. Ponieważ w tym programie liczba wejściowa z założenia nigdy nie przekroczy 59, zrezygnowano ze sprawdzania cyfry setek. Do zmiennej *cLitera* jest na początku wpisywana całkowita część wyniku z podzielenia liczby wejściowej przez 10. Oznacza to w praktyce, że otrzymujemy cyfrę dziesiątek. Cyfry w kodzie ASCI zaczynają się od kodu 30hex. Aby uzyskać z cyfry jej kod ASCI, należy dodać do niej stałą wartość przesunięcia równą 30hex. Dzięki sprawdzeniu, czy pierwsza cyfra nie jest równa zero, uzyskano wygaszenie zera nieznaczącego.

Druga procedura wypisuje na wyświetlaczu łańcuch podany jako argument. Wysyła ona odpowiednie kody do wyświetlacza, aż do napotkania zera, które kończy łańcuch znaków w języku C.

Resztę procedur z pliku *lcd.c* można by nazwać „procedurami niskiego poziomu". Ich kod jest niejako bezpośrednim przełożeniem na C lub asembler opisu obsługi interfejsu sterownika wyświetlacza alfanumerycznego, który pojawił się w archiwalnych numerach EdW. Nie będę ich tutaj omawiał, a zainteresowanych odsyłam do kodów źródłowych.

Wszystkie zmienne globalne upakowane zostały do jednej struktury zdefiniowanej jako nowy typ zmiennej *GLOBAL_VAR*. Następnie została zdeklarowana zmienna *g_var,* która zawiera wszystkie niezbędne pola. Dzięki takiemu rozwiązaniu zyskujemy możliwość wypełnienia wszystkich zmiennych stałą wartością za pomocą jednej instrukcji – *memset*. Jest to o tyle ważne, że dzięki nadaniu zmiennej *g_var* atrybutu *section* („*noinit*„) jest ona umieszczona w takim miejscu pamięci, która nie jest czyszczona po resecie układu. Przez to, gdy nastąpi reset wywołany przez *watchdog'a,* dane nie są tracone i praca może zostać natychmiast wznowiona.

Niestety w dokumentacji mikrokontrolera AT90S2313 nie udało mi się znaleźć informacji o tym, żeby zawierał on jakiś mechanizm umożliwiający wykrycie z jakiego źródła nadeszło zerowanie. Aby stworzyć jakąś namiastkę takiego mechanizmu, do struktury zmiennych globalnych zostało dodane pole *bWdt_55*. Jeśli zmienne są zainicjowane, powinno ono zawierać wartość 55hex. Została ona dobrana tak, aby sąsiadujące bity różniły się stanami. Teoretycznie jest bardzo mało prawdopodobne, aby taki stan mógł utrzymać się po zaniku zasilania. Dodatkowo sprawdzana jest poprawność pól zawierających aktualny czas. Przyjrzyjmy się **listingowi 2.** Pokazuje on fragment funkcji *main* odpowiedzialny za inicjacje zmiennych. Widać tutaj, że jeżeli struktura zmiennych globalnych przejdzie przez wszystkie wymienione testy, to jedyne, co zostanie ustawione, to pole *bOpcje.* Zostanie ono ustawione w taki sposób, żeby wyświetlany był bieżący czas oraz aby program był wykonywany. Jeśli przed przymusowym zerowaniem program był wstrzymany, tylko to zostanie zmienione. Jest to pewne uproszczenie, tym bardziej dopuszczalne, że *watchdog* w tym programie zadziała najwyżej w ekstremalnych warunkach, o których była mowa na początku artykułu.

Timer WDT został ustawiony na 60ms (dla 5V, do 180ms przy 3V). Tak krótki czas został podyktowany chęcią zminimalizowania błędu zegara w przypadku zawieszenia programu. Przy testowaniu najniższy czas, jaki umożliwiał prawidłowe działanie układu, wynosił 30ms. Jednak prawie dokładnie tyle trwa procedura inicjacji wyświetlacza. Wystąpiła obawa, czy w innym egzemplarzu mikrokontrolera nie okaże się, że nie jest on zdolny do pracy.

Program sterujący elementem wykonawczym umożliwia pracę z rozdzielczością do 15 minut. Całkowicie wystarczy to do potrzeb jakie ma zaspokoić omawiany układ. Odpowiednia sekwencja jest zapisana w 24-elementowej tablicy *abProgram.* Dla każdej godziny przeznaczony jest jeden bajt (element 0 – godzina pierwsza). Wykorzystane są tylko cztery najmłodsze bity każdego elementu tablicy. Bit najmłodszy zawiera stan wyjścia odpowiadającego pierwszemu, a czwarty – ostatniemu kwadransowi każdej godziny. Starsze połówki bajtów są w takim rozwiązaniu marnowane, jednak znakomicie upraszcza to część obliczeniową. Najlepiej ideę tę wyjaśni **listing 3** pokazujący wywoływaną co sekundę procedurę sterującą przekaźnikiem.

Po zapoznaniu się z tymi ciekawszymi elementami samodzielna analiza reszty programu nie sprawi już trudności. Warto tylko pamiętać, że nasz *„zły pies"* czuwa. W związku z tym w każdej występującej pętli, która może trwać dłużej niż 60ms, pojawia się komenda *„siad!".*

***Listing 1***

```
void WykonujProgram(void)
{
      unsigned char bDana;
      bDana = g_var.abProgram[g_var.bHour-1];
      bDana >>= (g_var.bMin / 15);
      if(bDana & 1)
            PK_PORT |= 1<<PK_PIN;
      else
            PK_PORT &= ~(1<<PK_PIN);
}
```

***Listing 2***

```
int main(void)
{
     // Start wdt
     wdt_reset();
     wdt_enable(WDTO_60MS);
     // Inicjacja zmiennych jeśli dane są błędne
     if(g_var.bWdt_55 != 0x55 || g_var.bHour > 24 ||
g_var.bHour == 0 ||
           g_var.bMin > 59 || g_var.bSec > 59)
     {
           memset((void*)&g_var, (int)0, sizeof(g_var));
           g_var.bHour = 12;
           g_var.bWdt_55 = 0x55;
     }
     g_var.bOpcje = OPCJA_WYSWIETLAJ_CZAS |
           OPCJA_WYKONUJ_PROGRAM;
[...]
```

## Montaż i uruchomienie

Zaprezentowana na **rysunku 2** płytka drukowana została zaprojektowany pod kątem umieszczenia jej w obudowie KM35 lub też – odpowiedniku innej firmy – Z5. Jeśli wyposażymy się w odpowiedni wyświetlacz LCD, jego płytka powinna idealnie pokrywać się z płytą czołową obudowy. Nie będzie także wymagane jego przykręcanie – zostanie utrzymany na miejscu przez kołki normalnie

przeznaczone do umocowania w obudowie płytki drukowanej. Przed przystąpieniem do montażu warto jest włożyć do obudowy płytkę drukowaną i używając jej jako wzorca nawiercić otwory pod cztery śruby. Uwaga: otwory w płytce nie pokrywają się z kołkami przeznaczonymi do jej zamocowania. Umieszczenie śrub „normalnie" kolidowałoby z wyświetlaczem oraz pojemnikiem na baterie.

Montaż przeprowadzamy w typowy sposób: zaczynając od elementów o najmniejszych gabarytach, kończąc na transformatorze i przekaźniku. Jako Z1 i Z2 zalecam zastosowanie złącz śrubowych typu ARK. Wszelkie połączenia do przycisków, wyświetlacza i diody LED warto wykonać tak, aby było możliwe ich łatwe rozłączenie. Pomocne w tym celu mogą być rzędy goldpinów i odpowiednie do nich złącza. Takie wykonanie uprości umieszczenie całości w obudowie.

Wykonaną płytkę wkładamy do obudowy i przykręcamy ją przygotowanymi śrubami M3. Od spodu umieszczamy nóżki, w których ukryją się łebki trzech z nich. Niestety problem pojawi się w okolicach pojemnika na baterie. Aby mogły się one zmieścić, konieczne jest umieszczenie śrubki łebkiem do góry. W takim przypadku w nóżce znajdzie się nakrętka. Wymaga to dokładnego przycięcia śruby.

W płycie czołowej wykonujemy otwór pod wyświetlacz w taki sposób, aby zmieściła się w nim cała szklana płytka razem z metalową oprawą.

Przyciski oraz diodę mocujemy na wierzchu obudowy. Diodę można umieścić w zasadzie w dowolnym miejscu. Uważać należy przy umieszczaniu przycisków, ponieważ mogą one blokować się o elementy układu. Zalecam umieszczenie ich środków w odległości 25mm od czoła obudowy, każdy przycisk w odległości 20mm od sąsiadujących. Przyporządkowanie przycisków do złącza jest następujące: 1-Plus, 2-Zegar, 3-Program, 4-Minus, 5-Wspólny.

W celu uproszczenia obsługi oraz zwiększenia estetyki układu można zastosować płytę czołową oraz opis do przycisków (**rysunki 3** i **4** ).

Po złożeniu i włączeniu zasilania układ jest w zasadzie gotów do pracy. Wymaga jeszcze tylko ustawienia czasu i zaprogramowania.

## Obsługa

Bezpośrednio po włączeniu układ znajduje się on w trybie wyświetlania czasu. Jest to podstawowy tryb pracy i sterownik wraca do niego automatycznie, zawsze gdy przez około 10 sekund nie zostanie naciśnięty żaden przycisk. Przyciski sterujące przyjmują funkcje, które zostały opisane w **tabeli 1**. Na początku powinniśmy zająć się ustawieniem zegara. W tryb ustawiania czasu wchodzimy przez naciśnięcie przycisku „Zegar". Czas zmieniamy przyciskami „+" i „-". Dłuższe przytrzymanie któregoś z nich powoduje szybką zmianę ustawień. Do normalnego wyświetlania czasu powracamy przez ponowne naciśnięcie przycisku „Zegar" bądź też odczekanie 10 sekund. Licznik sekund jest zerowany podczas trwania nastaw zegara. Jeśli chcemy zsynchronizować dokładnie sekundnik z innym zegarem, aby zapobiec automatycznemu powrotowi do wyświetlania czasu należy nacisnąć przycisk „Zegar" i puścić go w momencie wystąpienia synchronizacji.

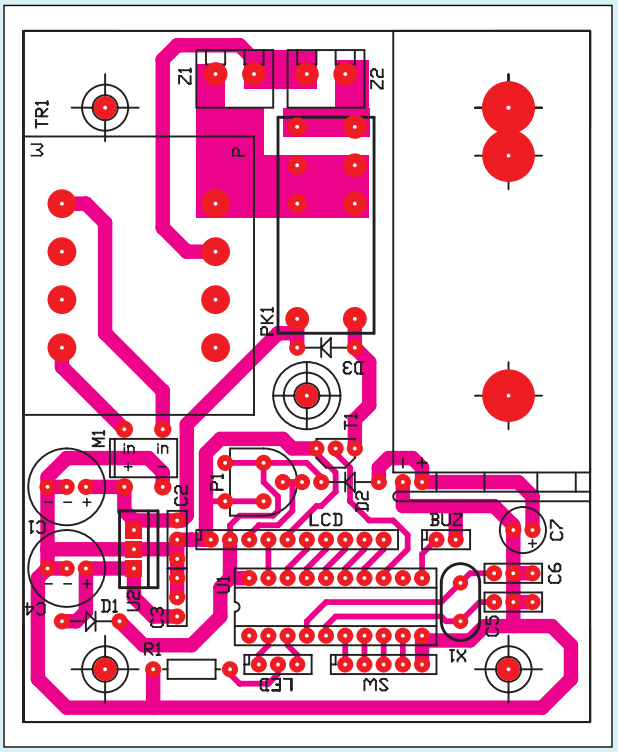
Gdy mamy już ustawione prawidłowo wskazania czasu możemy zabrać się za wpisywanie sekwencji, jaką sterownik będzie się posługiwał przy sterowaniu dołączonym odbiornikiem. Aby rozpocząć wpisywanie programu, należy nacisnąć przycisk „Program".

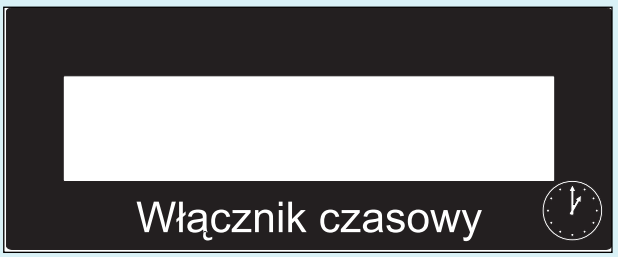
*Ciąg dalszy na stronie 33.*

| Tryb pracy: WYŚWIETLANIE CZASU | |
|---|---|
| + | Włączenie urządzenia wyjściowego i zatrzymanie wykonywania programu. |
| - | Wyłączenie urządzenia wyjściowego i zatrzymanie wykonywania programu. |
| Program | Wejście w tryb programowania. Dłuższe przytrzymanie – rozpoczęcie wykonywania wcześniej zatrzymanego programu. |
| Zegar | Ustawianie zegara. |

**Tab. 1**

**Rys. 2 Schemat montażowy**




**Rys. 3**

**Rys. 4**


*Listing 3*

```
void LcdPrintBYTE(unsigned char uDana)
{
      char cLitera;
      cLitera = uDana / 10;
      if(cLitera != 0)
      {
            cLitera += 0x30;
            LcdPut(cLitera);
      }
      cLitera = (uDana % 10) + 0x30;
      LcdPut(cLitera);
}

void LcdPrint(char* strText)
{
      for(;(*strText)!=0; strText++)
      {
            LcdPut((unsigned char)*strText);
      }
}
```

*Ciąg dalszy ze strony 20.*

Na wyświetlaczu pojawi się obraz jak w **tabeli 2**, która powinna wyjaśnić znaczenie poszczególnych cyfr. Przyjęto, że *1 oznacza wyjście włączone, 0 – wyłączone*. Edytowany kwadrans jest podświetlony kursorem.

| 1: 0000 | | | | |
|---|---|---|---|---|
| Programowana godzina: | Stan wyjścia w pierwszym kwadransie | Stan wyjścia w drugim kwadransie | Stan wyjścia w trzecim kwadransie | Stan wyjścia w czwartym kwadransie |

Tab. 2

Zmiany stanu wyjścia w tym czasie dokonujemy przyciskiem „Zegar”. Przyciski „+” i „-” służą do nawigacji. Zakończenie programowania jest analogiczne do wyjścia z trybu ustawiania czasu: można albo nacisnąć przycisk „Program”, albo odczekać 10 sekund. W trakcie programowania działanie programu zostaje wstrzymane. Po jego zakończeniu będzie automatycznie wznowione.

Tab. 3

| | |
|---|---|
| Czerwona | Urządzenie włączone, program nie wykonywany. |
| Zielona | Urządzenie wyłączone, program nie wykonywany. |
| Migająca | Program jest wykonywany. |

Niezależnie od trybu, w jakim znajduje się układ, dwukolorowa dioda informuje nas o stanie wykonywania ustawionego programu. Znaczenie jej wskazań tłumaczy **tabelka 3.**

**Radosław Koppel**

## Wykaz elementów

**Rezystory**
R1 ..... 510Ω
P1 ..... potencjometr montażowy 10kΩ

**Kondensatory**
C1 ..... 1000µF/25V
C2,C3 ..... 100nF ceramiczny
C4,C7 ..... 100µF/16V
C5,C6 ..... 33pF

**Półprzewodniki**
D1,D3 ..... diody krzemowe
D2 ..... dioda Schottky'ego
T1 ..... BS170 (BS107)
M1 ..... mostek prostowniczy 1A DF08
U1 ..... AT90S2313
U2 ..... 7805
dioda LED dwukolorowa

**Inne**
X1 ..... Rezonator kwarcowy 3MHz
B1 ..... Obudowa na baterie 2xR6
PK1 ..... Przekaźnik RM83
TR1 ..... Transformator TS 2/14 (8,2Vx0,22A)
Z1,Z2 ..... Złącza ARK2
Wyświetlacz alfanumeryczny LCD 16x1
(ze sterownikiem HD44780)
Cztery przyciski mocowane do płyty czołowej

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2704