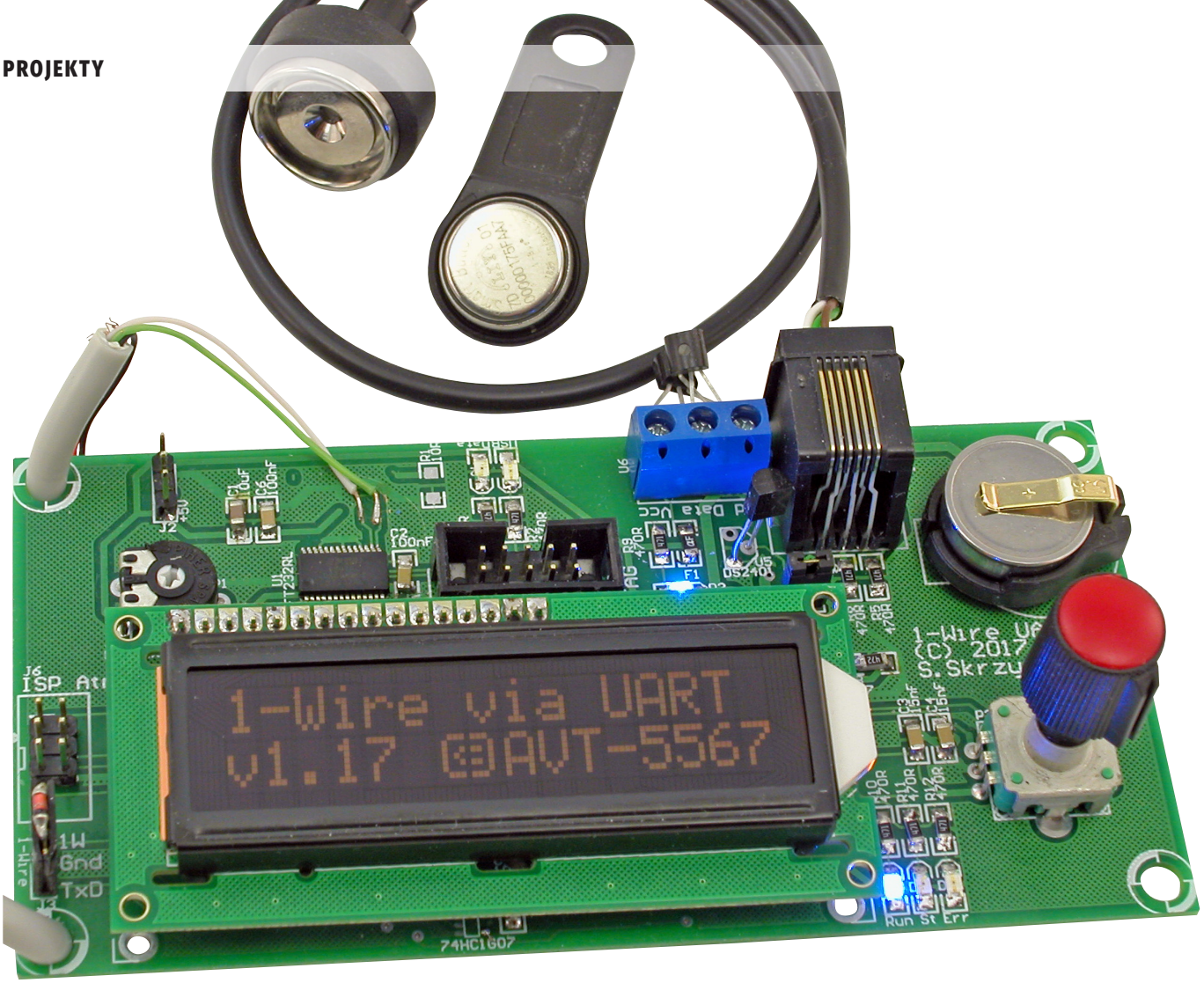


1-Wire za pomocą UART

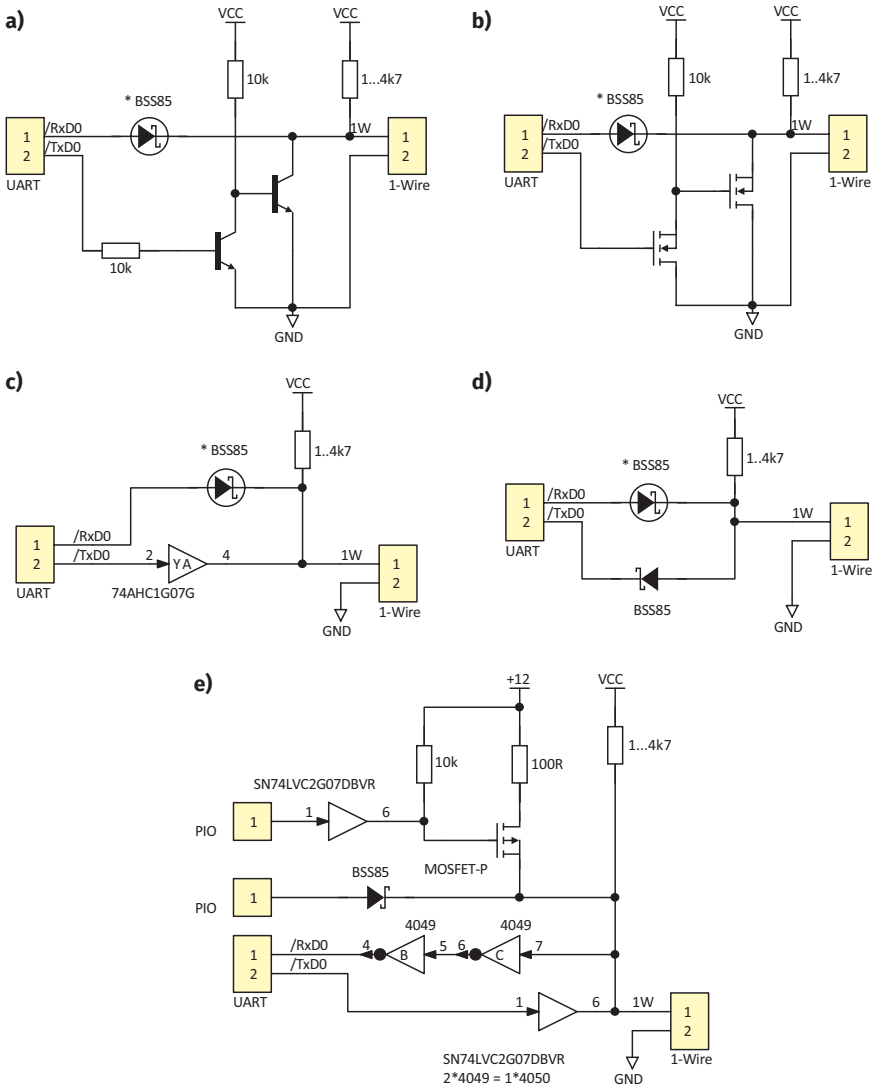
Płytka ewaluacyjna z mikrokontrolerem AVR

Interfejs 1-Wire firmy Dallas zdobył dużą popularność, zwłaszcza za sprawą termometrów DS18x20. Programowa obsługa transmisji jest stosunkowo łatwa, ale narzucone rygory czasowe powodują, że w czasie komunikacji przerwania muszą być wyłączone. Problem można rozwiązać, stosując układy DS2480 lub DS2482, ale podwyższa to cenę urządzenia. W artykule opisano sprzętową komunikację 1-Wire z wykorzystaniem UART-a. Dodatkowo opisano bibliotekę wyszukującą układy na magistrali. Opisano też płytkę ewaluacyjną umożliwiającą testowanie opisanych algorytmów.

***Rekomendacje:** Płytka przyda się do testowania urządzeń automatyki domowej lub może pracować w funkcji samodzielnego sterownika komunikującego się z czujnikami 1-Wire.*

Dane i zasilanie w trybie pasożytniczym są przesyłane za pomocą pojedynczej linii. W takiej sytuacji każdy projektujący układy elektroniczne domyśla się, że podczas transmisji bitu linia interfejsu nie może być zbyt długo wyzerowana, ponieważ dołączony do niej układ uzna, że wyłączono zasilanie i nastąpi jego restart, co będzie równoznaczne ze wznowieniem pracy układu. Dlatego też, pomimo nieskomplikowanej zasady działania, czas trwania bitu musi być ściśle określony. Jest to powodem, dla którego tworząc aplikację komunikującą się z otoczeniem przez 1-Wire, trzeba dobrze przemyśleć system przerwań, tak aby nie być w konflikcie z wymaganiami czasowymi transmisji. W praktyce zwykle na czas transmisji 1-Wire wyłącza się przerwania.

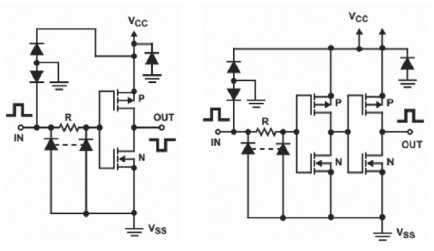
Programowa realizacja obsługi transmisji 1-Wire nie jest trudna, jednak wymaga używania timerów w celu odmierzania czasu i zwykle na czas obsługi transmisji angażuje CPU przez powstrzymywanie go od realizacji innych zadań. Gdyby tak udało się użyć interfejsu sprzętowego, zwolniłoby to nas od myślenia o zależnościach czasowych i uprościło oprogramowanie, ponieważ nadawaniem i odbiorem danych zająłby się sprzęt. Jak wspomniano we wstępie, można użyć układów DS2480 lub DS2482, ale podwyższają one koszt gotowego urządzenia i zajmują cenne miejsce na płytce. W nocie aplikacyjnej Microchipa AN187 opisano sposób użycia USART-a do obsługi transmisji 1-Wire. Od strony sprzętowej wymaga



Rysunek 1. Propozycje rozwiązań sprzętowych „przejściówki" UART/1-Wire: a) z użyciem 2 tranzystorów bipolarny, b) z użyciem 2 tranzystorów unipolarnych, c) z użyciem diody i 7407, d) z użyciem dwóch diod, e) z możliwością doprowadzenia impulsu programującego

on zastosowania jednego lub dwóch tranzystorów MOS (bipolarnych z rezystorami) lub jednej bramki 74xx07, np. 741G07 w miniaturowej obudowie SOT23-5. Wypróbowałem też rozwiązanie z diodą Schottky.

Propozycje rozwiązań sprzętowych pokazano na **rysunku 1**. Najlepsze wydaje mi się rozwiązanie z rysunku „C". Rozwiązanie z rysunku „D", jakkolwiek bardzo nieskomplikowane, nie jest zalecane. Na rysunku „E" przedstawiono interfejs, który umożliwia podanie impulsu programującego 12 V używanego przez pamięci EPROM. Na pierwszy rzut oka, wydawałoby się, że bramka CMOS



Rysunek 2. Wejście bramki w układach 4049 i 4050

4049 zostanie uszkodzona po podaniu na nią napięcia wyższego od jej napięcia zasilania. Tak by było w wypadku użycia standardowych bramek CMOS, jednak bufory 4049 i 4050 mają zmodyfikowane obwody zabezpieczające wejście bramki przed uszkodzeniem spowodowanym przez ESD. Nie są to dwie diody włączone w kierunku zaporowym, lecz dioda oraz dioda Zenera 15 V (**rysunek 2**). Dzięki temu, bez względu na napięcie zasilające, układ będzie on akceptował sygnały o napięciu do 15 V. Dlatego bramki 4049 i 4050 często są używane jako konwertery poziomów.

Przy generowaniu sekwencji 1-Wire za pomocą UART skorzystano z faktu, że przy prędkości transmisji wynoszącej 115200, czas trwania impulsu ujemnego podczas transmisji 0xFF wynosi około 8,7 µs, natomiast 0x00 około 78 µs. Umożliwia to wysyłanie zera i jedynki logicznej oraz odczyt bitu. Opisywaną sytuację zilustrowano na **rysunku 3** zaczerpniętym ze wspomnianej noty aplikacyjnej. Sekwencję reset i odczyt impulsu potwierdzającego obecność

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

W ofercie AVT* AVT-5567

Podstawowe parametry:
- Dwustronna płytka drukowana 126 mm×63 mm.
- Mikrokontroler ATmega162.
- Interfejs użytkownika złożony z 2 przycisków i enkodera.
- Komunikaty wyświetlane na LCD 2 linie × 16 znaków.
- Możliwość programowej emulacji 1-Wire za pomocą UART.
- Możliwość pracy funkcji centralki zamka, alarmu, czytnika sensorów 1-Wire itp.
- Programowanie za pomocą programatora szeregowego.

Projekty pokrewne na www.media.avt.pl:

AVT-5649	HUB 1-Wire z izolacją galwaniczną (EP 9/2018)
AVT-1949	Emulator DS18B20 (EP 4/2017)
AVT-1948	Interfejs termopary K z 1-Wire (EP 3/2017)
AVT-1787	Konwerter USB/1-Wire (EP 8/2013)

*** 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.

układu na magistrali przeprowadza się transmitując dane z prędkością 9600.

Na **listingu 1** zamieszczono fragment programu odpowiedzialny za nadawanie i odbiór danych przez 1-Wire. Niektóre układy o dużym poborze prądu podczas niektórych operacji (np. pamięci EEPROM w trakcie zapisu danych) mogą być zasilane pasożytniczo. Wymaga to „silnego" podciągania zasilania magistrali w czasie niektórych operacji. Jak to zrealizowano bez angażowania dodatkowych tranzystorów i wyprowadzeń mikrokontrolera pokazano na **listingu 2**.

W urządzeniu, do interakcji z użytkownikiem wykorzystano impulsator. Procedury jego obsługi dostępne w Internecie najczęściej są długie i skomplikowane. W modelu wykorzystano przerwania od impulsatora, dzięki czemu procedura jest bardzo prosta, nie zajmuje dużo czasu procesora i pamięci, działa poprawnie nawet przy szybkim obracaniu gałki impulsatora – pokazano ją na **listingu 3**.

Wyszukiwanie układów dołączonych do magistrali

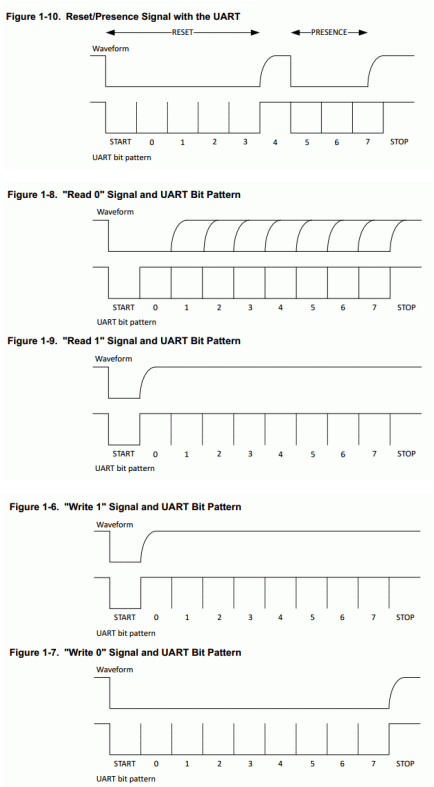
Liczba możliwych adresów 1-Wire wynosi 2^{56}. Zaadresowanie jednego układu zajmuje 960 µs (reset) + 8×60 µs (adres) = 1440 µs, więc przeskanowanie 2^{56} adresów w „normalny" sposób, nie licząc czasu przetwarzania danych, zajęłoby nieco ponad 3 miliony lat. Dlatego też firma Maxim-Dallas

zaimplementowała algorytm wyszukiwania układów, którego realizacja zajmuje 960 µs+(8+3×64)×61 µs =13,16 ms na znalezienie adresu pierwszego układu i 12,2 µs na każdy następny. Mechanizm wyszukiwaniu układów polega na odczytywaniu z układów bitów adresowych prostych i zanegowanych oraz adresowaniu jednym bitem układów włączonych do dalszego wyszukiwania. Procedurę tą powtarza się 64 razy. Ze względu na to, że jest stosunkowo skomplikowana, bardzo rzadko programiści implementują ją we własnych produktach. Językiem, w którym zaimplementowano wyszukiwanie w postaci funkcji bibliotecznej jest Bascom AVR. Niestety, w czasie jej pracy należy zapomnieć o przerwaniach. Przeszukując Internet znalazłem przykładową procedurę wyszukiwania układów. Zaimplementowałem ją do współpracy z UART-em. Ponadto, zoptymalizowałem procedurę obliczania CRC, dzięki czemu zajmuje mniej pamięci programu (**listing 4**). Warto wspomnieć, że poza funkcją wyszukiwania układów o kodzie komendy 0xF0 istnieje komenda ALARM SEARCH o kodzie 0xEC. Działa tak samo jak 0xF0 tyle, że swoją obecność zgłoszą układy w których wystąpiło kryterium alarmu (np. przekroczenie dopuszczalnej temperatury w termometrze).

Płytka testowa

Do testowania oprogramowania zaprojektowano płytkę uruchomieniową. Jej schemat ideowy pokazano na **rysunku 4**. Umieszczono na niej układy DS2401, DS2411, gniazdo dla pastylek, złącze czytnika oraz złącze ARK i SN25-6 z wyprowadzonymi sygnałami magistrali 1-Wire, zasilania zabezpieczonego bezpiecznikiem polimerowym oraz wyjściami LED (złącze SN25 i RJ12). Magistrala 1-Wire jest zabezpieczona układem DS9503.

Schemat montażowy płytki testowej zamieszczono na **rysunku 5**. Montaż jest typowy i nie wymaga szczegółowego omawiania. Pod wyświetlacz warto zastosować złącza goldpin (listwa i gniazdo). Bez tego dostęp do układów pod wyświetlaczem nie będzie możliwy, co znacznie utrudni lub



Rysunek 3. Zasada generowania impulsów 1-Wire przez UART

Listing 1. Podstawowe operacje 1-Wire realizowane za pomocą UART

```
//odczyt bitu
uint8_t onewireReadBit(void)
{
    uint8_t dsBit;
    byte ovt;
    int bit;

    onewireStrongPullup( FALSE );  // Wyłącz silne podciąganie
    UsartOW_Flush();
    UsartOW_Transmit( 0xFF );  // Wyślij impuls slotu odczytu
    ovt=OWOVT_BIT;  // Timeout = 80us
    while( (bit=UsartOW_Receive()) == NONE )
    {
        _delay_us(1);
        if (!(--ovt)) return(1);  // Timeout
    }
    if (ovt) while(ovt--) _delay_us(1);  // Czas na naładowanie pojemności magistrali
    #ifdef OWBTEST
        #warning „Tablica testowa odczytanych bitów."
        if (ow_pb <64) tab1wb[ow_pb++] = bit;     //test
    #endif
    if ((bit & 0xFF) != 0xFF) return(0);
    return(1);
}

//zapis bitu
byte onewireWriteBit(uint8_t bit)
{
    byte ovt;

    UsartOW_Flush();
    // Wyślij impuls slotu zapisu
    if (bit) UsartOW_Transmit(0xFF); else UsartOW_Transmit(0x00);
    ovt=OWOVT_BIT;  // Timeout = 100us
    while((bit=UsartOW_Receive()) == NONE)
    {
        _delay_us(1);
        if (!(--ovt)) return( 0x80 );  // Przekroczono Timeout
    }
    return( TRUE );
}

//reset
uint8_t onewireReset()
{
    uint8_t dsRst;
    byte ovt;
    int rst;

    OW_UBRRH = UBRR_RST >> 8;
    OW_UBRRL = UBRR_RST & 0xFF;
    UsartOW_Flush();
    UsartOW_Transmit( 0xF0 );
    ovt=OWOVT_RST;  // Timeout w 10us
    while( (rst=UsartOW_Receive()) == NONE )
    {
        _delay_us(10);
        if ( --ovt==0 )
        {
            OW_UBRRH = UBRR_BIT >> 8;
            OW_UBRRL = UBRR_BIT & 0xFF;
            UsartOW_Flush();
            return ( NONE );  // Timeout
        }
    }
    UsartOW_Flush();
    OW_UBRRH = UBRR_BIT >> 8;
    OW_UBRRL = UBRR_BIT & 0xFF;
    if ((rst & 0xFF) == 0xF0 ) return(FALSE);
    onewireStrongPullup( TRUE );  // Włącz silne podciąganie
    _delay_ms(1);
    onewireStrongPullup( FALSE );  // Wyłącz silne podciąganie
    return ( TRUE );
}
```

Listing 2. Załączenie wzmocnionego podciągania

```
void onewireStrongPullup( byte pullup )
{
    ONEWIREUP_PORT |= (1<<ONEWIREUP_DQ);  // Port stan wysoki (podciągaie silne lub słabe)
    if ( pullup )
    {
        #if defined( ONEWIRE_UART ) && defined( ONEWIREUP_RXD )
            OW_UCSRB &= ~(1<<RXEN0);  // Disable receiver
        #endif
        ONEWIREUP_DDR |= (1<<ONEWIREUP_DQ);  // Wyjście w stanie wysokim (silne podciąganie)
    }
    else
    {
        #if defined( ONEWIRE_UART ) && defined( ONEWIREUP_RXD )
            OW_UCSRB |= (1<<RXEN0);  // Enable receiver and transmitter
        #endif
        ONEWIREUP_DDR          &= ~(1<<ONEWIREUP_DQ);  // Wejście (słabe podciąganie)
    }
}
```

wręcz uniemożliwi uruchomienie urządzenia. W przypadku problemów ze zdobyciem lub wlutowaniem układu U3, można, zamiast niego zamontować diodę D9.

Poprawnie zmontowane urządzenie nie wymaga uruchamiania. Jedyną czynnością konieczną do przeprowadzenia po pierwszym uruchomieniu będzie wyregulowanie kontrastu wyświetlacza LCD. Kontrast można regulować na dwa sposoby, potencjometrem lub PWM-em. Jeśli regulacje przeprowadzamy potencjometrem, wtedy nie montuje się rezystora R4 albo w kodzie programu ustawiamy port PD5 jako wejście. Jeśli regulacja ma odbywać się z poziomu programu, to nie montujemy potencjometru P1. Można też pozostawić regulację zarówno potencjometrem jak i PWM-em. Zaletą takiego rozwiązania jest to, że wyświetlacz negatywowy będzie czytelny także po zatrzymaniu programu w trybie debug. Wymagane będzie jednak dobranie stałych KONTLCD_MIN, KONTLCD_DEF i KONTLCD_MAX.

Po włączeniu zasilania urządzenie skanuje magistralę w poszukiwaniu układów. Po skanowaniu wyświetlana jest informacja o liczbie znalezionych układów i czasie, w którym je znaleziono. Jak łatwo policzyć, mikrokontroler AVR taktowny częstotliwością 14,7 MHz jest w stanie zidentyfikować około 55 układów w ciągu sekundy w trybie UART (w trybie PIO 85). Po ekranie wyszukiwania jest wyświetlany ekran z danymi (typ i adres) pierwszego znalezionego układu. Kręcąc impulsatorem można obejrzeć dane kolejnych znalezionych układów. Jeśli wyświetlają się dane układu DS18B20, to krótkie naciśnięcie przycisku impulsatora wyświetli zmierzoną temperaturę. Komunikat „Pas” za temperaturą oznacza to, że układ jest zasilany pasożytniczo z magistrali 1-Wire, natomiast „Ext” informuje o dodatkowym zasilaniu.

Jeśli do magistrali zostanie dołączony kolejny układ wyświetlony zostanie komunikat o liczbie dodatkowo znalezionych układów. Podobnie dzieje się jeśli układy zostaną odłączone od magistrali. Naciskając na krótko impulsator przejdziemy do menu głównego. Przytrzymując przycisk impulsatora ponad sekundę wchodzimy do menu, w którym można ustawić filtr wyszukiwania układów. Kręcąc impulsatorem zmieniamy filtr. Wartość $00 oznacza wyszukiwanie wszystkich układów dołączonych do magistrali, $FF wyszukiwanie alarmów. Inne wartości spowodują wyszukiwanie układów wybranej rodziny, na przykład: $01 – numery seryjne DS1990, DS2401, DS2411, $28 – termometr DS24B20 itd.

Z menu wychodzimy naciskając przycisk impulsatora. Oprogramowanie ustawia temperaturę alarmu w układach zgłoszonych jako nieparzyste na TH=18°C, TL=8°C, jako parzyste na TH=40°C, TL=–5°C. Dzięki temu po podłączeniu dwóch lub więcej termometrów DS18B20 podczas wyszukiwania alarmów, zgłosi się część z nich. Jeśli oprogramowanie wykryje zmianę liczby układów, to wyświetli stosowny komunikat. Dołączając układ w trakcie wyszukiwania można zakłócić istniejącą transmisję, co może spowodować błędne komunikaty. Następuje to zwłaszcza podczas przykładania pastylki do czytnika. Wtedy to można zewrzeć nią linię danych z masą, co spowoduje chwilowe wyświetlenie błędu magistrali.

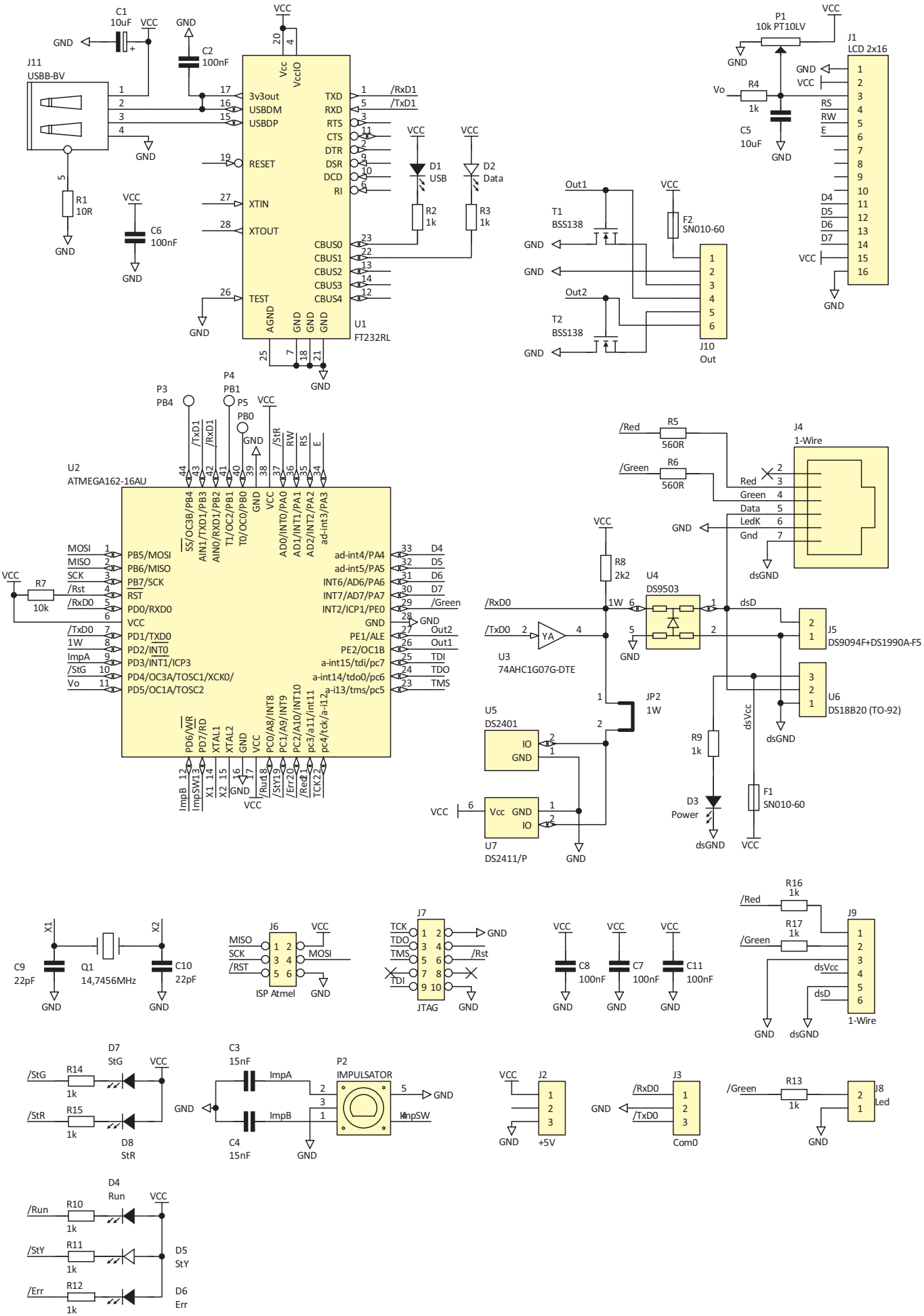
Aby ustawić kontrast wyświetlacza z poziomu programu, należy wcisnąć przycisk impulsatora i wywołać restart mikrokontrolera przez włączenie zasilania lub zwarcie linii reset procesora (piny 5-6 na J6). Kręcąc impulsatorem zmieniamy kontrast, który na bieżąco jest aktualizowany na wyświetlaczu. Dodatkowo, jego poziom jest wyświetlany na poziomym słupku Wciskając

Listing 3. Fragment procedury obsługi impulsatora

```
#define    RdImpA   (PIND & (1<<PD3) )
#define    RdImpB   (PIND & (1<<PD6) )
#define    RdImpSw  (PIND & (1<<PD7) )
#define    ddrImp() PORTD |=(1<<PD3) | (1<<PD6) | (1<<PD7)
byte volatile IMPdata;
byte IMPmin=0, IMPmax=255, IMProl=TRUE;

//  Inicjalizacja INT 1
void InitInt1()
{
    GICR   |= (1<<INT1);  // INT1   - Zezwolenie na IRQ od wejscia INT1
    MCUCR  |= (2<<ISC10);  // INT1   - Zbocze narastajace na INT1 wywołuje IRQ
    ddrImp();
}

//  Obsługa IRQ od wejścia INT 1
SIGNAL( INT1_vect )  // Musi być SIGNAL
{
int data;
    data = IMPdata;  // Zmienne volatile więc zapamiętujemy w rejestrach
    if ( !RdImpB )  // Wstawiając negację („!”) zmieniamy kierunek
    {
        data++;
        if ( IMProl && data > IMPmax ) data = IMPmin;  // ograniczenie zakresu
        if ( !IMProl && data >= IMPmax ) data = IMPmax;
    }
    else
    {
        data--;
        if ( IMProl && data < IMPmin ) data = IMPmax;  // ograniczenie zakresu
        if ( !IMProl && data <= IMPmin ) data = IMPmin;
    }
    IMPdata = data;
}
```

Listing 4. Wyszukiwanie układów 1-Wire

```
/* Funkcja szuka cyklicznie układów na magistrali. Komunikację po
   magistrali pożna przeprowadzic gdy wyszukiwanie zostało zakończone
  (funkcja zwraca zero lub liczbę znalexionych układów) w przeciwnymm
   wypadku wyszukiwanie zostanie przerwane. Proces wyszukiwania
   funkcja sygnalizuje zwracając -1 */

byte SearchStep( byte family )
{
    byte static StSearch=0, filtr=0;
    byte rslt, mem;

    if ( !StSearch )
    {
        if ( OWFirst() )  // Jeśli znaleziono pierwszy układ
        {
            rslt=SearchAdd( family, TRUE );  //dodaj do tablicy
            StSearch++;  // ustaw szukanie następnego
            if ( rslt ) filtr++;
            return( SEARCH_BUSY );  // zwróć wartość „szukam”
        }
        else  // Nic nie znaleziono
        {
            return( 0 );  // zwróć zero
        }
    }
    else
    {
        if ( OWNext() )  // Jeśli znaleziono następny układ
        {
            rslt=SearchAdd( family, FALSE );  //dodaj do tablicy
            StSearch++;  // Szukaj kolejnego
            if ( rslt ) filtr++; return( SEARCH_BUSY );
        }
        else  // Nie ma
        {
            if ( !family )  // Jeśli wyszukiwane wszystkie
            {
                mem = StSearch;  // Zwracamy liczbę wszystkich układów
                StSearch = filtr = 0;
                return( mem );  // Zwróć liczbę znalezionych układów
            }
            else  // Jeśli wyszukiwane przez filtr
            {
                mem = filtr;  // Zwracamy liczbę przefiltrowanych
                StSearch = filtr = 0;
                return( mem );  // Zwróć liczbę znalezionych układów
            }
        }
    }
    return( SEARCH_BUSY );
}
```



Rysunek 4. Schemat ideowy zestawu rozwojowego

Tabela 1. Funkcje diod LED

Nazwa	Oznaczenie	Funkcja
D1	USB	Świeci po wykryciu urządzenia przez Host USB.
D2	Data	Miga podczas przesyłania danych po USB.
D3	Power	Sygnalizuje zasilanie układów 1-Wire.
D4	Run	Pulsuje podczas pracy mikrokontrolera.
D5	StY	Miga podczas odbioru danych z magistrali 1-Wire. Dane wysłane są maskowane: LedOn jeżeli Rxd = 0 i Txd=1.
D6	Err	Świeci gdy nie wykryto żadnego układu 1-Wire.
D7	StG	Nieużywana.
D8	StR	Nieużywana.

przycisk impulsatora zapisujemy ustawienia w pamięci EEPROM, dzięki czemu ustawienia kontrastu są odtwarzane po włączeniu urządzenia.

Przydatne porady

Jeśli w procesorze brak portów szeregowych, można użyć zewnętrznych układów UART, np. SC16IS740/750/760 firmy NXP, które maja sprzętowy FIFO mieszczący 16 bajtów i są zgodne na poziomie rejestrów z 16C550. Komunikacja z układem odbywa się za pomocą interfejsu I²C lub SPI. Zewnętrzne układy UART są alternatywą dla konwerterów z serii DS.

Do komunikacji 1-Wire można także użyć interfejsu SPI w trybie master lub dwóch układów I²C w trybie master i slave. Przy zegarze 14,7456 MHz sygnał taktujący SPI należy podzielić przez 16 (rejestr SPCR i SPSR) tak jak i I²C (rejestr TWBR i TWPS). Wyprowadzenie SCLK (SCL w I²C) nie jest używane w tym zastosowaniu.

Podczas pracy mikrokontrolera dioda „Run" płynnie rozjaśnia się i ściemnia. PWM jest zrealizowany programowo w przerwaniach wywoływanych co 1 ms. Jak łatwo zauważyć, podczas obsługi 1-Wire praca diody nie jest zakłócana nawet w trybie PIO. Sekwencja reset magistrali też pracuje poprawnie. Błędy magistrali 1-Wire są sygnalizowane zaświeceniem diody „Err". Pojedynczy błąd zaświeci diodę na 0,1 s.

Układy na magistrali podczas wyszukiwania zgłaszają się według kolejności swoich numerów seryjnych. W przypadku podłączonych wielu termometrów do magistrali, przyłączenie kolejnego może zmienić ich kolejność (nie ma gwarancji, że nowo podłączony układ zgłosi się jako ostatni). Aby identyfikować układy można wykorzystać rejestry TH/USER1 i TL/USER2, które rzadko są używane w swoim pierwotnym zastosowaniu. W bibliotece „onewire_DS18B20" znajdują się funkcje obsługujące EEPROM i konfigurację.

Podczas skanowania magistrali określenie adresu jednego układu zajmuje ponad 12 ms. Jeśli w tym czasie przerwania działają i dane odbierane przez USART są interpretowane w przerwaniach, to nie ma problemu. Jeśli natomiast dane są interpretowane w programie głównym i są buforowane w RAM z użyciem przerwań, zależnie od prędkości, trzeba przydzielić odpowiednio duży obszar RAM na dane. Przykładowo, gdy do 1-Wire przyłączonych jest 10 układów, skanowanie zajmie około 135 ms. W tym czasie, przy prędkości 19200 (czas bajtu przy 8 bitach danych to 0,520 ms), UART może przyjąć 260 bajtów. Stosując prostą „sztuczkę" w wyszukiwaniu układów, wystarczy aby bufor na dane UART pomieścił tyle znaków, ile przyjdzie w 13 ms. Przy parametrach transmisji (19200, 8, n, 1) będzie to 25 znaków (10 razy mniejszy bufor na UART). Po zmianie prędkości na 115200 – 151 znaków zamiast. W tym celu należy funkcje „OWFirst()" i „OWNext()" wywoływać cyklicznie, a nie jedna po drugiej, jak pokazano na **listingu 4**. Trzeba pamiętać, aby pomiędzy „OWFirst()" a końcem szukania („OWNext()" zwraca FALSE) nie wykonywać innych operacji na magistrali, ponieważ przerwie to proces wyszukiwania. Inne operacje należy wykonywać przed wywołaniem „OWFirst()" lub po „OWNext()", jeśli ta zwróci FALSE. Aby wyszukać układy zgłaszające alarm należy użyć funkcji „OWFirstAlarm()" i „OWNextAlarm()" zamiast „OWFirst()" i „OWNext()".

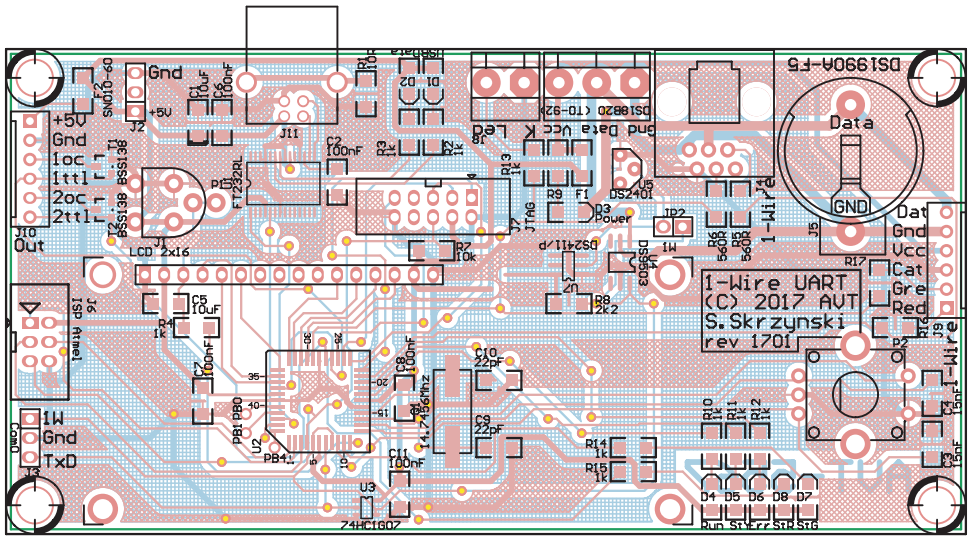
Istotny jest też sposób obsługi bufora FIFO i to zarówno programowego (na przerwaniach) jak i sprzętowego (w przypadku zewnętrznych UART). Sugeruję następujący sposób działania procedury obsługi FIFO:

1. Jeśli nie ma znaku w buforze odbiorczym, to opuścić funkcję.
2. Odczytać znak z FIFO i zapisać w buforze.
3. Zwiększyć wskaźnik bufora i skoczyć do punktu 1.

Oczywiście nic nie stoi na przeszkodzie aby, korzystając z UART-a do 1-Wire, obsługę 1-Wire przenieść do przerwania.

Obsługa 1-Wire przez USART jest szybka, co ma pewną wadę. Po transmisji bitu o wartości 0 urządzenia slave zasilane pasożytniczo muszą naładować swoje pojemności podtrzymujące zasilanie. Dlatego po transmisji bitu należy odczekać pewien czas oznaczony w nocie katalogowej jako T_{REC} wynoszący minimum 1 µs. Zależnie od liczby współpracujących układów i długości (a tym samym pojemności) przewodów połączeniowych ten czas będzie dłuższy. Można go znacznie zmniejszyć włączając na chwilę silne podciąganie.

Na koniec przydatna uwaga odnośnie do obsługi wyświetlacza. W urządzeniu zastosowano dynamiczne przydzielanie znaków, których wzorce nie znajdują się w ROM sterownika HD44780. Jak wiadomo, sterownik może przechowywać 8 znaków użytkownika. Aby zapamiętać wszystkie polskie znaki narodowe jest potrzebna pamięć na 18 symboli. Z tym problemem można sobie poradzić ograniczając znaki do „ąćęłńóśźż". Wykorzystuje się tu fakt, że wyraz rzadko rozpoczyna się od polskiej litery, a „ź" i „ż" są do siebie podobne. Co jednak zrobić z innymi symbolami, jak stopnie, PLAY,



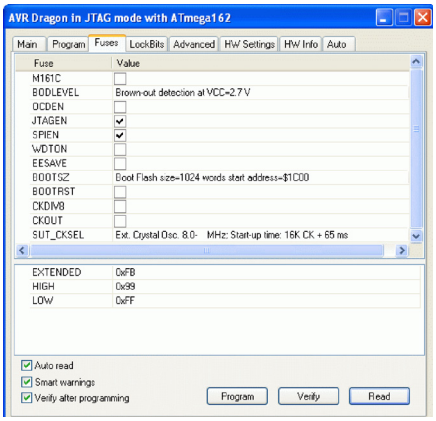
Rysunek 5. Schemat montażowy zestawu rozwojowego

Listing 5. Dynamiczne wyświetlanie polskich znaków

```
switch(znak)
{
//----- Polskie małe -----//
    case('ą'): nrCharFlash =  0; alt='a'; break;
    case('ć'): nrCharFlash =  1; alt='c'; break;
    case('ę'): nrCharFlash =  2; alt='e'; break;
    case('ł'): nrCharFlash =  3; alt='l'; break;
    case('ń'): nrCharFlash =  4; alt='n'; break;
    case('ó'): nrCharFlash =  5; alt='o'; break;
    case('ś'): nrCharFlash =  6; alt='s'; break;
    case('ź'): nrCharFlash =  7; alt='z'; break;
    case('ż'): nrCharFlash =  8; alt='z'; break;
//----- Polskie wielkie -----//
    case('Ą'): nrCharFlash =  9; alt='A'; break;
    case('Ć'): nrCharFlash = 10; alt='C'; break;
    case('Ę'): nrCharFlash = 11; alt='E'; break;
    case('Ł'): nrCharFlash = 12; alt='L'; break;
    case('Ń'): nrCharFlash = 13; alt='N'; break;
    case('Ó'): nrCharFlash = 14; alt='O'; break;
    case('Ś'): nrCharFlash = 15; alt='S'; break;
    case('Ź'): nrCharFlash = 16; alt='Z'; break;
    case('Ż'): nrCharFlash = 17; alt='Z'; break;
//----- Specjalne -----//
    case( LCDCHAR_STOPNIE_c ):  nrCharFlash = 18; alt='o';  break;
    case( LCDCHAR_PLAY_c ):  nrCharFlash = 19; alt='>';  break;
    case( LCDCHAR_PAUSE_c ):  nrCharFlash = 20; alt='|'; break;
    case( LCDCHAR_COPYRIGHTL_c ):  nrCharFlash = 21; alt='('; break;
    case( LCDCHAR_COPYRIGHTH_c ):  nrCharFlash = 22; alt='c'; break;
    case( LCDCHAR_RESERVEDL_c ):  nrCharFlash = 23; alt='('; break;
    case( LCDCHAR_RESERVEDH_c ):  nrCharFlash = 24; alt='r'; break;
}

if ( alt )  // Jeśli istniej znak alternatywny
{  // to znaleźiono znak charakterystyczny dla alfabetu
    for(x=0; x<8; x++)  // Szukanie znaku w tablicy dynamicznej
    {
        if ( tabChar[x] == znak ) return( x );
    }
    //----- Nie znaleziono znaku -----//
    for(x=0; x<8; x++)  // Szukanie wolnego miejsca
    {
        if ( !tabChar[x] )
        {
            tabChar[ x ] = znak;  // Kod znaku do tablicy
            LcdPutCGRAM( x, (byte*)(lcdCHAR+(nrCharFlash<<3)) );  // Wzór znaku do LCD
            return( x );
        }
    }
    return( alt );  // Brak miejsca, daj znak alternatywny
}
return( znak );  // Nie było konwersji, zwróć co dostałeś
```

FORWARD, PAUSE i innymi? Jest na to sposób. Zakładając, że na jednym ekranie wyświetlacza (zwłaszcza małego) nie pojawia się więcej niż 8 znaków spoza generatora w ROM, znaki można przydzielać dynamicznie. W tym celu, po inicjalizacji wyświetlacza lub komendzie jego czyszczenia tablica znaków CGRAM jest zerowana. Gdy na wyświetlaczu ma być wyświetlony znak specjalny tablica wzorców znaków jest przeszukiwana pod kątem jego występowania. Jeśli wzorzec zostanie odnaleziony, to jest odczytywany kod znaku z tablicy CGRAM i wysyłany do wyświetlacza. Gdy znaku nie ma, to na pierwszej wolnej pozycji w tabeli jest zapisywany kod znaku, a jego wzorzec wysyłany do CGRAM wyświetlacza. Gdy tablica jest pełna, wyświetlany jest znak alternatywny, np. zamiast „ą” „a”. Tym sposobem znaków specjalnych może być kilkanaście, ale na jednym ekranie można wyświetlić tylko 8 z nich. Opisywany sposób obsługi wyświetlacza ilustruje **listing 5**.

Na płycie dostępne są piny do dowolnego wykorzystania (PB0, PB1, PB4) oraz diody led „StG” i StY”. Ponadto, nie są oprogramowane wyprowadzenia sterujące sygnalizacją LED w czytniku pastylek (port PE0 LED „Green” i PC3 LED „Red”). Na złącze J10 wyprowadzono sygnały „Out1” i „Out2” o poziomach TTL i otwarty dren. Dzięki temu oraz po uwzględnieniu faktu zabezpieczenia magistrali 1-Wire i obecności pamięci EEPROM w mikrokontrolerze, po modyfikacji oprogramowania urządzenie można wykorzystać, na przykład, jako sterownik rygla w systemie kontroli dostępu.



Rysunek 6. Ustawienie fusebitów mikrokontrolera

Wykaz elementów:

Rezystory: (SMD 1206)
R1: 10 Ω
R2…R4, R9…R17: 1 kΩ
R5, R6: 560 Ω
R7: 10 kΩ
R8:2, 2 kΩ
P1: 10 kΩ lin. (potencjometr)

Kondensatory: (SMD 1206)
C1, C5: 10 µF
C2, C6…C8, C11: 100 nF
C3, C4: 12 nF
C9, C10: 22 pF

Półprzewodniki:
D1…D8: LED SMD
T1, T2: BSS138 (SOT-23)
U1: FT232RL (SSOP-28)
U2: ATmega162-16AU (PQFP44)
U3: 74AHC1G07G-DTE (SOT-23-5)
U4: DS9503 (SO-6)
U5: DS2401
U6: DS18B20 (TO-92)
U7: DS2411/P (TSOC-6)

Inne:
F1, F2: SN010-60
J1: moduł wyświetlacza LCD 2×16
J2, J3: SIP3
J4: złącze 8P8C
J5: DS9094F+DS1990A-F5
J6: IDC6
J7: IDC10
J8: złącze ARK2
J9, J10: HU06
J11: USBB-B
JP2: SIP2
P2: impulsator
Q1: rezonator 14,7456 MHz (HC49S)

Dostępne są dwie wersje programu. „1-Wire4usart” wykorzystuje UART do obsługi 1-Wire, natomiast „1-Wire4pio” korzysta z portu IO. Ustawienie fusebitów mikrokontrolera pokazano na **rysunku 6**.

ES
es@ep.com.pl