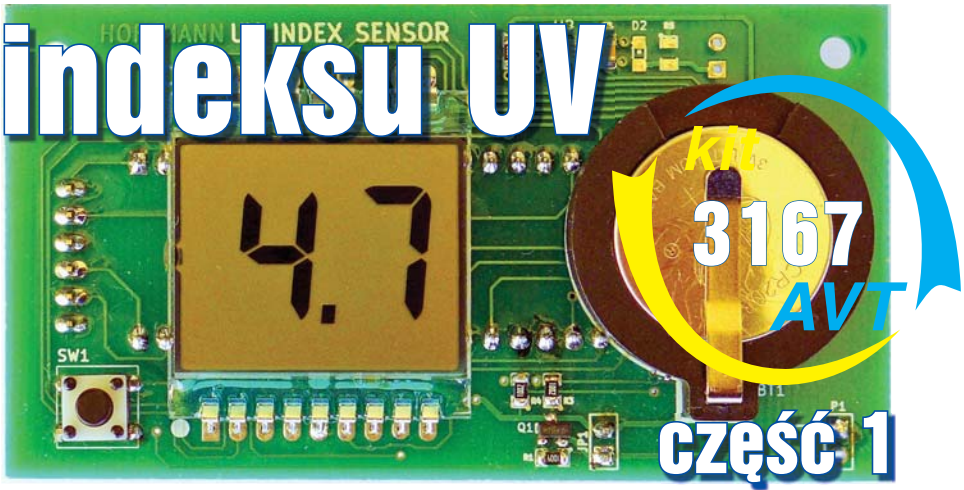
Miernik indeksu UV

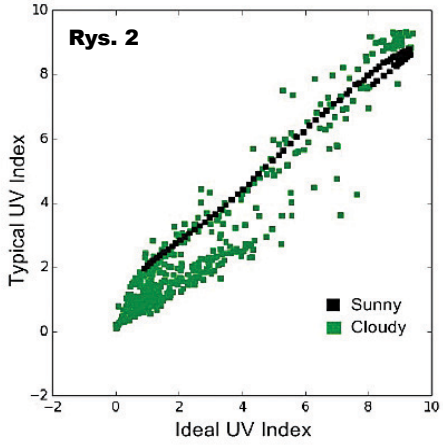


Początek wiosny oznacza coraz dłuższe i bardziej słoneczne dni. Słońce pozytywnie wpływa na nasze samopoczucie i zdrowie. Jednak promieniowanie UV ma również wpływ niekorzystny. Świadomość tego zagrożenia jest powszechna i wiele osób stosuje środki ochronne w postaci kremów i okularów słonecznych z filtrami UV. Kojarzą się one jednak głównie z urlopem i pobytem na plaży. Ale jak silne zagrożenie promieniowaniem UV występuje na co dzień? Czy np. wiosną w mieście też powinniśmy stosować środki ochronne? Odpowiedzi na to pytanie udziela wskaźnik nazwany indeksem UV. Jest to liczba z przedziału od 0 do 11+, **Rysunek 1** przedstawia barwną reprezentację poziomów zagrożenia promieniowaniem. Kolor zielony to brak zagrożenia, a fioletowy to zagrożenie „ekstremalne". Dla wartości powyżej 3 zalecane jest stosowanie środków chroniących przed promieniowaniem w postaci okularów przeciwsłonecznych, kremu z filtrem UV i nakrycia głowy. Im wyższa wartość indeksu, tym wyższy powinien być parametr SPF stosowanego kremu, a krótszy czas spędzany na słońcu. Szczegółów, jak zachowywać się na słońcu dla poszczególnych wartości indeksu, można szukać np. w Wikipedii.

Sposób wyliczania i pomiaru indeksu UV został ustandaryzowany przez Światową Organizację Zdrowia (WHO) i Światową Organizację Meteorologiczną (WMO). Indeks UV może zostać obliczony dla danego miejsca i czasu za pomocą modeli komputerowych, które biorą pod uwagę takie parametry jak: kąt padania słońca, zachmurzenie, grubość warstwy ozonowej czy wysokość nad poziomem morza. Indeks może być również mierzony bezpośrednio za pomocą specjalizowanych czujników.

Wartość indeksu UV nie jest jeszcze prezentowana w prognozach pogody w polskich mediach. Na stronach Instytutu Meteorologii i Gospodarki Wodnej przedstawiane są pomiary w czasie rzeczywistym dla czterech miast: Łeba, Legionowo, Katowice i Zakopane. Dostępne są tam również mapy prognozowanego indeksu UV dla całej Polski. Jednak gdy pisałem ten artykuł, na stronie widniała informacja: „Przerwa w prognozie indeksu UV do wiosny 2017". O silnym wpływie promieniowania UV również zimą przekonało się wielu narciarzy wracających ze stoku z opalenizną. W tym wypadku najważniejsze czynniki to odbijanie się promieniowania słonecznego od śniegu i rozrzedzone powietrze na większej wysokości. To tylko jeden z przykładów na to, że wartość indeksu UV jest zależna od konkretnego miejsca i sytuacji i dlatego warto mieć możliwość wykonania indywidualnego pomiaru za pomocą przenośnego urządzania.

Rys. 1



Najprostsze mierniki indeksu UV można kupić latem w dyskontach za cenę około dwukrotnie wyższą od ceny zastosowanego w projekcie czujnika. Jedną z odpowiedzi na pytanie, po co w takim razie budować własny miernik, może być pewność zastosowanego elementu pomiarowego. Budując własne urządzenie, wiemy, że pochodzi on od renomowanego producenta (Silicon Labs) i możemy się zapoznać z jego parametrami w nocie katalogowej. **Rysunek 2** pokazuje porównanie wartości odczytanych z czujnika (oś Y) z wartością indeksu UV wyliczoną według zaleceń WHO (oś X). Ale nigdy się nie dowiemy, jak wygląda analogiczna zależność dla taniego chińskiego urządzonka.

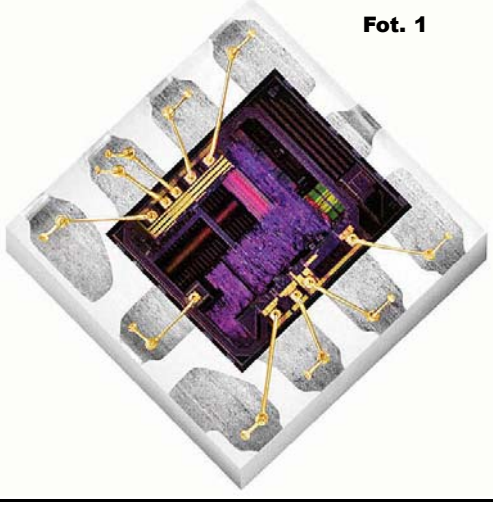
Powodem do zbudowania własnego miernika może też być chęć zapoznania się z programowaniem mikrokontrolerów PIC w języku C, obsługi wyświetlacza LCD bez wbudowanego sterownika oraz wykorzystania czujnika o wiele bardziej skomplikowanego od typowych przetworników temperatury czy wilgotności.

Zachęcam do budowy tego nieskomplikowanego urządzenia, które może pozytywnie wpłynąć na nasze bezpieczeństwo. Na pewno nie w takim stopniu jak czujniki tlenku węgla (których konstrukcję lepiej pozostawić profesjonalistom), ale jednak. Określenie, jak silne jest promieniowanie UV w słoneczny dzień, może nam pomóc podjąć decyzję czy schować się do cienia lub zastosować krem z filtrem, a przez to zadbać o zdrowie naszej skóry.

Opis układu

W urządzeniu wykorzystany jest bardzo zaawansowany, nowoczesny i wielofunkcyjny czujnik Si1147 produkcji Silicon Labs (**fotografia 1**). Jest on używany np. w smartfonach. Ale w przeciwieństwie

Fot. 1

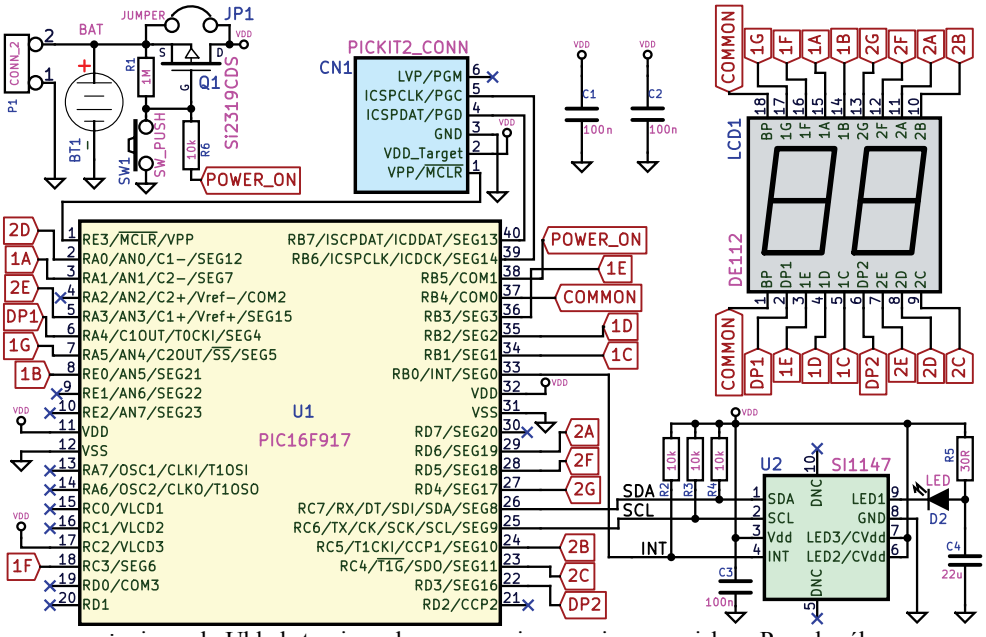


do smartfonów, które służą do wszystkiego, ja chciałem się skupić na jednej funkcjonalności – na pomiarze indeksu UV. Urządzenie z założenia miało być maksymalnie proste w użytkowaniu. Ma mieć wyłącznie jeden przycisk, który je włączy i po wystawieniu na słońce na wyświetlaczu pokaże liczbę reprezentującą indeks UV. I taka funkcjonalność została zrealizowana. Indeks UV jest odświeżany 10 razy na sekundę, a po 15 sekundach urządzenie wyłącza się całkowicie – nie pobiera w ogóle prądu z baterii.

Kolejne założenie projektowe to minimalne zużycie energii, ponieważ urządzenie ma być niewielkie i przenośne, a jedna bateria „pastylkowa"' ma wystarczyć na wiele pomiarów. Wybrane źródło zasilania to bateria typu CR2032. Ponieważ miernik ma pokazać wyłącznie jedną liczbę z zakresu od 0 do 11, wybrałem dwucyfrowy siedmiosegmentowy wyświetlacz LCD bez sterownika. Jest to wyświetlacz podobny do tych stosowanych np. w multimetrach cyfrowych czy prostych zegarach. Brak sterownika w wyświetlaczu stwarza wymaganie dla zastosowanego mikrokontrolera – powinien on mieć peryferia pozwalające na obsługę tego typu wyświetlaczy. Do takich mikrokontrolerów należy np. zastosowany PIC16F917.

W projektach, artykułach czy kursie C w EdW panują mikrokontrolery AVR, a ja w dalszej części artykułu przedstawię, jak zacząć przygodę z mikrokontrolerami PIC – również programowanymi w C. PIC16F917 znakomicie spełnia też wymaganie małego poboru prądu – całe urządzenia pobiera podczas pracy poniżej 1mA.

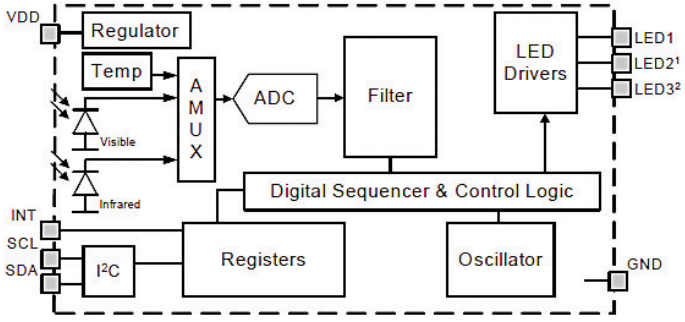
Urządzenie to typowy układ mikroprocesorowy (**rysunek 3** przedstawia jego schemat) składające się z czujnika (U2), mikrokontrolera (U1) i wyświetlacza (LCD1). U2 nie jest „gołym" czujnikiem, tylko skomplikowanym układem cyfrowym, wyposażonym w sensory światła widzialnego i podczerwonego, a także temperatury. Poza tym ma zintegrowany multiplekser, przetwornik analogowo-cyfrowy, filtry cyfrowe, a także... mikrokontroler. Producent nie podaje typu mikrokontrolera. Użytkownik nie ma też wpływu na jego oprogramowanie. Aby wykorzystać taki czujnik, musimy użyć drugiego mikrokontrolera, który z tym wewnętrznym będzie się komunikował poprzez magistralę I2C. Si114X zawiera też do trzech konfigurowalnych prądowych sterowników diod LED. Są one niezbędne do pomiarów zbliżeniowych. Układ steruje podczerwonymi diodami LED i na podstawie odbicia ich światła od obiektów oblicza odległość – nawet do 50 cm przy słabszym oświetleniu zewnętrznym. Wersje czujnika różnią się właśnie liczbą driverów LED: Si1145 ma tylko jeden, a Si1147 – trzy. W projekcie można wykorzystać dowolną wersję układu, ponieważ dla pomiarów indeksu UV drivery LED są nieistotne. **Rysunek 4** przedstawia schemat wewnętrznej struktury Si114X. Widoczne są tam dwie fotodiody: czuła na światło widzialne i na promieniowanie podczerwone. Nie ma za to elementu czułego na promieniowanie UV. A więc tak naprawdę indeks nie jest mierzony bezpośrednio, za to wewnętrzny mikrokontroler czujnika wylicza go na podstawie zależności pomiędzy zmierzonymi wartościami dla światła widzialnego i podczerwonego. Czujnik jest bardzo energooszczędny, pobiera zaledwie 3uA w stanie aktywności i <500nA w stanie czuwania.

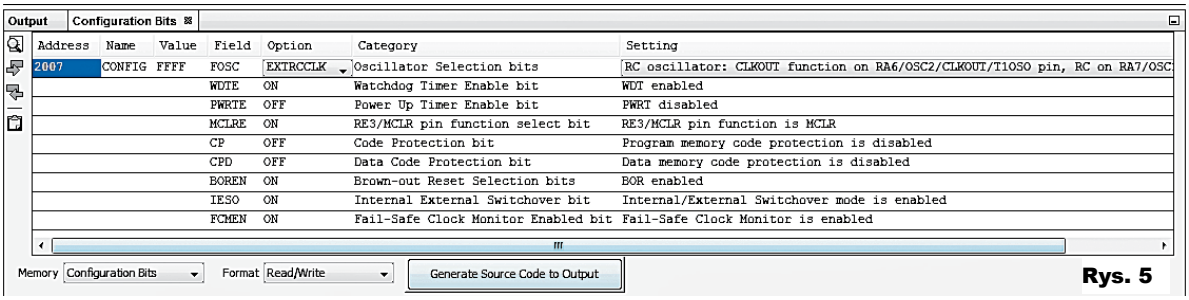


Rys. 3

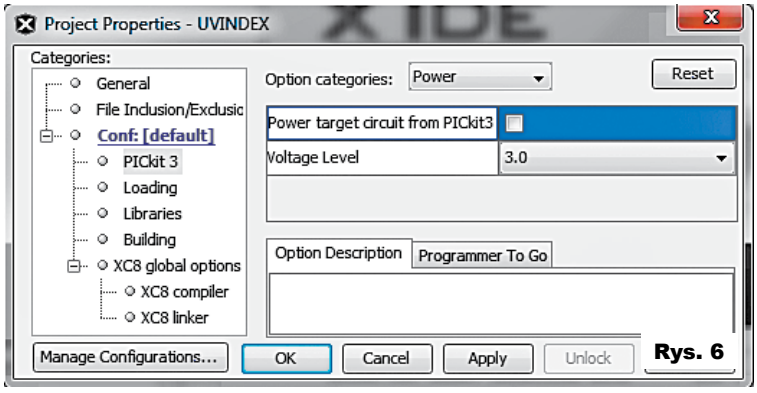
Źródłem zasilania całego urządzenia jest bateria CR2032. Zalecam wykorzystanie dokładnie takiego typu uchwytu baterii jak podany w wykazie elementów, ponieważ charakteryzuje się on bardzo dobrą jakością. W stanie spoczynku otwarty tranzystor P MOSFET Q1 odcina zasilanie całego urządzenia. Użytkownik naciskając przycisk SW1, otwiera tranzystor. Zasilony mikrokontroler wysterowuje pin 5 portu B do stanu niskiego, dzięki czemu zasilanie jest podtrzymane po puszczeniu przycisku. Po określonym czasie mikrokontroler zmienia stan pinu na wysoki, odcinając swoje własne zasilanie. Zworka JP1 pozwala dołączyć zasilanie na stałe – jest przydatna w trakcie opracowywania oprogramowania lub programowania mikrokontrolera. Złącze P1 zostało dodane jako opcjonalne do zasilania układu nie z baterii, tylko z zewnętrznego zasilacza. Czujnik promieniowania jest połączony z mikrokontrolerem magistralą I2C. Dodatkowo wyjście przerwania czujnika dołączone jest do odpowiedniego wejścia mikrokontrolera. Praca z przerwaniami nie jest wykorzystywana w oprogramowaniu, ale połączenie pozwala na jego rozwój. Mimo niewykorzystania przerwań, bardzo ważne jest, by nie zapomnieć o montażu rezystora R2 podciągającego wyjście przerwania czujnika do plusa zasilania. Podczas włączania Si1147 to wyjście pełni funkcję wejścia wpływającego na pracę wewnętrznego mikrokontrolera. Brak rezystora podciągającego może skutkować nieprawidłową pracą czujnika. Pozostałe rezystory podciągające są typowe dla magistrali I2C. Złącze CN1 służy do programowania mikrokontrolera w systemie (ISP) i jest zgodne z programatorami PICkit 2 i 3 produkcji Microchip.

Rys. 4





Rys. 5



Rys. 6

Jak zacząć z „PICami”? MPLAB X IDE i PICkit 3

Do programowania mikrokontrolerów PIC wykorzystuję środowisko MPLAB X IDE w wersji 3.40 oraz programator PICkit 3. IDE jest darmowe i dostępne do pobrania ze strony Microchipa. Mój programator na pewno jest oryginalny – zakupiony w TME. Obecnie kosztuje tam 205zł netto. Może jest to cena duża w porównaniu z niektórymi narzędziami dla AVR-ów, ale musimy pamiętać, że jest to oryginalny programator producenta układów. Wiele tego typu narzędzi dla ARMów przekracza cenę 1000zł. Oczywiście 200zł to też sporo – można za to kupić np. niezły multimetr. Na Allegro można kupić PICkit 3 za około 85zł, na Aliexpress – 10$. Obudowa wygląda identycznie, ale zapewne są to „klony” czy po prostu podróbki. Nie miałem z nimi osobiście do czynienia i nie wiem, jak dokładnie przebiegł proces klonowania. Zaletą oryginału jest np. możliwość programowania układów zasilanych napięciem 1,8V. Warto to sprawdzić, kupując kopię. Ja mogę tylko powiedzieć, że oryginał nigdy nie sprawił mi żadnych problemów.

Zaczynając pracę z programatorem, warto przejrzeć dokument „PICkit™ 3 In-Circuit Debugger/Programmer User’s Guide For MPLAB® X IDE”. Poza informacjami o wykorzystaniu narzędzia jest tam też zamieszczony jego pełny schemat. Oczywiście głównym elementem programatora jest mikrokontroler PIC!

Jeżeli programator był wcześniej używany z narzędziem służącym wyłącznie do flashowania PIC-ów, które również nazywa się PICkit 3, nie będzie on prawidłowo pracował ze środowiskiem MPLAB X IDE. Aby to naprawić, należy uruchomić to narzędzie i z menu Tools wybrać opcję *Revert to MPLAB mode*. Wewnętrzny mikrokontroler programatora zostanie przeprogramowany i jest on już gotowy do użycia w MPLAB.

MPLAB X IDE to dość typowe IDE „średniej wielkości”. Co mam na myśli? Otóż na pewno ma o wiele więcej opcji i okienek niż Arduino IDE, ale jest też skromniejsze i nie tak przytłaczające jak narzędzia oparte na Eclipse. Przy odrobinie dobrej woli bardzo łatwo się w nim odnaleźć, a ostatecznie można ograniczyć się do używania go w zakresie zbliżonym do Arduino, tzn. wykorzystując jedynie przyciski odpowiedzialne za zbudowanie projektu i wgrania go do mikrokontrolera. Największa różnica w porównaniu z Arduino to brak gotowych bibliotek do praktycznie każdego zastosowania. Tu programujemy w czystym C i musimy albo sami się postarać, albo… poszukać trochę w Internecie.

Zaczynającym użytkowanie MPLAB X IDE polecam bardzo dobrze napisany dokument MPLAB® XC8 Getting Started Guide, który objaśnia, jak stworzyć swój pierwszy projekt. Nie zamierzam tu przepisywać jego zawartości, ale chcę zwrócić uwagę na dwa szczegóły ważne szczególnie dla osób, które, tak jak ja, używały wersji MPLAB wcześniejszych niż X.

Wcześniej należało w swoich plikach źródłowych dołączać plik htc.h lub pic.h, teraz należy zapamiętać, by swoje źródła rozpoczynać od #include <xc.h>

Tak jak AVR-y mają swoje (nie)sławne fusebity, tak PIC-e też mają bity konfiguracyjne. Sposób ich ustawiania też się trochę zmienił. Obecnie na początku głównego pliku naszego programu powinniśmy zawrzeć linie typu: #pragma config FOSC = INTOSCIO. Ten przykład odpowiada za ustawienie źródła zegara mikrokontrolera na wewnętrzny oscylator. Oczywiście można wszystkie parametry bitów konfiguracyjnych wpisać ręcznie, ale jest to raczej kłopotliwie. Lepiej skorzystać z narzędzia, które niestety nie jest

umieszczone w najbardziej intuicyjnym miejscu. Aby się do niego dostać, należy wybrać w menu Window→PIC Memory Views→Configuration Bits. W narzędziu tym mamy przedstawione wszystkie dostępne bity konfiguracyjne dla wybranego mikrokontrolera i opcje, jakie możemy dla nich ustawić z dość dobrymi opisami. Jeżeli mamy wątpliwości, co wybrać, trzeba oczywiście skorzystać z dokumentacji układu. Gdy już wszystko ustawimy, klikamy przycisk *Generate Source Code to Output* i otrzymujemy zestaw komend # pragma, które wklejamy na początku głównego pliku źródłowego naszego projektu. Wygląd narzędzia przedstawia **rysunek 5**.

UWAGA! Bardzo ważną opcją, którą musimy ustawić dla naszego projektu, jest zasilanie programowanego układu. PICkit 3 może być źródłem zasilania dla układu, który programuje, co w wielu przypadkach może być bardzo wygodne. W naszym przypadku przy złych ustawieniach może być jednak niebezpieczne dla czujnika. PICkit może zasilać programowany układ napięciem do 5V. Jest to powyżej maksymalnej wartości dla Si1147, która wynosi 3,6V. Dlatego lepiej wyłączyć zasilanie z programatora. Wtedy podczas programowania układu bateria CR2032 będzie musiała być włożona do uchwytu, a zworka JP1 zwarta. PICkit 3 automatycznie dostosuje napięcie linii programujących do napięcia zasilania układu – czyli 3V. Aby skonfigurować programator, należy:

- kliknąć prawym klawiszem myszy na nazwie projektu (UVINDEX) i wybrać Properties
- po lewej stronie okna rozwinąć listę Conf: [default] i wybrać poniżej PICkit 3
- po prawej stronie okna z listy rozwijalnej wybrać Power
- odznaczyć opcję „Power target circuit from PICkit3”

Jeżeli jednak chcemy zasilać mikrokontroler z programatora, to w opcji poniżej możemy wybrać prawidłowe napięcie docelowe, czyli 3V. **Rysunek 6** pokazuje właściwą konfigurację programatora.

Oprogramowanie

Oprogramowanie do prezentowanego miernika indeksu UV w całości powstało w języku C. W wielu fragmentach wykorzystałem przykłady lub biblioteki dostępne w Internecie lub dokumentację firmy Silicon Labs.

Najważniejsze bloki oprogramowania to:

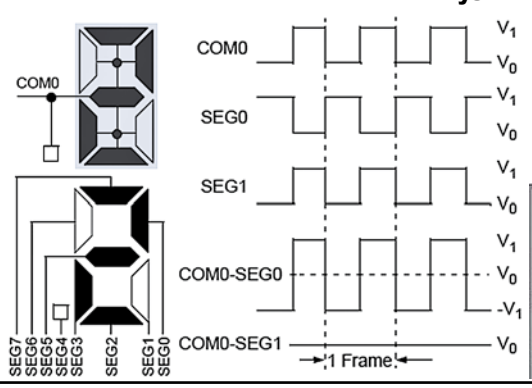
- obsługa wyświetlacza LCD
- obsługa komunikacji I2C
- obsługa protokołu komunikacyjnego czujnika

Wyświetlacz LCD1 ma wyprowadzenie wspólne (ang. backplane) dostępne na nóżkach 1 oraz 16, a także po jednym wyprowadzeniu dla każdego segmentu. W sumie segmentów jest 16: dwie cyfry siedmiosegmentowe oraz dwie kropki. Każdy segment jest połączony z pinem mikrokontrolera. Liczba połączeń jest znaczna w porównaniu z wyświetlaczami LCD z wbudowanym sterownikiem (tam mogą to być nawet tylko dwa piny w przypadku wykorzystania magistrali I2C). Gdyby dla większych wyświetlaczy bez sterownika chcieć stosować sterowanie statyczne, liczba wymaganych pinów mikrokontrolera byłaby ogromna. Dlatego wyświetlacze, mając więcej segmentów (np. 4-cyfrowe), mają większą liczbę wyprowadzeń wspólnych i sterowanie multipleksowane. W wyświetlaczu dwucyfrowym stosujemy sterowanie statyczne, co zdecydowanie upraszcza jego obsługę. Aby włączyć dany segment wyświetlacza LCD, należy podać napięcie pomiędzy jego pin a wyprowadzenie wspólne. Jednak nie może to być napięcie stałe, które mogłoby uszkodzić wyświetlacz. Ma to być przebieg prostokątny. W mierniku jego częstotliwość to ok. 100 Hz. **Rysunek 7** (z dokumentacji mikrokontrolera) przedstawia przykład sterowania wyświetlacza: przebieg prostokątny podany na segment numer 1 jest zgodny w fazie z tym podanym na wyprowadzenie wspólne – napięcie na segmencie jest równe 0V i jest on wyłączony, ciekłe kryształy przepuszczają światło – segment jest niewidoczny. Przebieg podany na segment 0 jest odwrócony w fazie, występuje na nim napięcie i jest on włączony – ciekłe kryształy są tak ustawione, że segment odbija światło. Odpowiednie przebiegi sterujące wyświetlaczem można by wytworzyć za pomocą dowolnego mikrokontrolera, ale byłoby to skomplikowane, wymagałoby zwłaszcza dobrej obsługi przerwań. Szczególnie dlatego, że napięcie stałe podane na wyświetlacz może go uszkodzić. Dlatego w praktyce do obsługi tego typu wyświetlaczy stosuje się mikrokontrolery ze specjalizowanymi peryferiami – takie jak właśnie PIC16F917. W tym przypadku obsługa wyświetlacza sprowadza się do właściwego skonfigurowania peryferii na początku programu, a później zapisu do odpowiednich rejestrów informacji, które segmenty chcemy włączyć. Ponieważ wyświetlacz jest sterowany statycznie, na wyprowadzenie wspólne nie potrzeba podawać napięć pośrednich pomiędzy plusem zasilania a masą. Dlatego wyprowadzenie VLCD3 mikrokontrolera jest podłączone do plusa zasilania, a VLCD2 oraz VLCD1 są niepodłączone. Dla sterowania multipleksowanego należy wykorzystać dzielniki rezystorowe i podać napięcia pośrednie. Szczegóły obsługi wyświetlacza można znaleźć w kodzie źródłowym dostępnym w Elportalu wśród materiałów dodatkowych do tego numeru. Można tam znaleźć również opis i przykład programowy sterowania wyświetlaczem multipleksowanym autorstwa Gaurav Singh ze strony *www.circuitvalley.com*, z którego korzystałem, opracowując program.

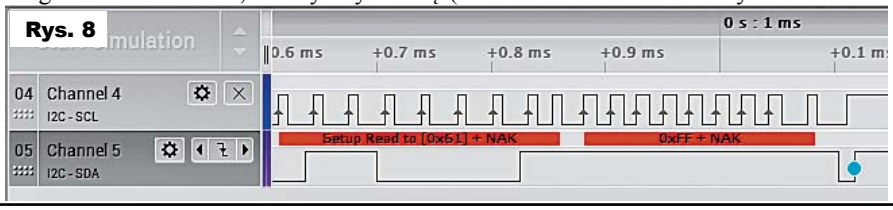
PIC16F917 ma sprzętowy sterownik wyświetlacza LCD, niestety dla magistrali I2C sprzętowo może być zrealizowana wyłącznie obsługa urządzenia podrzędnego (slave). Ponieważ tu mikrokontroler pełni funkcję urządzenia nadrzędnego (master), obsługa magistrali została zrealizowana programowo. Nie stanowi to problemu, ponieważ ilość przesyłanych danych jest minimalna, a mikrokontroler nie wykonuje wielu innych zadań.

Do programowej obsługi I2C chciałem wykorzystać biblioteki znalezione w Internecie. Protokół obsługi czujnika jest dość skomplikowany, dlatego pisząc program krok po kroku i chcąc sprawdzić komunikację I2C, dobrze jest zacząć od czegoś prostszego – najlepiej od odczytu rejestru o znanej wartości. Si1147 ma taki rejestr nazwany PART_ID pod adresem 0x00. Jego zawartość to 0x47, czyli końcówka nazwy układu. Niestety wszelkie moje próby zawsze zwracały 0xFF. Przeanalizowałem kod bibliotek I2C pobranych z Internetu, poprawiłem coś, co uważałem za nieścisłości i nic. Znalazłem inną bibliotekę (autorstwa Thứ Sáu ze strony *tri-iuh.*

Rys. 7



Rys. 8



blogspot.com), której kod wyglądał lepiej – to samo. Przy pracy z interfejsami cyfrowymi nieocenionym narzędziem jest analizator stanów logicznych. Mam dostęp do Saleae Logic, więc z niego skorzystałem. Zobaczyłem, że przebiegi zegara i danych I2C są zgodne z oczekiwaniami, jednak czujnik nie zwracał sygnału potwierdzenia (Acknowledge). Podłączyłem do mojego PIC-a inny układ z magistralą I2C, zegar RTC DS1340C. Tutaj układ zwracał prawidłowe potwierdzenie, a cykliczny odczyt z rejestru sekund – liczbę zwiększająca się od 0 do 59. Czyli biblioteka I2C była prawidłowa. Montaż czujnika jest dość skomplikowany, więc podejrzewałem złe połączenie. Poprawiłem luty – nic nie pomogło. Szukając rozwiązania w Internecie, dowiedziałem się, że nie ja pierwszy mam ten problem, a rozwiązanie jest banalne – adres czujnika podany w jego dokumentacji nie jest zapisany tak jak bym się tego spodziewał i tak jak podają tę informację inni producenci. Adres na magistrali I2C to liczba 8-bitowa, gdzie 7 starszych bitów to właściwy identyfikator, a najmłodszy bit oznacza, czy chcemy z urządzenia czytać (1), czy do niego pisać (0). I zazwyczaj adres jest podany jako taka 8-bitowa liczba z wyzerowanym najmłodszym bitem. Niestety Silicon Labs podało go jako 7 bitów identyfikatora dosunięte do prawej, czyli 0x60. A tak naprawdę adres to 0xC0, czyli 0x60 przesunięte o jeden bit w lewo. Po tej zmianie wszystko zaczęło działać i udało się odczytać upragnione 0x47. Na **rysunku 8** przedstawiony jest zrzut ekranu z narzędzia Saleae Logic pokazujący prawidłowy przebieg I2C, ale z brakiem potwierdzenia od urządzenia podrzędnego.

Zachęcam Czytelników obsługujących interfejsy cyfrowe do korzystania z analizatorów – to świetne narzędzia pozwalające wykluczyć źródła błędów lub potwierdzić prawidłowe działanie kodu.

Obsługę czujnika należy rozpocząć od zapisania pod adres 0x07 (rejestr HW_KEY) wartości 0x17. Bez tej operacji Si1147 nie będzie działał poprawnie. Czujnik może wykonywać pomiary cyklicznie i zgłaszać gotowość nowych wyników za pomocą przerwania. Częstotliwość pomiarów ustawia się za pomocą parametru MEAS_RATE (rejestry 0x08 oraz 0x09). Ja jednak pozostawiam tam domyślną wartość 0, co oznacza, że pomiary są wykonywane tylko na żądanie mikrokontrolera – gdy zapisze on odpowiednie polecenie do rejestru COMMAND. Taki sposób gwarantuje najniższy pobór prądu przez czujnik. Si1147 ma zestaw rejestrów dostępnych z magistrali I2C, ale część parametrów jego pracy jest konfigurowana w pamięci RAM, do której dostęp możliwy jest za pomocą prostego protokołu. Do jego obsługi służą rejestry COMMAND (0x18), PARAM_WR (0x17), PARAM_RD (0x2E) oraz RESPONSE (0x20). Aby zapisać daną w pamięci RAM czujnika, należy umieścić ją w rejestrze PARAM_WR, a następnie do rejestru COMMAND zapisać instrukcję PARAM_SET. Bitowo jest ona reprezentowana jako 101aaaaa, gdzie „aaaaa" to adres w pamięci RAM. W podobny sposób można odczytać wartość z RAMu – wydając komendę PARAM_QUERY; wartość zostanie zwrócona do rejestru PARAM_RD. Natomiast rejestr RESPONSE służy do sprawdzania poprawności wykonania wydanych komend. Jeżeli jego wartość zawiera się w przedziale od 0x00 do 0x0F, to wszystko poszło dobrze. Dostęp do pamięci RAM jest niezbędny, aby skonfigurować, jakie pomiary chcemy wykonywać. Pod adresem 0x01 znajduje się parametr CHLIST, którego poszczególne bity włączają pomiary. Trzeba w nim ustawić bit EN_UV. Należy również zapisać wartość „1" dla bitu VIS_RANGE parametru ALS_VIS_ADC_MISC (adres 0x12) oraz IR_RANGE w ALS_IR_ADC_MISC (0x1F). Te dwa parametry konfigurują czujnik do pracy w silnym świetle słonecznym. Do rejestrów kalibracyjnych UCOEF [0:3] zapisuję domyślne wartości podane w dokumentacji czujnika. Gdy czujnik już jest prawidłowo skonfigurowany, wystarczy cyklicznie zapisywać komendę ALS_FORCE (wartość 0x06) do rejestru COMMAND i 285us później można odczytywać wartość z AUX_DATA[0:1]. Po połączeniu zawartości dwóch rejestrów 8-bitowych w liczbę 16-bitową i podzieleniu jej przez 100 otrzymujemy wartość, która reprezentuje zmierzony indeks UV. Liczbę tę dzielimy na pojedyncze cyfry i wyświetlamy na odpowiednich pozycjach wyświetlacza.

Za miesiąc, w drugiej części artykułu przedstawione zostaną informacje dotyczące montażu i uruchomienia oraz możliwości zmian.

Paweł Hoffmann
pawelhoffmann@gmail.com