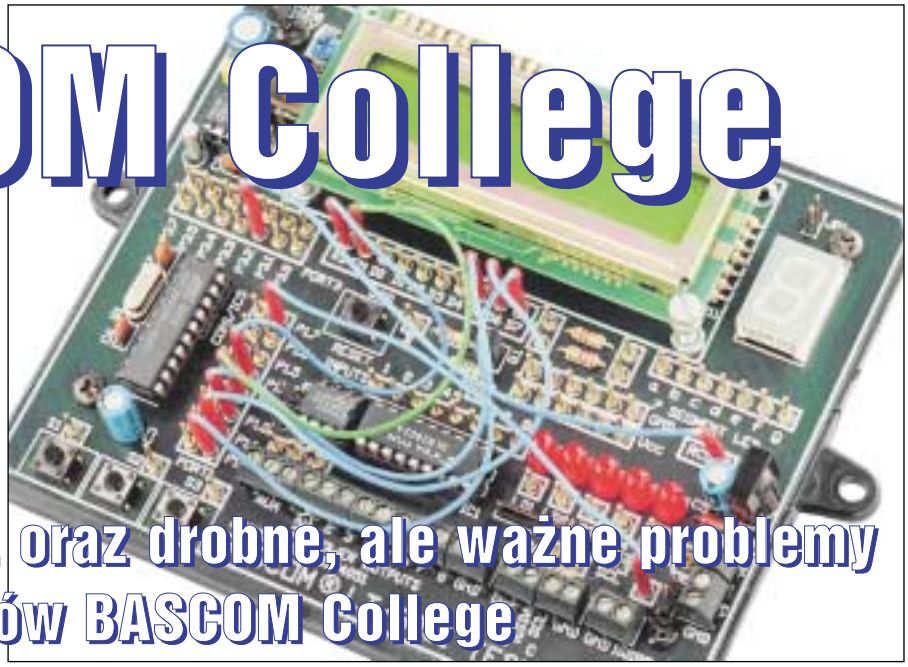
BASCOM College



Wykład 8

Magistrala 1WIRE oraz drobne, ale ważne problemy dręczące Studentów BASCOM College

Na kolejnym wykładzie BASCOM College zajmiemy się jednym z najciekawszych i najbardziej użytecznych elementów systemów mikroprocesorowych, jakim jest opracowana przez firmę DALLAS magistrala 1WIRE. Jest to zagadnienie, które chyba najczęściej wywołuje obawy początkujących programistów i dla wielu z Was z daleka pachnie "czarną magią". Rzeczywiście, porozumiewanie się procesora z układami peryferyjnymi za pomocą zaledwie jednego przewodu (nie licząc masy zasilania z pozoru wydaje się być niesłychane trudne i wymagające wysokich kwalifikacji od programisty. Można nawet zgodzić się z tym poglądem, dobrze pamiętam, jak próbowałem, z mizernym skutkiem, napisać w asemblerze procedurę odczytu temperatury z cyfrowego termometru DS1820. Cóż, wcale nie wstydzę się przyznać, że po pewnym czasie dałem sobie z tym spokój. Zainteresowanie transmisją 1WIRE powróciło dopiero po zapoznaniu się z językiem MCS BASIC, w którym wszystkie potrzebne do obsługi tego termometru, a także innych układów obsługiwanych transmisją 1WIRE czynności sprowadzają się do umiejętnego zastosowania ... trzech poleceń: 1WRESET, 1WREAD i 1WWRITE.

Dlaczego jednak tak zachęcam Was do nauczenia się obsługi magistrali 1WIRE? Głównie dlatego, że otwiera nam ona drogę do stosowania wyjątkowo atrakcyjnych podzespołów, o dostępie do których w przeszłości nie mogliśmy nawet marzyć. Podam Wam tylko jeden, prosty przykład: "żelaznym" tematem pism przeznaczonych dla elektroników jest budowa termometrów, obecnie najczęściej z cyfrową prezentacją wyników pomiaru. Budowa takich urządzeń jest, przy obecnym poziomie elektroniki, banalnie prosta i wysiłki konstruktorów idą już tylko najczęściej w stronę zwiększania liczby czujników służących pomiarowi temeratury. Ja sam mam na sumieniu kilka termometrów, w tym oczywiście także "Termometr cyfrowy z czterema czujnikami". Być może w najbliższym czasie ktoś opracuje termometr obsługujący pięć, sześć, a może nawet aż 10 punktów pomiarowych. Jednak wszystkie takie konstrukcje posiadają oczywiste ograniczenie, którym jest zawsze określona liczba punktów pomiarowych, i cechują się, niekiedy zupełnie niepotrzebną "nadmiarowością" konstrukcji. Po co bowiem mam budować przyrząd z czterema czujnikami, jeżeli będę wykorzystywał co najwyżej trzy z nich?

Zastosowanie magistrali 1WIRE i dołączanych do niej produkowanych przez firmę DALLAS czujników temperatury pozwoli nam na zupełnie inne podejście do zagadnienia. Wykorzystując procesor ’2051, wyświetlacz alfanumeryczny LCD i kilka drobnych elementów będziemy mogli zbudować termometr o N czujnikach, gdzie N jest wartością ograniczoną jedynie ilością wyprodukowanych przez DALLAS tanich i łatwo dostępnych termometrów, np. typu DS1820. Do jednoprzewodowej magistrali 1WIRE możemy bowiem dołączyć całkowicie dowolną ilość elementów z nią współpracujących, oczywiście nie tylko czujników temperatury. Na tej samej magistrali 1WIRE mogą jednocześnie pracować np. przełączniki dwustanowe typu DS2004, pozwalające na sterowanie dowolną liczbą układów wykonawczych.

Z magistralą 1WIRE współpracują także inne, bardzo interesujące elementy. Mam tu na myśli "magiczne" tabletki DALLASa, które mogą posłużyć do skonstruowania zamków elektronicznych nowej generacji. Dla konstruktorów planujących budowę urządzenia wykorzystującego 1WIRE niezwykle ważne jest to, że każdy układ z nią współpracujący ma własny, niepowtarzalny numer identyfikacyjny, mający postać liczby 64-bitowej. A zatem konflikt pomiędzy urządzeniami posiadającymi identyczne adresy nie może się nigdy zdarzyć.

Zanim jednak przejdziemy do szczegółowego opisu magistrali 1 WIIRE, chciałbym załatwić kilka bieżących spraw i pomóc niektórym Studentom w rozwiązaniu trudności, na jakie napotkali podczas eksperymentów z BASCOM-em. Jak zwykle, przeanalizowałem Wasze listy nadesłane pocztą elektroniczną i postarałem się wyłowić z nich te sprawy, o które najczęściej pytacie. I tu od razu ważna uwaga: nie jestem jedynym źródłem porad w sprawach dotyczących BASCOM-a i programowania procesorów ’51. Nie jestem nawet źródłem najlepszym. Do rozlicznych, znanych Wam już zalet BASCOM-a można dodać jeszcze jedną: jest to pakiet programowy posiadający chyba najlepszy serwis ON LINE ze wszystkich znanych mi programów. Jeżeli wyślecie zapytanie na bascom_mailist (www.mcselec.com), to po co najwyżej 24 godzinach otrzymacie odpowiedź nie tylko od Autora, ale także od całej grupy łebskich facetów. Na tej liście wszystko dzieje się zgodnie z zasadą Marka: "Nie ma głupich pytań, są tylko głupie odpowiedzi". Nikt tu nie zostanie wyśmiany, co często zdarza się na polskich listach, każde pytanie zostanie potraktowane z należną powagą. Ogromną przyjemność sprawiło mi pojawianie się na tej liście kilku nazwisk i nicków znanych mi z BASCOM College. Tak trzymać, Moi Drodzy i nie przejmować się niekiedy nieporadną angielszczyzną!

A oto problemy jakie sygnalizowaliście mi najczęściej:

1. Znacząca ilość pytań dotyczyła pamięci szeregowych EEPROM i sposobu ich

programowania poza systemem mikroprocesorowym. Ten temat już poruszaliśmy w jednym z poprzednich wykładów, ale widocznie podane przeze mnie informacje okazały się niewystarczające lub przedstawione w mało czytelny sposób. Z listów mogę wywnioskować, że Waszym zamiarem jest "wspomaganie" niezbyt pojemnej pamięci danych procesorów ’2051 za pomocą zewnętrznej, wstępnie zaprogramowanej pamięci EEPROM. No cóż, rozwiązanie trochę dziwne i w normalnej sytuacji polecałbym raczej zastosowanie procesora ’4051 i umieszczenie danych w tabelach. Jednak obecna sytuacja nie jest całkowicie normalna, ponieważ chipy ’4051 zniknęły z rynku i "mają zamiar" na niego powrócić dopiero w początku przyszłego roku. Dlatego też omówimy raz jeszcze temat programowania i kopiowania szeregowych pamięci EEPROM, ale uczynimy to podczas przerabiania ćwiczeń praktycznych.

2. Zauważyłem, że poważne kłopoty sprawia Wam prezentacja na wyświetlaczach liczb zmiennoprzecinkowych. Otrzymałem aż 12 listów w tej sprawie, co predestynuje ten temat do szerszego omówienia.

Wyświetlanie wyników operacji matematycznych dokonywanych na liczbach zmiennoprzecinkowych jest nieco kłopotliwe, ponieważ przed jakąkolwiek zmianą ich "wyglądu" musimy przekształcić je na zmienne tekstowe typu STRING. Dlaczego jednak tę prostą, wymagającą jedynie zadeklarowania dodatkowej zmiennej, czynność nazwałem kłopotliwą? Powód był prosty: zmienne typu STRING są wyjątkowo "RAM-żerne" i zabierają nam zwykle znaczne obszary pamięci danych, której w procesorach mamy zwykle niewiele. Zmienna typu STRING zajmuje w pamięci RAM tyle bajtów, ile znaków ma zawierać + jeden bajt, będący znacznikiem końca ciągu znaków. A zatem, jeżeli np. liczbę 245 zapiszemy pod postacią zmiennej tekstowej, to zajmie ona w pamięci 4 bajty, podczas kiedy pozostawiona jako zmienna typu BYTE zajęłaby tylko 1 cenny bajt. Największą wartością zmiennej typu LONG jest 2147483647 i zajmuje ona w pamięci 4 bajty. Łatwo policzyć, że jako zmienna tekstowa, przeznaczona do formatowania przed wyświetleniem na ekranie LCD zajmie aż 11 bajtów. Niemniej, niekiedy zmuszeni jesteśmy używać zmiennych STRING, ale zawsze powinniśmy to czynić z rozwagą, tak aby nie zobaczyć na ekranie monitora komunikatu: "Error 8: OUT OF INTERNAL MEMORY"!

Napiszmy sobie teraz krótki programik i prześledźmy jego działanie w symulacji programowej:

```
Config Lcd = 16*1a
Dim X As Single
Dim Z As Single
Dim Y As Single
X = 345988
Z = 278
Y = X / Z
Lcd  "Wynik = " ;Y
```

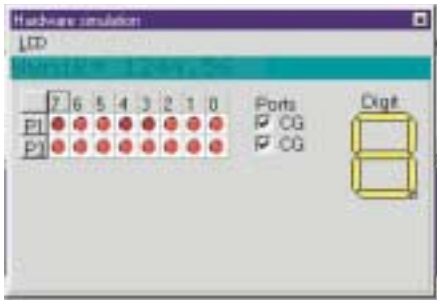
Po uruchomieniu tego programu na ekranie wyświetlacza ukazał się wynik obliczeń: 1244.5610352. Jest to rezultat jak najbardziej prawidłowy, ale korzystanie z niego może być nieco niewygodne. Na wyświetlaczu pozostały bowiem zaledwie ... 2 wolne pola, na których trudno by było umieścić jakikolwiek komentarz dotyczący dokonanych obliczeń. Ponadto, rzadko kiedy interesuje nas aż tak dokładny wynik obliczeń, zaprezentowany z precyzją 7 cyfr po przecinku. A zatem, wyjściem najprostszym będzie przekształcenie wyniku na postać tekstową i użycie funkcji FUSING w celu odpowiedniego sformatowania wyświetlanej liczby. Nasz nowy program będzie wyglądał następująco:

```
Config Lcd = 16 * 1
Dim Result As String * 8
Dim X As Single
Dim Z As Single
Dim Y As Single
X = 345988
Z = 278
Y = X / Z
Result = Fusing(y , ####.##)
Lcd "Wynik= " ; Result
```

Rezultat działania nowego programu został pokazany na **rysunku 1**. Wygląda to już całkiem przyzwoicie, wynik dzielenia został prawidłowo zaokrąglony do dwóch miejsc po przecinku, a na ekranie zmieścił się także odpowiedni komentarz. Omówmy teraz zasady stosowania polecenia FUSING, które w najnowszej edycji BASCOM-a zostało uzupełnione o polecenie FORMAT. Jego podstawowa składnia wygląda następująco:

X = FUSING (Z, ## (liczba miejsc przed przecinkiem). ## (liczba miejsc po przecinku)

Rys. 1



Warto zauważyć, że skutkiem działania tak wydanego polecenia będzie zaokrąglenie wyniku obliczeń do wskazanej liczby miejsc po przecinku. Tak więc liczba 33.59999 zostanie przedstawiona jako "33.60". Takie zaokrąglanie nie zawsze jest pożądane i możemy z niego zrezygnować stosując do oznaczenia liczby miejsc po przecinku znak "&" .

Wynikiem działania programu:

```
Z= 32.59999
X= FUSING (Z, ##.&&)
Print X
```

będzie wyświetlenie na ekranie " 32.59", czyli podanej liczby z dokładnością do 2 miejsc po przecinku, bez zaokrąglania.

Stosując polecenie FUSING mamy do dyspozycji jeszcze jedną funkcję, która w niektórych wypadkach może zwiększyć czytelność prezentowanych na wyświetlaczach wyników obliczeń. Funkcją tą jest dodawanie zer wiodących przed wyświetlaną liczbą. Posłużmy się kolejnym przykładem:

```
Z= 32.59999
X= FUSING (Z, 00##.##)
Print X
```

Po uruchomieniu tego programiku na ekranie ukaże się zadana liczba z dwoma wiodącymi zerami, czyli w postaci: 0032.60.

3. Kolejnym problemem, jaki napotkali Studenci BASCOM College podczas pisania własnych programów było operowanie większą ilością danych, które miały być wykorzystywane jako z góry określone stałe wartości, potrzebne do wykonywania przez program obliczeń matematycznych i wyświetlania komunikatów na ekranie. Otrzymałem kilka listingów programów, których autorzy definiowali dane potrzebne do wykonania pewnych obliczeń jako zmienne, z jednoczesnym nadaniem im konkretnych wartości. Takie działania prowadziły zwykle do bardzo szybkiego "zapchania" pamięci RAM i uniemożliwienia poprawnego funkcjonowania programu. Ponadto, dostęp do takich danych i operowanie nimi jest zwykle bardzo utrudnione.

Do szybkiego i sprawnego operowania danymi wykorzystywanymi przez program, stosowane są następujace instrukcje:

READ, RESTORE i DATA

Wszystkie potrzebne nam dane będziemy mogli starannie uporządkować i poukładać niczym książki na bibliotecznych półkach. Pamiętając, na jakiej półce i w jakiej szafce znajduje się potrzebna nam informacja będziemy mogli z łatwością ją odszukać i wielokrotnie wykorzystywać, zgodnie z aktualnymi potrzebami. Każda z szafek, w której będziemy przechowywać informacje musi mieć własną, niepowtarzalną nazwę i może być umieszczona w dowolnym miejscu programu. Jednak ze względu na konieczną przejrzystość listingu umieszczamy je zwykle na końcu programu. Nazwa grupy danych może składać się z dowolnych znaków dostępnych z klawiatury i musi być zakończona dwukropkiem, podobnie jak etykiety podprogramów. Na poniższym listingu został przedstawiony prosty programik demonstrujący sposób odczytywania danych zawartych w uprzednio zdefiniowanych tabelach. Program ten po skompilowaniu możemy uruchomić w symulacji programowej.

```
'pierwsza część programu, demonstrująca
                    odczyt kolejnych danych z tablicy

Dim X As Byte          'deklaracja zmiennej X
Dim Z As Byte          'deklaracja zmiennej Z
Dim Y As Byte          'deklaracja zmiennej Y

Restore Data1          'przygotuj do odczytu tabelę
 DATA1 z danymi. Odczytywanie danych rozpocznie się
                       teraz od pozycji zerowej
For X = 0 To 10        'dziesięciokrotnie:
```

```
Read Z                    'nadaj zmiennej Z wartość
                          odczytaną z kolejnej pozycji tabeli
Print X ; "  " ; Z        'wyślij na ekran terminala
          numer aktualnego pola tabeli i jego zawartość
Next X
Print                     ' wydrukuj pustą linię

'druga część programu, ukazująca sposób odczytywa
                                    nia wskazanej danej

Do          'początek głównej pętli programowej

Do          'początek pętli pomocniczej
Input "Która wartość? " , Y          'zapytanie o
     podanie kolejnego numeru danej zawartej w tabeli
  If Y < 11 Then                     'jeżeli podano
 istniejący numer (wiadomo, że tabela zawiera jedynie
                                    10 danych), to:
    Exit Do               'wyjdź z pętli pomocniczej
  End If                             'koniec warunku
Loop

Restore Data1             'rozpocznij odczytywanie
               danych z tabeli, począwszy on pozycji 0
  For X = 0 To Y          'odczytuj tabelę  aż do
            osiągnięcia poszukiwanego numeru danej
    Read Z
  Next X
Print : Print "Pozycja " ; Y ; "  Wartość= " ; Z
'wyślij na ekran terminala odczytaną na końcu daną
          (czyli tę poszukiwaną) i jej pozycję w tabeli
Loop

Data1:      'tabela z danymi
Data 1 , 234 , 122 , 37 , 56 , 89 , 70 , 44 , 67 , 88 , 33
```

Powyższy przykład ma charakter studialny i jego zadaniem było jedynie pokazanie możliwości, jakie daje operowanie danymi magazynowanymi nie w pamięci RAM, lecz w treści samego programu. Korzyści wynikające z zastosowania tej metody są oczywiste: nie obciążamy pamięci danych o z zasady niewielkiej pojemności, lecz znacznie obszerniejszą pamięć programu. Pamiętajmy, że pamięć tę zawsze możemy rozszerzyć przez ... zastosowanie innego typu procesora, niekoniecznie bardziej kosztownego. Jednak pamiętajmy, że naszą zasadą jest operowanie wyłącznie przykładami praktycznymi, wziętymi "z życia". Dlatego też przygotowałem dla Was drugi listing, będący fragmentem programu, nad którym kiedyś pracowałem. Program ten miał obsługiwać kolejny zegar, który od innych tego typy konstrukcji miał odróżniać się nietypowym sposobem wyświetlania aktualnego czasu i daty. Otóż zamiast prezentowania informacji w formie liczbowej miałem zamiar zastosować wyświetlanie czasu w formie tekstowej, tj. zamiast "23 : 57" zegar miał wyświetlać "Godzina dwudziesta trzecia pięćdziesiąt siedem". Jak dotąd, projekt tego zegara nie został ukończony, pomimo że napisanie programu realizującego zamierzone funkcje nie okazało się specjalnie skomplikowane. Powód był inny: zegar z wyświetlaniem tekstowym wymaga zastosowania wyświetlacza co najmniej 24*2 znaków, a lepiej 40*2 znaki. Wyświetlacze takie są obecnie bardzo kosztowne, co spowodowało odłożenie realizacji projektu do nie dającej się przewidzieć przyszłości.

Prezentowany fragment listingu jest podprogramem testującym sposób pobierania danych z tablic i składania z nich potrzebnych komunikatów przeznaczonych do wyświetlania na ekranie LCD. W wersji testowej teksty prezentowane są na ekranie monitora, co umożliwia sprawdzanie poprawności pracy programu wyłącznie w symulacji programowej. W wersji finalnej polecenia "PRINT" powinny zostać zastąpione poleceniami "LCD".

Program, pracując w pętli, zadaje pytanie o aktualną godzinę i po uzyskaniu poprawnej odpowiedzi wyświetla otrzymaną informację na ekranie monitora w formie tekstowej. W przypadku podania godziny większej niż 23, program pozwala sobie na niezbyt taktowne powątpiewanie w zdolności umysłowe osoby go obsługującej. Oczywiście, w przypadku budowy zegara program musi nieustannie sprawdzać bieżący czas i po wykryciu zmiany minuty lub godziny aktualizować informacje ukazujące się na wyświetlaczu.

Czytelnicy pozwolą, że ten program pozostawię bez komentarzy, dając tym samym pole do popisu dla Waszej dociekliwości. Rezultaty działania programu konwersji wartości liczbowej na jej odpowiednik tekstowy widoczne są na **rysunku 2.**

```
Dim X As Byte
Dim Z As Byte
Dim Y As String * 13
Const Cstring = "dwu-
dziesta"
Do
Do
Print
Input "Godzina? " , X
If X > 23 Then
Print "Are you OK????"
Exit Do
End If
If X = 20 Then
Print Cstring
Exit Do
End If
If X > 20 Then
Print Cstring ; " ";
X = X - 20
Restore Data_hours
For Z = 0 To X
Read Y
Next Z
Print Y
Exit Do
End If
Restore Data_hours
For Z = 0 To X
Read Y
Next Z
Print Y
Loop
Loop

Data_hours:
Data "zero zero " ,
"pierwsza" , "druga" ,
"trzecia" , "czwarta" ,
"piąta" , "szósta" ,
"siódma" ,
Data "ósma" , "dziewiąta" ,
"dziesiąta" , "jedenasta" ,
"dwunasta" , "trzynasta"
Data "czternasta" ,
"piętnasta" , "szesnasta" ,
"siedemnasta" ,
"osiemnasta",
Data "dziewiętnasta"
```

W tej części wykładu nie wyczerpaliśmy bynajmniej wszystkich tematów związanych ze składowaniem danych w tabelach i posługiwaniem się nimi. Mam jednak nadzieję, że powyższe wskazówki pozwolą Studentom BASCOM College na realizację własnych rozwiązań i pisanie coraz doskonalszych programów w MCS BASIC. Najwyższy więc czas, aby zabrać się za główny temat dzisiejszego wykładu, jakim jest:

Magistrala 1WIRE

Na jednym z poprzednich wykładów BASCOM College omówiliśmy już jeden z najpopularniejszych, jeżeli nie najpopularniejszy sposób wymiany danych pomiędzy elementami systemów mikroprocesorowych: magistralę I²C. Zakres stosowania tego sposobu transmisji informacji jest niezwykle szeroki, ale ogranicza się głównie do układów będących stałymi składnikami systemu i ulokowanych wewnątrz niego lub w małej od niego odległości. Dzisiaj chciałbym pokrótce omówić inny rodzaj przesyłania informacji, potrzebujący do prawidłowego działania zaledwie jednego przewodu (nie liczę tu, oczywiście przewodu masy, wspólnego dla całego systemu).

Rys. 2

Poznanie specyfikacji magistrali 1WIRE i poleceń umożliwiających jej obsługę z poziomu języka MCS BASIC otworzy nam dostęp do wielu, niezwykle atrakcyjnych elementów półprzewodnikowych, produkowanych głównie przez amerykańską firmę DALLAS. Mam tu na myśli przede wszystkim trzy typy takich elementów, wybrane z bogatej oferty DALLAS-a.

1. Cyfrowe termometry i stabilizatory temperatury.

2. Klucze elektroniczne, czyli tzw. tabletki lub pastylki, będące jednym z najskuteczniejszych zabezpieczeń systemów elektronicznych przed ingerencją w ich pracę niepowołanych osób. Układy te umożliwiają budowę zamków elektronicznych, których otwarcie "sposobem" nie jest nawet teoretycznie możliwe.

3. Przełączniki elektroniczne, których na magistrali 1WIRE możemy umieścić dowolną liczbę, a kierować każdym z nich osobna.

Szczerze namawiam Studentów BASCOM College do zapoznania się z obsługą magistrali 1WIRE i wykorzystania zdobytych wiadomości w praktyce. Komunikacja procesora z innymi układami realizowana za pomocą jednoprzewodowej szyny danych wydaje się być zadaniem bardzo skomplikowanym. Rzeczywiście, napisanie programu obsługi 1WIRE w asemblerze nie jest czynnością prostą i nawet dość zaawansowanemu programiście może zająć sporo czasu. Inaczej jest jednak, jeżeli będziemy posługiwać się naszym MCS BASIC. Z poziomu tego języka do obsługi transmisji 1WIRE wystarczają trzy proste polecenia: 1WRESET, 1WREAD i 1WWRITE, a ich umiejętne stosowanie pozwoli na napisanie potrzebnego programu w ciągu dosłownie kilku minut.

Pisząc ten tekst korzystałem z materiałów opracowanych przez moich PT Kolegów, Ryśka Szymaniaka i Piotra Zbysińskiego, opublikowanych na łamach Elektroniki Praktycznej.

Linia 1WIRE składa się z przewodu masy i przewodu sygnałowego. Wszystkie układy dołączone do linii muszą być typu otwarty dren lub kolektor. Poziom wysoki na linii sygnałowej wymuszany jest przez jej połączenie z napięciem zasilającym (+5VDC) poprzez opornik 4,7kΩ. Informacje między

ele-mentami dołączonymi do linii przekazywane są w formie binarnej przy pomocy zer, jedynek, sygnału potwierdzenia i zerowania, które są rozróżniane po czasie trwania. Redukcja połączeń sygnałowych między wieloma układami tylko do dwóch przewodów wymaga, aby wszystkie dołączone urządzenia przestrzegały wspólnego standardu. Ponieważ na dodatek urządzenia mogą czytać informacje i wysyłać je tą samą magistralą, nad przestrzeganiem porządku czuwa sterownik sieci. Zasada jest następująca: wszystkie urządzenia dołączone do wspólnej magistrali po włączeniu napięcia zasilającego albo po odebraniu z sieci impulsu resetu, ustawiane są w stan odbioru. Tylko sterownik sieci ma prawo zainicjować transmisję, inne układy "nie pytane" mogą tylko słuchać informacji przesyłanych magistralą. Stanem aktywnym linii sygnałowej jest poziom niski. Wynika to z faktu dołączenia do tej linii układów z otwartym drenem, które mogą jedynie zewrzeć ją do masy.

Jeżeli sterownik sieci chce wysłać bit informacji, który jest jedynką, zwiera do masy linię sygnałową na czas dłuższy od 15µs, a krótszy od 60µs. Dla bitu będącego zerem czas ten zawiera się między 60µs a 120µs, natomiast każdy poziom niski linii sygnałowej trwający dłużej od 480µs oznacza dla wszystkich dołączonych urządzeń polecenie resetu. Tak więc sterownik w każdej chwili może przerwać wszelką transmisję, wygenerować impuls resetu i wprowadzić wszystkie dołączone do magistrali układy w stan odbioru. Po resecie jedynym zadaniem układów dołączonych do magistrali jest zasygnalizowanie swojej obecności, czyli odpowiedź impulsem Presence. Impuls ten to oczywiście także stan niski trwający od 60µs do 240µs, który powinien się rozpocząć 15-60µs po zakończeniu resetu. Ponieważ wszystkie dołączone do sieci układy generują go niemal jednocześnie, sterownik nie potrafi rozróżnić ile i jakie układy odbierają sygnały. Wie jednak, że ktoś go słucha. Brak impulsu Presence oznaczać będzie albo awarię magistrali, albo niedołączenie do niej żadnego układu 1WIRE.

Z poziomu języka MCS BASIC inicjalizacja magistrali 1WIRE odbywa się po wydaniu polecenia:

1WRESET

bez jakichkolwiek parametrów dodatkowych.

W przypadku gdy sterownik zamierza odebrać dane z układu dołączonego do magistrali, także sterownik powinien zainicjować odbiór. Robi to w taki sposób, że przez czas dłuższy od 1µs a krótszy od 15µs zwiera linię sygnałową do masy, a potem bada jej stan. Układ transmitujący do sterownika bit danych podtrzymuje stan niski linii na czas równy jedynce lub zeru, zależnie od tego, jaki bit wysyła. Powtarza się to bit za bitem aż do zakończenia przesyłania oczekiwanej przez sterownik informacji. Poza wysyła-niem zer i jedynek układy podrzędne, oprócz sterownika, nie potrafią wyresetować linii, czyli wysłać na nią sygnału o czasie dłuższym niż 480µs.

Procesem adresowania konkretnego elementu dołączonego do sieci steruje zestaw rozkazów. Rozkazy są jednobajtowymi liczbami, które nadzorca wysyła magistralą po impulsie reset. Dla łatwiejszego zrozumienia zasady działania urządzenia, krótko opisane zostaną najważniejsze rozkazy, na jakie reagują układy współpracujące z magistralą 1WIRE. Wszystkie rozkazy wysyłane są na magistralę 1WIRE za pomocą polecenia:

1WWRITE, [bajt do wysłania]

READ ROM (33h) odczyt etykiety - numeru seryjnego układu, numeru rodziny układów i sumy kontrolnej CRC. Wydanie polecenia:

1WWRITE, 33H

spowoduje, że następne 64 bity, czyli osiem bajtów odczytywane przez sterownik będą zawierały kod rodziny, 6-cyfrowy unikatowy numer elementu i kod kontrolny etykiety CRC. Rozkaz ten można stosować, gdy do linii dołączony jest tylko jeden układ współpracujący. W przypadku obecności na linii większej liczby układów, ich odpowiedzi będą zakłócały się nawzajem. Jest to bardzo ważne polecenie, ponieważ pozwala na ustalenie adresów wszystkich układów dołączonych do magistrali 1WIRE i wykorzystywanie ich w przyszłości.

MATCH ROM (55h) czyli wstęp do podania na linię adresu konkretnego układu

Wydanie polecenia:

1WWRITE, 55H

pozwala wybrać konkretny układ, dołączony do linii danych. Po bajcie o wartości 55H procesor wysyła 64-bitową etykietę konkretnego układu 1WIRE (kod rodziny, numer, CRC, czyli w sumie 8 bajtów).

Wiecie co, Moi Drodzy? Ta forma rozmowy na temat magistrali 1WIRE, jaką w tej chwili stosujemy zdecydowanie przestaje mi odpowiadać. Wiemy już najważniejsze rzeczy dotyczące realizacji tej transmisji i najwyższa pora, aby rozpocząć konkretne działania w laboratorium, podczas których przyswoimy sobie w całkowicie "bezbolesny" sposób pozostałę część informacji teoretycznych. Wspomnimy tylko jeszcze o jednej sprawie: zasilaniu układów pracujących na magistrali 1WIRE i przejdziemy do laboratorium, w którym zbudujemy kilka ciekawych urządzeń testowych wykorzystujących "magiczne" układy DALLAS-a.

Z całą pewnością wielu z Was zadawało już sobie pytanie dotyczące sposobu zasilania układów pracujących na magistrali 1WIRE. Do dyspozycji mamy przecież tylko dwa przewody: sygnałowy i masy, a zatem w jaki sposób doprowadzane jest napięcie zasilania do układów peryferyjnych, szczególnie tych wyposażonych jedynie w dwa wyprowadzenia? Odpowiedź na to pytanie jest prosta: do zasilania układów współpracujących wykorzystywany jest także przewód sygnałowy, w czasie kiedy nie są nim przekazywane jakiekolwiek informacje. Taki sposób zasilania nosi nazwę zasilania pasożytniczego (ang. Parasite Power). Układy produkowane przez DALLAS pobierają znikomy prąd i to w dodatku tylko w momentach, kiedy zostały uaktywnione odpowiednim poleceniem wydanym przez procesor. Tak więc, przez cały pozostały czas, z wyjątkiem krótkich chwil, w których prowadzona jest wymiana danych, napięcie zasilania podawane jest do układów za pośrednictwem rezystora o wartości typowo 4,7Ωk. Ładuje ono miniaturowe kondensatory wbudowane w strukturę układów i pozwala na ich poprawną pracę w trakcie komunikacji z procesorem.

Na tym kończymy dzisiejszy wykład, a za miesiąc zapraszam Studentów BASCOM College do laboratorium, w którym zajmiemy się wyjątkowo interesującymi ćwiczeniami.

Zbigniew Raabe
e-mail: **zbigniew.raabe@edw.com.pl**