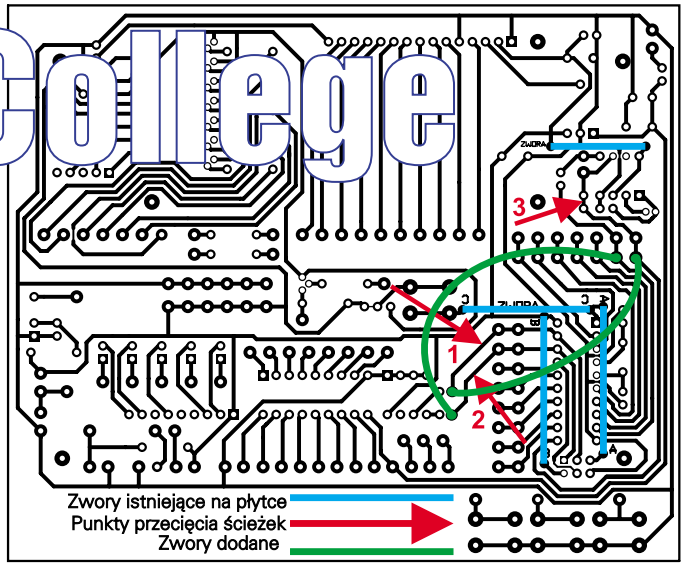
BASCOM College



Rys. 1

Ćwiczenie 6

Zegar z magistralą I²C

Dzisiaj rozpoczniemy ćwiczenia w nieco nietypowy sposób: musimy trochę przerobić nasze podstawowe narzędzie pracy, jakim jest płytka testowa AVT-2500. Nie, nie obawiajcie się, nie ma na niej żadnych fatalnych błędów (no może poza jednym malutkim, który zaraz przy okazji naprawimy). Płytkę musimy nieco zmodyfikować z innego powodu: podczas jej projektowania nie wszystko zostało do końca przemyślane i zoptymalizowane. To dość normalne: dopiero czas pokazuje, jakie rozwiązania sprawdziły się w praktyce, a jakie musimy zmienić. Zmiany na płytce będą drobne i proste do przeprowadzenia i polegać będą na przecięciu trzech ścieżek i dolutowaniu od spodu płytki dwóch dodatkowych zworek.

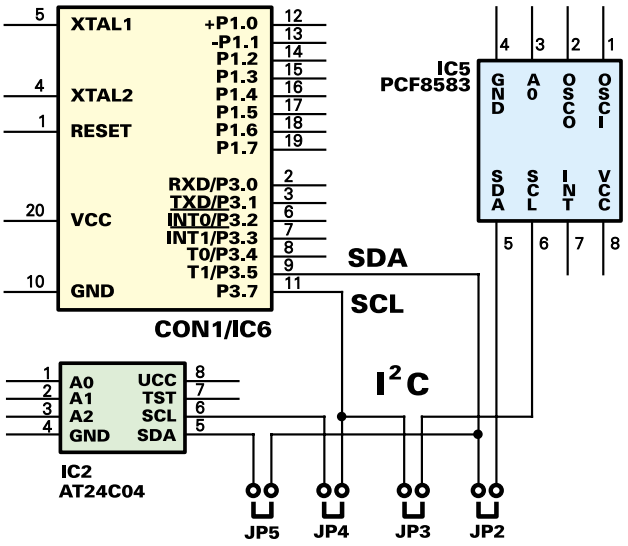
Popatrzcie na **rysunek 1**, na którym została pokazana **od strony lutowania** mozaika ścieżek naszej płytki testowej. Kolorem niebieskim zaznaczone są zworki już istniejące na płytce, a kolorem zielonym dwa dodatkowe połączenia, które za chwilę wykonamy. Natomiast czerwone strzałki wskazują miejsca, w których musimy przeciąć ścieżki. Przecięcia w punktach oznaczonych jako 1 i 2 umożliwią wykonanie nowych połączeń, a przecięcie ścieżki w punkcie 3 pozwoli przy okazji naprawić drobny błąd, który powstał w procesie produkcyjnym pierwszej serii płytek AVT-2500. Zwarty na stałe został na niej jumper JP4, uniemożliwiając w ten sposób odłączenie wyprowadzenia SCL układu IC5 od magistrali I²C.

Po przecięciu ścieżek musimy jeszcze wykonać dodatkowe połączenia, zaznaczone na rysunku kolorem zielonym. Połączenia wykonujemy odcinkami kynaru lub zwykłego przewodu w izolacji, lutując je do punktów zaznaczonych na rysunku zielonymi kropkami. Po sprawdzeniu poprawności wykonania przeróbek, wkładamy płytkę z powrotem do obudowy.

O co właściwie chodziło i co osiągnęliśmy w wyniku dokonania tych prostych przeróbek? Popatrzcie na **rysunek 2**, na którym został pokazany schemat przerobionego fragmentu płytki. Przed przeróbką dwa układy współpracujące z magistralą I²C zegar czasu rzeczywistego IC5 i pamięć szeregowa EEPROM IC2 mogły być dołączane za pomocą jumperów do różnych wyprowadzeń procesora. Takie rozwiązanie, aczkolwiek możliwe do zrealizowania na drodze programowej, nie miało najmniejszego sensu, a nawet zaprzeczało idei magistrali I²C. Po przeróbce do transmisji danych I²C potrzebne będą tylko dwa wyprowadzenia procesora: P3.5 i P3.7. Bardzo Was przepraszam, że musieliście wykonać dodatkową pracę, ale nie wszystko przemyślałem do końca podczas projektowania płytki.

Istnieje jeszcze jedna możliwość modyfikacji naszej płytki, o której można powiedzieć że jest "nadobowiązkowa" i dotyczy sposobu zasilania "bohatera" naszego dzisiejszego ćwiczenia: zegara czasu rzeczywistego PCF8583. Układ ten zasilany jest na płytce zupełnie prawidłowo: z szyny zasilającej +5VDC i będzie funkcjonował zupełnie poprawnie. Jednak po wyłączeniu zasilania układ ten "straci pamięć" i po powtórnym uruchomieniu rozpocznie zliczanie czasu od początku, czyli od ustawień zerowych. Czytelnicy, którzy mają zamiar wykonywać długotrwałe eksperymenty z RTC lub nawet chwilowo wykorzystywać płytkę testową w roli domowego zegara, mogą dokonać jeszcze jednej zmiany. Układ PCF8574 do poprawnej pracy wymaga podania napięcia zasilania nie mniejszego niż +5VDC. Jednak nawet po obniżeniu tego napięcia do 1V układ nie przestaje zliczać upływającego czasu, traci jedynie możliwość kontaktu z procesorem, pobierając za to prąd rzędu kilku mikroamperów. Wynika z tego, że warto dodać na naszej płytce dodatkowe źródło zasilania awaryjnego dla układu PCF8683, którym może być dowolnie mała bateryjka 1,5V. Na rysunku 2b został pokazany schemat zmian, jakie musimy wykonać na

Rys. 2



naszej płytce testowej, aby zapewnić nieprzerwane odliczanie czasu przez zegar RTC. Dodatkowe połączenia i elementy zostały zaznaczone kolorem czerwonym. Zmiany sprowadzają się do dodania do układu dwóch dowolnych diod, np. 1N4148 i bateryjki. Jako baterię proponowałbym zastosować miniaturowe ogniwko typu AAA, umieszczone pod wyświetlaczem LCD.

Jeszcze jedna, ostatnia prośba. Obojętne, czy dokonaliśmy zmiany w systemie zasilania zegara czasu rzeczywistego, czy nie, już teraz - zanim zaczniemy czytać dalszą część tego artykułu - włączmy zasilanie naszej płytki testowej.

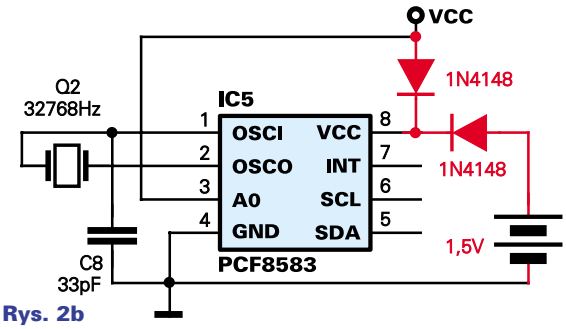
Ogromnym ułatwieniem w wykonywaniu dzisiejszych ćwiczeń będzie fakt, że wszystkie je możemy wykonać wyłącznie w emulacji sprzętowej, bez konieczności programowania procesora. Jest to jedna z zalet transmisji I^{2}C: szybkość transmisji nie jest w żaden sposób ograniczona "od dołu" i narzucana jest przez urządzenie nadrzędne, do którego "niewolnicy" muszą się bezwzględnie dostosować. W naszym przypadku urządzeniem nadrzędnym będzie komputer, który za pośrednictwem programu BASCOM i układu emulatora sprzętowego "udaje", że jest zaprogramowanym procesorem.

Na naszej płytce testowej umieszczone są dwa układy mogące współpracować z magistralą I^{2}C: zegar czasu rzeczywistego PCF8583 i pamięć szeregowa PCF8582 (AT24C04). Wielu z Was ma także do dyspozycji układ klawiatury szesnastkowej, także sterowany "I kwadratem".

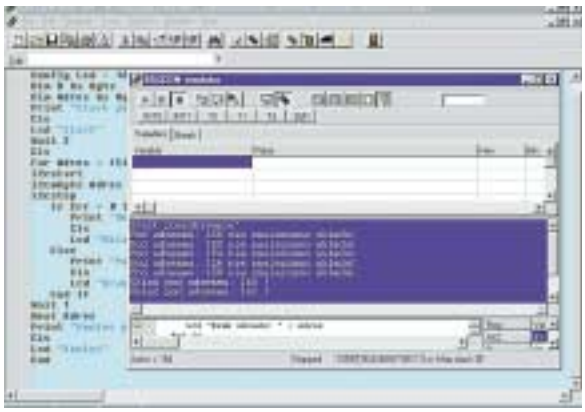
Cóż jednak z tego, że mamy te układy umieszczone na płytce, jeżeli nie wiemy o nich najważniejszego: jakie są ich adresy? Z wykładu wiemy, że aby "dobrać się" do jakiegokolwiek układu współpracującego z I^{2}C, musimy znać jego adres. A zatem musimy chyba zajrzeć do kart katalogowych .... nie, mamy też inne wyjście. Pamiętajmy o naszej zasadzie: praktyka, praktyka i jeszcze raz praktyka i spróbujmy doświadczalnie ustalić adresy obydwóch wymienionych układów. Napiszmy sobie krótki program i po skompilowaniu uruchommy w emulacji sprzętowej.

```
$sim
Config Lcd = 16 * 1a                    'konfiguracja wyświetlacza LCD
Dim Adres As Byte                       'deklaracja zmiennej ADRES
Print "Start poszukiwania"              'wyświetlenie komunikatu wstępnego
Cls                                     'wyczyść ekran wyświetlacza
Lcd "Start"                             'komunikat wysyłany na LCD
Cls                                     'wyczyść ekran wyświetlacza
For Adres = 150 To 200 Step 2           'przeszukaj 50 adresów
I2cstart                                'inicjalizacja magistrali I2C
I2cwbyte Adres                          'wyślij na magistralę zapytanie
                                                po kolejny adres
I2cstop                                 'koniec transmisji
        If Err = 0 Then                 'jeżeli wywoływane urządzenie
                                                odpowiedziało, to:
                Print "Układ pod adresem: " ; Adres ; " !" 'wydrukuj
                        komunikat o znalezieniu układu pod danym adresem
                Cls     'wyczyść ekran wyświetlacza
                Lcd "Układ pod: " ; Adres       'komunikat o znalezieniu
                                                        układu pod danym adresem
        Else
                Print "Pod adresem: " ; Adres ; " nie znaleziono układu!"
                Cls
                Lcd "Brak układu: " ; Adres
        End If
Wait 1
Next Adres
Print "Koniec przeszukiwania"
Cls
Lcd "Koniec"
End
```

Program testujący magistralę I^{2}C wysyła komunikaty jednocześnie na wyświetlacz LCD umieszczony na płytce testowej oraz do emulatora terminala w pakiecie BASCOM.



Rys. 2b



Rys. 3

W efekcie działania tego prostego programu uzyskaliśmy informację (patrz **rysunek 3**), że w naszym systemie znajdują się dwa układy mogące współpracować z magistralą I^{2}C i że ich adresy bazowe to 160 i 162. Moglibyśmy teraz zidentyfikować każdy z układów przez proste wyjęcie jednego z nich z podstawki, ale szkoda na to czasu. Powiem Wam, że układem o adresie 162 jest PCF8583, a pamięć szeregowa posiada adres o 2 mniejszy.

Od czego więc zaczniemy? O pamięci szeregowej EEPROM mamy już pewne pojęcie, ponieważ używaliśmy jej na poprzedniej lekcji. Zajmijmy się więc zegarem czasu rzeczywistego. Układ PCF8583 jest jednym z najpopularniejszych typów zegarów stosowanych w systemach mikroprocesorowych i komputerowych. Zaprojektowany został już dawno temu przez firmę PHILIPS, ale do tej pory nie ma sobie równych wśród mniej lub bardziej doskonałych jego odpowiedników. Wyjaśnijmy sobie najpierw termin: zegar czasu rzeczywistego RTC (Real Time Clock).

Układ RTC jest przede wszystkim wbudowanym w strukturę układu scalonego szeregiem liczników. Ich zadaniem jest jedynie zliczanie upływającego czasu, począwszy od setnych części sekundy do lat.

Zliczanie czasu nie jest jedyną funkcją pełnioną przez układu RTC. Muszą one także umożliwiać ustawienie stanu początkowego wszystkich liczników. Taka funkcja jest absolutnie niezbędna, ponieważ nie istnieje zegar idealny i wskazania każdego czasomierza musimy niejednokrotnie korygować.

Kolejnymi funkcjami realizowanymi przez układy RTC są rozbudowane układy budzików i timerów. Możemy tu wykorzystywać najróżniejsze kombinacje: alarmy pracujące w trybie codziennym, tygodniowym, miesięcznym lub nawet rocznym.

Układy RTC najróżniejszych typów znajdują się w każdym bez wyjątku komputerze klasy PC, w sprzęcie RTV wyposażonym w funkcje programowania (np. w magnetowidach) i w innych urządzeniach. Bywają też powodem poważnych problemów, o których wiele mówiono nie tak dawno temu. Mam tu na myśli "problem roku 2000", do której to daty RTC starszych komputerów nie były przystosowane i traktowały ją jako rok ... 1900!

Znamy już adres bazowy zegaru PCF 8583, ale jak dobrać się do przechowywanej w nim informacji?

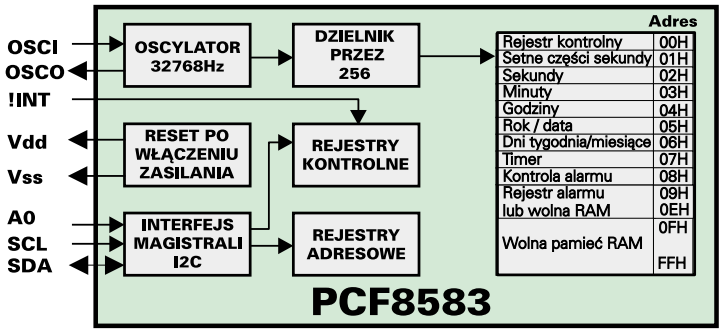
Schemat blokowy kostki PCF8583 został pokazany na **rysunku 4**. Wiemy zatem, pod jakim adresem wewnętrznym mamy szukać potrzebnych nam informacji o aktualnym czasie. Aby np. odczytać aktualną godzinę i minutę, musimy wykonać następujące czynności:

1. Zainicjalizować transmisję I^{2}C;
2. Podać na magistralę adres układu, z którym chcemy nawiązać łączność;

3. Poinformować układ, z którego rejestru mamy zamiar korzystać;
4. Poinformować układ, czy mamy zamiar odczytywać, czy zapisywać dane;
5. Odczytać potrzebne informacje;
6. Ukazać odczytane dane na wyświetlaczu lub emulatorze terminala.

A więc, do roboty! Napiszmy sobie następujący programik, którego zadaniem będzie cykliczne odczytywanie danych z rejestru sekundowego, minutowego i godzinnego układu PCF8583. Po skompilowaniu program uruchamiamy w emulacji sprzętowej.

```
$sim                    'praca w emulacji sprzętowej
Config Sda = P3.5       'konfiguracja linii SDA magistrali I2C
Config Scl = P3.7       'konfiguracja linii SCL magistrali I2C
Dim M As Byte           'deklaracja zmiennej określającej minuty
Dim S As Byte           'deklaracja zmiennej określającej sekundy
Dim H As Byte           'deklaracja zmiennej określającej godziny
Do
   I2cstart             'inicjalizacja magistrali I2C
   I2cwbyte 162         'podanie adresu PCF8583 dla zapisu
   I2cwbyte 2           'wybranie drugiego rejestru (sekund)
   I2cstart             'powtórna inicjalizacja magistrali I2C
   I2cwbyte 163         'podanie adresu PCF8583 dla odczytu danych
   I2crbyte S , Ack     'odczyt sekund z potwierdzeniem transmisji
   I2crbyte M , Ack     'odczyt minut z potwierdzeniem transmisji
   I2crbyte H , Nack    'odczyt godzin bez potwierdzania transmisji
   I2cstop              'koniec transmisji I2C
                        'wyślij na ekran terminala:
    Print "Godzina: " ; H ; " Minuta: " ; M ; " Sekunda: " ; S
Loop
```



Rys. 4

Rezultaty działania naszego programu pokazane są na **rysunku 5**. Niestety, kompletna klapa! Ta godzina 20 to jeszcze by mogła być, ale 87 minut? I na domiar złego jeszcze te 68 sekund! O co tu chodzi, błąd w programie czy też może ci projektanci od Philipsa coś sknocili w układzie? Ponieważ nic mądrego na razie nie możemy wymyślić, to pozostaje nam jeszcze jedna droga ratunku: zajrzeć do karty katalogowej układu PCF8583. No i co się okazuje? Jeszcze raz sprawdza się stara maksyma, że jeżeli wszystkie metody uruchomienia jakiegoś układu zawiodły, to należy ... zajrzeć do instrukcji, w której jak byk jest napisane: "... events are stored in **BCD** format...". A więc wszystko jasne, przed wyświetleniem otrzymanych z RTC danych należy je przekształcić z formatu BCD na normalną, zrozumiałą dla człowieka postać dziesiętną. Tylko jak to zrobić i co to właściwie jest ten format BCD? O kodzie BCD większość z Was z pewnością słyszała i wiecie, że jest to "okrojony" kod szesnastkowy, będący binarną reprezentacją cyfr od 0 do 9. Takim kodem sterowane są między innymi dekodery wyświetlaczy siedmiosegmentowych. Jednak w przypadku naszego zegara sprawa przedstawia się nieco inaczej: kod BCD ma tu postać liczby nie cztero- ale ośmiobitowej i zawiera informacje o wartości dwóch cyfr dziesiętnych, np. dziesiątek i pojedynczych minut. Najprościej wyjaśnić to na przykładzie.

Weźmy na przykład liczbę 57, składającą się z dwóch cyfr: 5 i 7. A zatem binarną reprezentacją dziesiątek w tej liczbie będzie 0101, a jednostek 0111. A zatem w kodzie BCD, stosowanym w naszym zegarze, ta liczba będzie miała postać: 0101 "sklejone" z 0111, czyli 01010111 (BCD) . Prawda, że proste?

Rzeczywiście, zasada przedstawiania liczb dwucyfrowych w kodzie BCD jest dość prosta, ale przeliczenie tak przedstawionych danych na kod dziesiętny z pewnością takim nie będzie. Prawdę mówiąc, nie bardzo wiedziałbym, jak się do tego zabrać "z marszu", ale na szczęście z pomocą przychodzi nam kolejny "fajerwerk" języka MCS BASIC. Aby otrzymać prawidłowe wyświetlanie aktualnego czasu, wystarczy dodać do naszego programu, bezpośrednio przed wysłaniem danych do terminala, tylko jedną linijkę:

```
H = Makedec(h) : M = Makedec(m) : S = Makedec(s)
```

Efektem działania polecenia MAKEDEC jest dokonanie przez program automatycznej konwersji liczb z kodu BCD na czytelną dla człowieka postać dziesiętną. Jak BASCOM to robi? Nie wiem i nie potrzebuję wiedzieć! Wiem tylko, że można to zrobić jeszcze prościej jedynie modyfikując jedną linię programu:

```
Print "Godzina: " ; Bcd(h) ; " Minuta: " ; Bcd(m) ; " Sekunda: " ; Bcd(s)
```

Zatrzymajmy się na chwilę przy poleceniach służących konwersji liczb na różne formaty, ponieważ w przyszłości będą nam one z pewnością bardzo potrzebne.

MAKEBCD [wartość] przetwarza podaną wartość, która musi być liczbą ośmiobitową, na jej reprezentację w kodzie BCD
MAKEDEC [wartość] przetwarza wartość podaną w kodzie BCD na jej postać w kodzie dziesiętnym

Pamiętajmy jednak, że nie należy nigdy wierzyć komuś wyłącznie na słowo, jeżeli mamy empiryczną możliwość weryfikacji głoszonych przez niego prawd. A zatem, piszemy sobie króciutki programik i po skompilowaniu uruchamiamy go w symulacji programowej:

```
Dim X As Word
X = 99
X = Makebcd(x)
Print "134 w kodzie BCD= " ; X
Print X ; " w kodzie BCD po konwersji na postać dziesiętną= ";
X = Makedec(x)
Print X
```

Wynik działania tego programu, pokazany na **rysunku 6,** świadczy dobitnie, że polecenia konwersji kodów działają poprawnie. Wracajmy jednak do naszego zegara, którego budowę dopiero rozpoczęliśmy.

Jak na razie uzyskaliśmy możliwość poprawnego odczytywania rejestrów układu PCF8583 i to tylko tych, w których zapisana jest informacja o aktualnej godzinie, minucie i sekundzie. Z pewnością zwróciliście uwagę na fakt, że wyświetlanie sekund odbywa się dziwnie nieregularnie. Zmiana wartości następuje, nie jak można by było się spodziewać, co sekundę, ale w zależności od szybkości procesora komputera, co kilka sekund. Jest to zjawisko całkowicie normalne i wynikające ze spowolnienia działania programu w emulacji sprzętowej. Gdybyśmy tak napisanym programem zaprogramowali procesor, to zjawisko to ustąpiłoby całkowicie. Nie sądzę jednak, aby było warto programować procesor na tym etapie pracy. Nasz program jest jeszcze "w powijakach" i jak na razie może służyć wyłącznie celom szkoleniowym.

Zobaczmy teraz, "co słychać" w wyższych rejestrach układu RTC, tam, gdzie powinny być zapisane informacje o aktualnej dacie. Do naszego programu dopisujemy następujące linijki:

```
Dim Day As Byte      'deklaracja zmiennej
                     określającej dzień miesiąca
Dim Month As Byte    'deklaracja zmiennej
                     określającej miesiąc roku
```

na początku programu,

```
I2crbyte Day , Ack          'odczyt dni
I2crbyte Month , Nack 'odczyt miesięcy
```

po odczycie czasu

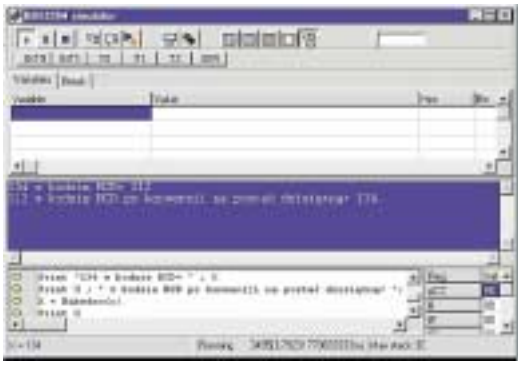
```
Day = Makedec(day) : Month = Makedec(month)
Print "Dzień: " ; Day ; "  Miesiąc:  " ; Month
```

i nie zapominając o zmianie NACK na ACK w linii odczytu godzin:

```
I2crbyte H , Ack      'odczyt godzin
```

Po wykonaniu tych drobnych zmian, powtórnie uruchamiamy nasz program w emulacji sprzętowej. Jak widać na **rysunku 7**, rezultat jest już nieco lepszy: na ekranie mamy już nie tylko pełną informację o aktualnym czasie, ale i o dacie. Tylko że te liczby jakieś dziwne: dlaczego mamy miesiąc 1, dzień 1 i do tego 0 godzin, 40 minut i jakieś tam sekundy? Odpowiedź na to pytanie jest zaskakująco prosta: ponieważ zasilanie płytki testowej zostało wyłączone i ponownie włączone właśnie 40 minut "z hakiem" temu!

Rys. 5



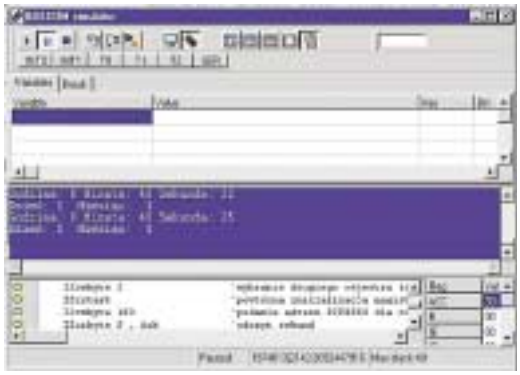
Rys. 6

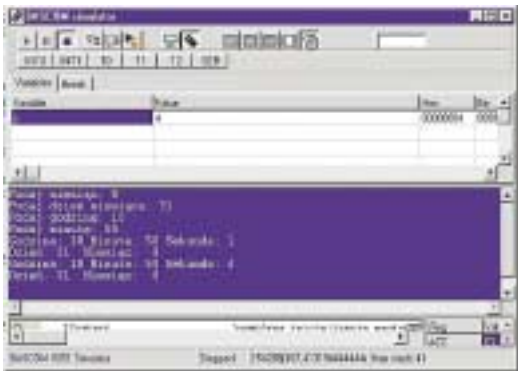
Mamy zatem zegar, ale o "rewelacyjnych" parametrach użytkowych. Potrafi on tylko odliczać czas, jaki upłynął od ostatniego włączenia zasilania płytki testowej. Gdyby jednak takie urządzenie nie zadowalało kogoś z Was (malkontenci zawsze się znajdą), to zapraszam do dalszej części ćwiczenia.

Wiemy, jak odczytywać czas z zegara RTC i najwyższa pora, aby dowiedzieć się, jak można nasz zegarek zsynchronizować z realnym czasem otaczającego nas świata. Okazuje się, że jest to równie proste, jak odczytywanie danych z PCF8583. Napiszmy sobie następujący program, który następnie dodamy do napisanego już wcześniej, zaraz po deklaracji zmiennych:

```
Input "Podaj miesiąc: " , Month
Input "Podaj dzień miesiąca: " , Day
Input "Podaj godzinę: " , H
Input "Podaj minutę: " , M
Month = Makebcd(month) 'konwersja
                miesiąca do formatu BCD
Day = Makebcd(day)  'konwersja
                dnia do formatu BCD
H = Makebcd(h)  'konwersja godziny
                doformatu BCD
M = Makebcd(m) 'konwersja minut
                do formatu BCD
S = 0  'wyzerowanie sekund
  I2cstart  'inicjalizacja magistrali I2C
  I2cwbyte 162  'podanie adresu
                PCF8583 do zapisu
  I2cwbyte 0  'wybranie pierwszego
                (kontrolnego) rejestru
  I2cwbyte 8  'ustawienie zapisu
                rejestru daty
  I2cstop  'koniec transmisji
  I2cstart  'inicjalizacja magistrali I2C
  I2cwbyte 162 'ustawianie trybu zapisu
  I2cwbyte 2  'wybranie drugiego
                rejestru (sekund)
  I2cwbyte S  'zapis sekund
  I2cwbyte M  'zapis minut
  I2cwbyte H  'zapis godzin
  I2cwbyte Day  'zapis dnia
  I2cwbyte Month  'zapis miesiąca
  I2cstop  'koniec transmisji
```

Rys. 7





Rys. 8

Po skompilowaniu uruchamiamy nasz program, jak zwykle w emulacji sprzętowej. Po starcie program zaczyna wypytywać nas o czas i datę, które podajemy z klawiatury komputera. Następnie dane zostają wprowadzone do rejestrów układu RTC i od tego momentu nasz zegar rozpoczyna prawidłowe zliczanie upływającego czasu.

Powyższe przykłady nie wyczerpują wszystkich możliwości, jakie oferuje sterowany kontrolą I^2C układ PCF8583. Nawet nie wspomnieliśmy o ustawianiu roku, dnia tygodnia, ani o wykorzystywaniu timerów i budzika. Nie możemy jednak poruszyć wszystkich tematów, bo co zostałoby dla Was, do samodzielnej pracy? Sądzę, że zarówno temat lekcji, jak i przerabiane ćwiczenia zachęciły Was do poznawania sposobu obsługi urządzeń współpracujących z magistralą I^2C.

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

Ciąg dalszy ze strony 22

Warto zauważyć, że polecenie I2CSEND nie wymaga wstępnego inicjalizowania magistrali, które wykonywane jest samoczynnie. Niestety, większa uniwersalność tego polecenia okupiona jest pewnym zwiększeniem długości kodu wynikowego.

Poleceniem komplementarnym do I2CSEND jest **I2CRECEIVE**. Składnia tego polecenia i jego możliwości są bardzo podobne do I2CSEND. W najprostszej postaci używamy tego polecenia do odczytywania jednej wartości z urządzenia podporządkowanego:

I2CRECEIVE [adres, wartość]

Jednak składnia polecenia I2CRECEIVE może być znacznie bardziej rozbudowana, a to samo polecenie może służyć zarówno do odbierania danych z magistrali I2C, jak i do ich wysyłania:

I2CRECEIVE [adres, wartość, liczbę bajtów do wysłania, liczbę bajtów do odebrania]

Na przykład:

```
Dim Wartosc(10) As Byte
Wartosc(1) = 1
Wartosc(2) = 3
I2creceive adres urządzenia, Wartosc(), 2, 1
                    'wysłanie na magistralę
        I2C dwóch bajtów i odebranie 'jednego bajtu
Print Wartosc(1)          'wydruk odebranej wartości
```

Na tym możemy zakończyć teoretyczne rozważania na temat magistrali I2C i sposobu jej obsługiwania z poziomu języka MCS BASIC. Zapraszam Was teraz do wykonania kilku ćwiczeń, które dadzą nam rzecz najważniejszą: praktyczną wiedzę o sposobach wykorzystywania magistrali I2C.

Zbigniew Raabe
e-mail: **zbigniew.raabe@edw.com.pl**
Konsultacje: **Sławomir Surowiński**
e-mail: **slawomir.surowinski@ep.com.pl**