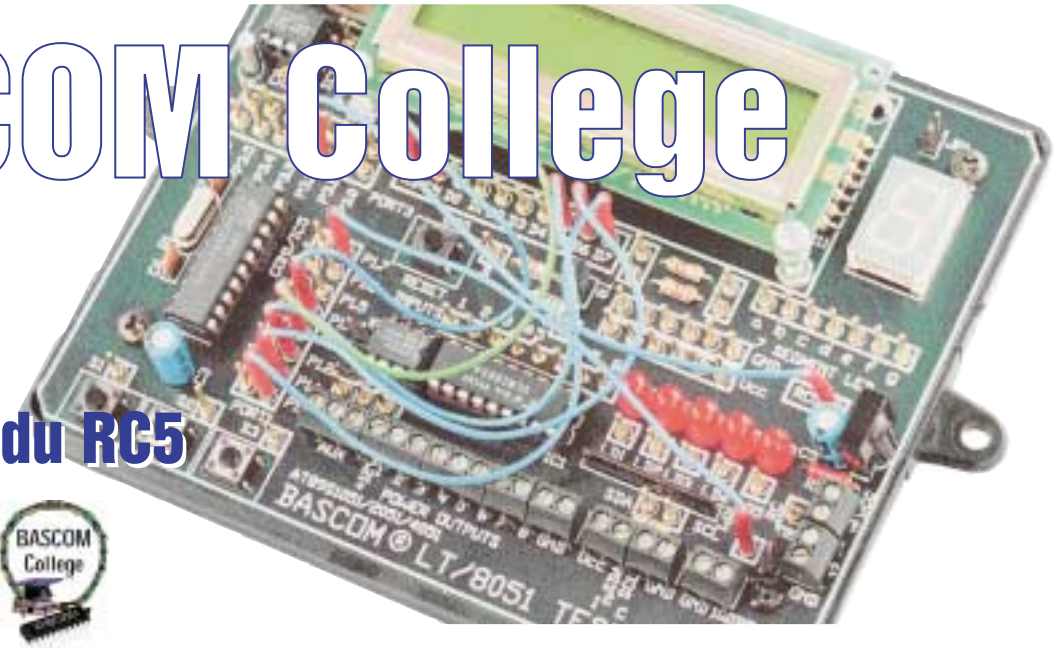
BASCOM College



Analizator kodu RC5

Ćwiczenie 3

Układ, z którego budową i sposobem działania zapoznamy się za chwilę, jest nie tylko ćwiczeniem praktycznym BASCOM College. Jest także pełnowartościowym urządzeniem laboratoryjnym, i to w dodatku takim, które w razie potrzeby możemy zmontować w ciągu kilku minut, a po wykorzystaniu zdemontować i nie zawracać nim sobie więcej głowy. Jest to nowy temat, który w najbliższej przyszłości będzie kontynuowany na łamach Elektroniki dla Wszystkich: **procesor jako uniwersalne narzędzie warsztatowe!** Właśnie tak, Moi Drodzy, w najbliższym czasie dowiecie się, jak w ciągu kilku minut, w awaryjnej sytuacji, możecie zmontować sobie miernik częstotliwości o całkiem przyzwoitych parametrach, układ do pomiaru rezystancji lub pojemności kondensatorów, obrotomierz czy nawet, o zgrozo: nawijarkę do cewek!

Układ jest prostym przyrządem laboratoryjnym, którego zadaniem jest ułatwienie pracy konstruktorom budującym urządzenie wykorzystujące kod RC5. Sądzę, że układ analizatora kodu RC5 może także oddać nieocenione usługi pracownikom serwisu zajmującym się naprawą sprzętu RTV.

Ponieważ sercem urządzenia jest procesor typu AT89C2051, układ jest niezwykle prosty w budowie i niezbyt kosztowny. Taki sam układ wykonany konwencjonalnymi (czytaj: archaicznymi) metodami z wykorzystaniem dekodera RC5 i wspomagających jego pracę układów cyfrowych, byłby nieporównywalnie bardziej skomplikowany i kosztowniejszy. Ponadto, wprowadzenie jakichkolwiek zmian w układzie wykonanym klasycznymi metodami jest najczęściej bardzo kłopotliwe. W przypadku naszego urządzenia i wielu innych, które zbudujemy w najbliższym czasie, wszelkie zmiany sprowadzają się najczęściej do dodania lub zmiany kilku linijek programu i ponownego zaprogramowania procesora. To jest właśnie nasza najbliższa przyszłość, a już teraźniejszość elektroniki: proste i tanie układy, których cała, niekiedy znaczna "inteligencja" skupiona jest w sterującym nimi programie.

Schemat układu został pokazany na **rysunku 1**. Nie ma tu wiele do skomentowania, zastosowałem w układzie tani i popularny procesor typu AT89C2051 i najprostszy wyświetlacz alfanumeryczny LCD mieszczący 16 znaków w pojedynczej linii. Wyświetlacz sterowany jest w trybie czterobitowym, a transmisja danych może odbywać się tylko w kierunku do wyświetlacza. Odbiornikiem kodu RC5 jest dobrze Wam znany układ TFMS5360, którego wyjście dołączone zostało do wejścia INT0/P3.2 procesora.

Częstotliwość pracy generatora zegarowego procesora stabilizowana jest rezonatorem kwarcowym o częstotliwości podstawowej 11,059 MHz, a procesor resetowany jest po włączeniu zasilania za pomocą układu z rezystorem R2 i kondensatorem C1.

Potencjometr montażowy PR1 służy do regulacji kontrastu wyświetlacza, a jumper JP1 pozwala na włączenie podświetlania wyświetlacza (o ile zastosujemy wyświetlacz LCD posiadający taką możliwość).

To wszystko, co można powiedzieć na temat hardware'owej części naszego analizatora. Skupmy się teraz na najważniejszym: programie sterującym jego pracą.

Programowanie

Program, program mikroprocesora, programowanie te słowa coraz częściej pojawiają się na stronach Elektroniki dla Wszystkich. Co jednak tak naprawdę oznaczają? Otóż, **program jest to zbiór instrukcji, których wykonanie przez procesor gwarantuje osiągnięcie celów przewidzianych przez programistę**. Najczęściej takie lub inne zachowanie się programu uzależnione jest od przewidzianych czynników zewnętrznych, takich jak upływ czasu, naciśnięcie klawisza lub odebranie przez procesor danych z układów peryferyjnych. Pojawia się teraz kolejne pytanie: jakie są podstawowe zasady konstruowania programów, obojętne czy komputerowych, czy przeznaczonych do umieszczenia w pamięci stałej procesora?

Bardzo rzadko piszemy program, którego zadaniem jest po prostu wykonanie szeregu instrukcji i zakończenie pracy. Oczywiście, takie programy niekiedy powstają, ale nawet trudno mi w tej chwili przypomnieć sobie jakiś sensowny przykład takiego rozwiązania. Możemy jednak wyobrazić sobie, że procesor po włączeniu zasilania i wyjściu ze stanu RESET wykonuje pewne czynności, a następnie kończy działanie, oczekując na wyłączenie i ponowne włączenie zasilania. Najczęściej jednak program pracuje w niekończącej się **pętli programowej** i zależnie od okoliczności zewnętrznych wykonuje określone przez programistę czynności. Instrukcje opisujące te czynności umieszczone są najczęściej w **podprogramach**, które także mogą mieć postać pętli, z której wyjście uwarunkowane jest zaistnieniem pewnych, określonych zdarzeń. No cóż. Moi Drodzy, z jednej strony mamy nowoczesną technologię, a z drugiej automatycznie nasuwające się skojarzenie z czymś bardzo już archaicznym: z klasycznym mechanizmem zegarowym! Program to właśnie taki mechanizm: jedno kółko napędza drugie, to z kolej zaczepia ząbkami o następne i tak w nieskończoność, aż do rozkręcenia się sprężyny lub w naszym przypadku do wyłączenia zasilania. Natomiast naszym, programistów zadaniem jest nie dopuścić, aby kiedykolwiek ten mechanizm przestał działać, przewidzieć wszystkie

zdarzenia, jakie mogą zajść i zabezpieczyć program przed każdym ziarenkiem piasku, które może wpaść pomiędzy tryby tego mechanizmu i zatrzymać jego ruch. Jest to zadanie bardzo trudne, wiecie przecież, że nie ma programu idealnego i że każdy z nich może się niekiedy zawiesić (nawet słynne z niezawodności WINDOWS98!).

Powiecie, że przecież program można przetestować i usunąć z niego wszystkie pluskwy. Otóż, nie zawsze!

Wszelkie testy mogą tylko stwierdzić istnienie błędu, ale nie mogą go wykluczyć!

No dobrze, postraszyłem Was trochę, ale mam nadzieję że nie ostudziło to Waszego zapału i chęci poznania fascynującego świata techniki mikroprocesorowej. Nie martwcie się, z poważnymi problemami spotkacie się na pewno, niejedną noc przesiedzicie nad "odpluskwianiem" napisanego programu, ale z pewnością nie w najbliższym czasie. Jeszcze jedna wskazówka: czy gracie może w szachy? Jeżeli tak, to dobrze wiecie, jak kończy się partia dla szachisty, który skupia uwagę na jakimś fragmencie szachownicy, zamiast z równą uwagą obserwować całe pole bitwy. Zupełnie tak samo ma się sprawa z pisaniem programu. Musicie pamiętać jednocześnie o wszystkich szczegółach, ponieważ gdzieś w dalekich zakamarkach programu może siedzieć wredna pluskwa i zakłócać jego działanie!

Bierzmy się więc za nasz program, który jak większość mu podobnych rozpoczyna się od ustalenia konfiguracji sprzętowej procesora i zadeklarowania zmiennych, stałych i podprogramów.

Najważniejszym poleceniem używanym w naszym programie jest słynne GETRC5, po którego wydaniu i odebraniu przez układ transmisji z pilota otrzymujemy dwie wartości: adres, pod jaki sygnału pilota został wysłany, i numer komendy, jaką sterowane urządzenie ma wykonać. W standardzie RC5 mogą istnieć 32 różne adresy i 64 komendy. A zatem, obydwie wartości są mniejsze od 255 i mogą być zapisane jako jeden bajt (wartość od 0 do 255). Musimy mieć także jakiś znacznik, świadczący o odebraniu poprawnej transmisji z pilota. Znacznik taki może mieć tylko dwie wartości: prawda i fałsz, czyli logiczna jedynka i logiczne zero. Możemy go zatem zadeklarować jako pojedynczy bit. A zatem piszemy pierwsze trzy linijki naszego programu:

```
DIM Kod as Bit
DIM Command as Byte
DIM Address as Byte
```

Co zostanie przez procesor zrozumiane jako konieczność zarezerwowania w pamięci RAM jednego bajtu na wartość adresu odebranego kodu, jednego bajtu na wartość komendy i jednego bitu na znacznik odebrania transmisji, nazwany "Kod".

Kolejnym krokiem będzie zawiadomienie procesora, a właściwie kompilatora, z jakim rodzajem wyświetlacza alfanumerycznego będzie miał do czynienia. Zastosujemy najtańszy wyświetlacz 16*1 znaków, a o tym fakcie zawiadomimy kompilator za pomocą instrukcji:

```
Config Lcd = 16*1a
```

Tu bardzo ważna uwaga: do określenia tego rodzaju wyświetlacza możemy zastosować zarówno powyższe polecenie, jak i :

```
Config Lcd = 16 * 1
```

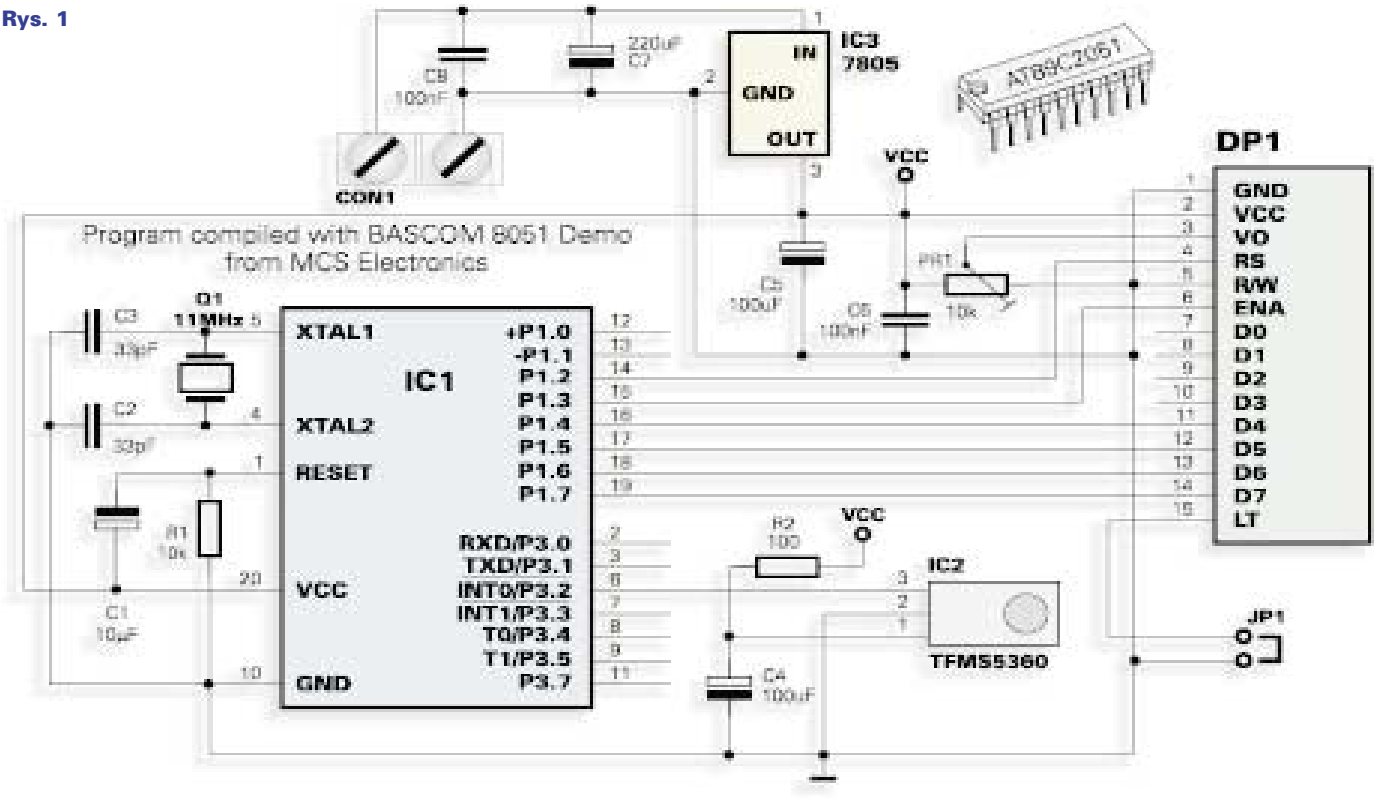
Rodzaj polecenia zależy od typu i producenta wyświetlacza i należy go ustalić doświadczalnie. Jeżeli okaże się, że na wyświetlaczu ukazuje się jedynie 8 pierwszych znaków żądanego tekstu, to należy zastosować parametr "16*1". Jest to jednak przypadek niezwykle rzadki: w 99 przypadkach na 100 stosujemy parametr podany jako pierwszy. Natomiast:

podczas symulacji programowej wyświetlacza 16-znakowego ZAWSZE stosujemy parametr 16*1!!!

Teraz uwaga, przechodzimy do bardzo ważnego fragmentu programu, decydującego o jego prawidłowym działaniu, a którego poprawne napisanie sprawia zwykle wiele kłopotów początkującym programistom.

Skąd właściwie procesor ma wiedzieć, że dołączony do niego układ TFMS5360 zaczął odbierać transmisję z pilota? Popatrzcie na schemat: wyjście układu odbiornika RC5 zostało tam dołączone do wyprowadzenia INT0/P3.2 procesora. Skrót INT0 oznacza wejście INTERRUPT 0, czyli wejście przerwania sprzętowego. Powstanie na tym wejściu stanu niskiego spowoduje zgłoszenie procesorowi sygnału przerwania, które zostanie

Rys. 1



obsłużone w przewidziany w programie sposób, Jest to niezwykle ważna funkcja procesora, a w naszym przypadku decydująca o prawidłowym działaniu napisanego programu.

Wystąpienie sygnału przerwania powoduje tymczasowe zawieszenie przez procesor wykonywania jakichkolwiek czynności i natychmiastowe wykonanie instrukcji związanej z przerwaniem (obsługę przerwania), a następnie powrót do uprzednio wykonywanego zadania.

A więc, piszemy instrukcję obsługi przerwania:

On Int0 Receiverc5

Co oznacza, że w momencie wystąpienia zgłoszenia przerwania, czyli stanu niskiego na wejściu INT0, procesor ma natychmiast przeskoczyć do podprogramu o nazwie Receiverc5 i po jego wykonaniu powrócić do miejsca w programie, w którym wystąpiło przerwanie. Wiemy, że wyjście układu TFMS5360 znajduje się zwykle w stanie wysokim, a stan niski przyjmuje dopiero w momencie odebrania wiązki modulowanej podczerwieni. Tak więc, odebranie sygnału z pilota spowoduje powstanie przerwania sprzętowego.

Podprogramem "Receiverc5", analizującym kod R5 zajmiemy się później, a teraz wydajemy polecenia zezwalające na obsługę przerwań:

Enable Int0

Enable Interrupts

Co oznacza włączenie obsługi przerwań w ogóle, a przerwania INT0 w szczególności. Należy bardzo uważać na te polecenia i anulować je, kiedy tylko przerwania przestaną być potrzebne. Niekontrolowane przerwania mogą narobić większego bigosu, niż używanie instrukcji GOTO (zgiń, przepadnij, siło nieczysta!). Tak więc, wyłączymy obsługę przerwań, kiedy tylko przestanie być potrzebna, a na razie zakończmy wreszcie czynności przygotowawcze i zajmijmy się jeszcze wyświetlaczem. Wydajemy kolejno polecenia:

```
Cls          'inicjalizacja i czyszczenie ekranu wyświetlacza
Lcd "*Waiting for RC5*"
'wyświetlenie komunikatu wstępnego
(nie jest konieczny, ale 'bajerancko wygląda)
Cursor Off   'usunięcie z ekranu kursora (bo "niebajerancko" wygląda)
```

Popatrzcie teraz na **rysunek 2**, na którym w poglądowy, graficzny sposób przedstawiłem działanie naszego programu. Po wykonaniu opisanych wyżej poleceń, program z łoskotem wpada w pułapkę, jaką jest pętla programowa zawarta w klamrach DO .... LOOP.

Instrukcja **DO .... LOOP** jest jedną z najważniejszych w każdym programie napisanym w MCS BASIC. Wprowadza ona program w niekończącą się pętle, z której trudno mu się wydostać (i bardzo mu tak dobrze!). Wszystkie instrukcje zawarte pomiędzy poleceniem **DO** (czyli "rób") a **LOOP** (czyli "pętlę") są cyklicznie wykonywane, a program może wyjść z pętli tylko w przypadku zaistnienia przewidzianego przez programistę wydarzenia. Taki wydarzeniem w naszym przypadku będzie wystąpienie zgłoszenia przerwania. Co jednak właściwie robi nasz program, kręcąc się jak pies za własnym ogonem w pętli programowej? Właściwie tylko jedno: sprawdza, czy znacznik **KOD** nie przyjął wartości "1". Ponieważ wszystkie instrukcje wewnątrz pętli programowej objęte są tym uwarunkowaniem, program nie robi właściwie niczego, wartego uwagi. Zajmijmy się jednak tym uwarunkowaniem, jak jest ono zbudowane?

Rys. 2

Wydanie instrukcji IF [warunek] THEN (jeżeli [warunek] to...) powoduje, że wszystkie następne polecenia, aż do instrukcji END IF, wykonywane są tylko po spełnieniu zadanego warunku.

A więc, program pracuje teraz w pętli, która może być przerwana jedynie po wystąpieniu zgłoszenia przerwania na wejściu INT0 procesora. W tym momencie następuje skok do podprogramu Receivec5, analizującego otrzymany kod, a następnie powrót do pętli. Jak jednak przeprowadzić dość skomplikowaną analizę kodu? Tu właśnie dochodzimy do kolejnego "highlight" programu BASCOM. Analiza kodu pobranego z wyjścia odbiornika IC2 zostanie przeprowadzona po wydaniu polecenia GETRC5, a podprogram " Receiverc5", realizujący tę funkcję i określający adres i numer komendy kodu RC5 wygląda następująco:

```
Receiverc5:
        Getrc5 (Subaddress, Command)
        Kod = 1
        Return
```

Bardzo ważna uwaga: polecenie GETRC5 jest "fabrycznie" przypisane do wejścia INT0 procesora i NIE MOŻE współpracować z jakimkolwiek innym wejściem, nawet jeżeli jest to wejście przerwania sprzętowego. Dekoder podczerwieni TFMS lub SFH MUSI być dołączony do nóżki procesora będącej wejściem przerwania INT0, czyli w przypadku układu 2051 do pinu 6.

Polecenie **GETRC5** analizuje odebrany kod i po stwierdzeniu jego poprawności zapisuje adres urządzenia jako zmienną "Subaddress", a numer komendy jako zmienną "Command." Jak to robi, nie wiem, nie muszę i nie chcę wiedzieć. To właśnie cała przyjemność pisania programu w języku wysokiego (w przypadku BASCOM-a "bardzo wysokiego") poziomu!

Po przeanalizowaniu odebranego kodu podprogram Receiverc5 ustawia wartość znacznika "KOD" na 1 i zgodnie z poleceniem **RETURN (wróć)** powraca do głównej pętli programowej.

Uwaga: Każdy skok do podprogramu, zasygnalizowany przez instrukcję **"GOSUB"** musi mieć swój finał w treści podprogramu, który musi kończyć się instrukcją **RETURN**!

Tym razem jednak program nie kręci się już bezcelowo w pętli. Zauważcie, że spełnione zostało uwarunkowanie **"If Kod =1"** i wszystkie instrukcje leżące pomiędzy **IF** a **END IF** mogą zostać wykonane!

```
Disable Int0  ' chwilowe zawieszenie obsługi przerwań
Cls           'czyszczenie ekranu
Lcd „Com: „ ; Command ; „„Adr: „ ; Subaddress   'wyświetlenie komunikatu
Kod = 0  'ponowne ustawienie flagi na "0"
Enable Int0   'ponowne zezwolenie na obsługę przerwań
```

Powrót do pętli programowej po ustawieniu flagi "KOD" na "1" spowoduje chwilowe wyłączenie obsługi przerwań, wyświetlenie na wyświetlaczu LCD komunikatu o adresie i numerze rozkazu kodu RC5, ponowne włączenie obsługi przerwań, ustawienie flagi na "0" i powrót do stanu oczekiwanie na odebranie kolejnego sygnału.

Cały program realizujący dwie dość skomplikowane funkcje: obsługę wyświetlacza LCD i analizę kodu RC5 zawiera jedynie 23 linie i można go napisać i skompilować dosłownie w ciągu kilku minut.

Po napisaniu programu musimy go zapisać jako plik *.bas, a następnie skompilować. Tę ostatnią czynność BASCOM wykonuje automatycznie po naciśnięciu klawisza F7, a w wyniku kompilacji otrzymujemy jednocześnie pliki w formacie *.BIN i *.HEX.

Zainteresowanym podaję kod wynikowy programu w formacie HEX. Program zajmuje w pamięci procesora 654 bajty, co daje nam swobodny wybór dowolnego typu procesora z podrodziny 'X051.

```
Dim Kod As Bit
Dim Command As Byte , Subaddress As Byte
Config Lcd = 16 * 1 ' lub Config Lcd = 16 * 1a
Reset Tcon.0
On Int0 Receiverc5
Enable Int0
Enable Interrupts
Cls
Lcd "Waiting for RC5"
Cursor Off
Do
  If Kod = 1 Then
      Disable Int0
          Cls
Lcd "Com: " ; Command ; ",Adr: " ; Subaddress
    Kod = 0
    Enable Int0
  End If
Loop

Receiverc5:
  Getrc5(Subaddress , Command)
  Kod = 1
Return
```

```
:10000000020231020C900000000003200000000BC
:1000100000000032000000000000032000000007C
:10002000000000320000000000000320000C0E0CC
:10003000741B1470FDD0E0002274C811451145D521
:10004000E0F9D8F522C0E07419112E1470FBD0E04D
:10005000220521AF21BF0908C0E074C01191D0E092
:10006000D292C293A2E49294A2E59295A2E69296CD
:10007000A2E79297D293C293A2E09294A2E19295C2
:10008000A2E29296A2E39297D293C293112E112EDE
:1000900022C29280CDC292C293C297C296D295D20A
:1000A00094D293C293112ED293C293112ED293C2A3
:1000B00093112EC294D293C293112E74281191746D
:1000C0000E11917406119122740111911457480E1
:1000D0001191752100 22E58104F5F0241EF581A817
:1000E000F0E4040000000001430B2F6F608E40406
:1000F0000000000006007 20B2F6F60880E4A8F0E6F1
:100100000826C313F9C31329FDA8F079017A0DE677
:1001100008C39DE4B33370010869F913EB33FBECBA
:1001200033FCDAEBEB543FFEEB33CC33CC33CC3344
:10013000541FFFE5F014F58122AE02760008DEFBC5
:1001400022780C3139200111C2D574082A14F8E63E
:1001500030E706B2D5780831BC20010F74102A149C
:10016000F8E630E706B2D5781031BC78107908751A
:10017000F008EAA4FBE626F6C0D0BA0104D0D0808D
:100180000BD0D0AE021E08E633F6DEFA780CAE02D3
:10019000E633F608DEFAAE02780CE6970908DEFAD6
:1001A0001978104012790 8C3780CAE02E697F60968
:1001B00008DEF9781019067908DBBA22AE02D3E618
:1001C000F43400F608DEF822C00179105115D00190
:1001D000C00174102A14F8E6C0E0780831397808B4
:1001E000760AE9240FF9E4F719C0013141780CD0FF
:1001F00001E62430F77810AE02E44608DEFC70E831
:10020000D0E020010730E704742D19F7A801D001D0
:100210007A10511E22AE02E6F70809DEFA22E6F74E
:1002200060040809DAF822E660061200510880F737
:10023000227581351195C288D2A8D2AF11C8780B2A
:10024000E88360061200510880F602025D5761697A
:1002500074696E6720666F722052433500740C110A
:10026000917 5F00030050375F0017401C395F070CD
:1002700030202770202C611C8C2A8780BE88360A5
:10028000061200510880F602028E433A20007833AD
:100290007A017922D20131C8C20178225127780B24
:1002A000E883600612005108 80F60202B22C413A3F
:1002B0002000078347A017922D20131C8C201782233
:1002C0005127C205D2A8020261C0D0C2D3C0E0C08B
:1002D000F0C083C082C000C001C002C003C004C01F
:1002E00005C006C007C008C009C00CC00DC010C0C2
:1002F000111200D67834EFF67833EEF6D205D0112D
:10030000D010D00DD00CD009D008D007D006D00521
:10031000D004D003D002D001D000D082D083D0F05E
:10032000D0E0D0D0320000000000000000000004B
:00000001FF
```

Po skompilowaniu programu możemy już przystąpić do programowania procesora, ponieważ przetestowanie programu w symulacji sprzętowej jest absolutnie niemożliwe. Emulator sprzętowy działa zbyt wolno, aby prawidłowo odebrać i zinterpretować kod RC5. Mam nadzieję, że wszyscy moi Czytelnicy zapatrzyli się już w programator procesorów rodziny 89CX051 - MCS Flashprogrammer, wykonany samodzielnie lub zakupiony gotowy z oferty handlowej AVT. Zapoznamy się najpierw ze wszystkimi czynnościami przygotowawczymi, a następnie przystąpimy do programowania pierwszego naszego procesora. Pragnę jeszcze tylko przypomnieć, że programator powinien być dołączony do komputera za pomocą typowego kabla drukarkowego i powinien być zasilany napięciem stałym, niekoniecznie stabilizowanym o wartości 14 ... 16VDC.

Przed rozpoczęciem programowania musimy wykonać następujące, rutynowe czynności:

1. Otworzyć panel "OPTIONS" w podmenu "COMPILER", sprawdzić następujące opcje:

- panel "OUTPUT": nie zmieniać ustawień domyślnych

- panel "COMMUNICATION": opcja "BAUDRATE" jest nieistotna w przypadku programów nie wykorzystujących komunikacji za pomocą portu RS, natomiast w okienku "FREQUENCY" (**rys.3**) powinniśmy ustawić częstotliwość wykorzystywanego rezonatora kwarcowego. Najczęściej będzie to najpopularniejszy kwarc 11.059MHz. **Nieprawidłowe podanie częstotliwości rezonansowej oscylatora kwarcowego może prowadzić do złego działania programu, w szczególności podczas wykonywania polecenia "GETRC5)!**

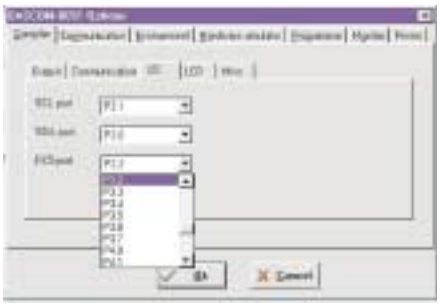
- w panelu "I^2C" (**rys. 4**) powinniśmy zaznaczyć, do którego wyprowadzenia procesora dołączony jest odbiornik kodu RC5 (w przypadku naszego programu jest to wyprowadzenie p3.2).

- w panelu "MISC" należy wybrać typ procesora, który zamierzamy zaprogramować. Najczęściej będzie to typ domyślny - AT89C2051.

- w panelu "PROGRAMMER" (**rys. 5**) powinniśmy przede wszystkim ustawić typ stosowanego programatora. Będzie nim, oczywiście, MCS FLASHPROGRAMMER. Drugą ważną czynnością jest zaznaczenie (lub pominięcie zaznaczenia) "ptaszkiem" okienka "AUTO FLASH". Jeżeli ta opcja zostanie zaznaczona, to programowanie procesora odbędzie się całkowicie automatycznie i jeżeli program nie napotka na jakieś nieprzewidziane przeszkody, to kilka sekund po naciśnięciu przycisku F4 otrzymamy zaprogramowany procesor. Niezaznaczenie opcji "AUTO FLASH" spowoduje, że po naciśnięciu F4 uzyskamy dostęp do okna obsługi programatora i możliwości ręcznego "grzebania" w procesorze, kasowania jego zawartości programu, odczytywania programu zapisanego w procesorze (o ile nie został on zabezpieczony przed odczytem), a także do możliwości zabezpieczania własnego programu. Jednak ręczna obsługa programatora jest na tyle skomplikowana, że poświęcimy temu zagadnieniu osobną lekcję i na razie radzę zaznaczyć okienko "AUTO FLASH" i wykonać programowanie procesora całkowicie automatycznie.

Rys. 3

Rys. 4

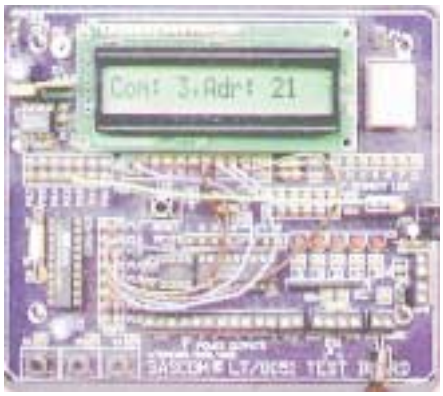


Rys. 5

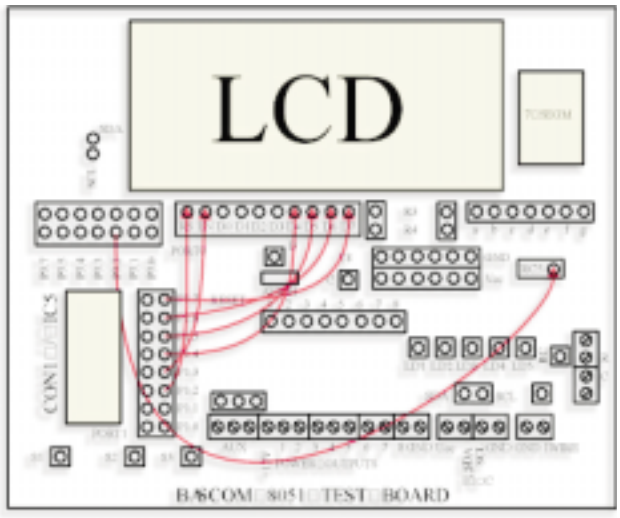
Jeżeli wszystkie powyższe czynności zostały wykonane poprawnie, a program został skompilowany (dla pewności naciskamy jeszcze raz F7), to wkładamy procesor w podstawkę i naciskamy F4 (lub klikamy myszką kolejno "PROGRAM" i "SEND TO CHIP" .

Sygnałem świadczącym o rozpoczęciu programowania procesora będzie "zapalenie się" diody LED w programatorze. Dioda ta będzie świecić z różną intensywnością przez kilka sekund (dłużej w przypadku większych programów), a jej wyłączenie będzie sygnałem, że już można wyjąć (dosłownie "jeszcze ciepły") procesor z podstawki programatora i zainstalować go w naszym analizatorze kodu RC5!

Mamy teraz dwie możliwości do wyboru: albo zmontować układ analizatora na naszej płytce testowej, albo wykorzystać nową płytkę uniwersalną do procesorów ‘X051, która opisana będzie w jednym za najbliższych numerów EdW.



Na **rysunku 6** został pokazany schemat połączeń, jakie musimy wykonać na naszej płytce testowej. Jest to alternatywne rozwiązanie w stosunku do montowania analizatora kodu RC5 jako samodzielnego układu. Rozwiązanie takie należy zastosować, jeżeli nie mamy w najbliższym czasie zamiaru budować wielu układów wykorzystujących kod RC5 i analizator będziemy wykorzystywać jedynie okazjonalnie. Tu właśnie wychodzi na jaw kolejna zaleta naszej płytki testowej: przy jej pomocy możemy w razie potrzeby błyskawicznie zmontować sobie prosty przyrząd pomiarowy, którego budowa jako samodzielnego układu nie miałaby ekonomicznego sensu.



Rys. 6

Technika montażu układów procesorowych na “uniwersalce X051” opisana jest w artykule traktującym o tej płytce.

Na zakończenie tego ćwiczenia mam dla Was ciekawą propozycję. Opanowaliśmy już technikę odbioru sygnałów RC5, które mogą znaleźć zastosowanie nie tylko w sterowaniu sprzętem RTV, ale także i w innych dziedzinach. Czy nie sądzicie, że warto by było napisać sobie procedurę (może jako wstawkę asemblerową?) o odwrotnym działaniu w stosunku do polecenia GETRC5? Mogło to by być np. SENDRC5 [adres, komenda]. Wizja dwóch układów mikroprocesorowych porozumiewających się ze sobą za pomocą łączności na podczerwień wydaje się być bardzo nęcąca! Zastanówcie się nad tym!

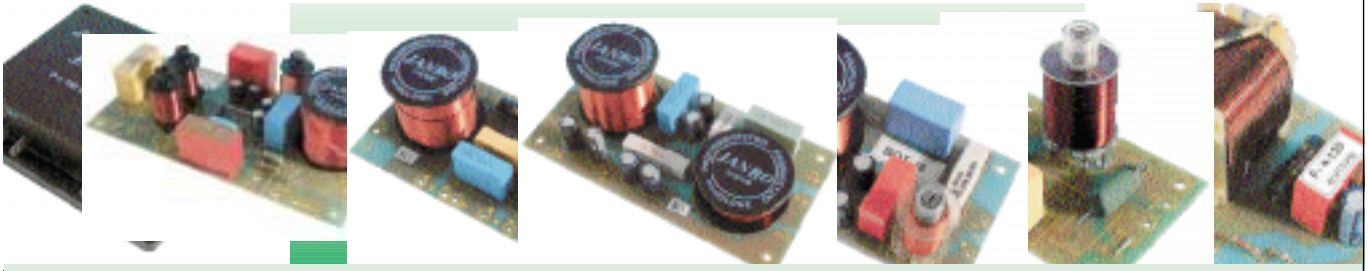
Zbigniew Raabe
e-mail: zbigniew.raabe@edw.com.pl

R E K L A M A · R E K L A M A · R E K L A M A · R E K L A M A · R E K L A M A · R E K L A M A

ZWROTNICE I FILTRY GŁOŚNIKOWE FIRMY JANE

Z OFERTY AVT

KOD	TYP	CZĘSTOTLIWOŚCI PODZIAŁU (Hz)	MOC (VA)	IMPEDANCJE OBCIĄŻEŃ (OHM)	UWA	CENA NETTO
ZWROTN01	B-2	5000	80	2 x 8	Dwudro¿na 12/6 dB/okt.	11.89 z³
ZWROTN02	B-2	5000	80	2 x 8	Dwudro¿na 12/6 dB/okt.	11.89 z³
ZWROTN03	BOT-2	1800	140	2 x 8 lub 2 x 4	Dwudro¿na, polecana przy mocnym g³oœniku wysokotonowym.	31.97 z³
ZWROTN04	P-2-18	2700 lub 4200	220	2 x 8 lub 2 x 4	Dwudro¿na, profesjonalna - 18 dB/okt.	54.10 z³
ZWROTN05	S£-OT-8	800 / 6500	110	3 x 8	Trójdro¿na, odpowiednik fabrycznych zwrotnic typu ALTUS	25.41 z³
ZWROTN06	S£-OT-U	850 / 6600	110	3 x 8 lub 3 x 4	Trójdro¿na, ulepszony ALTUS	31.97 z³
ZWROTN07	BOTES-8	520 / 5400	110	3 x 8	Trójdro¿na, z korekcj¹ fizjologiczn¹ - 4 dB	47.54 z³
ZWROTN08	J-3P/2	1160 / 6000	160	3 x 8 lub 3 x 4	Trójdro¿na, do g³oœników TONSIL - cewka powietrzna	41.80 z³
ZWROTN09	A-BW226	220 / 2350	210	3 x 8 lub 3 x 4	Trójdro¿na, do g³oœników TESLA ARN 226	59.84 z³
ZWROTN10	J-SB-FE	125	120	8 + 8 lub 4 + 4	Odga¾êŸnikowa, do zestawu superbasowego + satelity	63.93 z³
ZWROTN11	J-3AN	700 / 4600	160	3 x 8 lub 3 x 4	Trójdro¿na, z korektorem impedancji (GDN i GDM)	55.74 z³
ZWROTN12	X-3	400 / 5000	300	3 x 8 lub 3 x 4	Trójdro¿na, do zestawów estradowych	78.69 z³
ZWROTN13	Z-F130	130	120	4	Samochodowy fitr basowy - jeden kana³ w obudowie	38.52 z³
FILTR01	F-8	125 lub 400	300	8	Samochodowy filtr basowy do dodatkowego subwoofer'a	40.98 z³
FILTR02	F-4	85,130 lub 400	300	4	Samochodowy filtr basowy do dodatkowego subwoofer'a	40.98 z³
FILTR03	X-1H-PI	5000	160	8 lub 4	Filtr górnoprzepustowy do zestawów z g³oœnikami PIEZO	13.11 z³



Ceny netto bez 22% VAT. Prezentowane artykuły dostępne są w sprzedaży wysyłkowej i sklepach firmowych AVT, bliższe informacje na stronach z ofertą.