

Po dłuższej przerwie, spowodowanej po części wakacjami, zamieszczam dalsze komentarze dotyczące waszych listów, w których poruszacie problematykę opisywaną w serii artykułów o 8051. Dziś kolejna porcja listów oraz dodatkowe sprostowania błędów, które wkraśli się do druku podczas tworzenia artykułów klasy mikroprocesorowej. Przedstawiam również kilka nadesłanych rozwiązań zadania dotyczącego zegara czasu rzeczywistego. Na koniec znajdziecie prawdziwy kłosek dla bardziej zaawansowanych studentów klasy mikroprocesorowej mowa będzie mianowicie o dwóch aplikacjach na komputer edukacyjny, przysłanych do redakcji przez jednego z Czytelników. Jedna z nich pozwoli na programowanie pamięci EEPROM, a druga na programowanie mikrokontrolera 89C2051!



8051 Errare Humanum Est



LIST 1

Zenon Rakoczy z Chropaczowa słusznie zwrócił uwagę na błąd, który wkraśli się w program monitora komputerka edukacyjnego, zawartego w pamięci EPROM. Chodzi bowiem o błędne obliczanie sumy kontrolnej w procedurze monitora „SAVE”, która dostępna jest pod klawiszem „8” klawiatury komputerka.

W wyniku tego plik utworzony w formacie Intel-HEX z zawartością pamięci RAM komputerka jest nieprawidłowy, a w zasadzie nieprawidłowa jest tylko suma, będąca ostatnim bajtem w każdej linii tego zbioru. O formacie Intel-HEX pisałem w zeszłym roku na łamach bratniego pisma - Elektroniki Praktycznej.

W wyniku braku instrukcji

```
CLR    C
```

w procedurze, przed obliczeniem wspomnianej sumy kontrolnej podczas odejmowania (instrukcja SUBB ...) następuje także niezamierzone odjęcie znacznika przeniesienia C. Kiedy znacznik ten jest równy 0, wszystko jest w porządku, jednak kiedy równa się „1” wtedy obliczona suma kontrolna jest mniejsza o 1 od prawidłowej. Stąd po powtórным załadowaniu tak utworzonego zbioru Intel-HEX, do komputerka (komenda „LOAD” monitora), wystąpi komunikat „Err”, co świadczy o błędzie w linii zbioru.

Przyjrzyjmy się temu bliżej. Otóż w programie monitora obliczanie sumy kontrolnej odbywa się następująco:

```
.....
..... ;w zmiennej CRC znajduje się suma (mod 100h)
..... ;wszystkich bajtów rekordu (linii) danych
..... ;teraz nastąpi obliczenie sumy kontrolnej
CLR    A ;wyzierowanie akumulatora
SUBB   A, CRC ;obliczenie sumy wg wzoru: suma = 100h - CRC
..... ;ale ze względu na instrukcję SUBB tak naprawdę
..... ;wykonywane jest działanie: suma = 100h - CRC - C
LCALL  SEND ;i wysłanie sumy przez port szeregowy
```

Jak z tego widać w przypadku gdy znacznik C jest równy „1” suma kontrolna obliczona zostanie nieprawidłowo wg wzoru:

suma = 100h - CRC - 1,

czyli będzie o 1 mniejsza od prawidłowej. Dlatego przed instrukcją odejmowania SUBB należało dodać instrukcję kasującą przypadkowo ustawiony znacznik C.

```
CLR    A
CLR    C ;brakująca instrukcja
SUBB   A, CRC
LCALL  SEND ;wysłanie akumulatora - sumy
```

Prawidłową sumę kontrolną można także uzyskać stosując negację logiczną z inkrementacją otrzymanej sumy bajtów rekordu, oto instrukcje:

```
MOV    A, CRC
CPL    C
INC    A
LCALL  SEND ;wysłanie akumulatora - sumy
```

W dalszej części opisu będę posługiwał się tą metodą obliczenia sumy kontrolnej rekordu danych w zbiorze Intel-HEX.

W związku z tym, że błąd został poprawiony w oprogramowaniu monitora na przełomie maja i czerwca, nabywcy zestawów AVT-2250 z okresu przed tą datą mogą mieć problemy z prawidłowym generowaniem zbiorów HEX przez komputerka. Zenon Rakoczy pyta jak rozwiązać ten problem. Sposobów jest kilka.

ROZWIĄZANIE NR 1

Poniżej zamieszczony jest listing 1 prawidłowej procedury „SAVE”, którą można skompilować z dowolnym przesunięciem względem początku zewnętrznej pamięci operacyjnej komputerka, a następnie używać, ładując ją w razie potrzeby tak jak każdy inny program tworzony przez siebie (uruchamiając ją poleceniem „JUMP”).

Pogrubioną czcionką zaznaczono linie programu, które obliczają ostatni bajt transmitowanego rekordu danych. W tej procedurze rejestr R6 wykorzystywany jest jako zmienna CRC, o której mówiłem wcześniej. W procedurze wykorzystano dwie dodatkowe podprocedury umieszczone w ciele monitora, a mianowicie „NULLKEY” (adres: 02CDh) oraz procedurę RSTEXT (adres 02DEh) zawarte w pamięci EPROM monitora. Zadaniem pierwszej procedury jest oczekiwanie na zwolnienie klawisza klawiatury, druga procedura przesyła ciąg znaków ASCII tzw. „string” poprzez złącze portu szeregowego komputerka, którego adres jest podany w rejestrze DPTR.

Listing 1

```

1      CPU      '8052.def'
2      include 'const.inc'
3      include 'bios.inc'
4      ;*****
5      ;Poprawiona procedura wysylania zawartosci RAM
6      ;komputerka w formacie Intel-HEX
7      ;wersja relokowalna od adresu 8000h (w ext.RAM)
8      ;*****
9      8000      org      8000h
10     ;*****
11     8000      SAVEDATA:
12     8000 75786D      mov     DL1,#_5
13     8003 1202CD      lcall    02CDh          ;adres procedury NULLKEY
14     8006 C2D5        clr      F0              ;flaga dot. konca
15     8008 75F005      mov     B,#5
16     800B 7480        mov     A,#80h          ;#nullkey
17     800D 120295      lcall    DELAY
18
19     8010 1203B9      lcall    GETDPTR
20     8013 C083        push     DPH
21     8015 C082        push     DPL              ;zachowaj na stosie
22     8017 75F005      mov     B,#5
23     801A 1203B9      lcall    GETDPTR
24     801D A3          inc      DPTR
25     801E 858304      mov     04,DPH
26     8021 858205      mov     05,DPL
27     8024 D003        pop      03
28     8026 D002        pop      02
29     8028 7E00        mov     R6,#0              ;CRC=0
30
31     802A 1202C5      qwer:    lcall    CONIN
32     802D B40DFA      cjne     A,#klaw_OK,qwer
33
34     8030 8A83        xnowy:   mov     DPH,R2
35     8032 8B82        mov     DPL,R3
36     8034 75F005      mov     B,#5
37     8037 12025F      lcall    DPTR4HEX          ;wypisanie adresu
38
39     803A 743A        mov     A,#':'
40     803C 1202B9      lcall    OUTRS
41     803F 7F10        mov     R7,#10h          ;16 bajtow danych
42     8041 EF          mov     A,R7
43     8042 FE          mov     R6,A
44     8043 128090      lcall    send              ;CRC = liczba bajtow w rekordzie
45     8046 EA          mov     A,R2              ;wyslanie liczby bajtow
46     8047 128090      lcall    send              ;zaladowanie adresu MSB
47     804A EE          mov     A,R6              ;i wyslanie go
48     804B 2A          add     A,R2
49     804C FE          mov     R6,A
50     804D EB          mov     A,R3              ;CRC = CRC + MSB adresu
51     804E 128090      lcall    send              ;zaladowanie adresu LSB
52     8051 EE          mov     A,R6              ;i wyslanie go
53     8052 2B          add     A,R3
54     8053 FE          mov     R6,A              ;CRC = CRC + LSB adresu
55     8054 E4          clr      A
56     8055 128090      lcall    send              ;wyslanie: 00 - oznacza dane
57     8058 8A83        mov     DPH,R2
58     805A 8B82        mov     DPL,R3
59
60     805C E0          xznów:    movx    A,@DPTR          ;pobranie danej z komorki pamieci RAM
61     805D 128090      lcall    send              ;i wyslanie jej
62     8060 E0          movx    A,@DPTR          ;ponowne pobranie tej danej
63     8061 2E          add     A,R6              ;CRC = CRC + dana
64     8062 FE          mov     R6,A
65     8063 A3          inc      DPTR              ;zwiększenie adresu
66
67     8064 AA83        niewp:    mov     R2,DPH          ;i zapamietanie w R2.R3
68     8066 AB82        mov     R3,DPL
69
70     8068 EA          mov     A,R2
71     8069 6C          xrl     A,R4              ;porownanie MSB adresu pocz. i konca
72     806A 7006        jnz      dlej              ;czy takie same
73     806C EB          mov     A,R3              ;nie to skocz dalej (do etyk.'dlej')
74     806D 6D          xrl     A,R5              ;porownanie LSB adresu pocz. i konca
75     806E 7002        jnz      dlej              ;czy takie same
76     8070 D2D5        setb    F0              ;nie to skocz dalej (do etyk.'dlej')
77     ;flaga ozn. koniec obszaru
78     8072 DFE8        dlej:     djnz    R7,xznów          ;nastepny bajt danej z RAM
79     8074 EE          mov     A,R6              ;zaladowanie CRC
80     8075 F4          cpl      A              ;i obliczenie reszty wg. wzoru
81     8076 04          inc      A              ;reszta = 100h - CRC
82     8077 128090      lcall    send              ;i wyslanie jej
83
84     807A 740D        mov     A,#0Dh          ;wyslanie...
85     807C 1202B9      lcall    OUTRS          ;konca...

```

Listing 1, cd.

```

86 807F 740A      mov     A,#0Ah           ;wiersza, znaki #13#10...
87 8081 1202B9    lcall    OUTRS          ;tzw. eoln
88 8084 30D5A9    jnb     F0,xnowy          ;jesli nie koniec to nowy rekord
89
90 8087 C2D5      clr     F0           ;jesli koniec to...
91 8089 90809E    mov     DPTR,#rs_end        ;wyslanie konca pliku ':00000001FF'
92 808C 1202DE    lcall    02DEh          ;za pomoca dodatkowej procedury monitora
RSTEXT
93 808F 22        ret           ;i koniec podprogramu
94
95 8090 120235    send:    lcall    HEXASCII      ;procedura dodatkowa SEND:
96 8093 C5F0      xch     A,B           ;wysyla bajt jako 2 znaki ASCII
97 8095 1202B9    lcall    OUTRS
98 8098 E5F0      mov     A,B
99 809A 1202B9    lcall    OUTRS
100 809D 22       ret
101
102 809E 3A303030  80A2 30303030  80A6 3146460D  80AA 0A00
                        rs_end    db      ':00000001FF',13,10,0
103                      ;*****
104 80AC          END

Kompilacja zakonczona pomyslnie!
Zbior: „sav8000.s03”, 172 bajt(ow), 0.3 sekund(y).
```

ROZWIĄZANIE NR 2

Innym sposobem ominięcia tej trudności jest własnoręczne zmodyfikowanie programu monitora umieszczonego w pamięci EP-ROM. W tym jednak przypadku niezbędny jest dostęp do programatora pamięci EP-ROM lub posiadanie takiego urządzenia. Kroki, jakie trzeba podjąć w tym przypadku, pozwolą na modyfikację monitora w taki sposób, że naciśnięcie klawisza „8” wywoła potem prawidłową procedurę wysyłania zawartości pamięci RAM poprzez łącze szeregowe komputerka. Oto one.

1. Odczytać programatorem zawartość EP-ROM komputerka 27C64
2. Korzystając z opcji „Edit” programatora zmodyfikować następujące komórki pamięci spod podanych niżej adresów:

Adres:	Jest:	Ma być:
0701h	91 A9	F1 21
0721h	43 71 24	12 09 00

3. Od adresu 0900h załadować do bufora programatora www. kod programu zmodyfikowanej procedury SAVE (listing 1) skompilowany od tego adresu, za pomocą dyrektywy kompilatora

ORG 0900h ;zamiast ORG 8000h pozostawiając wcześniejszy – odczytany z EPROM kod programu monitora, bez zmian. Można także zamiast tego przepisać niżej wymienione dane z adresu 0900h do bufora programatora – listing 2.

Łatwiej jest jednak skorzystać z programu kompilatora PASM51.EXE i skompilować listing do postaci akceptowanej przez program obsługi programatora, czyli np. za pomocą instrukcji:

```
PASM51.EXE <zbiór> /h {Enter}
```

4. Zaprogramować ponownie pamięć EPROM tak zmodyfikowanym kodem monitora (wcześniej należy starą pamięć skasować promieniami ultrafioletowymi, lub posłużyć się inną, czystą kostką 27C64). Można także użyć pamięci EEPROM typu 28C64 (w dalszej części artykułu jeden z czytelników przedstawi prostą przystawkę do komputerka, służącą do programowania tej pamięci).

W ten sposób otrzymamy poprawiony program monitora, w którym procedura zapisu pamięci RAM komputerka SAVE będzie działać prawidłowo, a wywołanie jej będzie odbywać się tak jak poprzednio – za pomocą klawisza „8” klawiatury komputerka.

Listing 2

```

-----
Adres  Dane
-----
0900  75 78 6D0903  12 02 CD C2 D5 75 F0 05 74 80 12 02 95 12 03 B90913
C0 83 C0 82 75 F0 05 12 03 B9 A3 85 83 04 85 820923 05 D0 03 D0 02 7E 00
12 02 C5 B4 0D FA 8A 83 8B0933 82 75 F0 05 12 02 5F 74 3A 12 02 B9 7F 10
EF FE0943 12 09 90 EA 12 09 90 EE 2A FE EB 12 09 90 EE 2B
0953  FE E4 12 09 90 8A 83 8B 82 E0 12 09 90 E0 2E FE
0963  A3 AA 83 AB 82 EA 6C 70 06 EB 6D 70 02 D2 D5 DF
0973  E8 EE F4 04 12 09 90 74 0D 12 02 B9 74 0A 12 02
0983  B9 30 D5 A9 C2 D5 90 09 9E 12 02 DE 22 12 02 35
0993  C5 F0 12 02 B9 E5 F0 12 02 B9 22 3A 30 30 30 30
09A3  30 30 30 31 46 46 0D 0A 00
```

Podziękowania kieruję także w stronę p. Marka Lewandowskiego, od którego otrzymałem pocztą e-mailową informację o przedstawionym tu błędzie.



LIST 2

Zbigniew Wawryń z Wołowa w swoim liście słusznie zauważył pewną nieścisłość w wyjaśnieniach dotyczących obsługi bufora portu szeregowego – rejestru SBUF, o którym pisałem w kwietniowym numerze EdW na stronie 35 (górna prawa szpalta). Otóż stwierdzenie, że „... rejestr SBUF można adresować także za pomocą metody pośredniej (poprzez rejestr wskaźnikowy @Ri)...” jest oczywiście błędne, bowiem instrukcje operujące na tym wskaźniku z adresem powyżej 7Fh będą oczywiście operowały na górnej części pamięci RAM procesora, dostępnej tylko w kostkach 80C52 i pochodnych (87C52). Jak zapewne przypominacie sobie, w procesorach tych, w odróżnieniu od 80C51, znajduje się dodatkowe 128 bajtów wewnętrznej pamięci RAM umieszczonych pod adresami 80h...FFh. Dostęp do tych komórek jest jednak możliwy tylko za pomocą właśnie adresowania bezpośredniego, czyli poprzez rejestry Ro i R1 przy użyciu instrukcji np.

```
MOV R1, #90h      ; odczytaj z komórki o adresie 90h
MOV A,@R1         ; daną i umieść w akumulatorze
```

W przypadku chęci zaadresowania tej komórki za pomocą instrukcji np.

```
MOV A, 90h
```

do akumulatora zostanie załadowana zawartość rejestru 90h umieszczonego w obszarze SFR procesora, czyli w tym wypadku rejestr portu P1, a nie komórka dodatkowej pamięci RAM. Do adresowania rejestru SBUF należy oczywiście użyć adresowania bezpośredniego, czyli instrukcji np.

```
MOV A, SBUF      ;załadowanie odebranego znaku
                  ;do akumulatora
```


Też to potrafisz

Przyznam, że nie wiem co skłoniło mnie do przedstawienia takiego wywodu, bo w kilkuletniej praktyce nigdy nie zastosowałem błędnego adresowania.

Druga uwaga dotyczy niejasnego wywodu dotyczącego omawiania rejestrów odpowiedzialnych za przerwania, zamieszczonego w EdW nr 5/98 str. 38, prawa górna szpalta. Prawdą jest, że priorytet przerwań jest ustawiony fabrycznie i tak np. najwyższy priorytet ma przerwanie od wejścia INT0. „...Niestety dalsze stwierdzenie, że przerwanie jedno zostanie przerwane przez drugie dlatego, że jest dalsze w kolejności fabrycznie ustalonej, jest błędne. Szywny priorytet przerwań ma zastosowanie jedynie do rozstrzygania kolejności przerwań jednocześnie nadchodzących. Natomiast do ustalania, które przerwanie może być przerwane przez inne służy tylko rejestr priorytetu IP...” – koniec cytatu p. Zbigniewa.

Zbigniewie, przeczytawszy kilka razy ten fragment artykułu muszę przyznać, że opisując ten problem za pomocą języka prostego „do granic możliwości”, być może sam ugryzłem się w język i ... zamiast napisać to co miałem na myśli, przeleciałem tę myśl w sposób niewłaściwy. W każdym razie, obydwu nam chodzi o to samo. Ci spośród Czytelników, którzy niewłaściwie, zrozumieli wyjaśnienia proszeni są o zweryfikowanie swoich wiadomości. Postaram się pewnie dość skomplikowane dla Was, a jednocześnie oczywiste dla mnie problemy „8051”, opisywać bardziej jasno, tak aby nie było wątpliwości.



LIST 3

Pan podpisujący się jako **RYS** lat XX, pyta w swoim liście, dlaczego w listingu programu z lekcji 8 str. 46 szpalta 2, linia 74 używam instrukcji

ORL TMOD, #00h

która przecież nie zmienia zawartości rejestru TMOD. I faktycznie instrukcja nie wpływa na zawartość rejestru, bowiem jest to logiczne dodać liczbę „0” do rejestru TMOD. W programie instrukcja ta jednak się pojawia, bowiem w praktyce podczas pisania innych aplikacji wykorzystujących np. licznik T1 (lub inny) występuje potrzeba modyfikacji czwórki rejestru TMOD (starszych 4 bitów dla licznika T1, młodszych 4 bitów dla licznika T0) i najpierw należy wyzerować tę czwórkę, którą chcemy zmienić instrukcją ANL, a następnie ustawiamy bity tej czwórki właśnie za pomocą instrukcji ORL. W przykładzie podanym w artykule, do wymaganej pracy licznika T1 nie potrzeba ustawiania żadnego z 4 starszych bitów rejestru TMOD, ale instrukcję w linii 74 wstawiłem domyślnie w celu ukazania, że w innym przypadku wystarczy jej użyć jedynie ze zmodyfikowanym drugim jej argumentem, tak aby uzyskać zamierzony efekt.

Tak więc zastosowanie tej instrukcji ma tylko i wyłącznie znaczenie poznawcze i oczywiście nie wpływa w tym przypadku na zawartość rejestru TMOD.

Aby usunąć swoje wątpliwości, Pan Ryś, samodzielnie napisał program do testowania operacji logicznych, które wykonuje procesor 8051. Gratuluję świetnego pomysłu. Życzę powodzenia przy pisaniu kolejnych programów.



LIST 4

Tomasz Kutyla ze Stalowej Woli zauważył nieścisłości w druku dotyczące obliczeń liczników procesora, zamieszczonych w numerze EdW 5/98 (ach, to chyba naprawdę pechowy numer...!). I tak:

- na str. 38, szpalta 2, pod rysunkiem rejestru IP w opisie jego bitów dwukrotnie powtórzona została para bitów PX1 i PT1;
- w lekcji nr 8, str. 45, szpalta 1, wiersz 17 (od dołu) jest
 $fz = F_{xtal} / 12 / 32 = 28000 \text{ Hz}$
a powinno być oczywiście ...
28800 Hz!
- na tej samej stronie i kolumnie, wiersz 6 (od dołu) wydrukowano:
 $fz / 128 = 28800 / 256 = 225 (=THimp)$
a powinno być oczywiście:
 $fz / 128 = 28800 / 128 = 225 \dots itd.$
- na stronie 45, szpalta 2, wiersz 4 (od góry) wydrukowano:
 $TH1pocz = TH1max - TH1imp + 1 = 256 - 225 + 1 = 31$
a powinno być oczywiście napisane:
 $TH1pocz = TH1max - TH1imp + 1 = 255 - 225 + 1 = 31$
- na stronie 46 w listingu programu po linii 71 – z etykietą START, zabrakło przy okazji inicjacji układu przerwań, instrukcji
SETB EA ;uaktywnienie systemu przerwań

Program będzie oczywiście działał na komputerku edukacyjnym AVT-2250, bowiem sam monitor komputerka wcześniej wykonuje tę instrukcję i uruchamia system przerwań. Jednak przy pisaniu programów na samodzielne systemy z procesorem 8051 lub podobnymi, nie moż-

na zapomnieć o globalnym ich odblokowaniu, po odpowiednim ustawieniu bitów w rejestrze masek przerwań. W przeciwnym wypadku układ będzie przysłowiowo „martwy”.

Tak więc wspomniana wcześniej instrukcja powinna znaleźć się dla porządku po linii 79, kiedy to ustawione zostały bity maskujące przerwanie od licznika T1 i rejestru priorytetu przerwań.

Tomaszu, dziękuję za te uwagi i ... gratuluję wnikliwej lektury naszych artykułów oraz „dobrego oka”.



LIST 5

Marcin Wiązania z Kieleckiego zauważył nieścisłość w lekcji 8 z nr 5/98 EdW str. 44, szpalta 1, wiersz 30 (od dołu). Otóż stwierdzam tam, niesłusznie zresztą, że „... w przypadku wartości rezonatora 11059200 Hz zliczanie 1/100 sekundy byłoby dość kłopotliwe ze względu na to, że wartość tego kwarcu nie dzieli się przez 12 i dodatkowo przez liczbę całkowitą, tak aby dać liczbę 100...”. Otóż dzieli się - a tą liczbą, jak słusznie zauważa Marcin, jest przecież: 9261, bo:

$11059200 / 12 = 921600$,
 $921600 / 100 = 9216$ – szukana liczba całkowita

Stąd wartość początkową licznika T1 można obliczyć następująco:

$T1pocz = TH1.TL1 = 65535 - 9216 + 1 = 56320 = DC00h$

Przy tej wartości początkowej wpisywanej do licznika za pomocą instrukcji np.

MOV TH1, #0DCh
; olr TL1, #0 nie jest konieczne!

licznik T1 będzie przepelniany dokładnie 100 razy na sekundę, czyli tak jak chcieliśmy.

Pozdrawiam Pana, panie Marcinie.

Rozwiązanie zadania „ZEGAR”



LIST 6

To tyle uwag naszych Czytelników, co do treści lekcji nr 8. W dalszej części artykułu przedstawię najciekawsze rozwiązania zadania z lekcji nr 8, a mianowicie wyposażenie zegara czasu rzeczywistego w dodatkową funkcję wyświetlania daty.

Na szczególną uwagę zasługują rozwiązania dwóch naszych czytelników. Pierwszy list otrzymałem od p. **Marcina Wiązania** z woj. kieleckiego (**listing 3**). Jego program zegara z datownikiem oceniam na 5+, autor bowiem podjął się także kontroli i prawidłowego wyświetlania liczby dni dla każdego miesiąca. Wiemy przecież, że liczba dni różni się w każdym miesiącu, a lutym różnica ta wynosi nawet 3 dni!

Oto fragment listu p. Marcina:

„... Problem ilości dni w każdym miesiącu rozwiązałem za pomocą tabeli stałych z wartościami dni w każdym miesiącu. Jest to rozwiązanie bardzo proste, ale nie uwzględniające lat przestępnych i nie można go zastosować kiedy ograniczona jest ilość pamięci programu...” (to akurat nie jest problemem – przyp. redakcji). „...Po uruchomieniu programu należy najpierw wprowadzić datę w następującej kolejności „dzień” : „miesiąc” : „rok”, a następnie godzinę...” (godzinę, minuty i sekundy – przyp. redakcji). „Przełączenia na wyświetlanie daty dokonuje się za pomocą klawisza „1”, natomiast z powrotem na godzinę, za pomocą klawisza OK. Klawisz sterujący możemy bardzo łatwo zmienić wprowadzając wartość innego klawisza. Do odczytu klawisza wykorzystałem zmienną „klawisz”. Po wprowadzeniu daty, program zgodnie z wprowadzonym miesiącem odczytuje z „tabeli stałych” ilość dni, a następnie umieszcza je w zmiennej „mies”. Zmienna „mies” określa w procedurze przerwania ilość dni w danym miesiącu. Próbowalem także, aby data i godzina zmieniała się na przemian po naciśnięciu za każdym razem tego samego przycisku. Lecz nie udało mi się tego osiągnąć nawet po wprowadzeniu małego opóźnienia. Zbyt duże powodowało, że układ wcześniej wchodził do obsługi przerwania, przez co traciłem wartość akumulatora, natomiast zbyt mała wartość powodowała miganie wyświetlacza.

Specjalnie zamieściłem listing programu, a nie jego treść źródłową, bowiem dzięki temu pożytek mają z niego zarówno komputerowcy jak i „ręczniacy”.

Program p. Marcina działa bezbłędnie. Radzę zatem spróbować przetestować go na swoim sprzęcie w domowym zaciszu!

Listing 3

```

1          CPU      '8052.def'      2
;*****
3          ;Klasa mikroprocesorowa - LEKCJA 8
4          ;Program obsługi zegara czasu rzeczywistego
5          ;na komputer edukacyjny AVT-2250
6          ;*****
7          ;procedura korzysta z przerywania licznika T1 (tryb 0)
8          ;wykorzystywane sa 3 komorki wewn. RAM procesora
9          ;zegar liczy: godziny, minuty, sekundy, dni, miesiace i lata
10         ;w trybie 24-godzinnym
11         ;*****
12
13         include 'const.inc'
14         include 'bios.inc'
15
16         ;Definicje komorek w wewn. RAM procesora
17         ;zajmowane przez dane zegara
18
19 0060      GODZ      equ      60h          ;licznik godzin
20 0061      MIN       equ      61h          ;licznik minut
21 0062      SEK       equ      62h          ;licznik sekund
22 0064      DDequ     64h          ;licznik dni
23 0065      MMequ     65h          ;licznik miesiecy
24 0066      RRequ     66h          ;licznik lat
25 0067      MIES      equ      67h          ;zmienna ilosci dni w miesiacu
26 0063      licz128   equ      63h          ;licznik 1/128 sek
27
28         ;Definicje stalych wykorzystywanych w programie
29
30 001F      Czest      equ      31          ;wartosc pocz. licznika TH1
31
32         ;*****
33         ;Początek kodu programu
34 8000      org        8000h
35 8000 02807F  ljmp      START          ;petla glowna od etyk. START
36         ;*****
37         ;Wektor przerywania od licznika T1
38 801B      org        801Bh
39 801B      intT1:
40 801B 758D1F  mov      TH1,#Czest      ;początek proc. przer. T1
41 801E 0563    inc      licz128          ;przeladowanie licznika T1
42 8020 E563    mov      A,licz128        ;zwiększenie licznika 1/128sek.
43 8022 C2E7    clr      Acc.7
44 8024 7052    jnz      koniecT1
45 8026 E562    mov      A,SEK
46 8028 2401    add      A,#1            ;zwiększanie licznika sekund
47 802A D4      da      A              ;z korekcja dziesiętna
48 802B F562    mov      SEK,A
49 802D B46048  cjne     A,#60h,koniecT1 ;czy SEK > 59?, nie to skocz
50 8030 756200  mov      SEK,#0          ;tak to wyzeruj sekundy i koryguj minuty
51 8033 E561    mov      A,MIN
52 8035 2401    add      A,#1            ;zwiększenie licznika minut
53 8037 D4      da      A              ;z korekcja dziesiętna
54 8038 F561    mov      MIN,A
55 803A B4603B  cjne     A,#60h,koniecT1 ;czy MIN > 59?, nie to skocz
56 803D 756100  mov      MIN,#0          ;tak to zeruj minuty i koryguj godziny
57 8040 E560    mov      A,GODZ
58 8042 2401    add      A,#1            ;zwiększenie licznika minut
59 8044 D4      da      A              ;z korekcja dziesiętna
60 8045 F560    mov      GODZ,A
61 8047 B4242E  cjne     A,#24h,koniecT1 ;czy GODZ > 23?, nie to skoc
62 804A 756000  mov      GODZ,#0         ;tak to zeruj godziny
63 804D E564    mov      A,DD
64 804F 2401    add      A,#1            ;zwiększenie licznika dni
65 8051 D4      da      A              ;z korekcja dziesiętna
66 8052 F564    mov      DD,A
67 8054 B56721  cjne     A,MIES,koniecT1 ;czy mies.>ilosc dni, nie to skocz
68 8057 756401  mov      DD,#1          ;tak to koryguj dni
69 805A E565    mov      A,MM
70 805C 2401    add      A,#1            ;zwiększenie licznika miesiecy
71 805E C0E0    push     A
72 8060 90815F  mov      DPTR,#tab_mies   ;pobranie adresu tabeli ilosci dni w mies.
73 8063 93      movc     A,@A+DPTR       ;wpisanie do A ilosci dni w danym miesiacu
74 8064 F567    mov      MIES,A         ;a nastepnie przepisanie do zmiennej „MIES”
75 8066 D0E0    pop      A
76 8068 D4      da      A              ;korekcja dziesiętna miesiecy
77 8069 F565    mov      MM,A
78 806B B4130A  cjne     A,#13h,koniecT1 ;czy mies.>12, nie to skocz
79 806E 756501  mov      MM,#1          ;tak to koryguj miesiace i rok
80 8071 E566    mov      A,RR
81 8073 2401    add      A,#1
82 8075 D4      da      A              ;korekcja dziesiętna roku
83 8076 F566    mov      RR,A
84 8078      koniecT1:
85 8078 D082    pop      DPL            ;odtworzenie rejestrow

```

Listing 3, cd.

```

86 807A D083      pop     DPH                ;ze stosu
87 807C D0E0      pop     Acc
88 807E 32        reti
89               ;*****
90 807F          START:
91 807F C28E      clr     TR1                ;licznik T1 stop
92 8081 53890F    anl     TMOD,#0Fh          ;wyczyszczenie bitow T1
93 8084 438900    orl     TMOD,#00h          ;T1 jako 16-bitowy (tryb 1)
94 8087 758D1F    mov     TH1,#Czest         ;zaladowanie licznika
95 808A 756300    mov     licz128,#0        ;wyzerowanie licznika 1/256sek.
96 808D 757280    mov     intvec,#80h        ;zaladowanie MSB wektora przerwan
97 8090 D2AB      setb    ET1                ;odblokowanie przerwania od T1
98 8092 D2BB      setb    PT1                ;priorytet na to przerwanie
99 8094
100 8094 120274   lcall    CLS                ;wyczyszczenie displeja
101 8097 757840   mov     DL1,#_minus
102 809A 757940   mov     DL2,#_minus
103 809D 75F001   mov     B,#1
104 80A0 1203A7   lcall    GETACC              ;pobranie poczatkowych dni
105 80A3 F564     mov     DD,A
106 80A5 757A40   mov     DL3,#_minus
107 80A8 757B40   mov     DL4,#_minus
108 80AB 757C40   mov     DL5,#_minus
109 80AE 75F004   mov     B,#4
110 80B1 1203A7   lcall    GETACC              ;pobranie poczatkowych miesiacy
111 80B4 F565     mov     MM,A
112 80B6 757D40   mov     DL6,#_minus
113 80B9 90815F   mov     DPTR,#tab_mies        ;pobranie z tabeli ilosci dni zaleznej
114 80BC 93       movc    A,@A+DPTR          ;od wpisanego miesiaca
115 80BD F567     mov     MIES,A
116 80BF 757E40   mov     DL7,#_minus
117 80C2 757F40   mov     DL8,#_minus
118 80C5 75F007   mov     B,#7
119 80C8 1203A7   lcall    GETACC              ;pobranie poczatkowego roku
120 80CB F566     mov     RR,A
121 80CD 120274   lcall    CLS                ;wyczyszczenie displeja
122 80D0 757840   mov     DL1,#_minus
123 80D3 757940   mov     DL2,#_minus
124 80D6 75F001   mov     B,#1
125 80D9 1203A7   lcall    GETACC              ;pobranie poczatkowej godziny
126 80DC F560     mov     GODZ,A
127 80DE 757B40   mov     DL4,#_minus
128 80E1 757C40   mov     DL5,#_minus
129 80E4 75F004   mov     B,#4
130 80E7 1203A7   lcall    GETACC              ;pobranie poczatkowej minuty
131 80EA F561     mov     MIN,A
132 80EC 757E40   mov     DL7,#_minus
133 80EF 757F40   mov     DL8,#_minus
134 80F2 75F007   mov     B,#7
135 80F5 1203A7   lcall    GETACC              ;pobranie poczatkowej sekundy
136 80F8 F562     mov     SEK,A
137 80FA 757A40   mov     DL3,#_minus          ;zapalenie kresek w postaci
138 80FD 757D40   mov     DL6,#_minus          ;GG-MM-SS (godzina wprowadzona!)
139 8100 74FA     mov     A,#250
140 8102 120295   lcall    DELAY              ;odczekanie ok. 0,5 sekundy
141 8105 1202C5   lcall    CONIN              ;czekanie na start zegara (klawisz)
142 8108 D28E     setb    TR1                ;start licznika (zegara)
143 810A          pokaz:
144 810A E576     mov     A,klawisz          ;wpisanie zawartosci bufora klawisz do A
145 810C B43102   cjne    A,#'1',zegar        ;jezeli A rozne od 1 to skocz
146 810F 802D     sjmp     data              ;a jezeli rowne, to kolejny rozkaz
147 8111 E563     zegar:      mov     A,licz128
148 8113 30E608   jnb     Acc.6,pelne              ;co 1/2 sekundy pokazuj na zmiane
149 8116 757A00   mov     DL3,#0                ;puste DL3 i DL4
150 8119 757D00   mov     DL6,#0
151 811C 8006     sjmp     czas
152 811E 757A40   pelne:      mov     DL3,#_minus          ;i kreski na DL3 i DL6
153 8121 757D40   mov     DL6,#_minus
154 8124          czas:
155 8124 E560     mov     A,GODZ
156 8126 75F001   mov     B,#1                ;na DL1.DL2
157 8129 12024E   lcall    A2HEX              ;wypisz godziny
158 812C E561     mov     A,MIN
159 812E 75F004   mov     B,#4                ;na DL4.DL5
160 8131 12024E   lcall    A2HEX              ;wypisz minuty
161 8134 E562     mov     A,SEK
162 8136 75F007   mov     B,#7                ;na DL7.DL8
163 8139 12024E   lcall    A2HEX              ;wypisz na wyswietlacz
164 813C 80CC     sjmp     pokaz              ; i od poczatku
165 813E          data:
166 813E E576     mov     A,klawisz          ;przepisanie bufora klawisz do A
167 8140 B40D02   cjne    A,#klaw_OK,wypisz;jezeli A rozne od klaw_OK to skocz
168 8143 80C5     sjmp     pokaz              ;jezeli rowne, to wykon. kolejny rozkaz
169 8145 E564     wypisz:      mov     A,DD
170 8147 75F001   mov     B,#1                ;na DL1.DL2

```

```

171 814A 12024E    lcall    A2HEX        ;wypisz dni
172 814D E565      mov     A,MM
173 814F 75F004    mov     B,#4           ;na DL4.DL5
174 8152 12024E    lcall    A2HEX        ;wypisz miesiace
175 8155 E566      mov     A,RR
176 8157 75F007    mov     B,#7           ;na DL7.DL8
177 815A 12024E    lcall    A2HEX        ;wypisz na wyswietlacz rok
178 815D 80DF      sjmp     data        ; i od poczatku
179
180 815F 00322932   tab_mies db    00h,32h,29h,32h,31h,32h,31h    ;tablica dni w
181 8166 32323100   db    32h,32h,31h,00h,00h,00h,00h
182 816D 00003231   db    00h,00h,32h,31h,32h    ;kolejnych miesiacach
183 8172           END

Kompilacja zakonczona pomyslnie!
Zbior: „lekcja8a.s03”, 346 bajt(ow), 0.4 sekund(y).

```



LIST 7

Na zakończenie list od **Piotra Konopko** z Łomży, który także przesyła rozwiązanie problemu zadania z lekcji 8 oraz dodatkowy listing wersji zegara wyświetlającego setne części sekundy...

„... mam 19 lat i skończyłem właśnie technikum elektro-
niczne i szykuję się na studia. Chciałbym serdecznie Panu
podziękować za prowadzony na łamach EdW kurs progra-
mowania 8051. Do niedawna byłem, jak to się mówi, zielo-
ny na temat mikrokontrolerów, ale dzięki Panu i EdW szybko
przyswoiłem podstawy 8051. Z lekcji na lekcję odnoszę co-
raz większe sukcesy. Muszę Panu także powiedzieć, że wie-
dza, jaką zdobyłem na kursie, szybko przyniosła mi sukces
w szkole. Na zajęciach z programowania sterowników mik-
roprocesorowych nie miałem żadnych problemów. Mimo
nieco innych rozkazów i budowy sterowników założenia są
takie same...” (ciekaw jestem jakiego rodzaju sterowniki
Pan programuje w szkole? – przypis redakcji). „Dzięki temu,
że poznałem zasady działania 8051, czułem tzw. „bluesa”.
Wielka uniwersalność mikrokontrolerów, o jakiej pisał Pan
na początku klasy mikroprocesorowej, w moim przypadku
dała o sobie znać bardzo szybko. (...)

„Na zakończenie chciałbym stwierdzić, że klasa prowadzi-
na jest perfekcyjnie. Żaden belfer w dotychczasowej mojej
edukacji nie uczył mnie tak szybko i sprawnie jak Pan. Jesz-
cze raz Panu i całej redakcji EdW serdecznie dziękuję...”.

Bardzo dziękuję za te ciepłe słowa, szczególnie po lektu-
rze pechowego odcinka klasy mikroprocesorowej z nr 5/98.
Jestem rad, że tak wielu z Was odnosi pierwsze sukcesy
w programowaniu kontrolerów 8051 i to dzięki cyklowi
moich artykułów w EdW.

W nawiązaniu do listu p. Piotra zamieszczam listing jego
programu – zegara z wyświetlaniem setnych części sekun-
dy, zgodnie z zasadami opisanymi przy okazji omawiania lis-
tu 5. I chociaż program Piotra pracują znakomicie, to zwrac-
cam uwagę Panu i wszystkim początkującym programistom
na dobrą zasadę umieszczania jak największej liczby komen-
tarzy w programach źródłowych. Jak sami się bowiem prze-
konacie, ich brak często utrudnia, lub wręcz uniemożliwia
analizę programu (a nawet jego rozpoznanie) po pewnym
czasie.

Oto listing programu zegara ze zliczaniem setnych części
sekundy z kwarcem 11059200 Hz (**listing 4**).

Postarajcie się, Drodzy Czytelnicy, przeanalizować nie ko-
mentowany program p. Piotra i uzupełnić go o komentarze.
Z pewnością zrozumienie obcego kodu źródłowego, w do-
datku bez komentarzy, będzie pouczającą lekcją dla każdego
zapaleńca procesorów 8051 i nie tylko.

Na zakończenie korespondencji od p. Piotra Konopko
przytaczam dodatkowy listing programu zamka szyfrowego
(**listing 5**), który to p. Piotr wykonał samodzielnie jako swój
pierwszy program na 8051. Pomimo braku komentarzy i z
trochę zagmatwanej obsługi, program ten zasługuje na wy-
różnienie i pochwałę, ze względu na swoją prostotę i funk-
cjonalność, a przede wszystkim za to, że program „działa”
i potrafi sterować dołączonym do portu P1.0 układem załą-
czania rygla. Zanim przejdę do prezentacji listingu programu
zamka szyfrowego, posłuchajmy opisu sposobu użytkowa-
nia i działania programu. P. Piotr pisze w swoim liście:

Listing 4

```

1          CPU 8052.def    2      include    'const.inc'
3          include    'bios.inc'
4
5      0060      godz      equ     60h
6      0061      min       equ     61h
7      0062      sek       equ     62h
8      0063      liczn128  equ     63h
9      00DC      czest     equ     DCh
10
11     8000      org       8000h
12     8000 028056  ljmp     start
13     801B      org       801Bh
14     801B      inttl:
15     801B 758DDC  mov     TH1,#czest
16     801E E563   mov     A,liczn128
17     8020 2401   add     A,#1
18     8022 D4     da      a
19     8023 F563   mov     liczn128,a
20     8025 B40027 cjne    A,#0h,koniec
21     8028 E562   mov     A,sek
22     802A 2401   add     A,#1
23     802C D4     da      a
24     802D F562   mov     sek,a
25     802F B4601D cjne    A,#60h,koniec
26     8032 756200 mov     sek,#0
27     8035 E561   mov     A,min
28     8037 2401   add     A,#1
29     8039 D4     da      a
30     803A F561   mov     min,a
31     803C B46010 cjne    A,#60h,koniec
32     803F 756100 mov     min,#0
33     8042 E560   mov     A,godz
34     8044 2401   add     A,#1
35     8046 D4     da      a
36     8047 F560   mov     godz,a
37     8049 B42403 cjne    A,#24h,koniec
38     804C 756000 mov     godz,#0
39     804F      koniec:
40     804F D082   pop      dpl
41     8051 D083   pop      dph
42     8053 D0E0   pop      acc
43     8055 32     reti
44
45     8056      start:
46     8056 C28E   clr      TR1
47     8058 53890F anl      TMOD,#0fh
48     805B 438910 orl      TMOD,#10h
49     805E 758DDC mov     TH1,#czest
50     8061 756300 mov     liczn128,#0
51     8064 757280 mov     intvec,#80h
52     8067 D2AB   setb    ET1
53     8069 D2BB   setb    PT1
54     806B 120274 lcall    CLS
55
56     806E 757840 mov     DL1,#_minus
57     8071 757940 mov     DL2,#_minus
58     8074 75F001 mov     B,#1
59     8077 1203A7 lcall    GETACC
60     807A F560   mov     godz,a
61     807C 757A40 mov     DL3,#_minus
62     807F 757B40 mov     DL4,#_minus

```


Też to potrafisz

„...Po załadowaniu programu pojawi się napis „HASLO”. Na początku brak jest wprowadzonego hasła, dlatego należy wcisnąć klawisz OK., potwierdzający. Pojawi się wtedy napis „YES”, świadczący o prawidłowym hasle. Następnie ponownie wciskamy klawisz OK. w celu wprowadzenia nowego hasła. Zostanie wyświetlony napis „NEU” (nowy) po którym to możemy wprowadzić nowy szyfr (liczba cyfr zależy od wielkości wolnych komórek w wewnętrznej RAM powyżej adresu 22h). Hasło potwierdzamy klawiszem OK. Teraz układ jest zaszyfrowany, a na wyświetlaczu widnieje napis „HASLO”. Aby teraz móc sterować wyjściem zamka (w moim przypadku jest to P1.0), należy podać prawidłowy szyfr. Pięciokrotne wprowadzenie błędnego hasła blokuje zamek, uniemożliwiając dalsze manipulacje. Żle wprowadzony kod sygnalizowany jest napisem „Error”. Po wprowadzeniu właściwego hasła zatwierdzamy je klawiszem OK. Wówczas mamy 2 możliwości: albo możemy odblokować wyjście (P1.0) naciskając dowolny klawisz oprócz OK. i M, lub możemy wprowadzić nowe hasło wciskając klawisz OK. Przez cały czas odblokowania zamka wyświetlany jest napis „YES”.

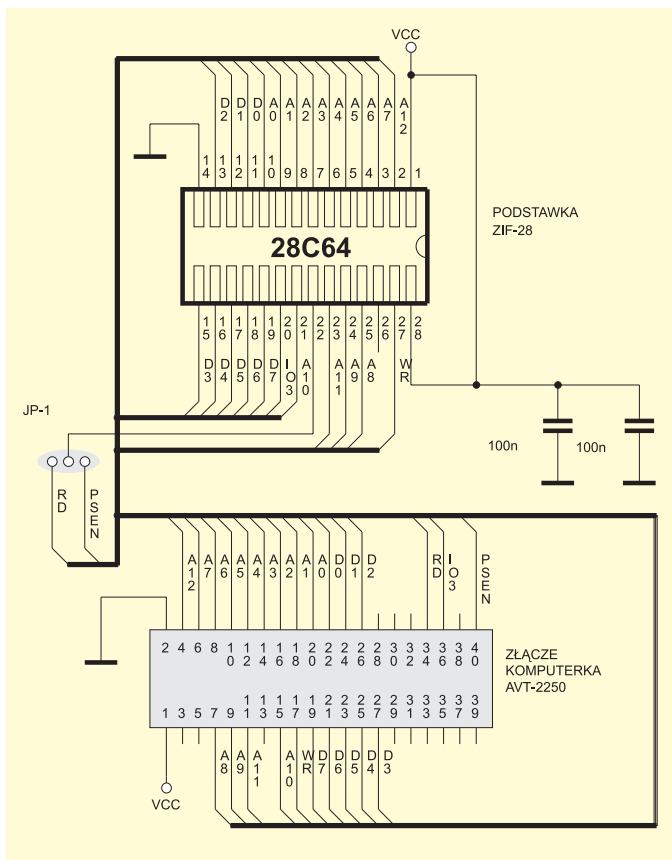
W celu ponownej aktywacji zamka wystarczy wcisnąć klawisz OK. Blokada automatycznie zostanie uaktywniona, a na wyświetlaczu ponownie zaświeci się napis HASLO...”

Brawo Piotrze!, życzę dalszych sukcesów i wielu ciekawych pomysłów na programy na 8051.

Na zakończenie naszej rubryki prezentuję prawdziwy „hit” tego numeru, mianowicie projekty: programatora pamięci EEPROM typu 28C64 (kompatybilna z 27C64, z tym że programowana i kasowana elektrycznie) oraz programatora mikroprocesora 89C2051 (który jest ograniczoną wersją procesora 87C51) w postaci przystawki do komputerka edukacyjnego AVT-2250. Projekt ten wykonał p. Tadeusz Kałuża z Podłęża (woj. krakowskie). Brawo panie Tadeuszu! Oto list:

„...Jestem 40-letnim rękodzielnikiem, zajmującym się amatorsko elektroniką. Analizując budowę komputerka edukacyjnego oraz uczestnicząc w prowadzonym przez Pana kursie programowania mikrokontrolerów jednoukładowych, doszedłem do wniosku, że funkcja edukacyjna dla komputerka to mało i wykonałem dwie proste przystawki, które jak uważam zdecydowanie zwiększają jego atrakcyjność. Pierwsza przystawka jest bardzo prosta i służy do programowania pamięci typu EEPROM np. 28C64. Przystawka ta składa się z zaciskowej podstawki podłączonej do łącza systemowego komputerka w sposób uwidoczniiony na schemacie (rysunek 1).

Rys. 1. Schemat przystawki programującej pamięć EEPROM 28C64



Listing 4, cd.

```
63 8082 75F003      mov     B,#3
64 8085 1203A7      lcall   GETACC
65 8088 F561        mov     min,a
66 808A 757C40      mov     DL5,#_minus
67 808D 757D40      mov     DL6,#_minus
68 8090 75F005      mov     B,#5
69 8093 1203A7      lcall   GETACC
70 8096 F562        mov     sek,a
71 8098 74FA        mov     A,#250
72 809A 120295      lcall   DELAY
73 809D 1202C5      lcall   CONIN
74 80A0 D28E        setb    TR1
75 80A2              czas:
76 80A2 E560        mov     A,godz
77 80A4 75F001      mov     B,#1
78 80A7 12024E      lcall   A2HEX
79 80AA E561        mov     A,min
80 80AC 75F003      mov     B,#3
81 80AF 12024E      lcall   A2HEX
82 80B2 E562        mov     A,sek
83 80B4 75F005      mov     B,#5
84 80B7 12024E      lcall   A2HEX
85 80BA E563        mov     A,licz128
86 80BC 75F007      mov     B,#7
87 80BF 12024E      lcall   A2HEX
88 80C2 80DE        sjmp    czas
89
90 80C4              END
```

Kompilacja zakończona pomyślnie!
Zbiór: „zegar_2.s03”, 172 bajt(ow), 0.3 sekund(y).

Kod programu wpisany do komputerka wygląda następująco:

```
START:
90 C0 00
E0
C0 83
53 83 9F
F0
D0 83
A3
74 02
12 02 95
75 F0 03
12 02 5F
74 E0
B5 83 E6
02 00 00
KONIEC
```

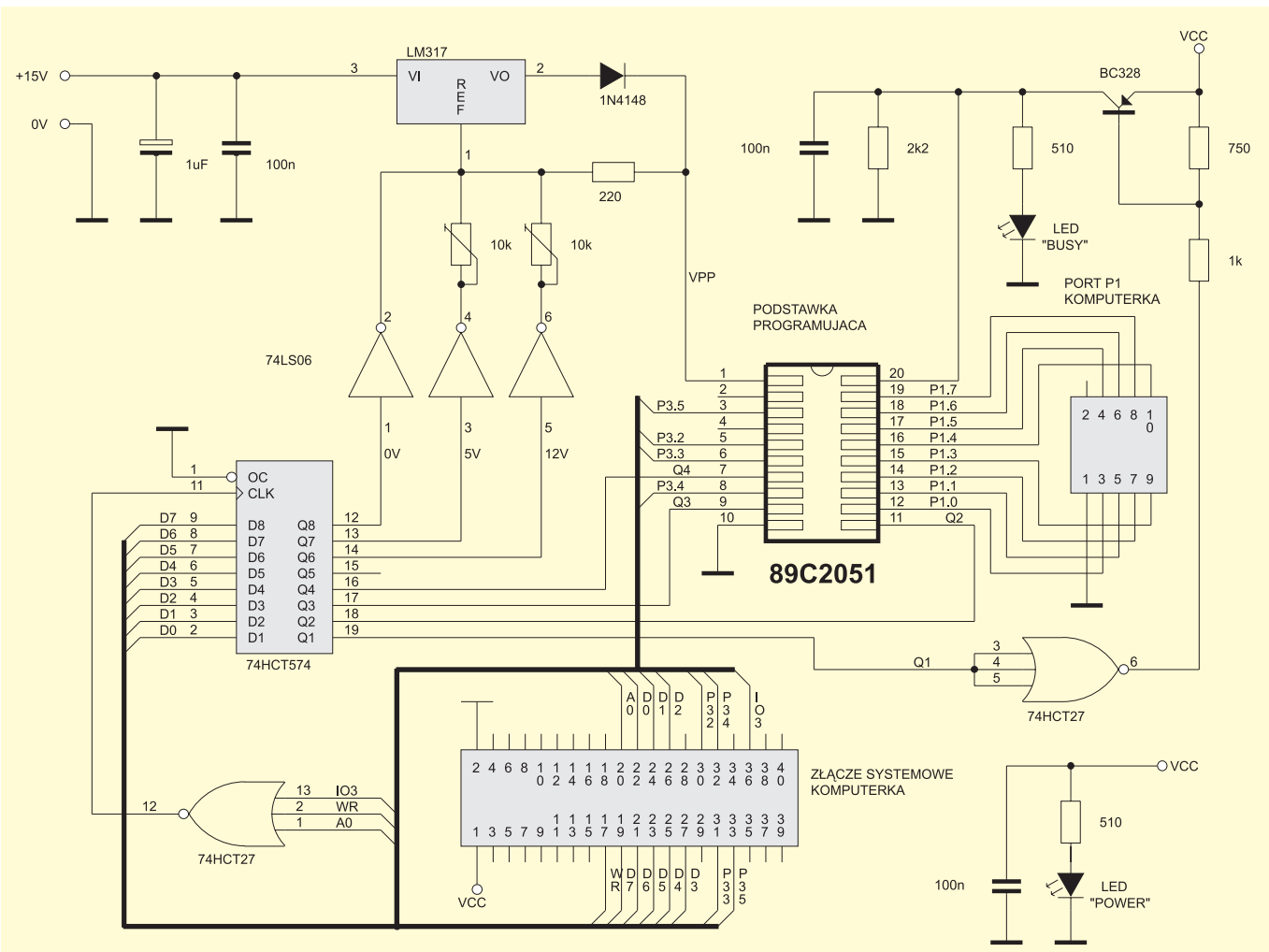
Po podłączeniu przystawki w komputerku JUMPER JP-3 należy ustawić na poz. C000h, a program wpisać w wolnym obszarze pamięci programu U3 lub w końcowym fragmencie aktywnego obszaru pamięci U4. Proces programowania rozpoczyna się wywołaniem wpisanego programu funkcją „JUMP” i powoduje przekopiowanie danych z obszaru adresowego C000h – DFFFh do programowanej pamięci, jednocześnie wyświetlając aktualny stan rejestru DPTR (czyli adresu aktualnie programowanej komórki – przyp. redakcji).

Następna przystawka jest bardziej złożona i służy do programowania mikrokontrolerów typu AT89C2051 firmy Atmel. Przy jej wykonaniu wzorowałem się na pańskich projektach AVT-320 i AVT-2250. Schemat tej przystawki przedstawia rysunek (rysunek 2).

Program uruchamiający przystawkę najlepiej wpisać w wolny obszar pamięci U3 komputerka (ostatecznie może być zapisany w pamięci U4 w obszarze adresowym D000h – DFFFh). Jumper JP-3 należy ustawić w pozycji C000h. Wywołanie programu funkcją „JUMP” powoduje ustawienie wszystkich istotnych w programowaniu wyprowadzeń podstawki w stan niski, a następnie oczekuje na naciśnięcie odpowiedniego klawisza w celu wywołania pożądanej procedury. W tym też czasie należy włożyć w podstawkę procesor 89C2051. Kod programu przedstawia listing 6.

Wywołanie programu powoduje wyświetlanie sekwencji – P1. 02. C3. i oczekuje na naciśnięcie następujących klawiszy :

klawisz (1) – programowanie – odczyt, powoduje wpisanie do pamięci programowanego mikroprocesora danych umieszczonych w pamięci



Rys. 2. Schemat przystawki programującej procesor 89C2051 (89C1051)

Listing 5

```

1                                     CPU  8052.def
2
3                                     include  'CONST.INC'
4                                     include  'BIOS.INC'
5  8000                                org      8000h
6  8000  7A00                          mov     R2,#00h
7  8002  752000                        mov     20h,#00h
8  8005  752100                        mov     21h,#00h
9
10 8008                                poczatek:
11 8008  120274                        lcall   cls
12 800B  75F002                        mov     B,#2
13 800E  9080E1                        mov     dptr,#tablica
14 8011  120285                        lcall   text
15 8014  7878                          mov     R0,#78h
16 8016  7922                          mov     R1,#22h
17 8018                                skok10:
18 8018  1280DB                        lcall   wait
19 801B  1202C5                        lcall   conin
20 801E  E576                          mov     A,76h
21 8020  B40D37                        cjne    A,#13,skokpom2
22 8023  E520                          mov     A,20h
23 8025  6521                          xrl     A,21h
24 8027  704F                          jnz     skok2
25 8029  120274                        lcall   cls
26 802C  757B6E                        mov     dl4,#_Y
27 802F  757C79                        mov     dl5,#_E
28 8032  757D6D                        mov     dl6,#_5
29 8035  7878                          mov     R0,#78h
30 8037  7922                          mov     R1,#22h
31 8039  1280DB                        lcall   wait
32 803C  1202C5                        lcall   conin
33 803F  E576                          mov     A,76h

```

34	8041	B40D55		cjne	A,#13,skok3
35	8044	757B54		mov	dl4,#_n
36	8047	757C79		mov	dl5,#_E
37	804A	757D3E		mov	dl6,#_U
38	804D		skok5:		
39	804D	1280DB		lcall	wait
40	8050	1202C5		lcall	conin
41	8053	E576		mov	A,76h
42	8055	B40D05		cjne	A,#13,skok4
43	8058	80AE		sjmp	poczatek
44	805A		skokpom2:		
45	805A	0280B0		ljmp	skok1
46	805D		skok4:		
47	805D	F7		mov	@R1,A
48	805E	C000		push	00h
49	8060	E578		mov	A,dl1
50	8062	B40003		cjne	A,#0,skok6
51	8065	120274		lcall	cls
52	8068		skok6:		
53	8068	D000		pop	0h
54	806A	09		inc	R1
55	806B	0520		inc	20h
56	806D	B88002		cjne	R0,#80h,skok9
57	8070	80DB		sjmp	skok5
58	8072		skok9:		
59	8072	7440		mov	A,#_minus
60	8074	F6		mov	@R0,A
61	8075	08		inc	R0
62	8076	80D5		sjmp	skok5
63	8078		skok2:		
64	8078	120274		lcall	cls
65	807B	757979		mov	dl2,#_E
66	807E	757A50		mov	dl3,#_r
67	8081	757B50		mov	dl4,#_r
68	8084	757C3F		mov	dl5,#_0

Też to potrafisz

Listing 5, cd.

```

69 8087 757D50      mov     dl6,#_r
70 808A 1280DB      lcall  wait
71 808D 1202C5      lcall  conin
72 8090 0A          inc     R2
73 8091 752100      mov     21h,#00
74 8094              skok7:
75 8094 BA0550      cjne   R2,#5,skokpom
76 8097 80FB        sjmp   skok7
77 8099              skok3:
78 8099 C290        clr     90h      ;clr P1
79 809B 1280DB      lcall  wait
80 809E 1202C5      lcall  conin
81 80A1 E576        mov     A,76h
82 80A3 B40DF3      cjne   A,#13,skok3
83 80A6 7A00        mov     R2,#00h
84 80A8 752100      mov     21h,#00h
85 80AB D290        setb   90h
86 80AD 028008      ljmp   poczatek
87 80B0              skok1:
88 80B0 67          xrl     A,@R1
89 80B1 701C        jnz     skok8
90 80B3 0521        inc     21h
91 80B5 09          inc     R1
92 80B6 B88003      cjne   R0,#80h,skok11
93 80B9 028018      ljmp   skok10
94 80BC              skok11:
95 80BC C000        push    00h

```

```

96 80BE E578        mov     A,d11
97 80C0 B40003      cjne   A,#0,skok12
98 80C3 120274      lcall  cls
99 80C6              skok12:
100 80C6 D000        pop     0h
101 80C8 7440        mov     A,#_minus
102 80CA F6          mov     @R0,a
103 80CB 08          inc     R0
104 80CC 028018      ljmp   skok10
105 80CF              skok8:
106 80CF 740D        mov     A,#13
107 80D1 2521        add     A,21h
108 80D3 F521        mov     21h,A
109 80D5 B880E4      cjne   R0,#80h,skok11
110 80D8 028018      ljmp   skok10
111 80DB              wait:
112 80DB 74FF        mov     A,#255
113 80DD 120295      lcall  delay
114 80E0 22          ret
115
116 80E1 76776D38     tablica db _H,_A,_5,_L,_0,0
117 80E7              skokpom:
118 80E7 028008      ljmp   poczatek
119
120 80EA              END

```

Kompilacja zakonczona pomyslnie!
Zbior: „szyfr.s03”, 234 bajt(ow), 0.3 sekund(y).

U4 w obszarze adresowym C000h ... C7FFh oraz kontrolny odczyt do obszaru adresowego C800h ... CFFFh;

klawisz (2) – odczyt – kopiowanie pamięci programu mikroprocesora do obszaru adresowego C800h ... CFFFh w pamięci U4;

klawisz (3) – kasowanie pamięci programu mikroprocesora;

zakończenie każdej procedury m sygnalizowane jest na wyświetlaczu komputerka i wtedy można wyjąć z podstawki mikroprocesor lub klawiszem (0) powrócić do miejsca wywołania procedur.

Opisane programy umieściłem w prosty sposób w pamięci U3 moje-go komputerka, wykorzystując przystawkę do programowania pamięci 28C64 oraz prosty programik odczytu monitora.

Jeszcze raz dziękuję za wspianą zabawę, serdecznie pozdrawiam Pana i cały zespół redakcyjny EdW ...”.

Tym z Czytelników, którzy mają wątpliwości skąd wziąć te dane do programowania, wyjaśniam, że można je oczywiście załadować przed uruchomieniem programu przystawki programatora z komputera za po-

mocą instrukcji monitora „LOAD”, lub wpisać ręcznie (ręczniacy) za pomocą polecenia „EDIT” programu monitora.

Jeszcze raz brawa dla Pana, panie Tadeuszu, nie tylko za pomysł, ale i za wytrwałość w stworzeniu dość długiego, jak na ręczniaka, programu. Szkoda, że nie przysłał nam Pan komentarzy dotyczących swoich listingów. Proponuję Czytelnikom uzupełnienie tego samodzielnie i przeanalizowanie programu na poziomie mnemoników procesora 8051. Przekonajcie się, że proces tłumaczenia kodu maszynowego na źródłowy nie jest taki trudny. Tak a propos, to tłumaczenie nazywa się fachowo disasemblacją (operacja odwrotna do asemblacji, czyli tłumaczenia kodu źródłowego na maszynowy).

Wszystkim Autorom listów serdecznie dziękuję za wspiane pomysły i uwagi dotyczące moich artykułów, zaś pozostałym Czytelnikom życzę wielu ciekawych pomysłów. Piszcie do mnie!

Sławomir Surowiński

Listing 6

```

START:
Adres      Dane
x8B0       75 90 00 (*)
x8B3       C2 B2
x8B5       C2 B3
x8B7       C2 B4
x8B9       90 80 00
x8BC       74 80
x8BE       F0
x8BF       12 02 74
x8C2       75 78 F3
x8C5       75 79 06
x8C8       75 7B BF
x8CB       75 7C 5B
x8CE       75 7E B9
x8D1       75 7F 4F
x8D4       12 02 C5
x8D7       B4 31 03
x8DA       02 x8 E9 (*)
x8DD       B4 32 03
x8E0       02 x9 62 (*)
x8E3       B4 33 EE
x8E6       02 x9 70 (*)
PROGRAMOWANIE:
x8E9       12 02 74
x8EC       75 78 F5
x8EF       12 x9 AD (*)
x8F2       D2 B4
x8F4       74 47
x8F6       F0

```

```

x8F7       75 90 FF
x8FA       74 27
x8FC       F0
x8FD       90 C0 00
x900       F5 90
x903       C2 B3
x905       00 00
x907       D2 B3
x909       A3
x90A       75 F0 05
x90D       12 02 5F
x910       74 02
x912       12 02 95
x915       D2 B2
x917       C2 B2
x919       74 C8
x91B       B5 83 E2
x91E       90 80 00
x921       74 47
x923       F0
x924       75 90 FF
x927       90 C8 00
x92A       C2 B4
x92C       00 00
x92E       E5 90
x930       D2 B4
x932       F0
x933       75 F0 05
x936       12 02 5F
x939       D2 B2
x93B       C2 B2
x93D       A3

```

```

x93E       74 D0
x940       B5 83 07
x943       90 80 00
x946       74 87
x948       E0
x949       C2 B3
x94B       C2 B4
x94D       75 90 00
x950       74 80
x952       F0
x953       75 79 3F
x956       75 7A DC
x959       12 02 C5
x95C       B4 30 FA
x95F       02 x8 B0 (*)
(*) - „x” pierwsza cyfra
ustalonego adresu
x97F       74 29
x981       F0
x982       00 00 00
x985       C2 B3
x987       74 0A
x989       12 02 95
x98C       D2 B3
x98E       74 0A
x990       12 02 95
x993       74 49
x995       F0
x996       C2 B3
x998       74 81
x99A       F0
x99B       75 90 00
x99E       74 80
x9A0       F0
x9A1       75 79 DC
x9A4       12 02 C5
x9A7       B4 30 FA
x9AA       02 x8 B0 (*)
x9AD       74 81
x9AF       F0
x9B0       74 0A
x9B2       12 02 95
x9B5       74 41
x9B7       F0
x9B8       D2 B3
x9BA       22
KONIEC
x97C       75 90 FF

```