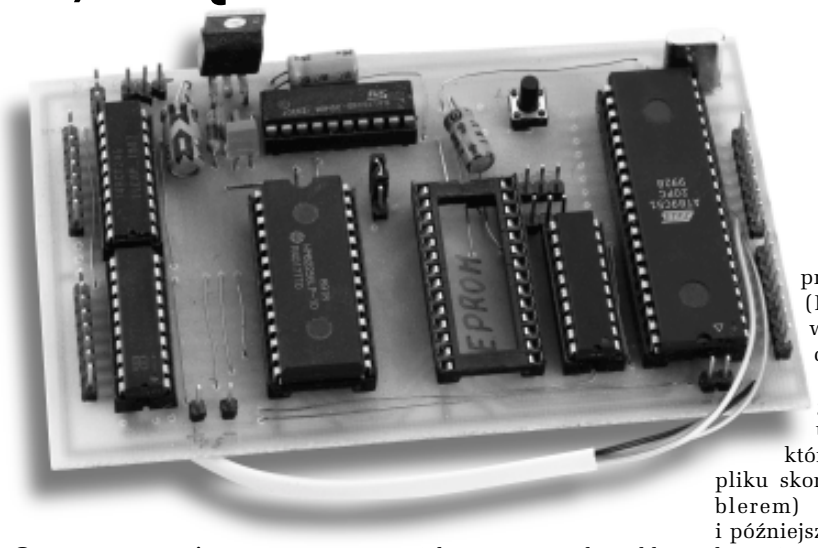
Dział „Projekty Czytelników" zawiera opisy projektów nadesłanych do redakcji EP przez Czytelników. Redakcja nie bierze odpowiedzialności za prawidłowe działanie opisywanych układów, gdyż nie testujemy ich laboratoryjnie, chociaż sprawdzamy poprawność konstrukcji.

Prosimy o nadsyłanie własnych projektów z modelami (do zwrotu). Do artykułu należy dołączyć podpisane ***oświadczenie, że artykuł jest własnym opracowaniem autora i nie był dotychczas nigdzie publikowany****. Honorarium za publikację w tym dziale wynosi 250,- zł (brutto) za 1 stronę w EP. Przysyłanych tekstów nie zwracamy. Redakcja zastrzega sobie prawo do dokonywania skrótów.*

Układ prototypowy dla mikrokontrolerów z serii 8051, część 2

Projekt 094

W drugiej (ostatniej) części artykułu przedstawiamy oprogramowanie układu uruchomieniowego. Jego obsługa jest uproszczona, gdyż we współpracującym komputerze PC zastosowano zwykły program terminalowy z systemu Windows.



Oprogramowanie

Oprogramowanie przygotowane dla zestawu zostało podzielone na dwie części:
- oprogramowanie płytki prototypowej,
- oprogramowanie dla komputera PC zapewniające transmisję danych z i do płytki EVM51.

Oprogramowanie płytki składa się z kilku zbiorów: programu monitora i zestawów dodatkowych komend dla monitora oraz opis układu wyjściowego (z buforowaniem szyny adresowej lub nie) implementowanego w układzie programowalnym GAL. Dwa listingi (**list. 1** i **2**) zawierają równania logiczne opisujące ten układ (w GAL'u). Pierwszy jest wykorzystywany w przypadku, gdy nie zamierzamy dołączać dodatkowych pamięci czy układów rozszerzeń, jak programowalne układy wejścia/wyjścia, itp. Układ U8 jest wtedy 8-bitowym portem wejściowym, a układ U6 8-bitowym portem wyjściowym. Drugi listing przedstawia równania logiczne dla układu GAL w przypadku, gdy zamierzamy dołączać zewnętrzne układy. W tym przypadku U8 jest dwukierunkowym buforem szyny danych, a U6 jest zatrzaskiem młodszej części szyny adresowej. Należy pamiętać, że w tym przypadku wyprowadzenie 11 układu U6 trzeba połączyć z nóżką ALE mikrokontrolera. Chętni mogą oczywiście eksperymentować i zmieniać we własnym zakresie równania dla układu programowalnego GAL.

Oprogramowanie komputera PC to standardowo włączony do systemu Windows, bądź innego systemu operacyjnego program terminalowy (Hyper Terminal w przypadku Windows).

Testowanie programów wymaga użycia programu, który umożliwi transfer pliku skompilowanego (asemblerem) do pamięci RAM i późniejszą kontrolę jego wykonania. Rolę tego programu spełnia monitor.

Płytka EVM51 używa programu monitora o bardzo dużych możliwościach, a są to:

1. Automatyczna detekcja szybkości transmisji przez port szeregowy.
2. Można samemu dodawać nowe komendy monitora bez potrzeby kompilacji programu monitora, czy przeprogramowywania mikrokontrolera 89C51.
3. Download/Upload programów w formacie Intel hex (plik po kompilacji asemblerem).
4. Edycja wewnętrznej pamięci RAM.
5. Edycja zewnętrznej pamięci RAM (ponieważ pamięć RAM umieszczona na płytce może być zarówno pamięcią danych jak i pamięcią programu, dołączana komenda monitora umożliwia łatwą edycję obu ich rodzajów).
6. Wbudowany disassembler.
7. Możliwość pracy krokowej mikrokontrolera.
8. Ustawianie pułapek (*breakpoints*).
9. Wbudowana funkcja umożliwia programowanie pamięci Flash. Na razie odblokowanie/zablokowanie funkcji programowania pamięci Flash jest możliwe tylko podczas kompilacji, a to powoduje ko-

Tab. 2.

00:	*00*	*00*	*00*	*00*	*43*	*03*	*F0*	*FF*	55	55	55	55	AA	AA	AA	AA
10:	*55*	*55*	*55*	*55*	*AA*	*AA*	*AA*	*AA*	*55*	*55*	*55*	*55*	*AA*	*AA*	*AA*	*AA*
20:	55	55	55	55	AA	AA	AA	AA	55	55	55	55	AA	AA	AA	AA
30:	55	*A0*	*01*	*E6*	*07*	*0A*	*00*	*01*	*43*	*31*	*0F*	*48*	*0B*	*00*	*0B*	*AA*
40:	*55*	55	55	55	AA	AA	AA	AA	55	55	55	55	AA	AA	AA	AA
50:	55	55	55	55	AA	AA	AA	AA	55	55	55	55	AA	AA	AA	AA
60:	55	55	55	55	AA	AA	AA	AA	55	55	55	55	AA	AA	AA	AA
70:	55	55	55	55	AA	AA	AA	AA	*E2*	*33*	*AA*	*FF*	AA	AA	AA	AA

Wykorzystanie wewnętrznej pamięci RAM przez monitor. Komórki pamięci zaznaczone na szaro są wykorzystywane przez program monitora.

Rys. 5.

Tab. 3.

Adres	Zastosowanie
00...07	Pierwszy bank rejestrów
10...1F	Wykorzystywane podczas ładowania programów (komenda download)
31...40	Stos (uwaga: zewnętrzne komendy, a zwłaszcza praca krokowa i disassembler potrzebują znacznie więcej pamięci na stos)
78...7B	4 bajty przechowujące wartość wpisaną do timera T1 ustalające szybkość transmisji. Wykorzystywane w czasie restartu i tylko przez procedurę autodetekcji szybkości transmisji szeregowej. Programy mogą „zamazywać" te komórki, ale trzeba pamiętać, że wywołanie podprogramu monitora ustalającego szybkość transmisji przez port szeregowy, nadpisze te cztery bajty.

nieczność posiadania dwóch układów 89C51 z zaprogramowanym monitorem jeśli równocześnie chcemy wykorzystywać płytkę jako zwykłą płytkę uruchomieniową. Standardowo wszystkie funkcje obsługujące pamięć Flash są wyłączone, a użycie ich wymaga drobnych zmian na płytce (rozkład wyprowadzeń pamięci Flash różni się nieco od rozkładu wyprowadzeń typowej pamięci RAM.

Do komunikacji z monitorem może służyć jeden z dowolnych programów terminalowych: ProComm (MSDOS), Seyon (X11, Linux), Terminal (Windows 3.1) lub HyperTerminal (Windows). Program terminala powinien być skonfigurowany do pracy w następujący sposób: 8N1, prędkość transmisji jest ustalana automatycznie (zalecam na początek 9600 dla kwarcu 11,059MHz).

Uruchomienie przeprowadzamy następująco:

- Łączymy płytkę z komputerem za pomocą kabla z układem MAX232.
- Uruchamiamy terminal.
- Włączamy napięcie zasilające płytkę.
- Naciskamy klawisz „ENTER".

Nastąpi autodetekcja szybkości transmisji i wyświetlenie komunikatu powitalnego. Wpisanie „?" wyświetli pomoc ze spisem komend (**rys. 5**). Program monitora w czasie wykonywania komend używa wewnętrznych zasobów mikrokontrolera - niektóre zewnętrzne komendy mogą dotyczyć zewnętrznych zasobów płytki, np. zewnętrznej pamięci RAM, a więc przy wykorzystywaniu niektórych zewnętrznych komend muszą być dostępne zewnętrzne zasoby. W tym celu należy zapoznać się z dokumentacją zewnętrznych komend. Mapę wykorzystania wewnętrznej pamięci RAM mikrokontrolera przedstawiono w **tab. 2**, a przeznaczenie komórek opisano w **tab. 3**. W zasadzie większość z nich, oprócz stosu, może być wykorzystywana przez programy użytkownika, a ich nadpisanie nie powinno powodować problemów w pracy monitora. Jednak jest to ingerencja w dane monitora i w pewnych sytuacjach może powodować dziwne zachowanie się programów.

Opis standardowych komend monitora

M - List programs

Wyświetla wszystkie programy zapisane w zewnętrznej pamięci programu (przypominam, że pamięcią programu może być też pamięć RAM umieszczona na płytce i zazwyczaj to właśnie w tej pamięci są przechowywane dodatkowe programy). Program musi się zaczynać na stronie 256 bajtowej i posiadać specjalny 64 bajtowy nagłówek. Długość programu jest dowolna. Reguły pisania programów są opisane dalej.

R - Run program

Wykonuje program, który ma specjalnie zdefiniowany nagłówek.

D - Download

Download pliku w formacie Intel hex. Plik zostaje zapisany do pamięci RAM, skąd może zostać uruchomiony, a jego działanie może być monitorowane. Komenda sprawdza poprawność transmisji i wyświetla ewentualne komunikaty o błędach. Przykład:

```
Begin ascii transfer of In-
tel hex file, or ESC to
abort
.........................
Download completed

Summary:
 249 lines received
 3923 bytes received
 3923 bytes written
No errors detected
```

U - Upload

Komenda w działaniu odwrotna do poprzedniej. Przechwytując transmitowane dane do pliku możemy „zgrać" plik po modyfikacjach na nasz komputer PC. Tą funkcją można także zgrywać EPROM-y 27C256 umieszczone na płytce EVM51. Przykład pokazano na **list. 3**.

N - New location

Zmienia aktualny wskaźnik do pamięci (aktualny wskaźnik jest wyświetlany po znaku gotowości).

J - Jump to memory location

Skok do określonej lokalizacji. Umieszczony w tej lokalizacji program zostaje wykonany. Przed skokiem jest odkładany na stos adres 0000, więc jeśli program kończy się instrukcją RET to nastąpi restart monitora. Jeśli program przestaje działać, to jedyną szansą na powrót do monitora jest zerowanie mikrokontrolera.

H - Hex dump memory

„Zrzuca" na ekran 256 bajtów pamięci poczynając od aktualnej lokalizacji wskaźnika pamięci. Uwaga: wykonywane jest to komendą *movc*.

I - Hex dump internal memory

Wyświetla zawartość wewnętrznej pamięci danych. Nie wyświetla rejestrów specjalnego przeznaczenia.

E - Editing external RAM

Edycja zewnętrznej pamięci począwszy od aktualnego wskaźnika pamięci.

List. 1.

```
;74HC573 jest 8 bitowym portem wyjściowym
Equations
U245_DIR = !((!uP_P4 & !uP_RD & uP_A15) # (uP_P4 & !uP_RD & !uP_A15));
U245_G   = !((!uP_P4 & uP_A15) # (uP_P4 & !uP_A15));
U573_CLK =   (!uP_P4 & !uP_WR & uP_A15) # (uP_P4 & !uP_WR & !uP_A15);
IO_WR    = !((!uP_P4 & !uP_WR & uP_A15) # (uP_P4 & !uP_WR & !uP_A15));
IO_CS    = !((!uP_P4 & uP_A15) # (uP_P4 & !uP_A15));
RAM_OE   = !((uP_P4 & !uP_RD & uP_A15) # (!uP_P4 & !uP_RD & !uP_A15) # (!uP_P5 & !uP_PSEN & !uP_A15));
RAM_WE   = !((uP_P4 & !uP_WR & uP_A15) # (!uP_P4 & !uP_WR & !uP_A15 ));
ROM_OE   = !((uP_P5 & !uP_PSEN) # (!uP_PSEN & uP_A15 ));
```

List. 2.

```
;74HC573 jest buforem szyny adresowej
Equations
U245_DIR = !((!uP_P4 & !uP_RD & uP_A15) # (uP_P4 & !uP_RD & !uP_A15));
IO_RD    = !((!uP_P4 & !uP_RD & uP_A15) # (uP_P4 & !uP_RD & !uP_A15));
U245_G   = !((!uP_P4 & uP_A15) # (uP_P4 & !uP_A15));
IO_CS    = !((!uP_P4 & uP_A15) # (uP_P4 & !uP_A15));
IO_WR    = !((!uP_P4 & !uP_WR & uP_A15) # (uP_P4 & !uP_WR & !uP_A15));
RAM_OE   = !((uP_P4 & !uP_RD & uP_A15) # (!uP_P4 & !uP_RD & !uP_A15) # (!uP_P5 & !uP_PSEN & !uP_A15));
RAM_WE   = !((uP_P4 & !uP_WR & uP_A15) # (!uP_P4 & !uP_WR & !uP_A15 ));
ROM_OE   = !((uP_P5 & !uP_PSEN) # (!uP_PSEN & uP_A15 ));
```

C - Clear memory

Zapisuje zerami zewnętrzną pamięć. Funkcja ułatwia wychwycenie właściwych danych w pamięci, gdyż po włączeniu zasilania pamięć RAM jest zazwyczaj wypełniona przypadkowymi wartościami.

Powyższy opis jest bardzo skrótowy i dotyczy komend standardowo wbudowanych w program monitora. Nie zostały opisane funkcje umożliwiające programowanie/kasowanie pamięci Flash, gdyż prezentowana płytka nie jest przystosowana do umieszczenia tego typu pamięci, ale przystosowanie jej do tego celu jest bardzo proste i nie wymaga więcej niż 20 minut.

Dołączenie specjalnego nagłówka do pisanych programów umożliwia wykorzystanie funkcji wbudowanych w monitor. Większość z nich to funkcje obsługujące transmisje szeregową.

Jak to zostało wcześniej wspomniane, możliwe jest dodawanie komend użytkownika. Aby były one „widziane" przez monitor, muszą być właściwie skonstruowane. Oto zasady pisania tych komend:

1. Muszą się zaczynać na 256-bajtowej stronie.
2. Aby były rozpoznawalne muszą zawierać specjalnie skonstruowany nagłówek.
3. Muszą się kończyć instrukcją RET.

Punkty 1 i 3 są oczywiste. Wyjaśnienia wymaga konstrukcja nagłówka. Są 3 podstawowe typy programów rozpoznawalnych przez monitor:

- Normalny program. Nagłówek programu pojawi się podczas próby wykonania komendy *R* - „run".
- Start-up program. Użyteczny wtedy, gdy jest wbudowany w pamięć nieulotną (EPROM lub Flash). W tej wersji płytki nie jest wykorzystywany. Wykonywany jest podczas restartu oprogramowania płytki.
- Komenda.

Oczywiście każdy inny program może być wykonany przez mikrokontroler. Nie będzie on jednak rozpoznawany i wyświetlany przez program monitora. W takim przypadku do wykonania programu należy użyć instrukcji *jump* monitora.

Standardowo na dołączane komendy przeznaczona jest przestrzeń pomiędzy adresami 1000...1FFF, ale monitor wykryje także programy umieszczone w innej lokalizacji, jeśli spełniają one przedstawione tu wymagania. Pamiętać należy, że wektory przerwań mikrokontrolera są przekierowywane na adresy powyżej 2000H (np. przerwanie timera T1 zostanie przekierowane na adres 201BH). Należy o tym pamiętać pisząc programy i odpowiednio przesunąć ich początek.

List. 3.

```
First Location: 0000
Last Location: 003A
Sending Intel hex file from 0000 to 003A Press any key:
:100000000208A60220037420116A2202200B742D1C
:10001000116A22022013020A72FFFF02201BFFFF57
:10002000FFFFFF0220230209AAFFFF02202B216EFF
:0B003000016A0162215D217F218C012B
:00000001FF
```

Obecnie dostępne dodatkowe zestawy komend zawierają między innymi (**rys. 6**):

- Pracę krokową.
- Edytor pamięci (wymaga emulacji terminala VT100 - większość programów terminala to potrafi).
- Listowanie programów za pomocą programu disassemblera.
- Ustawianie pułapek (*breakpoints*).
- Edycję rejestrów specjalnego przeznaczenia.

Nie będę w tym miejscu opisywał wszystkich komend. Dokładniejsza dokumentacja jest dostępna. Jeżeli Czytelnicy wykażą zainteresowanie szerszym opisem przedstawionego tu programu monitora i możliwościami dostępnego oprogramowania płytki, z przyjemnością opiszę je w następnym artykule. Niektóre z nich stawiają pewne wymagania, ale wszystkim jest w stanie sprostać przedstawiana tu płytka. Stale przybywa nowych komend, a fakt, że mogą być umieszczane w pamięci RAM sprawia, że jest je niezwykle łatwo modyfikować i wymieniać na inne.

Trzeba tu także dodać, że ta łatwość dodawania komend jest bardzo ważna. Dysponuje bowiem programami zamieniającymi przedstawioną tu płytkę w programator mikrokontrolerów czy programator pamięci z interfejsem I^2C. Dla przykładu, programowanie pamięci 24C04 wymaga połączenia płytki z układem scalonym pamięci tylko czterema przewodami, a program do programowania tego typu pamięci, podobnie jak większość innych programów do opisywanej tu płytki, jest dostępny wraz z kodami źródłowymi za darmo.

Przemysław Dmochowski
pdmochow@wp.pl

Tab. 4.

Normalny program:

```
.equ    locat, 0x0FFF           ;Location for this program
.org    locat
.db     0xA5,0xE5,0xE0,0xA5     ;signiture bytes
.db     35,255,0,0              ;id (35=prog)
.db     0,0,0,0                 ;prompt code vector
.db     0,0,0,0                 ;reserved
.db     0,0,0,0                 ;reserved
.db     0,0,0,0                 ;reserved
.db     0,0,0,0                 ;user defined
.db     255,255,255,255         ;length and checksum (255=unused)
.db     "Program Name",0        ;max 31 characters, plus the zero
.org    locat+64                ;executable code begins here
```

Start-up program:

Jak powyżej. Zmienić należy jedynie id na wartość 253

Komenda:

```
.equ    locat, 0x8000           ;Location for this program
.org    locat
.db     0xA5,0xE5,0xE0,0xA5     ;signiture bytes
.db     254,'A',0,0             ;id (254=command,
                                ;key='A'- wielka litera !!)
.db     0,0,0,0                 ;prompt code vector
.db     0,0,0,0                 ;reserved
.db     0,0,0,0                 ;reserved
.db     0,0,0,0                 ;reserved
.db     0,0,0,0                 ;user defined
.db     255,255,255,255         ;length and checksum (255=unused)
.db     "Command Name",0        ;max 31 characters, plus the zero
.org    locat+64                ;executable code begins here
```

Rys. 6.