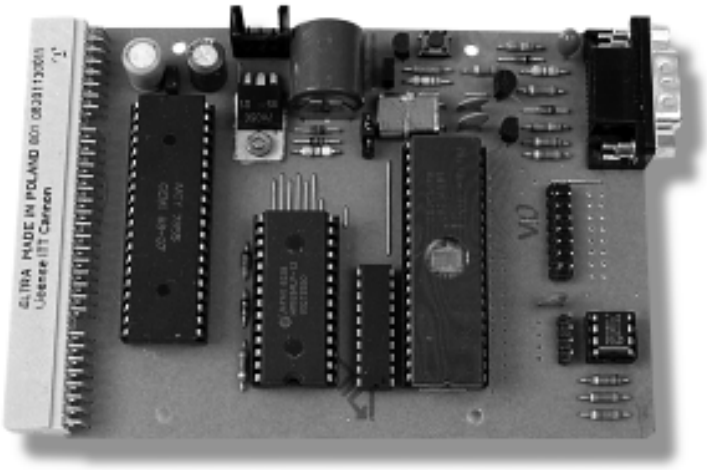
Mikroprocesorowy system edukacyjny, część 3

kit AVT-353

Jest to przedostatnia część artykułu poświęconego opisowi konstrukcji mikroprocesorowego systemu edukacyjnego. Prezentujemy w niej oprogramowanie i procedury opracowane przez autora.



Możliwości i wykorzystanie systemu operacyjnego

Na dostarczonej wraz z zestawem dyskietce znajdują się dwa pliki tekstowe - *sysop.def* oraz *sysop.ref*. Pierwszy z nich zawiera definicje adresów procedur systemu i powinien być dołączony do treści źródłowej programu użytkownika. Drugi plik - *sysop.ref* jest ściągawką zawierającą krótki opis tych procedur i dalej traktowany będzie jako przewodnik podczas opisu. Przedstawienie każdej procedury składa się z nagłówka (słowo kluczowe - DZIAŁANIE), prezentującego w zwięzły sposób wykonywaną przez nią operację, następnie określone zostają ewentualne parametry wejściowe oraz wyjściowe (słowa kluczowe - IN, OUT), po czym (po słowie UŻYWA) wyszczególniono angażowane rejestry procesora wraz z opcjonalnym określeniem zajmowanego banku rejestrów. Zaznaczyć należy, że system operacyjny generalnie używa zerowego banku rejestrów (bity RS0, i RS1 w słowie PSW równe zero), jakkolwiek możliwe jest użycie dowolnego spośród czterech dostępnych, ale tylko dla procedur oznaczonych inskrypcją „BANK aktualny". Procedury systemu operacyjnego używają także wskaźników operacyjnych w słowie stanu programu (CY,AC,OV i P), co nie jest zaznaczone w ich skróconym opisie.

Kolejną częścią opisu procedury w pliku *sysop.ref* jest podanie liczby zajmowanych bajtów na stosie systemowym (po słowie kluczowym STOS). Jest to niezbędna rezerwa wymagana przez daną procedurę, jednak w kalkulacji rozmiaru stosu uwzględnić należy jego obciążenie wnoszone dodatkowo przez procedury obsługi przerwań zaimplementowane w programie użytkownika. Ostatnim elementem opisu każdej procedury jest jej nazwa zastrzeżona wraz z definicją adresu wywołania. Nadmienić należy, że jakakolwiek zmiana adresu podanego w definicji uniemożliwi działanie tak zmodyfikowanych procedur systemu, prowadząc najczęściej do zawieszenia się programu użytkownika.

Tak więc plik *sysop.ref* można zamiennie, z plikiem *sysop.def*, dołączać do treści programu użytkownika - z punktu widzenia programu użytkownika oba spełniają identyczną rolę. Różnica polega jedynie na przekazywanej treści dla samego użytkownika. System operacyjny zajmuje także jedną komórkę wewnętrznej pamięci RAM mikrokontrolera o adresie 20h. Program użytkownika nie powinien modyfikować jej zawartości w jakikolwiek sposób, poza dopuszczonym przez te procedury systemu, które wykorzystują ją do własnych celów.

Generalnie ogólna struktura programów użytkownika powinna wyglądać następująco:

```
ORG 8000h ;adres bazowy pamięci RAM U2
SJMP START
;skok do początku programu
ORG 8003h
;podprogram obsługi przerwania INT0
...          ;treść podprogramu
RETI
;
ORG 800Bh
;podprogram obsługi przerwania CT0
...          ;treść podprogramu
RETI
;———————
START:
x         ;początek programu głównego
x
LCALL RESETLcd
;odwołanie do procedur systemu
MOV A,#28h
;inicjacja wyświetlacza dwuliniowego
LCALL WRITEord
MOV A,#0Ch
;zapis rozkazu do kontrolera LCD
LCALL WRITEord
;odpowiada sekwencji !@0C@01
LCALL CLEARdisp   ;w programie lcd4.exe
...
;———————
$include(sysop.def)
;dołączenie definicji adresów systemu
;———————
END       ;koniec programu
```

Czytelnicy zauważą zapewne, że odwołania do procedur systemu operacyjnego są realizowane jako dalekie wywołania podprogramów. Specyfika ta spowodowana jest faktem umieszczenia programów użytkownika w górnym segmencie pamięci programu, natomiast system operacyjny rezyduje w segmencie dolnym. Jedynym rozkazem mikrokontrolera zdolnym do obsługi takiego wywołania podprogramu jest LCALL. Dodatkową koniecznością staje się wprowadzenie stałego przesunięcia do adresów wektorów zerowania i przerwań. Tak więc jest konieczna zamiana według poniżej opisanego klucza:

```
0|—> 8000h    ;zerowanie systemu
3|—> 8003h    ;przerwanie INT0
0Bh —> 800Bh  ;przerwanie CT0
13h —> 8013h  ;przerwanie
              ; INT1 1Bh —> 801Bh
              ; przerwanie
              ; CT1 23h —> 8023h
              ; przerwanie UART
```

Po treści programu użytkownika dyrektywą makroasemblera dołączony zostaje plik definicji adresów systemu operacyjnego. Oczywiście, można tę operację wykonać poprzez zwykłe „sklejenie" dwóch plików testowych, przy pomocy opcji zastosowanego edytora - do pliku programu dołączyć należy plik definicji.

Opis procedur systemu operacyjnego przeprowadzony zostanie w kolejności ich występowania w pliku *sysop.ref*. Jako pierwszy znajduje się tam blok operacji arytmetycznych:

DIVIDE - procedura dzieląca czterobajtową dzielną przez dwubajtowy dzielnik, w wyniku czego otrzymywany jest dwubajtowy wynik oraz, jeżeli wynik dzielenia nie jest przepełnieniem, dwubajtową resztę modulo. Przepełnienie uzyskiwane jest jeżeli wynik dzieienia będzie większy od szesnastobitowej liczby binarnej (czyli 65535) i sygnalizowane jest wynikiem składającym się z samych jedynek (czyli FFFFh).

DIVI - procedura dzieląca czterobajtową dzielną przez jednobajtowy dzielnik, w wyniku czego otrzymywany jest czterobajtowy wynik, oraz jednobajtowa reszta modulo. Procedura nie sygnalizuje wystąpienia przepełnienia.

MULTIPLE - procedura mnożenia dwubajtowej mnożnej i mnożnika, w wyniku czego otrzymywany jest czterobajtowy wynik.

MULTI - procedura mnożenia czterobajtowej mnożnej i jednobajtowego mnożnika, w wyniku czego otrzymywany jest czterobajtowy wynik, wraz z ewentualnym jednobajtowym przeniesieniem do R6.

Nadmienić należy, że wszystkie procedury arytmetyczne systemu posługują się liczbami naturalnymi (nieujemnymi) w zapisie dwójkowym lub opcjonalnie szesnastkowym (heksadecymalnym).

Para rejestrów R3, R2, stanowi akumulator szesnastobitowy, przechowujący liczbę według schematu: R3 - starszy bajt, R2 - młodszy. Przyjęty sposób zapisu, w którym rejestry o coraz większych numerach identyfikacyjnych przechowują coraz bardziej znaczące cyfry lub bajty, jest zgodny z naturalnym sposobem zapisu liczb dziesiętnych (zawsze zaczynamy od najbardziej znaczącej pozycji). Dysponując tym zestawem, zrealizować można obliczenie według wzoru, np.

```
3275x25
—— = ?
     1087
```

co w asemblerze zastosowanego mikrokontrolera zapisać można:

```
PRZYKŁAD_1:
MOV R2,#0CBh
   ; załadowanie mnożnej (3275)
MOV R3,#0Ch
MOV R4,#0
MOV R5,#0
MOV R6,#25 ; załadowanie mnożnika (25)
LCALL MULTI
; wywołanie procedury systemu,
; wynik w|R5,R4,R3,R2 (R6 - ignorowany)
MOV R6,#3Fh
;załadowanie dzielnika (1087)
MOV R7,#4
LCALL DIVIDE   ;R3,R2 - wynik, czyli 75
          ;(część ułamkowa jest tracona)
```

czy też np.

```
1278x0.52 = ?
```

co zapisać można:

```
PRZYKŁAD_2:
MOV R2,#0FEh ;załadowanie mnożnej (1278)
MOV R3,#4
MOV R6,#52    ;załadowanie mnożnika (52)
MOV R7,#0
LCALL MULTIPLE    ;wynik w|R5,R4,R3,R2
MOV R6,#100
LCALL DIVI    ;po podzieleniu przez 100
;otrzymujemy w R5,R4,R3,R2 właściwy wynik
```

Zauważyć należy, że w tym przypadku zastosowano przeskalowanie ułamków, czyli mnożenie ma postać 1278x52=66456. Więc, aby uzyskać właściwy wynik, należy dodatkowo wykonać dzielenie przez współczynnik przeskalowywujący ułamek (w tym wypadku jest to 100), aby otrzymać wymagane 664.

Kolejne procedury arytmetyczne powodują zwiększenie i zmniejszenie zawartości zespołu rejestrów, tworzących licznik binarny:

INCR - zwiększenie zawartości licznika o jeden (inkrementacja);

DECR - zmniejszenie zawartości licznika o jeden (dekrementacja).

Zwiększenie lub zmniejszenie zawartości licznika zapisać można następująco:

```
PRZYKŁAD_3:
MOV R1,#LICZNIK_0
;adres (nie wartość) najmłodszego bajtu
MOV R2,#3     ;trzy bajty
LCALL INCR (lub DECR)
;———————
LICZNIK_0 EQU 30h
;trzy bajty pamięci definiujące
```

```
LICZNIK_1 EQU 31h  ;licznik binarny
LICZNIK_2 EQU 32h
```

Dodatkowo, w przypadku procedury dekrementacji, jest sygnalizowana przez stan wskaźnika CY zmiana znaku liczby przechowywanej w liczniku, występująca przy przejściu z zera na -1. Odpowiada to, w kodzie uzupełnień do dwóch (U2), zapisaniu do wszystkich bajtów tworzących licznik samych jedynek. Dzięki temu istnieje prosta możliwość odliczania zadanej liczby zdarzeń - na początku należy załadować do licznika wartość pomniejszoną o jeden, po czym kolejno dekrementować licznik, aż do chwili, gdy wskaźnik CY będzie równy 1.

Kolejnym blokiem procedur są podprogramy konwersji dziesiętno-binarnej oraz heksadecymalno-binarnej:

BIN2BCD - procedura zamienia liczbę binarną umieszczoną w rejestrach R3,R2 na liczbę dziesiętną, składowaną w postaci rozpakowanej w rejestrach R7,R6, R5,R4, zawierających kolejno cyfry tysięcy, setek, dziesiątek, oraz jednostki w postaci: 00 do 09 (zapis skrótowy - 0x). Dodatkowo, jeżeli wyjście z procedury odbywa się z ustawionym wskaźnikm CY, to konwertowana liczba jest większa od 9999, czemu towarzyszy umieszczenie w R2 cyfry dziesiątek tysięcy (postać 0x), a jeżeli wskaźnik CY pozostaje wyzerowany, to zawartość R2 będzie przypadkowa.

BCD2BIN - zamienia liczbę dziesiętną, umieszczoną w rejestrach R7, R6, R5, R4 (według powyżej opisanej konwencji), na liczbę binarną, zwracaną w R3, R2.

HEX2B - dokonuje zamiany znaku ASCII, reprezentującego cyfrę heksadecymalną umieszczonego w rejestrze A, na odpowiednik binarny, zwracany w A w postaci od 00h do 0Fh (zapis skrótowy - 0xh). Litery małe i duże traktowane są równorzędnie. Podanie znaku nie będącego cyfrą heksadecymalną da w odpowiedzi nieokreśloną wartość.

B2HEX - zamiana czterech najmłodszych bitów liczby binarnej umieszczonej w rejestrze A na znak ASCII reprezentujący cyfrę heksadecymalną. Cztery najstarsze bity A pozostają nieistotne dla wyniku (zapis skrótowy - Nxh).

W zestawie procedur sterujących alfanumerycznym wyświetlaczem LCD znajdują się:

RESETlcd - bezparametrowa procedura inicjacji kontrolera wyświetlacza dokonuje przełączenia w tryb komunikacji cztrobitowej, ustawia obsługę jednej linii o długości 80 znaków, odpowiada <!> z programu lcd4.exe.

CRSRleft - bezparametrowa procedura realizująca przesunięcie kursora o znak w lewo, odpowiada <@10>.

CRSRright - bezparametrowa procedura realizująca przesunięcie kursora o znak w prawo, odpowiada <@14>.

CRSRhome - bezparametrowa procedura realizująca ustawienie kursora i okna wyświetlacza od pozycji początkowej linii, odpowiada <@02>.

CLEARdisp - bezparametrowa procedura realizująca kasowanie wyświetlacza oraz ustawienie kursora i okna wyświetlacza od pozycji początkowej linii, odpowiada <@01>.

WRITEspc - bezparametrowa procedura zapisująca znak spacji na aktualnie wskazywane kursorem pole wyświetlacza.

WRITEdata - zapis kodu ASCII umieszczonego w rejestrze A na wskazywane kursorem pole wyświetlacza. Podanie kodu z zakresu 0 do 7 spowoduje wyprowadzenie na wyświetlacz określonego znaku definiowanego przez użytkownika.

WRITEord - zapis rozkazu umieszczonego w rejestrze A do kontrolera wyświetlacza, odpowiada ogólnej sekwencji <@hh>.

CHARdef - procedura definiująca znaki użytkownika. Definicja rozpoczyna się zawsze od znaku zerowego. Po wywołaniu procedury należy podać liczbę definiowanych znaków, po czym kolejno ośmiobajtowe definicje kolejnych znaków. Ostatni bajt definicji znaku musi zawsze być równy zero, procedurę należy wywoływać zawsze po pełnej sekwencji inicjacji kontrolera wyświetlacza.

Użycie według przykładu:

```
PRZYKŁAD_4:
LCALL CHARdef ;wywołanie procedury
DB 3      ;definicja trzech znaków
DB 0Eh,1,0Fh,11h,0Fh,2,1,0
   ;definicja znaku numer 0
DB , , , , , ,0 ;definicja znaku numer 1
DB , , , , , ,0 ;definicja znaku numer 2
...|; ciąg dalszy programu
```

Uważny Czytelnik zauważy z pewnością, że opisywane powyżej procedury realizują rozkazy zaliczone podczas opisu programowania kontrolera wyświetlacza do grupy drugiej i trzeciej.

Przykładowa inicjacja sterownika wyświetlacza po restarcie systemu uwidoczniona została wcześniej pod hasłem „ogólnej struktury programów użytkownika“. Sekwencja ta jest analogiczna do stosowanej podczas inicjacji wyświetlacza, wymaganej na początku pracy z programem lcd4.exe. Instrukcje asemblera:

```
MOV A,#28h
LCALL WRITEord
```

są wymagane jedynie przy wyświetlaczu o organizacji dwuliniowej. Dla jednoliniowego spowodują znaczne obniżenie kontrastu wyświetlanych znaków.

Krzysztof Kuryłowicz