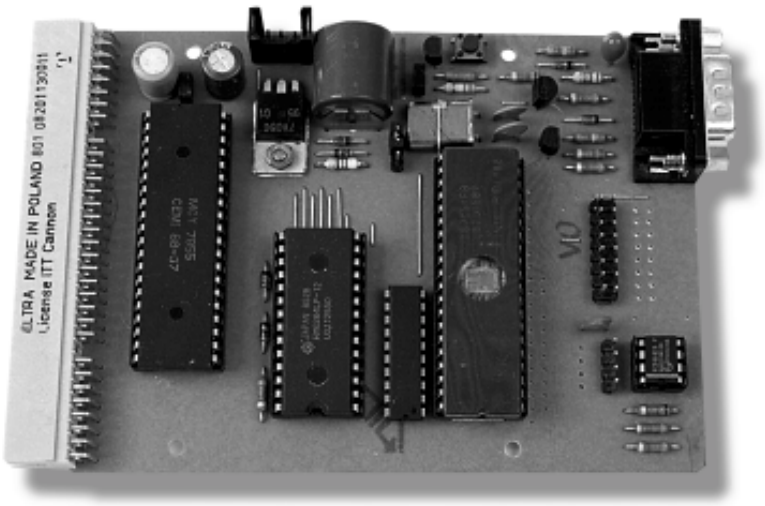
Mikroprocesorowy system edukacyjny, część 4

kit AVT-353



Jest to ostatnia część artykułu prezentującego konstrukcję mikrokomputera edukacyjnego z procesorem rodziny '51. Omawiamy w niej dalsze szczegóły związane z oprogramowaniem "zaszytym" w pamięci procesora.

Kolejne procedury powodują inicjację oraz obsługę wymiany danych łączem szeregowym RS232:

INITrs - procedura inicjacji układu UART oraz licznika/czasomierza T1 określającego właściwą prędkość transmisji. Można wybrać dwie prędkości zależne od stanu wskaźnika CY: 2400Bd dla CY=0 lub 4800Bd dla CY=1.

Zasadniczo system operacyjny powinien realizować transmisję z większą prędkością, np. podczas komunikacji z programem recv51.exe. Możliwość pracy z niższą częstotliwością pozostawiono użytkownikom dysponującym większym doświadczeniem w dziedzinie programowania, gdyż wymaga ona własnoręcznego oprogramowania protokołu od strony urządzenia współpracującego z systemem uruchomieniowym.

Przy opisie procedury inicjacji zaznaczyć należy konieczność umieszczenia jej wywołania dopiero po inicjacji licznika/czasomierza T0. Fakt ten tłumaczyć należy możliwością naruszenia parametrów konfiguracyjnych licznika/czasomierza T1 używanego do określenia prędkości transmisji, jakie może mieć miejsce przez nieumiejętnie przeprowadzony zapis parametrów licznika/czasomierza T0.

Jeżeli użytkownik zamierza w swoich programach korzystać z procedur obsługi łącza szeregowego, należy zwrócić uwagę na bit blokujący zgłoszenie przerwania od licznika/czasomierza T1, który generalnie powinien pozostawać wyzerowany - co wyłącza korespondujące z nim przerwanie. Oczywiście, można wykorzystać je w sposób celowy, jednak należy mieć na uwadze częstotliwość powtarzania, równą około 76,8kHz, co stawia bardzo wysokie wymagania czasowe dla procedury jego obsługi. Bardziej sensownym sposobem użycia przerwania od licznika/czasomierza T1 jest zaprogramowanie licznika/czasomierza T0 w trybie 3. (mode 3), gdyż został on wprowadzony celem obsługi powyżej opisywanego przypadku.

Postać inicjacji liczników oraz systemu przerwań, realizowana na początku programu użytkownika, powinna wyglądać następująco:

```
PRZYKŁAD_5:
x  ;inicjacja licznika T0
SETB C
LCALL INITrs
   ;inicjacja UART, oraz licznika T1
...
MOV IE,#xxxx0xxx
   ;przewijanie licznika T1 pozostaje
x  ;zablokowane
```

W grupie procedur systemu

operacyjnego, realizujących elementarną komunikację szeregową znajdują się jeszcze:

RCHAR - odbiera pojedynczy znak, zwraca w znaczniku CY stan świadczący o braku (wyzerowany) lub obecności (ustawiony) znaku w buforze odbiornika. W razie obecności znak przekazywany jest w rejestrze A.

Najprostszy sposób adaptacji tej informacji wyglądać będzie następująco:

```
PRZYKŁAD_6:
...
WAITchar:
LCALL RCHAR   ;odbierz znak
JNC WAITchar  ;brak znaku
...           ;A - odebrany znak,
```

gdzie fragment programu, oznaczony jako WAITchar, pozostanie w pętli tak długo, aż odebrany zostanie znak, przekazany następnie w rejestrze akumulatora.

SSPC - bezparametrowa procedura nadająca łączem szeregowym znak spacji.

SCHAR - nadanie łączem szeregowym znaku o kodzie umieszczonym w rejestrze A.

RHEX - procedura odbiera łączem szeregowym dwa znaki ASCII traktowane jako cyfry heksadecymalne, z których skompletowany zostanie bajt w postaci binarnej, w którym pierwszy znak odpowiada czterem najstarszym bitom, a drugi czterem najmłodszym bitom bajtu. Po kompletacji bajt zwrócony zostanie w rejestrze A - w razie przerwania transmisji w trakcie odbioru znaków procedura spowoduje zatrzymanie wykonywania programu użytkownika aż do czasu ponownego ustanowienia łączności.

Kolejna grupa procedur systemu operacyjnego jest przeznaczona do prezentacji wyników pracy programów użytkownika, ujętych w koncepcję strumieni danych:

SETdev - procedura w zależności od stanu wskaźnika CY określa urządzenie wyprowadzające, do którego przesyłane będą wypracowane przez program dane. Dla CY=0 strumień kierowany jest do łącza RS232, a dla CY=1 do panela LCD. Zmienną przechowującą stan logiczny określający aktualnie wybrane urządzenie wyjściowe jest bit GF0 (rejestr specjalny PCON).

PUTspc - bezparametrowa procedura wyprowadzająca znak spacji na aktualnie wybrane urządzenie.

PUTchar - wyprowadza znak o kodzie umieszczonym w rejestrze A. Jeżeli urządzeniem jest panel LCD, to znak o kodzie 10 (LF) przesunie kursor na pozycję początkową wyświetlacza (odpowiada <@02>), natomiast znak o kodzie 13 (CR) skasuje bieżącą zawartość wyświetlacza (obpowiada <@01>).

SHEX - umieszczony w rejestrze A bajt zamieniony zostanie na dwa znaki ASCII reprezentujące heksadecymalną postać kolejno czterech starszych oraz czterech młodszych bitów, tak uformowany dwuznakowy ciąg zostanie następnie przesłany przez aktualnie wybrane urządzenie wyprowadzające.

Zawartość pary rejestrów R3,R2 przechowującej 16-bitową liczbę binarną, można zobrazować w postaci heksadecymalnej następująco:

```
PRZYKŁAD_7:
...
LCALL PUTspc
;przesłanie do strumienia spacji
MOV A,R3      ;starszy bajt
LCALL SHEX    ;przesłanie do strumienia
MOV A,R2      ;młodszy bajt
LCALL SHEX    ;przesłanie
...
```

SMESS - wyprowadza ciąg znaków umieszczony za wywołaniem procedury na aktualnie wybrane urządzenie wyjściowe. Ciąg musi być zakończony bajtem o wartości 0 (zero). Zakończenie ciągu znaków bajtem o innej wartości spowoduje wyprowadzanie kolejnych bajtów treści programu, aż do napotkania komórki o zerowej wartości.

Poniższy przykład ilustruje poprawne wywołanie procedury:

```
PRZYKŁAD_8:
LCALL SMESS   ;wywołanie procedury
DB 'Tekst przykładowy',0
; ciąg znaków zakończony zerem
...           ; dalsza treść programu
```

SNUMmess - spośród kilku ciągów tekstowych, umieszczonych za wywołaniem procedury, wyprowadza ciąg określony numerem przekazywanym w rejestrze A. Podanie numeru przewyższającego zadeklarowaną liczbę, jak i podanie bezpośrednio po wywołaniu procedury innej niż zdeklarowana, liczba ciągów spowoduje zawieszenie programu użytkownika, ciągi zakończone muszą być znakiem w wartości 0.

Poniższy przykład ilustruje poprawne wywołanie procedury:

```
PRZYKŁAD_9:
MOV A,#2
   ;wyprowadzony zostanie ciąg numer 2
LCALL SNUMmess
DB 3          ;liczba zdeklarowanych ciągów
;deklaracje kolejnych trzech ciągów
DB 'Tekst pierwszy',0
DB 'Tekst drugi',0
DB 'Tekst trzeci',0
...  ; kontynuacja programu użytkownika
```

Sposób działania powyższych dwu procedur uniemożliwia wyprowadzenie na wyświetlacz LCD znaku definiowanego o numerze 0. Możliwe jest to tylko przez wywołanie procedury <WRITEdata>, po wcześniejszym zapisaniu do rejestru akumulatora (A) wartości zero, np. rozkazem <CLR A>. Pozostałe siedem znaków definiowanych jest dostępne po podaniu stosownego numeru (od 1 do 7).

Następne dwie procedury służą do formatowego wyprowadzania liczb dziesiętnych:

DIGIdispl - wyprowadza rozpakowaną liczbę dziesiętną umieszczoną w rejestrach R7, R6, R5, R4, przekazujących kolejno cyfry tysięcy, setek, dziesiątek oraz jednostek, według klucza określonego zawartością rejestru A, odpowiadającego bezpośrednio opisowi statusu wyświetlania DIGstat (adres 20h), do którego zapisana zostanie następnie zawartość rejestru akumulatora. Procedura realizuje wyprowadzenie liczby w formacie ustalonym, tj. wszystkie cztery cyfry, znak liczby, jak i separator zostaną wyprowadzone według wspomnianego klucza, gdzie kolejne linie odpowiadają bitom rejestru A:

```
0|- pozycja separatora
1|- ––– ––––– : cyfra wg wagi - 3|2|1|0
2|- ksztalt separatora
3|- ––– ––––– : 0-(.)/1-(:)/2-(-)/3-(_)
4|- znak liczby : 0-plus/1-minus
5|- nieistotne
6|- nieistotne
7|- nieistotne;
```

Wpisanie do rejestru A wartości np.6 wyprowadzi liczbę w postaci: <_ts:dj>, a np. 19h spowoduje wyprowadzenie liczby w po-

staci: <-tsd-j>. Najprostszy przypadek ma miejsce po wpisaniu do rejestru A wartości zero, kiedy to liczba wyprowadzona zostanie w naturalnej postaci: <_tsdj.>. Powyższy symboliczny zapis określa kolejno cyfry <t>ysięcy, <s>etek, <d>ziesiątek oraz <j>ednostek. Dodatkowo znak <_> symbolizuje spację, znak <-> symbolizuje minus, bądź myślnik, a znak <.> kropkę dziesiętną. Nadmienić należy, że opisywana procedura wyposażona została także w funkcję wygaszania zer nieznaczących, tj. np. liczba 327 wyprowadzona zostanie jako <_327>, a nie jako <0327>. Działaniu jej towarzyszy zawsze wyprowadzenie sześciu znaków, co zostało zaznaczone w opisie jako przemieszczenie kursora o tyleż pozycji.

Opis powyższy nie wyczerpuje wszystkich możliwości procedury <DIGIdispl>. Konieczne staje się więc przeprowadzenie własnych doświadczeń według wzoru:

```
PRZYKŁAD_10:
MOV R2,#xxh ;<--- młodszy bajt liczby
MOV R3,#0xh ;<--- starszy bajt liczby
LCALL BIN2BCD
; konwersja binarno/dziesiętna
MOV A,#xxh ;<--- klucz wyprowadzania
LCALL DIGIdispl
;wyprowadzenie do wcześniej określonego
;procedurą <SETdev> urządzenia
...       ;kontynuacja programu,
```

lub:

```
PRZYKŁAD_11:
MOV R4,#0xh
;cyfra jednostek dec/hex tj. 0|do Fh
MOV R5,#0xh   ;cyfra dziesiątek
MOV R6,#0xh   ;cyfra setek
MOV R7,#0xh   ;cyfra tysięcy
MOV A,#xxh ;<--- klucz wyprowadzania
LCALL DIGIdispl
;wyprowadzenie do wcześniej określonego
;procedurą <SETdev> urządzenia
...       ;kontynuacja programu
```

Pamiętać jednak należy, że procedura formatowego wyprowadzania liczby obsługuje jedynie cztery cyfry. Tak więc liczba większa od 9999 przedstawiona zostanie, po odcięciu cyfry dziesiątek tysięcy, jako cztery mniej znaczące pozycje (np. 65535 jako 5535). Pozbawiona tej wady jest kolejna procedura:

DIGIdis - realizuje wyprowadzenie liczby według formatu nieokreślonego. Przed wywołaniem wymaga wpisania do rejestru statusu wyświetlania DIGstat (adres 20h) wartości określającej sposób prezentacji liczby. Wpisanie do rejestru R1 pozycji początkowej wskaźnika np. wartości 6 wyprowadzi liczbę przekazywaną kolejno w rejestrach: R6,R5,R4.

Przykładowe wyprowadzenie liczby większej od 9999 wyglądać będzie następująco:

```
PRZYKŁAD_12:
MOV R2,#6Eh   ;młodszy bajt liczby 45678
MOV R3,#0B2h  ;starszy bajt liczby 45678
LCALL BIN2BCD
;konwersja binarno/dziesiętna
JNC PRZ12_1   ;dla liczby <10000
MOV A,R2
;liczba >9999 - cyfra dziesiątek tysięcy
ADD A,#48
;przesunięcie cyfry do znaku ASCII
LCALL PUTchar
;wyprowadzenie cyfry dziesiątek tysięcy
MOV A,#40h
;kolejne zera znaczące, liczba bez znaku
SJMP PRZ12_2

PRZ12_1:
MOV A,#0
;znak liczby przedstawiony będzie jako
;spacja, analiza znaczenia zer PRZ12_2:
MOV DIGstat,A
;rejestr statusu wyświetlania
MOV R1,#7
;od cyfry tysięcy (w rejestrze R7)
LCALL DIGIdis
;wyprowadzenie pozostałych czterech cyfr
...       ;kontynuacja programu
```

Zasadnicza różnica pomiędzy procedurą <DIGIdispl>, a <DIGIdis> polega na implementacji dodatkowego, 6. bitu w rejestrze statusu <DIGstat>. Jego wyzerowanie wyprowadzi kolejne cyfry o wartości zero, znajdujące się na pozycjach nieznaczących (z strony lewej pierwszej cyfry o wartości większej od zera) w postaci spacji. Dodatkowo przed pierwszą cyfrą większą od zera umieszczony zostanie znak liczby stosownie do stanu czwartego bitu rejestru <DIGstat>. Cyfry o wartości większej od zera, lub zerowej, lecz występujące po prawej stronie pierwszej cyfry znaczącej zostaną wyprowadzone stosownie do swojej wartości. Ustawienie wyżej wymienionego bitu spowoduje, że wszystkie kolejne cyfry będą traktowane w sposób identyczny, tj. bez rozdziału na równe czy większe od zera - wszystkie zera przedstawione będą jako znaczące. Pominięty zostanie także zapis znaku liczby, dzięki czemu liczba nie będzie poprzedzona spacją czy znakiem minus. Umożliwia to proste „sklejanie“ liczb o większej niż cztery liczbie cyfr (uwidoczniono to w ostatnim przykładzie).

Procedura <DIGIdis> umożliwia wyprowadzenie mniejszej niż cztery liczby cyfr, co jest osiągane przez wpisanie do rejestru R1 wartości wskazującej na pierwszy interesujący nas rejestr. Przykładowo wpisanie np. 5 spowoduje wyprowadzenie cyfr dziesiątek oraz jednostek przechowywanych odpowiednio w rejestrach R5 i R4.

Ostatnią grupą procedur systemu operacyjnego są podprogramy współpracy z urządzeniami podłączonymi do magistrali I2C:

I2Copen - realizuje przesłanie jednego bajtu danych lub otwarcie połączenia przy przesyłaniu większej niż jeden liczby bajtów. Wymaga podania adresu urządzenia z wyzerowanym bitem najmniej znaczącym (w postaci binarnej: xxxxxxx0). Podanie adresu urządzenia nie odpowiadającego fizycznie podłączonemu do magistrali I2C układowi spowoduje zawieszenie programu użytkownika.

I2Cwrite - realizuje zapis kolejnych bajtów danych do urządzenia, dla którego uprzednio otwarto połączenie.

I2Cread - realizuje odczyt kolejnych bajtów danych z urządzenia, dla którego uprzednio otwarto połączenie.

Procedury obsługi łącza I2C automatycznie dekrementują wskaźnik liczby pozostałych do przesłania bajtów (rejestr R1) oraz inkrementują wskaźnik subadresu urządzenia (rejestr R3), ułatwiając dostęp do kolejnych komórek, np. pamięci. Zapis jednego bajtu, do umieszczonej na druku systemu uruchomieniowego pamięci EEPROM (U5), zrealizuje sekwencja rozkazów:

```
PRZYKŁAD_13:
MOV A,nn  ;dana do zapisania w|rejestrze A
SETB bitDIR   ;zapis do EEPROM
MOV R1,#1     ;transmisja jednego bajtu
MOV R2,#0A0h  ;adres grupowy pamieci U5
MOV R3,#nn    ;subadres,czyli właściwa
              ;komórka pamięci
ACALL I2Copen
;wybranie EEPROM, zapis bajtu
...       ;kontynuacja
```

Analogicznie odczyt zrealizuje sekwencja:

```
PRZYKŁAD_14:
```

```
CLR bitDIR    ;odczyt z|EEPROM
MOV R1,#1     ;transmisja jednego bajtu
MOV R2,#0A0h  ;adres grupowy pamieci U5
MOV R3,#nn    ;subadres,czyli właściwa
              ;komórka pamięci
ACALL I2Copen ;wybranie EEPROM,
              ;odczyt bajtu
MOV nn,A      ;odczytany bajt w|A
...       ;kontynuacja
```

Aby przesłać większą niż jeden liczbę bajtów należy posłużyć się sekwencją:

```
PRZYKŁAD_15:
SETB bitDIR   ;zapis danych do EEPROM
MOV R1,#0     ;otwarcie połączenia
MOV R2,#0A0h  ;adres grupowy pamięci
MOV R3,#nn    ;subadres, adres pierwszej
              ;komórki
ACALL I2Copen ;otwarcie połączenia
...  ;inne czynności, połączenie otwarte
MOV R1,#2     ;transmisja dwu bajtów
MOV A,nn      ;pierwszy
ACALL I2Cwrite    ;zapis
...           ;inne czynności
MOV A,nn      ;drugi
ACALL I2Cwrite    ;zapis i|zerwanie
              ;połączenia
...           ;kontynuacja
```

W treści powyższych przykładów zauważyć można konieczność każdorazowego określenia kierunku transmisji przed wywołaniem procedury <*I2Copen*>. Jest to spowodowane wymaganiem przesłania, jako pierwszego, bajtu wybierającego dane urządzenie podłączone do magistrali I2C (w tym przypadku pamięć EEPROM - (U5) - jest wybierana przez adres grupowy A0h) wraz z określeniem kierunku przyszłej transmisji danych. Jako kolejny bajt jest wysyłany subadres pamięci określający bezpośrednio komórkę pamięci, do (bądź z-) której realizowana będzie transmisja. Po przekazaniu bajtu danych połączenie zostanie samoczynnie zerwane.

Nieco odmienna sytuacja występuje przy realizacji transmisji kilku bajtów.

W pierwszej kolejności procedura <*I2Copen*> wykonuje otwarcie połączenia, wybierając dane urządzenie. Kolejno zostanie przesłany subadres pamięci, określający dostęp do pierwszego bajtu, dla którego realizowana będzie transmisja. Następnie do rejestru R1 zostaje wpisana liczba transmitowanych bajtów. Sam proces przesyłania danych realizują procedury: <*I2Cwrite*> dla zapisu do wybranego urządzenia oraz <*I2Cread*> dla odczytu. Po przesłaniu zadanej liczby bajtów, połączenie zostanie automatycznie przerwane. Liczba transmitowanych w jednym połączeniu bajtów zależy od typu obsługiwanego w danej chwili układu i tak np. dla zastosowanej w systemie pamięci EEPROM wynosi ona maksymalnie 8 bajtów na cykl.

Oczywiście, dostęp do umieszczonej na płytce systemu pamięci nie wyczerpuje wszystkich możliwości interfejsu I2C. W zależności od aplikacji zastosowanych podzespołów, odpowiednio skonfigurować należy przebieg transmisji łączem I2C. Wspomnieć należy jeszcze o występującym ograniczeniu, którym jest rezerwowanie tej samej komórki wewnętrzej pamięci RAM mikrokontrolera przez procedury formatowego wyprowadzania liczby (<DIGstat>), jak i obsługi łącza (<I2Cstat>). Powoduje to konieczność rozdziału tych funkcji, tj. w trakcie otwarcia połączenia przy wielobajtowej transmisji nie należy używać procedur formatowego wyprowadzania liczb. Jeżeli jednak konieczności tej nie da się uniknąć, przed wywołaniem procedur wyprowadzania liczb ochronić należy komórkę wspólną pamięci o adresie 20h, przy pomocy rozkazu:

```
MOV nn,I2Cstat,
```

natomiast po zrealizowanym wyprowadzeniu liczby komórkę tę należy odtworzyć:

```
MOV I2Cstat,nn,
```

gdzie: nn - dowolna inna komórka wewnętrznej pamięci RAM.

Ograniczenie powyższe nie dotyczy jednobajtowych transmisji łączem I2C.

Celem ilustracji niektórych procedur systemu operacyjnego, jak i sposobu ich użycia, na dostarczonej wraz z zestawem dyskietce znajduje się osiem programów przykładowych, umieszczonych w plikach oznaczonych kolejno <*test_1*> do <*test_8*>.

Krzysztof Kuryłowicz