

W kolejnym odcinku naszego cyklu zapoznamy się z pracą dodatkowego układu czasowo licznikowego, który występuje w mikrokontrolerach 8052/C. W drugiej części artykułu przedstawię specjalne tryby pracy procesorów, dzięki którym możliwe jest konstruowanie energooszczędnych autonomicznych układów bateryjnych. I choć na tym etapie nauki panowania nad mikrokontrolerem 8051 i jemu podobnymi, za wcześnie na podawanie przykładowych rozwiązań konstrukcyjnych, to tę część teorii warto poznać, zanim zabierzemy się do budowania „inteligentnych urządzeń elektro-nicznych”... a raczej mikroelektro-nicznych.



Poprzedni odcinek szkoły mikroprocesorowej poświęciłem omówieniu strony praktycznej układów czasowo-licznikowych T0 i T1 procesora 8051/52. Podałem też praktyczne wskazówki na powiązanie pracy liczników z układem przerwań procesora. Dzięki takiemu połączeniu udało nam się wspólnie stworzyć i przeanalizować przykładowy programik odmierzający czas. To podstawa do zrozumienia prawidłowego programowania mikrokontrolerów, bowiem prawie 100% programów pisanych na procesory wykorzystuje procedury (podprogramy) obsługi przerwań do kontroli pewnych funkcji samego procesora jak i układów peryferyjnych.

Umiejętne zaprogramowanie układu licznikowego oraz układu przerwań gwarantuje sukces w działaniu programu a dodanie funkcji zabezpieczających – tzw. „programowych haczyków” (których omówieniem zajmę się przy okazji następnego odcinka klasy mikroprocesorowej) gwarantuje poprawną pracę mikrokontrolera nawet w sytuacjach z góry nieprzewidzianych, jak np. przypadkowe przeładowanie liczników, czy automatyczne rozpoznawanie prędkości transmisji szeregowej z urządzenia zewnętrznego. Wróćmy jednak do tematu i zajmijmy się układem czasowo-licznikowym (potocznie nazywanym licznikiem) T2.

TIMER T2 w 8052/C

Jak powiedziałem wcześniej, układ ten nie występuje w procesorach 8051/C51, jest natomiast integralną częścią struktury procesorów 8052/C. Z pewnych względów przy niektórych aplikacjach dwa układy czasowo-licznikowe (T0 i T1) to za mało, wtedy warto sięgnąć właśnie po kostkę '52.

Licznik T2 podobnie jak poprzednie T0 i T1 jest 16-bitowy, a zliczanie odbywa się przy pomocy dwóch 8-bitowych rejestrów z grupy SFR o adresach:

0CDh: TH2 – starsze 8 bitów licznika T2

0CCh: TL2 – młodsze 8 bitów licznika T2

Rejestry operacyjne licznika T2:

nazwa:									adres:
TH2	d7	d6	d5	d4	d3	d2	d1	d0	CDh
nazwa:									adres:
TL2	d7	d6	d5	d4	d3	d2	d1	d0	CCh

Tak jak poprzednio rejestry te można odczytywać jak i modyfikować, przeładowując (zmieniając) ich zawartość np. za pomocą instrukcji:

```
mov TH2, #wartH
mov TL2, #wartL
```

Z układem czasowo-licznikowym T2 związane są dwa dodatkowe 8-bitowe rejestry, (tworzące 16-bitowy rejestr **RLD**) a mianowicie:

0CBh: RLDH – starsze 8 bitów rejestru RLD

0CAh: RL DL – młodsze 8 bitów rejestru RLD

Rejestry dodatkowe licznika T2:

nazwa:									adres:
RLDH	d7	d6	d5	d4	d3	d2	d1	d0	CBh
nazwa:									adres:
RLDL	d7	d6	d5	d4	d3	d2	d1	d0	CAh

Rejestr RLD (RLDH.RL DL) pełni dwojaką rolę w zależności od trybu pracy licznika T2. Po pierwsze może być rejestrem wartości początkowej licznika T2 (TH2.TL2). Wtedy to w trybie pracy, który za chwilę omówię, po przepelnieniu licznika, wartość początkowa tego licznika zostaje automatycznie (bez programowego przeładowywania) przepisana z RLDH do TH2 oraz z RL DL do TL2.

W innym przypadku rejestr RLD może pełnić rolę rejestru zatraskowego, w którym zapamiętywana jest zawartość licznika T2 (TH2.TL2) w pewnych szczególnych momentach, o tym także za chwilę.

Podobnie jak omawiane w poprzednim odcinku układy T0 i T1, licznik T2 może pełnić dwie różne funkcje:

- może pracować jako czasomierz, czyli zliczać impulsy wewnętrzne o częstotliwości równej częstotliwości oscylatora Fxtal podzielonej przez 12 (zastosowanie takiego trybu pracy już poznaliśmy na przykładzie z lekcji nr 8)
- może także zliczać impulsy zewnętrzne z specjalnego wejścia T2, którym jest pin 1 procesora (alternatywna funkcja bitu 0 portu P1). Zwiększenie zawartości licznika T2 następuje w tym przypadku w momencie wykrycia zbocza opadającego na wejściu T2 (P1.0). Z licznikiem T2 związane jest także dodatkowe wejście T2EX – końcówka procesora P1.1 (pin 2). Końcówka ta może pełnić rolę sygnału strobującego zatrzaśnięcie zawartość licznika T2 (TH2.TL2) w rejestrach RLD (RLDH.RL DL), w innym przypadku umożliwia zdalne załadowanie wartości początkowej z rejestrów RLD do rejestrów licznika T2.

We wszystkich trybach pracy licznika T2 zasada jego pracy jest podobna jak dla układów T0 i T1. A więc obowiązują zasady dotyczące np. wykrywania stanów niskich i wysokich (na przemian) w przypadku zliczania impulsów zewnętrznych, ich częstotliwości maksymalnej, w zależności

od częstotliwości oscylatora procesora. Wszystkie omówiłem w poprzedniej części klasy mikroprocesorowej, warto więc je sobie przypomnieć.

Warto także wspomnieć o możliwości pracy licznika T2 w trybie taktowania portu szeregowego. Wtedy licznik T1 może być użyty do innych celów, natomiast T2 ze względu na swoją 16-bitową „naturę” pozwala na generowanie bardzo niskich częstotliwości taktowania portu, a daczego, o szczegółach powiem za chwilę.

Główny rejestr sterującym trybami oraz pracą licznika T2 jest rejestr T2CON (SFR adres 0C8h). Z pewnością pamiętasz drogi Czytelniku, że w przypadku liczników T0 i T1 był to rejestr TMOD (oraz częściowo TCON), natomiast w przypadku układu czasowo-licznikowego T2 wszystkimi funkcjami licznika T2 steruje tylko jeden rejestr T2CON.

bity:	CFh	CEh	CDh	CCh	CBh	CAh	C9h	C8h	
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	C8h
	7	6	5	4	3	2	1	0	

Rejestr jest adresowany bitowo (w odróżnieniu od TMOD w przypadku T0 i T1), co znacznie ułatwia operacje modyfikacji poszczególnych jego bitów. Oto ich znaczenie:

TF2 (bit T2CON.7, adres: CFh) – bit ustawiany w momencie przepełnienia licznika T2, jest znacznikiem zgłoszenia przerwania przez T2

EXF2 (bit T2CON.6, adres CEh) – bit ustawiany w momencie wykrycia opadającego zbocza na wejściu T2EX (pin 2 procesora), aktywny gdy bit EXEN2 = 1, jest sygnałem zgłoszenia przerwania

RCLK (bit T2CON.5, adres CDh) – ustawienie tego bitu powoduje przyłączenie licznika T2 jako taktującego (przepełnieniem) odbiornika portu szeregowego procesora

TCLK (bit T2CON.4, adres CCh) – ustawienie tego bitu powoduje przyłączenie licznika T2 jako taktującego (przepełnieniem) nadajnika portu szeregowego procesora

EXEN2 (bit T2CON.3, adres CBh) – bit uaktywniający wejście T2EX (pin 2 procesora), ustawienie – uaktywnienie, wyzerowanie – dezaktywacja

TR2 (bit T2CON.2, adres CAh) – bit sterujący zliczaniem licznika T2, ustawienie (TR2=1) powoduje pracę licznika, wyzerowanie (TR2=0) zatrzymanie zliczania

C/T2 (bit T2CON.1, adres C9h) – bit określający funkcję pracy T2, i tak :
– ustawienie (C/T2=1) przełącza T2 w tryb zliczania impulsów zewnętrznych
– wyzerowanie (C/T2=0) włącza funkcję czasomierza (zliczanie impulsów wewnętrznych Fxtal/12)

CP/RL2 (bit T2CON.0, adres C8h) – bit określający tryb pracy licznika, i tak:
– ustawienie (CP/RL2=1) powoduje aktywację trybu z zatraskiwaniem zawartości licznika (rejestrów T2 – TH2.TL2 w rejestrach RLD – RLDH.RLDL)
– wyzerowanie (CP/RL2=0) powoduje pracę z automatycznym wpisywaniem wartości początkowej (z rejestrów RLD do rejestrów T2)

W zależności od kombinacji niektórych bitów z rejestru T2CON możliwe są różne stany pracy układu czasowo-licznikowego T2, oto kilka wskazówek:

- 1) W przypadku gdy ustawimy bit CP/RL2, licznik T2 będzie zliczał modulo 2^{16} , po każdym przepełnieniu będzie ustawiany znacznik zgłoszenia przerwania TF2.
Jeżeli ustawimy dodatkowo bit EXEN2, co spowoduje aktywację wejścia T2EX procesora (P1.1), to możemy zatrzasnąć podając na to wejście opadające zbocze, zawartość licznika T2 w rejestrach RLD (RLDH.RLDL).
- 2) Jeżeli bit CP/RL2 jest wyzerowany, licznik T2 będzie pracował w trybie z automatycznym wpisywaniem wartości początkowej z rejestrów RLD w momencie przepełnienia. W tym miejscu nasuwa się podobieństwo pracy liczników T0 lub T1 w trybie 1, z tym, że przeładowanie licznika następuje w przypadku T2 automatycznie, co zwalnia nas od programowego wpisywania wartości początkowej do rejestrów TH2 i TL2. W momencie przepełnienia ustawiany jest znacznik TF2, co może być także sygnałem zgłoszenia przerwania. Dodatkowo przeładowanie zawartości rejestrów roboczych licznika (TH2.TL2) może nastąpić pod wpływem zewnętrznego zbocza opadającego, podanego na wejście T2EX, trzeba tylko dodatkowo ustawić bit EXEN2 (EXEN2=1). Takie wymuszenie powoduje także ustawienie znacznika EXF2. Dzięki temu możliwe jest zsynchronizowanie pracy

wewnętrznego układu licznikowego T2 z zewnętrznym sygnałem zegarowym (dołączonym do wejścia T2EX).

- 3) W każdym trybie pracy ustawienie znacznika TF2 może być sygnałem zgłoszenia przerwania. W tym przypadku ustawiane znaczniki w słowie T2CON nie są automatycznie zerowane po przyjęciu przerwania, toteż należy o tym pamiętać w procedurze obsługi przerwania. Z drugiej strony niezerowanie znaczników umożliwia programiście ich analizę w procedurze obsługi przerwania, a dopiero po tym ich wyzerowanie.
- 4) Jak wspomniałem wcześniej, licznik T2 może taktować port szeregowy. W tym celu należy ustawić bity TCLK i RCLK. Taktowany w ten sposób port szeregowy będzie mógł pracować w trybie 1 (znaki 8-bitowe, prędkość określana programowo) oraz w trybie 3 (znaki 9-bitowe, prędkość określana programowo). Fizycznie w trybie taktowania portu licznika T2 pracuje zliczając impulsy z automatycznym ładowaniem wartości początkowej z rejestrów RLD (RLDH.RLDL). Zliczane mogą być impulsy wewnętrzne (czasomierz) o częstotliwości Fxtal/2, lub impulsy zewnętrzne z wejścia T2 (P1.0). Po podzieleniu impulsów taktujących (Fxtal/2 lub zewnętrznych z wejścia T2) dodatkowo przez 16 traktują one odbiornik (RCLK=1) lub nadajnik (TCLK=1) portu szeregowego. W tym trybie pracy licznika, jego przepełnienie nie ustawia znacznika jego przepełnienia TF2. Dzięki temu w przypadku gdy ustawimy bit EXEN2 (w słowie T2CON), to pod wpływem opadającego zbocza sygnału na wejściu T2EX (końcówka P1.1 procesora) ustawiony zostaje znacznik EXF2, co może być sygnałem zgłoszenia przerwania. Jak widać możliwe jest zatem wykorzystanie wejścia T2EX jako dodatkowego (obok INTO i INT1) wejścia przerywającego procesora.

W przypadku taktowania portu szeregowego (T2 taktowany sygnałem wewnętrznym Fxtal / 2) prędkość transmisji (n) można określić wzorem:

$$n = \frac{F_{xtal}}{(65536 - RLD) \times 2 \times 16}$$

stąd łatwo po przekształceniu wzoru wyznaczyć wartość początkową rejestrów RLD przy zadanej prędkości transmisji:

$$RLD = 65536 - \frac{F_{xtal}}{2 \times 16 \times n} = 65536 - \frac{F_{xtal}}{32 \times n}$$

gdzie RLD to oczywiście wartość początkowa wpisana do rejestrów RLDH.RLDL. Poniżej podaję przykładowe prędkości transmisji dla rezonatora kwarcowego 11,0592 MHz.

Fxtal (MHz)	RLD	n (bodów)
11,0592	FFFFh	345600
11,0592	FFFDh	115200
11,0592	FFFCh	86400
11,0592	FFFAh	57600
11,0592	FFF7h	38400
11,0592	FFF4h	28800
11,0592	FFEEh	19200
11,0592	FFDCh	9600
11,0592	FFB8h	4800
11,0592	FF70h	2400
11,0592	FEE0h	1200
11,0592	FDC0h	600
11,0592	FB80h	300
11,0592	F700h	150
11,0592	EE00h	75

Jak widać, zakres możliwych do uzyskania prędkości transmisji jest o wiele szerszy, niż w przypadku taktowania portu licznikiem T1. Najmniejsza szybkość w przypadku rezonatora kwarcowego jak w tabeli czyli 11,0592 MHz przy użyciu licznika T2, to 5 bitów na sekundę, a więc prawdziwy żółw! Sprawdźmy:

$$n_{min} = 11059200 / (32 \times (65536 - 0)) = 11059200 / (32 \times 65536) = 5 \text{ (bitów/sek.)}$$

Wtedy oczywiście wartością początkową będzie zero (RLDH.RLDL=0). W tabeli pogrubioną czcionką zaznaczono typowe, spotykane w komputerach PC wartości transmisji szeregowej, realizowanej poprzez port RS232c.

A oto przykład programowania rejestru sterującego T2CON oraz roboczych i dodatkowych w celu uzyskania kilku trybów i funkcji pracy licznika T2.

Założymy, że nasz licznik T2 będzie pracował jako czasomierz z automatycznym ładowaniem wartości początkowej z RLD po przepełnieniu.

Założymy, że przepełnienie ma następować dokładnie co 1ms, a do procesora dołączony jest rezonator kwarcowy o częstotliwości 6 MHz. Obliczenia:

- przy fxtal = 6 MHz licznik pracując jako czasomierz będzie inkrementowany co 2µs

Też to potrafisz

– w czasie 1 ms (milisekundy) zawiera się 1000 μ s (mikrosekund), czyli 500 okresów zegara procesora
– wobec tego wartość początkowa licznika można obliczyć jako:
wart.początkowa = wartość maksymalna – 500 = 65536 – 500 = 65036 = FE0Ch (heksadecymalnie)
można więc zapisać komendy inicjujące licznik T2:

```
mov    T2CON, #0          ;czasomierz z automat.
                                ;ładowaniem wart. początkowej z RLD

mov    RLDH, #0FEh
mov    RLDL, #0Ch          ;załadowanie wartości FE0Ch
                                ;(początkowej)

mov    TH2, RLDH
mov    TL2, RLDL          ;aby prawidłowo zainicjować
                                ;pierwsze przepełnienie

setb   TR2                ;start licznika T2
.....
.....                    ;dalsze instrukcje
```

Analizując linie poleceń warto szczególną uwagę zwrócić na pierwszą komendę, która ustawia bity w słowie T2CON. Zauważmy, że wszystkie bity tego słowa zostały ustawione na zero, co zgadza się z opisem rejestru T2CON, który omówiłem przed chwilą.

Jeżeli ktoś teraz zechce dopisać procedurę obsługi przerwania, może to zrobić analogicznie jak w przypadku opisanej wcześniej procedury obsługi przerwania od przepełnienia licznika T0 (T1), pamiętając jednak o adresie wektora przerwania, który w tym przypadku wynosi:

002Bh

a w przypadku naszego komputerka edukacyjnego (AVT-2250) procedura powinna zaczynać się od adresu

802Bh

zgodnie z zasadami pisania takich podprogramów, które omówiłem w poprzednim numerze EdVw.

Program na nasz komputerka mógłby więc zaczynać się następująco:

```
org    8000h
ljmp   START
org    802Bh

intT2: .....
        .....
        .....
pop     DPL
pop     DPH
pop     Acc
reti

;*****
START:  mov    T2CON, #0          ;czasomierz z automat.
                                ;ładowaniem wart.
                                ;początkowej z RLD

mov    RLDH, #0FEh
mov    RLDL, #0Ch          ;załadowanie wartości FE0Ch
                                ;(początkowej)

mov    TH2, RLDH
mov    TL2, RLDL          ;aby prawidłowo zainicjować
                                ;pierwsze przepełnienie

setb   TR2                ;start licznika T2
.....
.....                    ;dalsze instrukcje
.....

END
```

Nie należy jednak zapominać że w licznik T2 wyposażony jest procesor 8052/C, toteż jeżeli masz w komputerku AVT-2250 procesor 8051/C51, to musisz go po prostu wymienić na właściwą kostkę. W sklepie nie powinna ona kosztować więcej jak 5zł.

Specjalne tryby pracy

W tej części artykułu omówię tryby pracy procesora, dzięki którym możliwe jest realizowanie ciekawych rozwiązań układowych, np. urządzeń zasilanych z baterii – czyli takich, w których kwestia poboru energii jest elementem krytycznym.

W obszarze rejestrów SFR procesora (przypominam – jest to wewnętrzna pamięć danych RAM adresowana bezpośrednio, o adresach 80h...FFh) znajduje się jeszcze jeden ciekawy rejestr specjalnego przeznaczenia. Jego funkcją jest kontrola specjalnych trybów pracy procesora, a mianowicie:

- trybu tzw. „jałowego”,
- trybu tzw. „uśpienia”.

Z grubsza rzecz ujmując tryby te różnią się od siebie stopniem poboru mocy przez procesor, oraz funkcji, jakie pozostają aktywne w tych trybach pracy w odróżnieniu od normalnego trybu pracy procesora.

Przejdźmy zatem do omówienia rejestru PCON, bo o nim jest mowa. Poniżej przedstawione jest znaczenie poszczególnych bitów tego rejestru. Warto przy tym zauważyć, że rejestr nie może być adresowany bitowo, to też nie da się sterować jego poszczególnymi bitami poprzez instrukcje typu:

SETB bit

lub

CLR bit

nazwa:

PCON	SMOD	-	-	-	GF1	GF0	PD	IDL	adres: 87h
------	------	---	---	---	-----	-----	----	-----	------------

Rejestr PCON jest umieszczony pod adresem 87h w obszarze SFR procesora. Zawiera 5 istotnych dla użytkownika bitów.

SMOD (bit .7) – bit podwojenia szybkości transmisji poprzez port szeregowy w trybach 1,2 lub 3 pracy. Ustawienie tego bitu (SMOD=1) powoduje dwukrotne zwiększenie częstotliwości taktowania portu szeregowego poprzez licznik T1, kiedy ten pracuje w trybie taktowania tego portu. W odcinku, w którym omawiałem port szeregowy, w zamieszczonej tam tabeli podałem przykłady wartości początkowych dla standardowych prędkości transmisji asynchronicznej. Wszystkie odnosiły się właśnie do podwojonego trybu prędkości transmisji, kiedy bit ten jest ustawiony. Jeżeli nie chcemy pracować w trybie podwojonej prędkości, bit ten powinien być wyzerowany (PCON=0).

Ustawienie bitu SMOD w rejestrze PCON można wykonać za pomocą instrukcji, np.:

ORL PCON, #80h

wyzerowanie zaś za pomocą instrukcji:

ANL PCON, #7Fh

– (bity: 6...4) – nie wykorzystane

GF1 (bit .3) – bit programowy do dowolnego wykorzystania przez programistę

GF0 (bit .2) – bit programowy do dowolnego wykorzystania przez programistę

PD (bit .1) – bit włączający tryb obniżonego poboru mocy – „uśpienia”. Ustawienie tego bitu powoduje wprowadzenie procesora w tryb uśpienia, kiedy to pobór prądu spada o około 500 razy, a napięcie zasilające Vcc może być obniżone do 2,0V. Wykonanie instrukcji ustawiającej ten bit jest ostatnim poleceniem wykonanym przez procesor w programie.

IDL (bit .0) – bit włączający tryb „jałowy” procesora.

Poniżej nieco dokładniej omówię oba tryby pracy.

Tryb jałowy

Instrukcja, która ustawia bit PCON.0 powoduje wprowadzenie procesora w ten tryb. Jest ona ostatnią wykonywaną przez procesor instrukcją. Wewnętrzny sygnał zegarowy zostaje odłączony od jednostki centralnej (CPU), ale układ przerwań, port szeregowy i licznikowy pracują dalej, jeżeli wcześniej były odpowiednio skonfigurowane i ustawione. Stan całego procesora, a więc stan:

- rejestrów specjalnych SFR,
- pamięci wewnętrznej RAM użytkownika
- pinów portów P0...P3

pozostaje bez zmian i jest taki sam jak był tuż przed wejściem procesora w tryb jałowy.

Końcówki ALE i /PSEN procesora ustawiają się w stan wysoki.

Istnieją dwa sposoby na wyjście z tego stanu:

- 1) Nadejście dowolnego przerwania – oczywiście jeżeli było ono wcześniej uaktywnione w rejestrze IE. Pojawienie się przerwania zeruje automatycznie (bez udziału programu użytkownika) flagę PCON.0 – i procesor powraca do normalnej pracy, z tym, że następną instrukcją po wyjściu ze stanu jałowego pod wpływem przerwania będzie pierwsza znajdująca się w procedurze obsługi danego przerwania, aż do instrukcji RETI, kiedy to procesor automatycznie powraca do instrukcji następnej po tej, która wprowadziła procesor w stan jałowy, czyli tej, która ustawiła bit IDL w rejestrze PCON.
- 2) Drugim sposobem na wyjście z tego stanu jest zerowanie procesora. Ze względu na fakt, że podczas trybu „jałowego” procesor pracuje nadal zegar systemowy, do prawidłowego zresetowania potrzebny jest impuls zerujący o długości co najmniej 24 okresów oscylatora.

Tryb uśpienia – obniżonego poboru mocy

W tym trybie, obecnie stosowanym przez producentów tylko w kostkach typu CMOS, czyli np. 80C51 lub 80C, cały mikrokontroler pobiera znacznie mniej energii, oraz dodatkowo napięcie zasilające układ może zostać zmniejszone od standardowych 5V do 2,0V. Instrukcja ustawiająca bit PD (PCON.1) jest ostatnią wykonywaną przez procesor. W trybie tym oscylator procesora

zostaje wyłączony (po prostu staje). Zostają odłączone wszystkie układy funkcjonalne procesora, takie jak układy licznikowe, port szeregowy, układ przerwań. Pozostaje jedynie niezmienniona zawartość wewnętrznej pamięci RAM, w tym pamięci użytkownika oraz rejestrów specjalnych SFR. Piny portów pozostają zgodne ze stanami odpowiadających im bitów w rejestrach P0...P3 w obszarze SFR. Końcówki ALE i PSEN znajdują się w stanie niskim. W tym trybie pracy procesora, a raczej nie trybie pracy, co uśpienia, procesor pobiera około 500 razy mniej prądu niż w stanie normalnej pracy. Dla przykładu podam, że dla kostki 80C51 (czyli w wersji CMOS) :

- w trybie pracy normalnej pobór prądu wynosi ok. 20mA (przy fxtal=12MHz)
- w trybie „jałowym” pobór prądu spada do około 3,0 mA (przy fxtal=12MHz)
- w trybie uśpienia pobór prądu przez układ wynosi ok. 50µA (mikroamper!), przy obniżeniu zasilania do 2,0V.

Jedyną metodą na opuszczenie trybu uśpienia i powrót do normalnej pracy jest wyzerowanie mikroprocesora poprzez podanie impulsu resetującego na wejście RST (pin 9) o czasie trwania ok. 10ms.

Reset procesora

Popularne zresetowanie odbywa się poprzez podanie impulsu dodatniego na wejście RST kostki (pin 9) zgodnie z zasadami, które omówiłem przed chwilą. Najprostsze i bardziej rozbudowane układy resetowania mikrokontrolerów serii MCS-51 podałem w jednym z pierwszych odcinków klasy mikroprocesorowej.

W wyniku zresetowania rejestru układy specjalne procesora (SFR) zostają automatycznie zainicjowane wartościami, jak podaję w tabeli poniżej:

Rejestr	Wartość po „RESET”	Uwagi
PC	0000h	licznik rozkazów
ACC	00h	akumulator
B	00h	rejestr B
PSW	00h	słowo stanu programu
SP	07h	wskaźnik stosu

DPTR	0000h	wskaźnik danych
P0...P3	FFh	porty
IP	xxx00000b	rejestr priorytetu przerwań
IE	0xx00000b	rejestr masek przerwań
TMOD	00h	rej. liczników T0 i T1
TCON	00h	rej. ster. liczników i przerwań
TH0	00h	
TL0	00h	
TH1	00h	
TL1	00h	
SCON	00h	rejestr portu szeregowego
SBUF	zmienny	bufor portu szeregowego
PCON (MOS)	0xxxxxxx	dla układów NMOS (8051/2)
PCON (CMOS)	0xxx0000b	dla ukl. CMOS (80C51/52)

Uwaga: litera „x” oznacza, że dany bit przyjmuje wartość przypadkową lub nie jest implementowany w danym rejestrze. Wartość po „RESECE” przedstawiono w formacie szesnastkowym lub binarnym (z „b” na końcu) celem ułatwienia analizy i porównania z opisem rejestrów SFR na wkładce z numeru Edw 11/97.

Programując mikrokontrolery 8051/52, trzeba pamiętać o tym fakcie, toteż szczególnie w przypadku wykorzystania specjalnych trybów pracy należy mieć na względzie fakt, że rejestry specjalne zostają utracone po resetie procesora, a więc konieczne jest ich odtworzenie, oczywiście tylko w razie takiej konieczności.

Jeżeli ktoś interesuje się szczegółami, dotyczącymi architektury wewnętrznej mikrokontrolera 80C51 lub podobnych, które nie za często są wykorzystywane w projektach, przynajmniej przez początkujących programistów, może sięgnąć do literatury [1] i [2]. Obie pozycje wymagają jednak znajomości podstaw terminologii technicznej jęz. angielskiego. Chętnych zapraszam do lektury, pozostałym proponuję poczekać na kolejny odcinek naszego cyklu.

Sławomir Surowiński

Literatura:

- [1] 80C51-based 8-bit Microcontrollers, Data Handbook, Philips 1995
[2] Microcontroller Data Book, Atmel 1995,6,7

Lekcja 9

Na początku lekcji zajmijmy się rozwiązaniem zadań z poprzedniego numeru EdW i odcinka klasy mikroprocesorowej.

Rozwiązanie zadania nr 1

Oto krótka analiza głównej części listingu od etykiety START.

W linii (72) profilaktycznie zatrzymujemy licznik T1. W dwóch kolejnych liniach ustawiamy tryb pracy licznika T1, jako czasomierza zliczającego wewnętrzne impulsy zegarowe (tryb pracy 0). W linii (75) wpisujemy obliczoną wartość początkową, przy której licznik będzie przepelniany okładnie co 1/128 sekundy.

W linii (76) wyzerowany zostaje licznik zliczający 1/128 części sekundy, a w kolejnej linii (77) załadowany zostaje do zmiennej systemowej komputerka starszy bajt 16-bitowego adresu tabeli wektorów przerwań w zewnętrznej pamięci programu.

Należy jeszcze w linii (78) odblokować przerwanie od licznika T1 (przepelnienie licznika) oraz ustawić priorytet na to przerwanie (linia 79) inaczej bowiem odmierzenie czasu może być zakłócone pracującym przecież stale licznikiem T0 i wywołaną przez jego przepelnienia procedurą obsługi przerwania, która zajmuje się obsługą wyświetlacza i klawiatury komputerka edukacyjnego AVT-2250.

Po zainicjowaniu układu licznikowego T1 oraz przerwania od tego licznika, w linii 81 profilaktycznie czyścimy wyświetlacz, aby potem w liniach (82)...(85) pobrać z klawiatury godzinę początkową i w linii 86 zapamiętać ją w komórce GODZ. Podobnie dzieje się dla minut – linie (88)...(91) i sekund – linie (94)...(98).

Po wprowadzeniu (ustawieniu czasu) zapalone zostają poziome kreski oddzielające godziny od minut (linia 100) i minuty od sekund (linia 101), a następnie po koniecznym ze względu na drgania styków klawiatury opóźnieniu (ok. 0,5 s.) – linie (102),(103), komputerka oczekuje na naciśnięcie klawisza przez użytkownika celem uruchomienia zegara. Uruchomienie następuje w kolejnej linii (105), a dalej znajduje się część programu, której zadaniem jest wyświetlanie upływającego czasu na wyświetlaczu – od etykiety „pokaz:” w linii (107).

Dzięki badaniu bitu nr 6 w zmiennej „licz128”, który przecież zmienia swój stan co 0,5 sekundy (linia 109), możliwe jest naprzemienne gaszenie kresek (linie 110,111) oraz ich zapalenie – linie (113) i (114).

Dalej od etykiety „czas:” rozpoczyna się wypisanie czasu, czyli wyświetlane są godziny – linie (116...118), minuty – linie (119...121) oraz sekundy – linie (122...124).

W linii (125) następuje skok do początku, gdzie następuje kolejne uaktualnienie wyświetlanego czasu.

Rozwiązanie zadania nr 2

Aby wyświetlić aktualny czas w trybie jak pokazano w zadaniu, czyli:

1 2 – 3 4 . 5 8

wystarczy zmodyfikować podane linie na postać jak poniżej:

linia	adres	kod	instrukcja
101	8099	75D80	mov DL6,#_kropka
111	80AE	75D80	mov DL6,#_kropka
114	80B6	75D80	mov DL6,#_kropka

lub w zależności od swoich upodobań posłużyć się zmienną systemową: „blinks”, której każdy z bitów określa atrybut wyświetlanego znaku, czy ma migać, czy być normalnie wyświetlanym.

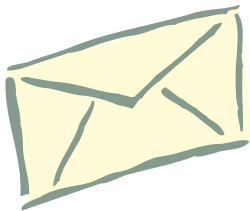
Ponieważ nie dotarły do mnie jeszcze listy z rozwiązaniami zadania nr 3, pozwolę sobie zamieścić najciekawsze propozycje w kąciaku początkowym 8051 w kolejnym numerze EdW.

Uwaga!

W poprzedniej lekcji nr 8 do listingu wkładły się drobne błędy, a mianowicie, komentarze w liniach o podanych numerach powinny wyglądać następująco:

- (74) ;T1 jako 8-bitowy z preskalarem 5-bitowym
(76) ;wyzerowanie licznika 1/128 s.

Sławomir Surowiński



Kącik pocztowy 8051

Zasypany wieloma listami, dotyczącymi cyklu artykułów poświęconych programowaniu mikrokontrolerów 8051, postanowiłem uruchomić kącik pytań i odpowiedzi, które kierujecie do mnie w swoich listach.

Na wstępie chciałem bardzo podziękować za każdy list, zarówno te pochwalne jak i krytyczne. Przyznam że, cieszy mnie bardzo fakt, iż tak wielu z Was zdecydowało się sięgnąć po „mikroprocesory”, a co najważniejsze odnosi już małe, ale jak ważne w nauce sukcesy.

Dziś kolejna porcja listów od Czytelników i odpowiedzi na niektóre pytania.

List 1

Zenon Rakoczy z Chropaczowa ma wątpliwości co do instrukcji

`MOV adres1, adres2`

a ponieważ, jak pisze, jest „ręczniakiem” problem pojawił się w transkrypcji na języka maszynowy bez korzystania z komputera i assemblera 8051.

Jak pisałem wcześniej w artykułach instrukcje typu

`MOV X, Y`

kopiują zawartość po stronie Y do literału X, czyli następuje przeniesienie typu

`X ← Y`

i wszystko się zgadza. Zenon ma wątpliwości, jak należy przetłumaczyć np. instrukcję:

`MOV DPH, B`

Jak wynika z wcześniejszych analiz instrukcji tego typu, najpierw należy zapisać bajt określający instrukcję MOV tego typu, czyli zgodnie z tabelą we wkładce będzie to:

85 – kod instrukcji „MOV adres1, adres2”

83 – adres rejestru DPH

F0 – adres rejestru B, czyli

instrukcję tę można zapisać jako ciąg bajtów: 85 83 F0.

Niestety, jak podawałem wcześniej, podczas opisu instrukcji procesora ten typ instrukcji jest wyjątkiem i kolejność rejestrów po przetłumaczeniu będzie odwrotna, czyli: **85 F0 83**.

Oczywiście nie zmienia to działania tej instrukcji, po prostu tak tłumaczy się ją na język maszynowy.

Przy okazji podczas omawiania tej instrukcji w EdW w opisie wkraśli się błąd, było zatem:

MOV adres1, adres2

– przepisanie zawartości komórki o adresie „adres1” do komórki o adresie „adres2”

a powinno być:

MOV adres1, adres2

– przepisanie zawartości komórki o adresie „adres2” do komórki o adresie „adres1”

dalsza część opisu jest bez błędów, czyli:

(adres1) ← (adres2)

- kod: 1 0 0 0 0 1 0 1

- cykle: 2 bajty: 3 (kod instrukcji + adres2 + adres1)

- przykład:

```
MOV    7Fh, 7Eh ;przepisanie zawartości dwóch
          ;sąsiadujących komórek w
          ;wew. RAM procesora
```

List 2

Marcin Jurzak nadesłał wiadomość przez Internet, że ma kłopoty z przesyłaniem programów z komputera do komputerka. Za każdym razem, kiedy transmisja zaczyna się na wyświetlaczu komputerka edukacyjnego, pojawia się napis „Err” – czyli komunikat błędu.

Jeżeli taki komunikat pojawia się oznacza to, że kabel wykonał poprawnie, i dane przesyłane są z PC-ta do komputerka AVT-2250, z tym, że nie są zrozumiałe dla BIOS-a, stąd komunikat o błędzie. Powodów takiego stanu rzeczy może być kilka:

a) nie ustawione parametry portu szeregowego w PC-cie – powinien wydać komendę ustawiającą je, jak opisywałem przy okazji opisu Bios-a komputerka, a mianowicie z poziomu DOS-a wywołać komendę:

`MODE COM2: 4800, n, 8, 1 {Enter}`

Jeżeli korzysta z Windows 95, należy parametry portu ustawić w Panelu sterowania w opcji System – Menedżer Urządzeń, ustawiając parametry:

Bitów na sekundę : 4800

Bitów danych: 8

Parzystość: brak

Bitowy stopu: 1

Sterowanie przepływem: Brak

oraz dodatkowo w opcji „Zaawansowane...” koniecznie wyłączyć (odhaczyć) opcję buforowania poprzez FIFO.

b) inną przyczyną może być fakt, że próbuje wysłać zbiory binarne, a nasz komputer akceptuje tylko zbiory w formacie Intel-HEX, radzę sprawdzić.

Czytelnik ma także wątpliwości co do rysunku kabla połączeniowego PC z komputerkiem AVT-2250 w wersji 9 na 25 pinów. Informuję, że rysunek jest prawidłowy, a skrzyżowanie 2 z 3 występuje tylko w kablu „9 na 9”. We wtyku DB25 znaczenie końcówek 2 i 3 jest dokładnie odwrotne niż na końcówce DB9, stąd brak krzyżowych połączeń.

List 3

Łyżka miodu od Czytelników: Wiesław Kusek z Mielca pisze:

“...Otrzymałem niedawno obiecany bezpłatnie zestaw AVT-2250. Od razu chcę bardzo podziękować za ten zestaw. I w związku z tym mam kilka uwag i spostrzeżeń, oraz co nie udało mi się uniknąć – pytań. ... Kit AVT-2250 – bardzo dobra ocena. Rzecz droga, ale wiadomo pewnych rzeczy nie da się przeskoczyć. Układ przemyślany i dopracowany do końca, co świadczy o dużej fachowości autora opracowania i co dla mnie – początkującego w tym temacie najważniejsze – w dość jasny i czytelny sposób opisany. Chociaż kilku zdań nie rozumiem. Brawo za wysoką jakość płytek drukowanych. Układ został przeze mnie zmontowany i po włączeniu zasilania ruszył bez żadnego problemu.”

Adam Szendzielorz z Wodzisławia Śląskiego napisał do nas:

“... Niedawno otrzymałem od Was w/w układ „mikrokomputerka edukacyjnego.... Po odebraniu go na pocztę, czym prędzej wziąłem się do montowania go. W jakieś 3 godziny układ stał na stole gotowy – trzeba było tylko jeszcze sprawdzić czy działa... Po podłączeniu zasilania na wyświetlaczach ukazał się napis „HELLO” !!! Sam nie mogłem uwierzyć – po raz pierwszy w mojej karierze już po pierwszym uruchomieniu (w sumie dość skomplikowanego układu) działał on poprawnie!!! – to chyba dzięki bardzo starannie wykonanych płytek drukowanych i sprawnych elementów, choć moja staranność przy jego wykonywaniu też na pewno temu sprzyjała. (“Ależ oczywiście! – przyp. red.) Po pierwszych emocjach zacząłem przerabiać wszystkie lekcje EdW dotyczące procesorów – nie jest to łatwe, ale myślę, że w drodze praktyki będzie to prostsze i lepiej to zrozumieć.

Ponieważ posiadam komputer PC, wykonałem kabel łączący go z układem. I tu mała uwaga i jednocześnie prosba... Ręczne wklepywanie to chyba strata czasu (który można by wykorzystać do innych celów), pisanie na komputerze to już coś – można to zapisać, często budzący odtworzyć i przesłać do komputerka, łatwiej coś zmienić, a przede wszystkim wykonuje się to „duuuużo” szybciej.”

Takich listów otrzymuję bardzo dużo. Cieszymy się z tego, że większość z Was nie ma problemów z uruchamianiem układu komputerka edukacyjnego AVT-2250. Jednak do redakcyjnego serwisu trafiają czasem z reklamacją zestawy nie działające. Powodem takiego stanu rzeczy i często zażenowania nabywcy jest niestaranny, często budzący zgrozę sposób montażu. Apeluję więc, nie starajcie się lutować elementów „byle czym” i mierzcie swoje siły na zamiary, a każdy układ elektroniczny odpali bez problemów. Jeżeli nie czujecie się na siłach w samodzielnym zmontowaniu, proponuję zamówienie w AVT zmontowanego komputerka – kit AVT-2250/C.

P.S. Adamie z Wodzisławia, brakującą dyskietkę AVT-2250/D otrzyma jak tylko pojawią się one w magazynie AVT. W razie kłopotów proszę o kontakt z Działem Handlowym AVT i potwierdzenie swego zamówienia.

List 4

Tomasz Jabłonowski z Moniek pisze:

- że, po pierwsze, nie może znaleźć „we wkładce” (zapewne EdW) np. *CLS, A2HEX* itp.
- po drugie, „inne” kompilatory nie rozumieją argumentów niektórych publikowanych w cyklu komend, np. *DL1,#_minus*

Szanowny kolego, wyrażenia typu *CLS, A2HEX, DPTR4HEX*, to nie są komendy procesora 8051, ale zaproponowane przez autora cyklu szkoły mikroprocesorowej nazwy procedur (podprogramów) umieszczonych w pamięci EPROM komputerka. Pisałem o tym wiele razy. Pod nazwami tymi kryją się konkretne adresy, proszę zajrzeć do zbiorów BIOS.INC, które znajdują się na dyskietce AVT-2250/D.

To samo dotyczy niektórych zdefiniowanych konkretnie dla naszego komputerka edukacyjnego stałych i zmiennych, a więc:

DL1 – pod tą nazwą kryje się adres komórki w wewn. RAM procesora, która przechowuje znak do wyświetlenia na pozycji nr 1 (pierwszy wyświetlacz).

_minus – pod tą nazwą kryje się stała (liczba), która określa kolejność zapalonych segmentów wyświetlacza podczas wyświetlania znaku „-”.

Toteż jeżeli zechcesz używać innego kompilatora, powinienes zadeklarować na początku programu przed instrukcjami następujące przypisanie:

```
DL1    equ    78h
_minus equ    01000000b
```

To samo dotyczy wszelkich innych nazw, którymi operujemy w naszych artykułach, a które można znaleźć w zbiorach CONST.INC i BIOS.INC na dyskietce do systemu AVT-2250.

List 5

Bardzo ciekawy list od Czytelnika z Katowic Marek Joniec pyta mianowicie:

- 1) „... dlaczego wszelkie napisy adresów i danych dokonuję w kodzie heksadecymalnym, skoro procesor i układy zewnętrzne operują danymi binarnymi, w którym momencie i gdzie np. dana „D9h” jest transkodowana na liczbę dwójkową „11011001”.
- 2) Czy jeżeli system działa tylko z zewnętrzną pamięcią RAM, która w naszym przypadku posiada przestrzeń adresową 8000h...FFFFh, to gdzie mam w niej szukać obszaru rejestrów specjalnych SFR? Czy obszar ten jest automatycznie gdzieś umiejscowiony, czy też odpowiedzialny jest za to monitor systemowy?
- 3) Jeżeli w specyfikacji rejestrów podane jest, że rejestr PSW posiada adres D0h, to gdzie się on znajduje w moim RAM-ie?
- 4) Dlaczego chcąc wpisać jakąś liczbę na dany wyświetlacz odwołuję się do adresów 78h...7Fh (przecież są to adresy pamięci programu EPROM, a nie pamięci RAM).
- 5) Większość rozkazów assemblerowych wymaga podania adresu bezpośredniego „xx”, co to znaczy i jak mam adresować obszar 8000h...FFFFh za pomocą dwóch pozycji?
- 6) W jaki sposób mam wprowadzać dane z klawiatury do pamięci nie używając gotowego polecenia GETACC zawartego w monitorze?
- 7) Na czym polega w praktyce adresowanie bitowe rejestrów specjalnych?

Oto odpowiedzi

Ad.1 Zapisywanie wszelkich wartości liczbowych oraz kodu maszynowego procesora w zapisie szesnastkowym (heksadecymalnym) jest najbardziej naturalnym i czytelnym, jak się wkrótce przekonasz, sposobem. Szesnastkowy zapis, szczególnie w ujęciu techniki cyfrowej, mikroelektroniki, mikrokontrolerów i wreszcie komputerów jest tym, czym język narodowy danego kraju.

Prawdą jest, że układy komputerowe przetwarzają wszystko w kodzie binarnym, ale nam ludziom trudno byłoby „czytać” i analizować kod programu w takiej postaci, aczkolwiek jeszcze przed kilkudziesięciami laty taki sposób programowania był jedynym – lecz były to lata 50. i początki techniki komputerowej.

Prosty przykład, łatwiej odnaleźć (wzrokowo) i rozróżnić dwie liczby, np.:

1234h i ABCDh

niż te same zapisane w postaci binarnej, a więc:

0001001000110100b i 1010101111001101b

Jeżeli chodzi o drugą, dość oryginalną część pytania, to mogę mieć pewien problem z odpowiedzią, bowiem fakt zamiany, o której Czytelnik mówi, można by porównać z czymś tak oczywistym, a jednocześnie trudnym do opisanie jak np. „słuchanie czyjeś mowy i wyciąganie z niej wniosków”.

Aby jednak zaspokoić ciekawość Czytelnika można powiedzieć, że w przypadku mikrokontrolerów zamiana liczby z kodu szesnastkowego na binarny fizycznie odbywa się w momencie:

- w przypadku pobierania rozkazów z pamięci EPROM komputerka – w momencie programowania pamięci EPROM w urządzeniu zewnętrznym zwanym programatorem pamięci EPROM, który wczytuje zbiory np. Intel-HEX (takie jakie tworzy nasz kompilator) a następnie zapisuje dane o programie w pamięci EPROM korzystając z linii adresowych kostki pamięci oraz z linii (np. 8) danych D0...D7, których przecież jest 8 tak jak jest 8 bitów w jednym bajcie informacji.
- w przypadku przesyłania danych z komputera PC do komputerka – już w momencie transmitowania danych z komputera, który zamienia informację na szeregową, czyli ciąg bitów z odpowiednimi sygnałami sterującymi.

Zresztą najlepszą odpowiedzią na to pytanie może być fakt, że sam mikrokontroler 8051 i wszystkie pozostałe układy tego typu pobierają i wysyłają informację już w postaci binarnej poprzez swoje końcówki w obudowie, których może być więcej lub mniej. W przypadku 8051 i podobnych mu są to 4-8-bitowe porty P0...P3.

Tak więc informacja trafia do procesora – wychodzi z niego zawsze w postaci binarnej, to tylko peryferyjne układy zewnętrzne transformują ją na postać bardziej czytelną dla użytkownika czy programisty i o to przecież chodzi, tak działa postęp.

Dlatego w komputerze PC używamy monitora czy klawiatury, które przecież też za pomocą dodatkowych urządzeń (takich jak karta graficzna) zamieniają informację bitową na postać bardziej czytelną np. szesnastkową i wyświetlają ją na ekranie, czy zamieniają wciśnięty klawisz na sekwencję bitów odpowiadającą kodowi danego klawisza.

Ad.2 Jeszcze raz wyjaśniam.

a) Istnieją dwa rodzaje pamięci danych – **wewnętrzna i zewnętrzna**

b) **wewnętrzna zawarta jest fizycznie w strukturze procesora 8051** (w kostce).

Procesor ten zawiera dokładnie: 128 komórek (bajtów) pamięci użytkownika o adresach: 00h...7Fh oraz 128 komórek (nie wszystkie aktywne) o adresach 80h...FFh, pod którymi to znajdują się rejestry specjalne SFR procesora.

To wszystko zawarte jest w strukturze procesora, pamiętajmy!

Adresowanie tej pamięci RAM (wewnętrznej) odbywa się za pomocą wszystkich rozkazów procesora typu MOV oraz rozkazów wykonujących operacje arytmetyczne lub logiczne na komórkach tej pamięci, czyli: ANL, ORL, XRL, ADD, SUBB, INC, DEC, itd.

c) zewnętrzna pamięć RAM to pamięć, jak sama nazwa wskazuje, dołączana z zewnątrz w postaci kostek SRAM o rozmiarze maksymalnie 64kB (65536 bajtów). Adresowanie tej pamięci odbywa się za pomocą tylko 2 rozkazów, a mianowicie:

MOVX A, @DPTR – odczyt danej z komórki w **zewnętrznej** pamięci danych o adresie zawartym w DPTR (czyli pokrywającym całe 64kB, 0...FFFFh) i umieszczenie jej w akumulatorze (Acc), czyli rejestrze w **wewnętrznej** RAM procesora o adresie bezpośrednim E0h.

MOVX @DPTR, A – zapis danej z komórki z akumulatora do **wewnętrznej** pamięci danych pod adres wskazany w rejestrze DPTR (uwaga! jw.).

Ad.3 Po odpowiedzi na pyt. poprzednie odpowiedź na to pytanie jest oczywista – w PSW znajduje się w wewnętrznej pamięci RAM procesora (w kostce) pod adresem D0h.

Ad.4 To nieprawda, że adresy „odwołań do wyświetlacza” znajdują się w EPROM-ie. Pisząc program wykorzystujący procedury BIOS-a komputerka edukacyjnego AVT-2250 programista może dla ułatwienia skorzystać z usług prowadzonych przez monitor.

I tak, fizycznie monitor komputerka w procedurze obsługi przerwania (pojawiającej się okresowo dokładnie 512 razy na sekundę) pobiera cyklicznie zawartość kolejnych rejestrów DL1...DL8, czyli o adresach 78h...7Fh w **wewnętrznej** RAM procesora, a następnie przepisuje je do zatrasku (74574) na płycie wyświetlacza, który to steruje układem ULN2803A – patrz schemat komputerka edukacyjnego. Sekwencję tę można zilustrować następująco:

MOV A, DL1 ;załadowanie do akumulatora

;zawartości rejestru DL1

MOV DPTR, #4000h

;załadowanie do DPTR adresu rejestru 74574

MOVX @DPTR, A

;wreszcie przesłanie zawartości akumulatora

;do rejestru celem wyświetlenia

Ad.5 Po przeczytaniu odpowiedzi na pytanie 2, wiesz już, drogi Czytelniku, że nie da się zaadresować zewnętrznej pamięci RAM

