Programowanie '51
Procedury uzależnień czasowych

W numerze 11/94 EP przedstawiliśmy mini-moduł 8051. Wielu Czytelników, szczególnie początkujących projektantów urządzeń sterowanych mikroprocesorami, czuje zapewne wyraźny niedosyt informacji o sposobach wykorzystania tego modułu do konkretnych zastosowań. Każdy mikroprocesor jest bowiem ożywiany przez oprogramowanie. Dlatego postanowiliśmy zająć się zaganieniem programowania „pięćdziesiątki jedynki". Czynimy to, jak na EP przystało, od strony praktycznej. W cyklu kilku kolejnych artykułów przedstawiamy gotowe procedury. Wybrano takie procedury, które mogą być często używane w różnych aplikacjach. Po zakończeniu tego cyklu powstanie biblioteka procedur, którą udostępnimy na dyskietce.

Mikroprocesor 8051 pracuje z cyklem maszynowym równym dwunastu okresom zegara. Lista rozkazów obejmuje rozkazy wykonywane w ciągu jednego, dwóch lub czterech cykli maszynowych.

Timery T0 i T1 potrafią zliczać cykle maszynowe, czyli wielokrotności dwunastek okresów zegara. Wykorzystajmy tę właściwość mikroprocesora do odmierzania czasu rzeczywistego. Przykładem niech będzie procedura sterująca migotaniem diody LED. Dioda ta będzie sterowana dowolną linią portu P1. Zmianę stanu jej świecenia na komplementarny będziemy realizować co 1s. Procedurę tę przedstawia listing jej semblacji (**list. 1**).

Zakładamy, że częstotliwość zegara wynosi 12MHz. Wtedy czas trwania jednego cyklu maszynowego wynosi

$$\frac{1}{12 \cdot 10^6} \cdot 12 = 10^{-6}\,\mathrm{s} = 1\,\mu\mathrm{s}$$

co zdecydowanie uprości nam obliczenia i procedurę.

Jedna sekunda to w tej sytuacji dokładnie jeden milion cykli maszynowych, które trzeba zliczyć, a potem zmienić stan wyjścia sterującego na przeciwny. Interesuje nas zatem licznik modulo 1000000. Pojedynczy, 16-bitowy timer oczywiście nie wystarczy, należy więc zliczać ilość przerwań od tego timera. Można wprawdzie zatrudnić drugi timer do kontynuacji zliczania, jednak procedura staje się skomplikowana. Trzeba bowiem zapewnić obsługę przerwań od każdego z timerów z osobna oraz połączyć jeden z drugim, tracąc tym samym dodatkowo jeden z nich oraz dwie linie portu P3. O ile stratę linii portu P3 można jeszcze przeboleć, o tyle brak jeszcze jednego układu czasowego do realizacji innej zależności w czasie rzeczywistym może być znacznym utrudnieniem, a taką konfigurację układową trzeba uznać za rozrzutność.

Zliczanie liczby przerwań od timera zrealizujemy w module obsługi przerwania. Wybieramy tryb 1 pracy timera T0 zliczającego impulsy zegarowe. W tym trybie układ czasowy jest 16-bitowym licznikiem zliczającym w przód, który ustawia przerzutnik TF0 (bit 5 rejestru TCON) w momencie przepełnienia, żądając tym samym obsługi przerwania.

Kolejną ważną sprawą jest przygotowanie timera T0 do realizacji zamierzonego celu. W tym celu należy odpowiednio skonfigurować system przerwań oraz sam układ czasowy T0. Zakładamy, że w naszym przykładzie procesor zajmuje się obsługą tylko tego przerwania.

Najpierw programujemy system przerwań. Ponieważ w naszym układzie nie ma potrzeby stosowania wyróżnienia układu czasowego T0, priorytet przerwań będzie ustawiony na poziomie niskim, zatem rejestr IP, przechowujący informację o priorytecie przerwań, będzie wyzerowany (linia 15 programu). Korzystamy więc z narzuconego przez producenta podstawowego, sprzętowego ustawienia kolejności wyboru obsługi przerwań, nadchodzących jednocześnie.

Do wyboru trybu pracy układów czasowych służy rejestr programujący o nazwie TMOD, którego poszczególne bity definiują działanie tych układów, po cztery bity na układ czasowy. A zatem układ czasowy T0 programujemy następująco, w kolejności poszczególnych bitów:

- parę bitów M1M0 (bity 1 i 0), określających numer trybu pracy układu czasowego, ustawiamy w stan 01 - tryb 1 układu,
- bit C/T (bit 2), określający źródło impulsów zliczanych, zerujemy - źródłem

List. 1.

```
1   0030              N PART_SEC   EQU  30H
2
3
4   0000              C            ORG  0
5   0000    2100      C            AJMP  RESTART
6
7
8   000B              C            ORG  11
9   000B    211A      C            AJMP  T0_SERVIS
10
11
12  0100              C            ORG  100H
13
14  0100    75A800    C RESTART:   MOV  IE,#0;  BLOKADA  WSZYSTKICH  PRZERWAN
15  0103    75B800    C            MOV  IP,#0B;  JEDNAKOWE  POZIOMY  PRZERWAN
16  0106    758901    C            MOV  TMOD,#01H;  TRYB  1  DLA  T0,  NAPEDZANY  Z  ZEGARA
17  0109    758C3C    C            MOV  TH0,#03CH
18  010C    758AAF    C            MOV  TL0,#0AFH
19  010F    753000    C            MOV  PART_SEC,#0
20  0112    758B10    C            MOV  TCON,#00010000B  ;  T0  JAKO  TIMER,  JUZ  LICZY
21  0115    75A882    C            MOV  IE,#10000010B;  ODBLOKOWANIE  T0
22                                 ;
23                                 ;PROGRAM  GLOWNY
24  0118              C MAIN:
25  0118    80FE      C            SJMP  MAIN
26
27                                 ;  OBSLUGA  PRZERWANIA  T0
28  011A    C0E0      C T0_SERVIS: PUSH  ACC          ;  |  ZEGAR  |  TH0    |  TL0   |  PART_SEC |
29  011C    758C3C    C            MOV  TH0,#03CH     ;  |  12MHZ  |  03CH   |  0B7H  |    40     |
30  011F    758AB7    C            MOV  TL0,#0B7H     ;  |   6MHZ  |  03CH   |  0B7H  |    20     |
31  0122    0530      C            INC  PART_SEC      ;  |   3MHZ  |  03CH   |  0B7H  |    10     |
32  0124    7428      C            MOV  A,#40
33  0126    B53008    C            CJNE  A,PART_SEC,T0_S1
34  0129    E590      C            MOV  A,P1
35  012B    F4        C            CPL  A
36  012C    F590      C            MOV  P1,A
37  012E    753000    C            MOV  PART_SEC,#0
38  0131              C T0_S1:
39  0131    D0E0      C            POP  ACC
40  0133    32        C            RETI
```

tych impulsów jest układ zegara procesora,

- bit GATE (bit 3), który definiuje możliwość bramkowania zliczanych impulsów za pomocą zewnętrznego wejścia INT0, zerujemy - impulsy nie będą bramkowane.

Pozostałe bity rejestru TMOD, dotyczące układu czasowego T1 zerujemy, ten układ nie będzie uruchamiany. Takie ustawienie realizuje linia 16.

Po określeniu charakteru pracy timera T0 pozostaje nam włączenie go do pracy oraz uaktywnienie systemu przerwań, który po restarcie procesora pozostaje w stanie pełnej blokady obsługi przerwań.

Rejestr TCON, sterujący pracą układów czasowych, zawiera m.in. bit TR0, którego ustawienie odblokowuje zliczanie przychodzących impulsów. Pozostałe bity tego rejestru zerujemy (linia 20).

Na koniec odblokowujemy system przerwań, czyli ustawiamy bit 1 rejestru IE zezwalający na obsługę przerwania od T0 oraz bit 7 rejestru IE, który zezwala na obsługę wszystkich włączonych przerwań indywidualnie (linia 21).

Obsługę przerwania od T0 realizuje się w osobnej procedurze adresowanej za pomocą tzw. wektora czyli rozkazu skoku do tej procedury ze ściśle określonego miejsca w programie. W naszym przypadku tym miejscem programu jest komórka pamięci o adresie 0BH. Taka organizacja pamięci pozwala na oszczędną relizację odwołań do procedury przerwań poprzez wystawienie nie więcej niż ośmiobitowego adresu na szynę adresową. Stamtąd natomiast można już wykonać skok do właściwej procedury obsługi przerwania.

Zastanówmy się teraz nad programową konstrukcją licznika zliczającego do 1000000. Zostało już powiedziane, że w obsłudze przerwania będzie zawarty również licznik tych przerwań, co zwielokrotni pojemność timera, a dodatkowo tutaj będziemy przełączać diodę LED. Należy wykorzystać pojemność licznika timera do maksimum po to, aby rzadziej było generowane przerwanie. Taką liczbą, która się sama nasuwa jest 50000, bowiem 1000000=20*50000. Oznacza to, że co dwudzieste przerwanie trzeba zmienić stan wyjść portu P1. Zatem naszym zadaniem będzie odpowiedź na pytanie, jaką wartość początkową ustawić w rejestrach TH0 i TL0, żeby obsługa przerwania następowała dokładnie po 50000 cykli maszynowych.

Timer T0 w trybie 1 jest licznikiem zliczającym w przód i generującym przerwanie w momencie przepełnienia. Przerwanie to zostanie obsłużone dopiero po zakończeniu wykonywania bieżącej instrukcji programu głównego, zaś następny cykl będzie przeznaczony na wypracowanie decyzji o wyborze przerwania. Pozostaje 65535 cykli. Licznik powinien zliczać od wartości 65535-50000=15535, pod warunkiem, że tę liczbę wpisuje się do pary rejestrów TH0TL0 natychmiast. Jest to oczywiście nieprawdą, minie nieco czasu potrzebnego na wykonanie wszystkich niezbędnych operacji. Czas na ich wykonanie skróci okres zliczania przez timer T0, zwiększając tym samym liczbę początkową nastawy liczników. Szczegółowe rozliczenie czasu jest następujące:

15535 + 2 (AJMP T0_SERVIS) + 2 (PUSH ACC) + 2 (MOV TH0,#....) + 2 (MOV TL0,#...) = 15543. Reprezentacja szesnastkowa tej liczby to 03CB7H.

Dalsza część procedury obsługi przerwania zajmuje się zliczaniem liczby przerwań i zmianą stanu linii portu P1 co dwudzieste przerwanie. Jeśli do procedury restartującej wpiszemy dodatkową linię

MOV P1,#55H

to możemy uzyskać efekt poruszających się punktów. Oczywiście, skracając czas zliczania, czyli wpisując do pary rejstrów TH0TL0 większą liczbę początkową i/lub mniejszą liczbę, z którą będzie porównywana komórka licznika przerwań PART_SEC, zwiększa się prędkość ruchu takich punktów.

Zmiana częstotliwości zegara na 6MHz czy 3 MHz powoduje, że czas pomiędzy przerwaniami ulegnie wydłużeniu odpowiednio dwu- i czterokrotnie, czyli w celu zachowania pomiaru tego samego odstępu czasu trzeba również odpowiednio zmniejszyć liczbę przerwań pomiędzy zmianami stanu portu, co pokazano w tabeli zawartej w listingu 1.

System przerwań układu 80C51 ma tę właściwość, że jest dwupriorytetowy i wielopoziomowy, czyli nie zawsze może obsłużyć żądanie przerwania, co więcej, trudno precyzyjnie określić moment rozpoczęcia realizacji tej obsługi. Zatem, gdy inne źródła przerwań będą uaktywnione, to ta procedura będzie powodować spóźnianie się układu czasowego. W niektórych sytuacjach może to nie mieć istotnego znaczenia i ta dokładność nam wystarczy. Jeśli będzie nam potrzebna procedura znacznie dokładniejsza, to musimy skorzystać z nieco innego algorytmu.

Wskaźniki zgłoszenia przerwań, w tym też interesujący nas TF0, są próbkowane zawsze w dziesiątym okresie zegara każdego cyklu maszynowego. Wyniki tego testu są brane pod uwagę w następnym cyklu maszynowym. Jeśli jeden ze wskaźników był ustawiony, system

List. 2.

```
1   0030          N     PART_SEC  EQU  30H
2   0000          N     R0REG     EQU  0
3   0001          N     R1REG     EQU  1
4
5
6   0000          C               ORG  0
7   0000  2100    C               AJMP  RESTART
8
9
10  000B          C               ORG  11
11  000B  2120    C               AJMP  T0_SERVIS
12
13
14  0100          C               ORG  100H
15
16  0100  75A800  C     RESTART:  MOV  IE,#0;  BLOKADA  WSZYSTKICH  PRZERWAN
17  0103  75B800  C               MOV  IP,#0B;  JEDNAKOWE  POZIOMY  PRZERWAN
18  0106  758901  C               MOV  TMOD,#01H;  TRYB  1  DLA  T0,  NAPEDZANY  Z  ZEGARA
19  0109  758C3C  C               MOV  TH0,#03CH
20  010C  758AAF  C               MOV  TL0,#0AFH
21  010F  753000  C               MOV  PART_SEC,#0
22  0112  758810  C               MOV  TCON,#00010000B  ;  T0  JAKO  TIMER,  JUZ  LICZY
23  0115  75A882  C               MOV  IE,#10000010B;  ODBLOKOWANIE  T0
24                                          :
25                                ;PROGRAM  GLOWNY
26  0118          C     MAIN:
27  0118  00      C               NOP
28  0119  2400    C               ADD  A,#0
29  011B  08      C               INC  R0
30  011C  09      C               INC  R1
31  011D  A4      C               MUL  AB
32  011E  80FB    C               SJMP  MAIN
33
34                                ;  OBSLUGA  PRZERWANIA  T0
35  0120  C0E0    C     T0_SERVIS: PUSH  ACC
36  0122  C000    C               PUSH  R0REG
37  0124  C001    C               PUSH  R1REG
38  0126  00      C               NOP
39  0127  A98C    C               MOV  R1,TH0
40  0129  A88A    C               MOV  R0,TL0
41  012B  74D2    C               MOV  A,#0D2H
42  012D  C3      C               CLR  C
43  012E  98      C               SUBB  A,R0
44  012F  F8      C               MOV  R0,A
45  0130  743C    C               MOV  A,#03CH
46  0132  99      C               SUBB  A,R1
47  0133  F9      C               MOV  R1,A
48  0134  898C    C               MOV  TH0,R1
49  0136  888A    C               MOV  TL0,R0
50  0138  0530    C               INC  PART_SEC
51  013A  7428    C               MOV  A,#40
52  013C  B53008  C               CJNE  A,PART_SEC,T0_S1
53  013F  E590    C               MOV  A,P1
54  0141  F4      C               CPL  A
55  0142  F590    C               MOV  P1,A
56  0144  753000  C               MOV  PART_SEC,#0
57  0147          C     T0_S1:
58  0147  D001    C               POP  R1REG
59  0149  D000    C               POP  R0REG
60  014B  D0E0    C               POP  ACC
61  014D  32      C               RETI
```

przerwań wygeneruje instrukcję LCALL z adresem właściwego wektora obsługi, ale jej realizacja będzie wstrzymana w jednym z trzech następujących przypadków:

1. Aktualnie wykonywany jest podprogram obsługi przerwania o równym bądź wyższym priorytecie.
2. Bieżący cykl nie jest ostatnim w wykonywanej właśnie instrukcji.
3. Wykonywana jest instrukcja powrotu z obsługi przerwania RETI albo instrukcja zapisu do rejestru IE lub IP.

Warunek 1 powoduje, że zagnieżdżanie przerwań, ze względu na niewielką pojemność stosu, jest ograniczone do dwóch. Drugi przypadek zapewnia dokończenie bieżąco wykonywanej instrukcji. Blokada obsługi przerwania w trzecim przypadku umożliwia zakończenie obsługi innego przerwania lub zmiany ustawienia systemu przerwań. Należy pamiętać, że jeśli żądanie obsługi przerwania z wymienionych powodów nie zostanie zrealizowane w bieżącym cyklu maszynowym, to o jego obsłudze decyduje ustawienie wskaźnika przerwania. Fakt, że przerwanie zostało przyjęte, nie jest zapamiętywany, czyli proces arbitrażu i wyboru przerwań jest procesem bez pamięci.

Biorąc pod uwagę te informacje, należy ustalać dynamicznie wartość liczby, od której będzie zliczał timer T0. Zmodyfikowaną procedurę przedstwaiono na **listingu 2**.

W module obsługi przerwania od liczby 50000 jest odejmowana liczba impulsów już zliczonych przez układ czasowy. Dodatkowo awansem liczba 50000 została pomniejszona o czas wykonania niezbędnych operacji odejmowania i przesyłania danych pomiędzy komórkami pamięci.

Gorzej przedstawia się sprawa w przypadku stosowania zegara o częstotliwości, która nie daje czasu trwania cyklu maszynowego równego wielokrotności 1µs, np. 10MHz. Wtedy należy nieco inaczej realizować pomiar czasu. Można wprawdzie pogodzić się z niewielką niedokładnością i stosować przedstawione algorytmy, niemniej warto wiedzieć, jak napisać program dla mikroprocesora taktowanego zegarem o takiej częstotliwości.

Jeśli iloraz z dzielenia częstotliwości zegara przez 12 jest liczbą ułamkową, to co pewną, określoną liczbę sekund musimy zliczyć pewną liczbę impulsów więcej lub mniej. Np. dla częstotliwości 10MHz dostajemy do zliczenia 8333333 i jedną trzecią czasu trwania cyklu maszynowego. Jeśli w ciągu trzech sekund zliczymy dwa razy po 833333 cykle i jeden raz 833334, to tak jak byśmy zliczyli w ciągu jednej sekundy wymaganą, ułamkową lich iczbę.

W tym arykule przedstawiliśmy tylko elementarne procedury uzależnień czasoywch. Niemniej, umiejętność pisania tych procedur będzie bardzo przydatna w samodzielnym programowaniu mikroprocesorów '51 w różnorodnych aplikacjach.

Mirosław Lach