АРХИТЕКТУРА МИКРОПРОЦЕССОРОВ

ПРАКТИЧЕСКИЙ КУРС

Часть 5

Типовые программы

Пример 18. Организовать устранение дребезга контактов. Метод подсчета заданного числа совпадающих значений сигнала. Проверяют N-раз подряд, что контакт замкнут. Если хоть один раз при этом продетектировали, что он разомкнут, то начинают проверку снова.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Процедура определения, что контакт разомкнут
; (=1) методом многократной проверки
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ORG   30H         ;
Begin:                    ;
DBNC:   MOV   R3,#20      ; Инициализация счетчика
DBNC1:  JNB   P3.4,DBNC   ; Если =0, то начать заново
        DJNZ  R3,DBNC1    ; N раз проверить, что =1
        END               ;
```

В методе временной задержки программа, обнаружив размыкание контакта (=1), запрещает доступ к нему на время задержки. Время задержки заведомо больше длительности переходных процессов в датчике (1–10 мс). Задержку формирует подпрограмма DELAY.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Процедура определения, что контакт разомкнут
; (=1) методом задержки
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ORG   30H         ;
Begin:                    ;
DBNC:   JNB   P3.4,DBNC   ; Если =0, то начать заново
        CALL  DELAY       ; Вызов п/п задержки
        END               ;
```

Пример 19. Организовать подсчет числа импульсов. Пусть импульсы поступают на вход T1 таймера/счетчика T/C1 (режим 1) и их число не превышает 65535. Логика работы управляющих цепей счетчика представлена на рис. 1.3. По рисунку выбираем GATE=0, $\overline{\text{INT1}}$=1 (исходный уровень) и C/$\overline{\text{T}}$=1. В этом случае старт счетчика будет осуществляться сигналом TR1=1, а стоп – TR1=0. Сигналы GATE, C/$\overline{\text{T}}$ и выбор режима задаются состоянием битов регистра TMOD (рис. 1.3). Сигнал TR1 задается состоянием бита TCON.6.

Рассмотрим три варианта организации программы. Для настройки счетчика 1 выбрано управляющее слово 50H = 01010000B. Это соответствует значениям старшей тетрады TMOD: TMOD.7 (GATE) = 0; TMOD.6 (C/T) = 1 (счетчик внешних событий); TMOD.5=0, TMOD.4 = 1 (режим 1).

1. Подсчет числа импульсов между внешними событиями. В каче-

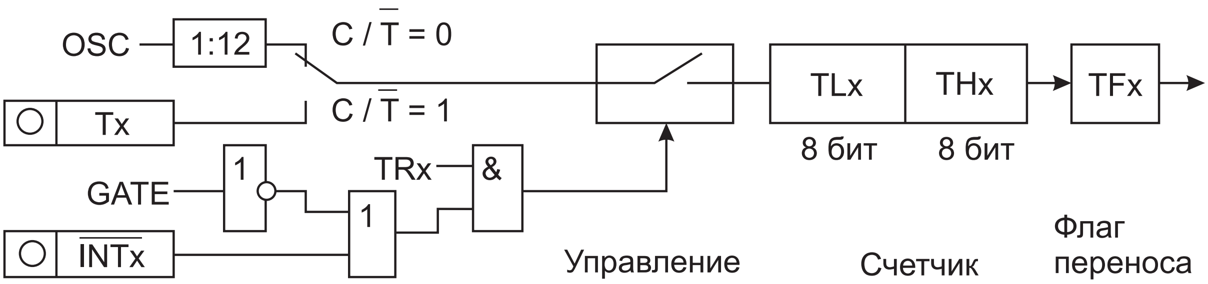

Рис. 1.3. Логика управления каналом таймера/счетчика T/Cx в первом режиме. Параметр $x = 1$ для T/C1 и $x = 0$ для T/C0

стве внешних событий используем сигнал на ножке P3.5 (T1), а сигнал управления «Вкл. конвейер» подается на ножку P3.4: 0 – пуск, 1 – стоп. Состояние ножки P3.4 будем опрашивать программно. Выбираем первый режим работы счетчика-таймера. Ниже приведен текст программы, реализующий поставленную задачу.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Импульсы поступают на ножку T1.
; 'Вкл. конвейер' (=0) поступает на ножку P3.4
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       ORG   30H        ;
Begin: MOV   TMOD,#50H  ; Настройка счетчика 1
       CLR   A          ; Очистка (A)
       MOV   TH1,A      ; Сброс счетчика
       MOV   TL1,A      ;
       JB    P3.4,$     ; Ожидание вкл. конвейера
       SETB  TCON.6     ; Пуск счетчика 1
       JNB   P3.4,$     ; Ожидание выкл. конвейера
       CLR   TCON.6     ; Останов счетчика
       MOV   B,TH1      ; Сохранение результата
       MOV   A,TL1      ; подсчета
       END              ;
```

2. Подсчет числа импульсов за заданный промежуток времени. В этом случае пуск счетчика происходит при запуске программы. Особенностью программы является использование дополнительной подпрограммы временной задержки с именем DELAY. Отличие от предыдущей программы – лишь в организации старта и стопа счетчика. Останов счетчика определяется программной временной задержкой, которую формирует подпрограмма DELAY. Ниже приведены тексты основной программы и подпрограммы DELAY.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Импульсы поступают на ножку T1.
; Программная задержка
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       ORG   30H        ;
Begin: MOV   TMOD,#50H  ; Настройка счетчика 1
       CLR   A          ; Очистка (A)
       MOV   TH1,A      ; Сброс счетчика
       MOV   TL1,A      ;
       SETB  TCON.6     ; Пуск счетчика 1
       CALL  DELAY      ; Задание промежутка времени
       CLR   TCON.6     ; Останов счетчика
       MOV   B,TH1      ; Сохранение результата
       MOV   A,TL1      ; подсчета
       SJMP  $          ; Конец основной программы
;Подпрограмма задержки на заданное время
DELAY: MOV   R1,#099H   ; Задержка около 150 мкс
       DJNZ  R1,$       ; Организация цикла
       RET              ; Возврат из подпрограммы
       END              ;
```

Принцип работы подпрограммы DELAY ясен из приведенного текста. Для формирования временной задержки используется оператор цикла DJNZ с однобайтным счетчиком цикла. Для увеличения времени задержки возможно использование как программных циклов с двухбайтным счетчиком цикла, так и нескольких вложенных программных циклов с однобайтными счетчиками цикла.

3. В данном случае используется полностью аппаратный метод подсчета. Временную задержку формируем таймером T/C0. Для таймера T/C0 выбираем режим 1, GATE=0, $\overline{\text{INT0}}$=1, C/$\overline{\text{T}}$=0.

При тактовой частоте 12 МГц изменение состояния таймера будет происходить каждую микросекунду. Для заданного времени измерения, например 10 мс, необходимо отсчитать 10000 импульсов. Поскольку счет ведется до переполнения таймера, то исходное число в таймере должно быть дополнением до состояния «все единицы» +1. Дополнительная единица обусловлена тем, что флаг переполнения таймера устанавливается не при переходе в состояние «все единицы», а при следующем импульсе, т.е. при переходе в состояние «все нули». Остальное – как в предыдущем примере.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Импульсы поступают на ножку T1.
; Интервал счета задается таймером T0.
; Тактовая частота F = 12 МГц.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       ORG   30H               ;
;Константа TIME для отсчета интервала времени 10 мс
TIME   EQU   NOT(10000)+1      ;
;Маска для одновременного пуска счетчика и таймера
Start  EQU   01010000B         ;
;Маска для останова счетчика и таймера
Stop   EQU   NOT(Start)        ;
Begin: MOV   TMOD,#50H         ; Настройка счетчика
       CLR   A                 ; Очистка (A)
       MOV   TH1,A             ; Сброс счетчика
       MOV   TL1,A             ;
       MOV   TH0,#HIGH(TIME)   ; Загрузка в TC0
       MOV   TL0,#LOW(TIME)    ; константы TIME
;Одновременный пуск счетчика и таймера
       ORL   TCON,#Start       ;
WAIT:  JBC   TCON.5,EXIT       ; Ожидание флага TF0,
                               ; Переход и сброс TF0
       SJMP  WAIT              ; Организация цикла
;Одновременный останов счетчика и таймера
EXIT:  ANL   TCON,#Stop        ;
       MOV   B,TH1             ; Результат счета
       MOV   A,TL1             ; в регистрах (BA)
       END                     ;
```

В данной программе использованы зарезервированные слова ассемблера ASM-51.

NOT – означает дополнение до единицы (инверсию) операнда, указанного в круглых скобках;

HIGH, LOW – означают операции выделения старшего или младшего байта операнда, указанного в круглых скобках.

Отметим, что операции NOT, HIGH и LOW выполняет программа ассемблер, а не микроконтроллер. Итоговый машинный код содержит только результаты выполнения этих операций, которые для микроконтроллера являются константами.

Пример 20. Произвести опрос группы двоичных датчиков. Сравнить входные сигналы порта P0 с эталонным кодом в аккумуляторе. Запрограммировать ожидание заданной комбинации сигналов группы датчиков. Решение поставленной задачи основано на использовании особенности команды CJNE. Данная команда проводит сравнение однобайтовых значений первого и второго операндов и осуществляет переход по указанному адресу при неравенстве этих операндов.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Ожидание заданного кода MYCODE в P0
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       ORG   30H         ;
MYCODE EQU   11001110B   ; Эталонный код
Begin:                   ;
WTCODE:MOV   A,#MYCODE   ; Загрузка кода
       MOV   P0,#0FFH    ; Все линии P0 на ввод
       CJNE  A,P0,$      ; Ожидание кода
       END               ;
```

Рассмотрим другую программу. Передача управления одной из семи подпрограмм в зависимости от кода на P0.0 – P0.2.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Передача управления одной из семи программ
; в зависимости от кода на P0.0 -- P0.2
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
       ORG   0H                 ;
       SJMP  Begin              ;
       ORG   30H                ;
Begin:                          ;
       MOV   DPH,#HIGH(PROG0)   ; ст. байт адреса
       MOV   P0,#0FFH           ; прогр. на ввод
       MOV   A,P0               ; чтение порта
       ANL   A,#07H             ; маскирование битов
       ADD   A,#5               ; 5 доп. байтов
       MOVC  A,@A+PC            ;
       MOV   DPL,A              ; 2Б
       MOV   A,#0               ; 2Б
       JMP   @A+DPTR            ; 1Б
;---------------------------------------
```

```
;--------------------------------------
BASE:   DB   LOW PROG0  ;
        DB   LOW PROG1  ;
        DB   LOW PROG2  ;
        DB   LOW PROG3  ;
        DB   LOW PROG4  ;
        DB   LOW PROG5  ;
        DB   LOW PROG6  ;
        DB   LOW PROG7  ;
;--------------------------------------
        ORG  100H       ;
PROG0:  NOP             ;
;--------------------------------------
        ORG  110H       ;
PROG1:  NOP             ;
;--------------------------------------
        ORG  120H       ;
PROG2:  NOP             ;
;--------------------------------------
        ORG  130H       ;
PROG3:  NOP             ;
;--------------------------------------
        ORG  140H       ;
PROG4:  NOP             ;
;--------------------------------------
        ORG  150H       ;
PROG5:  NOP             ;
;--------------------------------------
        ORG  160H       ;
PROG6:  NOP             ;
;--------------------------------------
        ORG  170H       ;
PROG7:  NOP             ;
;--------------------------------------
        END             ;
```

В данной программе для организации передачи управления использована команда перехода по косвенному адресу JMP @A+DPTR. Отметим, что в системе команд микроконтроллера x51 это единственная команда, допускающая переменную в адресе перехода.