ОБРАБОТКА СИГНАЛОВ НА OCHOBЕ MCS-51

**Введение**

В данном курсовом проекте рассматриваются практические примеры и их программная реализация на языке ассемблера для микроконтроллера семейства MCS-51 (МК51).

При рассмотрении решения задач, связанных с генерацией временных интервалов и работой последовательного порта предполагается использование МК с частотой задающего генератора равной 12 МГц.

**1. Использование команд передачи данных**

*Пример 1.* Передать содержимое буфера последовательного адаптера в резидентную память данных по косвенному адресу в R0:

MOV @R0, SBUF; передача принятого по последовательному каналу; байта в РПД.

*Пример 2.* Записать в ячейки РПД с адресами 39 и 40 число FA1BH:

LOAD: MOV 39H,#0FAH

MOV 40H,#1BH

*Пример З.* Загрузить в указатель данных начальный адрес 4300Н массива данных, расположенного во внешней памяти данных:

MOV DPTR,#4300Н ; загрузка начального значения указателя данных.

*Пример 4.* Загрузить управляющее слово в регистр управления таймером:

MOV TCON,#00000101В; разрешение внешних прерываний по низкому уровню сигнала.

*Пример 5.* Сбросить все флажки пользователя (биты) в резидентной области памяти от 20Н до 2FH:

MOV R0,#20Н ; заданный начальный адрес области памяти

MOV Rl,#0FH ; загрузить счетчик (длина области памяти)

LOOP: MOV @R0,#0 ; снять один байт (8 флажков)

INC R0 ; переход к следующему байту

DJNZ R1, LOOP ; цикл, если не все флажки сброшены

*Пример 6.* Передать управление по метке L0, если счетчик 0 достиг значения 128:

MOV A, TL0 ; передача содержимого счетчика в аккумулятор

JB А.7, L0 ; перейти на L0, если А.7=1

…

L0: ; продолжение программы

*Пример* Запомнить во внешней памяти данных содержимое регистров   
банка 0. Начальный адрес ВПД – 2000Н:

MOV PSW,#00010000B; выбор банка регистров 1

MOV R0,#8 ; счетчик ← 8

MOV DPTR, #2000H; определение начального адреса ВПД

MOV Rl,#0 ; определение начального адреса РПД

LOOP: MOV A,@R1 ; (А) ← (регистр)

MOVX @DPTR, A; передача из аккумулятора в ВПД

INC Rl ; переход к следующему регистру

INC DPTR ; приращение указателя адреса

DJNZ R0, LOOP ; R0=R0–1, если R0>0 то повторить цикл

*Пример 8.* Обращение к памяти программ, где сохраняется готовая таблица значений. Для этого используется специальная команда MOVC. Например: программа для выбора значений функции синуса из таблицы с точностью 0,4% и дискретом 1°. Начальный параметр для подпрограммы есть значения угла х, которое находится в аккумуляторе. Данная программа работает без указателя данных DPTR. Инкремент аккумулятора перед обращением к таблице необходим в связи с использованием однобайтной команды возврата. Таблица синусов занимает в памяти 90 байтов.

; вычисление sin(x) по таблице значений: вход (А)←(х), х∈(0,89°);

; выход (А) ← дробная часть значения синуса

SIN: INC A ; инкремент аккумулятора

MOVC А,@А+РС ; загрузка значения синуса из таблицы

RET ; возвращение

; таблица синусов

SINUS: DB 00000000В; SIN (0)=0

DB 00000100В ; SIN (1)=0,017 DB 00001001В; SIN (2)=0,035

………………………

DB 11111111В ; SIN (89)=0,999

*Пример 9.* Операции со стеком и организация прерываний. Механизм доступа к стеку MCS-51: перед записью в стек содержимое регистра-указателя стека SP инкрементируется, а после чтения данных из стека декрементируется.

После начальной установки ОМЭВМ в SP заносится начальное значение 07Н. Для переопределения начального значения SP можно использовать команду MOV SP, #data16.

Таким образом, стек может быть расположен в любом месте РПД. Стек используется для организации обращения к подпрограммам и при обработке прерываний. Кроме того, может использоваться для временного хранения значений регистров специальных функций.

Подпрограмма обработки прерываний.

ORG 3 ; задание адреса вектора прерываний

SJMP SUBINO ; переход на подпрограмму обработки

ORG 30Н

SUBINO: PUSH PSW ; сохранение в стеке PSW

PUSH A ; сохранение в стеке аккумулятора А

PUSH В ; сохранение в стеке дополнения-аккумулятора В

PUSH DPL ; сохранение в стеке DPTR

PUSH DPH ; сохранение в стеке DPTR

MOV PSW,#00001000B; выбор другого банка регистров (1)

.

обработка прерываний

.

POP DPH ; восстановление DPTR

POP DPL ; восстановление DPTR

POP В ; восстановление В

POP A ; восстановление аккумулятора

POP PSW ; восстановление PSW и банка регистров 0

RETI ; возвращение в основную программу

Если SP=1FH, размещение регистров в стеке после входа в подпрограмму обработки будет таким, как показано на рис. 1.

|  |  |  |
| --- | --- | --- |
| 7FH |  |  |
| 26H | DPH | ←(SP) |
| 25H | DPL |  |
| 24H | В |  |
| 23H | А |  |
| 22H | PSW |  |
| 21H | РСН | Указатель  ←(счетчик)  команд PC |
| 20H | PCL |
| 1FH |  |
| 00H |  |  |

Рисунок 1 – Распределение памяти при вызове подпрограммы

*Пример 10.* Передать управление одной из восьми подпрограмм при появлении нулевого уровня на соответствующем входе порта 1. Высший приоритет входаР1.3.

ORL P1,#FFH ; настройка Р1 на ввод

L1: MOV А, Р1 ; ввод данных из порта

CPL А ; инверсия аккумулятора

JZ L1 ; ожидание появления первого нуля

JNB Р1.3, SUBR1; переход на первую подпрограмму

JNB P1.4, SUBR2; переход на вторую подпрограмму

…………………………………………

JNB P1.2, SUBR8; переход на восьмую подпрограмму

Порядок приоритетов определяется порядком проверки нулевого уровня на соответствующем входе и может быть любым.

**2 Арифметические операции в MCS-51**

*Пример 11.* Сложить десятичные двоично-кодированные числа, размещенные в А и R5:

ADD A, R5 ; двоичное сложение

DA A ; десятичная коррекция результата

*Пример 12.* Вычитание байтов. Данная операция может выполняться двумя способами: 1) перевести уменьшаемое как отрицательное в дополнительный код и выполнить операцию добавления; 2) перевести уменьшаемое в обратный код и произвести инверсию суммы.

Например, из данных в А необходимо отнять данные регистра R3. Вычитание выполняем по алгоритму:

А ← 

CPL A ; инверсия аккумулятора

ADD A, R3 ; добавление байтов

CPL A ; получение разности

*Пример 13.* Сложить два двоичных многобайтных числа. Оба слагаемых находятся в РПД, начиная из младшего байта. Начальный адрес слагаемых находится в R0 и R1, формат в R2. Результат размещают на месте первого слагаемого.

CLR С ; сбрасывание переноса

LOOP: MOV A,@R0 ; загрузка в аккумулятор; бегущего байта первого слагаемого

ADDC A,@R1 ; добавление байтов с учетом переноса

MOV @R0, A ; размещение байта результата

INC R0 ; смещение указателя

INC Rl ; смещение указателя

DJNZ R2, LOOP ; цикл, если не все байты просуммированы

Время суммирования составляет (1+7N) мкс, где N – длина в байтах.

*Пример 14.* Перемножить число в аккумуляторе на число 2х, где х ≤ 8 – значение в R6. Умножение на 2 заменяется арифметическим сдвигом аккумулятора и R0:

MOV R0,#0 ; сброс R0

CLR С ; сброс переноса

LI: RLC А ; сдвиг влево 16-разрядного данного в А и R0

ХСН А, R0

RLC A

ХСН А, R0

DJNZ R6, L1 ; цикл

*Пример 15.* Умножение (MUL). Выполняется для двух беззнаковых чисел, которые находятся в регистрах А и В. После выполнения операции младший байт произведения размещается в А, а старший – в В.

Умножение числа любого формата на константу 168. Число размещено в РПД, адрес младшего байта находится в R0. Формат числа в байтах задан в R2.

MOV А,#00Н ; сброс аккумулятора А

LOOP: ADD A,@R0 ; загрузка множимого

MOV B,#168D ; загрузка множителя

MUL AB ; перемножение

MOV @R0, A ; запись младшего байта частичного произведения

INC R0 ; прирост адреса

MOV А, В ; пересылка старшего байта частичного произведения в аккумулятор А

ХСН A,@R0 ; предшествующее формирование очередного байта произведения

JNZ R2, LOOP ; цикл, если не все байты начального числа перемноженные на константу.

Полученный результат находится на месте начального числа и занимает в РПД на 1 байт больше. Время вычисления произведения составляет (1+13N) мкс, где N – длина числа в байтах.

*Пример 16.* Деление (DIV) – делится содержимое аккумулятора на значение в В. После деления в А находится целая часть, в В-остаток. Команда может быть использована для быстрого преобразования двоичного числа в двоично-десятичный формат.

Программа переводит двоичное число, находящееся в аккумуляторе, в двоично-десятичный код, который может быть трехразрядным (в десятичной системе счисления). В этом случае число сотен будет размещено вR0**,** десятки и единицы – соответственно в А и В.

MOV B,#100D ; (B) ← 100 для вычисления числа сотен в числе

DIV AB ; в А є число сотен (то есть старшая цифра)

MOV R0, A ; пересылка в R0 старшей цифры

ХСН А, В ; пересылка остатка от деления входного числа в А

MOV B,#10D ; (В)←10 (определяется число десятков в числе)

DIV AB ; в А – число десятков, В-число единиц

SWAP A ; размещение числа десятков в старшей тетраде А

ADD А, В ; суммирование остатка (числа единиц) аккумулятор сохраняет две младших цифры.

Время преобразования – 16 мкс.

*Пример 1* Сравнение двух четырехразрядных чисел, которые подаются на входы порта Р2.

Сигнал равенства выводится на Р1.1, большее – Р1.2, меньшее **–** Р1.3, разрешение сравнения подается на Р1.0.

ANL P1,#11110001В ; сброс всех сравнений

L0: JNB Р1.0, L0 ; ожидание разрешения

MOV A, P2

ANL A,#F0H ; выделение первого числа

SWAP A

MOV В, A ; запоминание первого числа

CLR C ; сброс переноса

MOV А, P2

ANL A,#0FH ; выделение второго числа

SUBB А, В ; вычисление В

JZ LR ; переход, если А=В

JC LL ; переход, если А<В

SETB P1.2 ; результат А>В

JMP EXIT

LR: SETB P1.1 ; результат А=В

JMP EXIT

LL: SETB P1.3 ; результат А<В

EXIT: ; продолжение выполнения программы

Аналогично выполняется сравнение чисел большей разрядности. Если необходимо сравнивать многобайтные числа, то сравнение следует начинать со старших байтов.

**3. Логические операции в MCS-51**

*Пример 18.* Вывод управляющих сигналов из МК.

Формирование статических сигналов используется для управления исполнительным механизмом по принципу «включен-выключен». При этом соответствующие линии портов просто устанавливаются в 0 или 1.

Если имеем группу исполнительных механизмов, то для их управления необходимо сформировать управляющее слово (УС), каждый бит которого – соответствующий управляющий сигнал.

Для изменения УС достаточно выполнять логические операции над теми битами портов, которые необходимо изменить.

Например: ANL – для сброса тех битов УС, какие заданные нулями

ORL – для установки бит

XRL – инверсия бит по времени

*Пример 19.* Определение переполнений. При накоплении большого количества слагаемых может возникнуть переполнение разрядной сетки. Для определения этого используют модифицированный дополнительный код, который отличается от обычного дополнительного введением дополнительного знакового разряда. При добавлении k слагаемых таких разрядов должно быть r = log2 k.

Признаком переполнения служит отличие знаковых разрядов. Отметим, что использование r знаковых разрядов суживает диапазон изменения сигналов в 2r раз.

*Пример 20.* Сбросить биты 0,2,4,6 порта 2:

ANL Р2,#10101010В; сбрасывание бит 0,2,4,6 порта 2

*Пример 21.* Установить биты 0…3 из порта 1

ORL P1.#00001111В; (P1.0…P1.3) ← 1111

Можно также записать

ORL P1,#0FH

*Пример 22.* Выбрать нулевой регистровый банк:

ANL PSW,#11100111В; сбрасывание бит RS0 и RS1

*Пример 23.* Проинвертировать биты порта Р1, которые отвечают единичным битам в аккумуляторе:

XRL Р1, А ; сумма по модулю два значения порта 1 и аккумулятора

*Пример 24.* Проинвертировать биты 7,6,5,4 порта 0:

XRL Р,#11110000В; сумма по модулю два значений порта 0 и константы

Можно также записать XRL Р0, #0F0H

*Пример 25.* Проинвертировать биты 0…3 в аккумуляторе:

XRL A,#0FH ; сумма по модулю два значения аккумулятора и константы

*Пример 26.* Настроить биты 1,3,5,7 порта 1 на ввод:

ORL Р1,#10101010В; установление P1.1, P1.3, Р1.5, Р1.7

*Пример 2* Маскировка данных при вводе. Ввести в регистр R3 информацию из линий 1,3,5,6,7 порта 1:

MOV А, Р1 ; ввод байта с Р1

ANL А,#11101010В; маскировка

MOV R3, А ; передача в R3

*Пример 28.* Выполнить логический сдвиг вправо двухбайтового числа, которое размещено в R5, А:

SHIFTR: CLR С ; сбрасывание переноса

CPL С ; установление переноса

ХСН А, R5 ; обмен байтами

JNB А.7, L1 ; если R5.7=1, то снять флажок переноса

CLR C

RRC A ; сдвиг флажка переноса

L1: XCH A, R5 ; обмен

RRC A ; сдвиг младшего байта

*Пример 29.* Выполнить сдвиг влево двухбайтового числа, которое размещено в R5 и А:

SHIFT L: RLC А ; сдвиг младшего байта

XCH А, R5 ; обмен А и R5

RLC A ; сдвиг старшего байта

XCH А, R5 ; обмен

*Пример 30.* Управление группой бит порта.

В РПД находится массив распакованных десятичных цифр. Необходимо передать их внешнему устройству в соответствии с протоколом (рис. 2). Для передачи 4-х бит данных используются младшие линии порта 1. Линии Р1.4 и Р1.5 используются как сигналы квитирования, то есть передачу сигналов на выход МК сопровождает стробирующий сигнал на линии Р1.4. Внешнее устройство, которое приняло данные, сообщает об этом сигналом на входе Р1.5. Биты Р1.6 и Р1.7 не должны изменять своих значений.

Исходные данные программы: начальный адрес массива – (R0), длина массива – (R1).

ORL P1, #00100000В; настройка

; Р1.5 на ввод

LOOP: MOV A, @R0; загрузка байта

; в аккумулятор

ANL P1,#11100000В; сброс данных

; и строба

ORL P1, A ; выдача данных

ORL P1,#00010000В; выдача строба

WAIT: JNB Р1.5, WAIT; ожидание ответа

INC R0 ; продвижение указателя адреса

JNZ Rl, LOOP ; цикл, если не все данные переданные

**4. Битовые операции в MCS-51**

*Пример 31.* Операция Исключающее ИЛИ используется очень часто для сравнения бит, или их сброса. В кодах битового процессора такой операции нет, но легко осуществляется наложение логической операции Исключающее ИЛИ на флажок переноса:

JNB bit, LZ ; исключительное ИЛИ для флажка переноса

CPL С ; инверсия флажка

LZ: ; продолжение программы

*Пример 32.* Преобразовать байт в последовательный код и передать его через Р1.0, не воздействуя при этом на остальные разряды порта. Передачу вести, начиная с младшего бита:

MOV R7,#8D ; инициализация счетчика циклов

LOOP: RRC А ; присвоение переноса значение бита А.0

MOV Р1.0, С ; передача бита

DJNZ R7, LOOP ; цикл, если не все биты переданы

Время выполнения программы 41 мкс, время передачи – 5 мкс (скорость передачи – 200 кбит/с).

*Пример 33.* Вычислить булеву функцию 3-х переменных Y=(X⋅)+W (X+V). Переменные X, V, W поступают на линии 2, 1, 0 порта 1; результат Y необходимо вывести на линию 3 порта 1. Для сохранения промежуточных значений использовать бит F0H.

Y BIT P1.3 ; спецификация бит порта 1

Х BIT P1.2

V BIT Р1.1

W BIT Р1.0

MOV С, X ; ввод Х

ANL С**,**/V ; X⋅

MOV F0, С ; запоминание результата в F0

MOV С, Х ; ввод Х

ORL С, V ; X+V

ANL С, W ; W (X+V)

ORL С, F0 ; (W (X+V))+(X⋅)

MOV Y, С ; вывод результата

Время выполнения программы 14 мкс.

*Пример 34.* Организовать последовательную передачу данных из аккумулятора на 0 вывод порта 2. Пересылку вести манчестерским кодом (каждый бит кодируется двумя интервалами: первый интервал имеет инверсию бита, второй – его прямое значение):

MOV R0,#8D ; инициализация счетчика бит

LOOP: RRC А ; (С)←(сдвиг мл. бита из акк-ра в перенос)

CPL С ; инверсия бита

MOV Р2.0, С ; передача инверсии бита

CPL С ; восстановление бита

NOP;

NOP ; выравнивание длины интервалов

NOP;

MOV P2.0, С ; передача прямого значения бита

DJNZ R0, LOOP ; цикл, если счетчик не нулевой

Передача начинается с младших битов. Продолжительность одного интервала 6 машинных циклов (6 мкс), время передачи одного бита – 12 мкс, время передачи байта – 96 мкс (скорость передачи 83 кбит/с, или 10,4 кбайт/с).

*Пример 35.* Пошаговый режим работы. Для его реализации необходимо:

* запрограммировать одно из внешних прерываний (к примеру INT0) на активизацию по уровню;
* в подпрограмме обработки прерывания ожидать последовательность «1» – «0» на входе INT0 (вывод P3.2), задаваемую, например, с помощью кнопки.

В основной программе необходимо дописать следующее:

SETB ІЕ.0 ; разрешение прерывания уровня 0

CLR TCON.0 ; прерывания разрешены по нулевому уровню

…; продолжение основной программы

Подпрограмма обработки прерывания должна закончиться следующими командами:

…; подпрограмма обработки

L1: JNB P3.2, L1 ; ожидание уровня 1

L2: JB P3.2, L2 ; ожидание уровня 0

RETI ; возвращение и выполнение одной; команды основной программы; после чего снова происходит ; возвращение в подпрограмму.

*Пример 36.* Обращение к медленным микросхемам внешней памяти. Программным путем можно задать необходимую длительность импульсов WR и RD. Например, если сигнал  должен длиться 50 мкс, то это осуществляется так:

CLR P3.7 ; =0

MOV R3,#24D ; инициализация счетчика (2 мкс)

L0: DJNZ R3**,** L0 ; цикл (24\*2 мкс)

SETB P3.7 ; =1

**5. Взаимодействие МК с объектом управления**

*Пример 3* Опрашивание двоичного датчика, например, конечного выключателя.

Ключ подключен к входу MCS-51: порт 1, разряд 3.

WAIT0: JNB P1.С, WAIT0; ожидание размыкания датчика

WAITC: JB P1.3, WAITC; ожидание замыкания датчика

*Пример 38.* Опрашивание группы двоичных датчиков (аналогично нахождению заданного кода или комбинации датчиков).

WTCODE: MOV A,#10D ; загрузка в А эталонного кода 00001010В

WAIT: CJNE A, P1, WAIT; если кодовая комбинация не совпала с заданной, то ждать

EXIT: ; вывод

*Пример 39.* При достижении значения параметра равного 135, передать управление на подпрограмму с меткой LABEL А, иначе LABEL В.

MOV A,#135D ; загрузка значения параметра

CJNE A, P1, LABEL В; сравнение и передача управления

LABEL А:

LABEL В:

*Пример 40.* Зафиксировать импульс, поступающий на вход ОМЭВМ (P1.3) и его окончание.

WAITC: JB P1.3, WAITC; ожидание Р1.3=0

WAIT0: JNB P1.3, WAIT0; ожидание Р1.3=1

Зафиксированный импульс имеет вид .

Для импульса обратного вида  нужно поменять WAIT0 и WAITC местами.

Таблица 1 – Минимальная продолжительность сигнала, которую определяет МК

|  |  |  |
| --- | --- | --- |
| Подключение датчика к выводам | Минимальная длительность импульса, мкс | |
| (отрицательного) | (положительного) |
| P1, P2, BUS/P0 | 10/2 | 12,5/2 |
| T0, Т1 | 5/2 | 5/2 |
| ЗПР | 10/2 | 5/2 |

*Пример 41.* Передать управление на метку TEST и установить Р3.7, если на Т0 поступит 30 импульсов.

MOV R1,#30D ; загрузка числа импульсов

L1: JB P3.4, L1 ; ожидание нуля

L0: JNB P3.4, L2 ; ожидание единицы

JMP L0

L2: DJNZ R1, L1 ; повторить 30 раз

JMP TEST

TEST: SETB Р3.7 ; установление бита

*Пример 42.* Схема для фиксации короткого импульса: D-триггер устанавливается коротким импульсом, а сбрасывается программно, после определения наличия сигнала на входе Т0.

*Пример 43.* Устранение дребезжания контактов. Дребезжание состоит в том, что благодаря упругим свойствам контактов возможен отрыв контактов, который приводит к переходному процессу. При этом сигнал может быть прочитан многократно как случайная последовательность нулей и единиц. Это можно устранить с помощью триггера (рис. 6).

Чаще всего с помощью MCS51 это делается программно так, как показано на рис. 7 и рис. 8.

Реализация первого метода, количество совпадений задано в R3.

DBNC: MOV R3,#3 ; инициализация счетчика

DBNC1: JB P3.4, DBNC; если контакт замкнутый,   
 ; то начать отсчет опрашиваний   
 ; сигнала сначала

DJNZ R3, DBNC1 ; повторять, пока значение в R3

; не станет равным 0

JB P3.4, DBNC; конечная проверка

Временная задержка подбирается экспериментальное (1–10 мс) в зависимости от типа датчика.

*Пример 44.* Формирование импульсных сигналов делается по принципу включен / выключен с необходимой временной задержкой:

PULLS:ON: ANL P1,#11110111B; выдача импульса в линию 3 порта 1

CALL DELAY ; временная задержка

OFF: ORL Р1,#00001000В; сброс

*Пример 45.* Считывание данных из таймера. Для устранения возможных сбоев при считывании сначала читается старший байт, потом – младший, после чего подтверждается, что старший байт за это время не изменился.

RDTIME: MOV A, TH0

MOV R0, TL0

CJNE A, TH0, RDTIME

MOV R1, А

RET

*Пример 46.* Подсчет числа импульсов, например, радиодеталей, которые движутся по конвейеру, зафиксированных фотоэлементом (рис. 9).

MOV TMOD,#01000000B; настройка счетчика 1

MOV TH1,#00H ; сброс счетчика

WAIT0: JB P3.4, WAIT0; ожидание включения транспортера

SETB TCON.6 ; запуск счетчика 1

WAITC: JNB P3.4, WAITC; ожидание отключения транспортера

CLR TCON.6 ; остановка счетчика 1

MOV A.TH1 ; A ← число деталей

EXIT: ; выход

Максимальное число радиодеталей – 255.

*Пример 4* Реализация функций времени на основе таймеров.

В MCS-51 при частоте генератора тактовых импульсов 12 МГц таймер / счетчик изменяет свое значение с частотой 1 Мгц (в режиме таймера), или при поступлении сигнала от внешнего источника через счетный вход T0, T1 (в режиме счетчика).

Если использовать таймер / счетчик в режиме 1 полного формата (16 бит), то получим задержки в границах (1 – 65536 мкс).

Программа задержки на 50 мс в MCS-51, прерывание разрешено. Организация перехода к метке NEXT при переполнении Т/С0:

ORG 0ВН ; адреса вектора прерываний от Т/С0

CLR TCON.4 ; остановка Т/С0

RETI ; выход из подпрограммы обработки прерываний

ORG 100H ; начальный адрес программы

MOV TMOD, #01Н; настройка Т/С0

MOV TL0, #LOW (NOT(50000) – 1); загрузка таймера

MOV TH0, #HIGH (NOT(50000) – 1)

SETB TCON.4 ; старт Т/С0

SETB IE.1 ; разрешение прерываний от Т/С0

SETB PCON.0 ; перевод в режим холостого хода

NEXT: ; продолжение программы.

*Пример 48.* Организация задержки программным путем:

DELAY: MOV R2, Х ; (R2)←(Х)

COUNT: DJNZ R2, COUNT; декремент R2 и цикл, если не нуль

RET ; возвращение из подпрограммы

*Пример 49.* Подпрограмма задержки на 50 мс, используя Т/С0, погрешность не превышает 2 мкс.

DELAY: MOV TMOD,#00000001B; настройка Т/С0

MOV TH0,#HIGH (NOT(50000–16))

MOV TL0,#LOW (NOT(50000–16))

SETB TCON.4 ; запуск Т/С0

JNB TCON.5,$ ; ожидание

ANL TCON,#NOT(30H); остановка Т/С0, сброс TF0

RET ; возвращение из подпрограммы

*Пример 50.* Подпрограмма задержки на 1 секунду. Погрешность не больше, чем 123 мкс.

OS: MOV R3,#20D ; счетчик циклов

S1: ACALL DELAY ; задержка на 50 мс

DJNZ R3, S1 ; организация цикла

RET ; возвращение из подпрограммы

*Пример 51.* Измерение временных интервалов. При применении таймера используется вход разрешения счетчика (). Тогда измеренный сигнал подается на вход , а измерение продолжительности выполняется в Т/С0.

«Положительный» импульс:

MOV TMOD,#00001001В; настраивание Т/С0

MOV TH0,#0 ; сбрасывание таймера

MOV TL0,#0

SETB TCON.4 ; запуск Т/С0

WAIT0: JNB P3.2, WAIT0; ожидание единичного уровня

WAITC: JB P3.2, WAITC; ожидание нулевого уровня

CLR TCON.4 ; остановка Т/С0

RET ; выход из процедуры

Управление программе должно передаваться при условии =0. Прерывание от Т/С0 и внешние от  – должны быть запрещены. По завершению программы в Т/С0 будет находиться число, пропорциональное продолжительности импульса на входе . Максимальная продолжительность импульса 65536 мкс, погрешность 1 мкс.

Если необходимо измерять большую продолжительность импульса, то можно программным путем подсчитывать число переполнений от таймера.

*Пример 52.* Преобразование параллельного кода в последовательный в MCS-51 сводится к передаче одного байта в буфер приемо-передатчика:

MOV SBUF, А

*Пример 53.* Обратное преобразование последовательного кода в параллельный инициируется стоповым битом и выполняется аппаратно без участия программы. Основная программа должна выполнить команду

MOV A, SBUF.

*Пример 54.* Преобразование однобайтовых чисел в дополнительный код и наоборот. Числа со знаком можно представить в дополнительном коде в границах от -128 к +12 Алгоритмы перевода чисел из прямого кода со знаком в дополнительный и обратное преобразование одинаковы:

DOD-PR: JNB А.7, EXIT; проверка знакового разряда

CPL А ; инверсия аккумулятора

ADD A, #1 ; добавление единицы

SETB А.7 ; установление знака

EXIT: ; выход

*Пример 55.* Преобразование чисел из одной системы исчисления в другую методом «двух счетчиков». При этом методе из начального кода отнимается, а к новому коду прибавляется единица до обнуления начального кода. Причем отнимание осуществляется в старой системе счисления, а добавление – в новой.

Программа перевода двоичного числа в двоично-десятичное.

MOV R5, А ; передача начального числа в R5

CLR А ; сброс аккумулятора

REV: ADD A,#1 ; добавление единицы

DA A ; десятичная коррекция

DJNZ R5, REV ; декремент начального кода и цикл, если не нуль

Результат преобразования находится в аккумуляторе.

*Пример 56.* Подсчет числа импульсов за заданный промежуток времени. Решается 4-я способами:

1. Программная реализация временного интервала и программный подсчет числа импульсов.

2. Программная реализация временного интервала и аппаратный подсчет числа импульсов (на внутреннем таймере / счетчике).

3. Аппаратная реализация временного интервала и программный подсчет числа импульсов.

4. Аппаратная реализация временного интервала и аппаратная реализация подсчета импульсов.

Реализация на MCS-51: в таймере-счетчике Т/С1 осуществляется подсчет числа импульсов; в Т/С0 заданный временной интервал. Датчик импульсов должен быть подключен к Т1.

TIME: EQU NOT(10000)+1; определение константы TIME для отсчета времени в 10 мс

MOV TMOD,#01010001B; настройка таймеров-счетчиков:

1-й – счетчик 16 бит, 0-й – таймер

CLR A ; сброс аккумулятора

MOV ТН1, А ; сброс Т/С1

MOV ТL1, А;

MOV ТН0,#НІGH(ТIМE); загрузка в Т/С0 константы TIME

MOV TL0,#LOW (TIME);

ORL TCON,#50H; запуск Т/С1 и Т/С0

WAIT: JBC TCON.5, EXIT; проверка переполнения Т/С0

SJMP WAIT ; цикл, если TF=0

EXIT: MOV В, ТН1 ; (В) – (А), число импульсов за время 10 мс

MOV A, TL1

программный команда управление микроконтроллер

**6. Реализация ЦАП и АЦП**

*Пример 5* Преобразование информации из цифровой формы в аналоговую осуществляется с помощью подключения интегральных микросхем цифроаналогового преобразователя (ЦАП) к одному из портов MCS-51. Выдача информации при этом осуществляется с помощью одной команды, например: MOV P1, А.

Если необходимо генерировать какую-то сложную функцию, то лучше записать ее таблично и передавать с соответствующим периодом в заданный порт.

Пример *58.* Аналого-цифровой преобразователь (АЦП) последовательного приближения (рис. 10).



Рисунок 10 – Реализация АЦП последовательного приближения

МК выдает через порт P1 байт данных, что превращается в аналоговую форму и сравнивается с входным сигналом компаратора, после чего анализируется результат сравнения. В зависимости от значения сигнала на входе Т0 MCS-51 или оставляет старший бит в единичном состоянии, или сбрасывает в 0. Потом аналогично проверяются все остальные биты.

R4 – регистр цифрового эквивалента;

R3 – регистр бегущего значения 1 для указания весового бита.

CONVRT: MOV R5#08Н ; загрузка счетчика циклов

MOV R3,#01Н

MOV R4,#00Н

LOOP: MOV А, R3 ; сдвиг R3 вправо на 1 разряд

RR A

MOV R3, А

ORL А, R4 ; логическое ИЛИ весового разряда и цифрового эквивалента

MOV Р1, А ; выдача в Р1

JBC T0, ENDUGH; если Т0=1, то аналоговый эквивалент выданного байта больше чем UBX и сохраняется в R4, если наоборот, то установленный бит запоминается в R4

MOV R4, А

ENDUGH: DJNZ R5, LOOP; декремент R5 и если не 0, то переход к анализу следующего бита

*Пример 59.* Реализация АЦП методом двойного интегрирования.

Метод двойного интегрирования заключается в следующем (рис. 11). Сначала интегрируется входной опорный сигнал ЕОП. На выходе интегратора устанавливается отрицательный потенциал, после чего интегрируется входной аналоговый сигнал за строго определенное время Т1. Т1 отсчитывается от момента перехода сигнала интегратора через 0. Потом подается UОП и измеряется время интегрирования Т2, которое пропорционально входному сигналу.



Рисунок 11 – Реализация АЦП двойного интегрирования

Максимальное время преобразования для точности 16 разрядов 2×65.536 мс. Если необходима точность преобразования 8 разрядов, то Т/С0 переключается в режим 8-битного таймера, а время преобразования составляет 2×256 мкс.



Рисунок 12 – Временные диаграммы сигналов АЦП  
двойного интегрирования

MOV TMOD,#01H; настройка Т/С0 на режим 16 бит

MOV TH0,#HIGH (NOT(T1)+1); загрузка Т/С0

MOV TL0,#LOW (NOT(T1)+1)

SETB P1.1 ; настройка Р1.1 на ввод

SETB Pl.0 ; подача ЕОП на интегратор

WAIT: JB P1.1, WAIT; ожидание появления на выходе интегратора отрицательного уровня сигнала

CLR P1.0 ; подача UBX на интегратор

WAITT0: JNB P1.1, WAIT0; ожидание момента Т0

SETB TCON.4 ; запуск Т/С0

WAITT1: JNB TCON5, WAITT1; ожидание момента Т1

SETB P1.0 ; подача ЕОП на интегратор и начало обратного интегрирования

WAITT2: JB P1,1, WAITT2; ожидание момента Т2

CLR TCON.4 ; остановка Т/С0

CLR TCON.5 ; сброс флажка TF0

MOV В, TH0 ; формирование результата в регистровой паре А, В

MOV A, TL0

**7 Локальная управляющая микросеть на основе MCS-51**

При управлении сложными технологическими объектами используются системы с распределенным управлением, которые состоят из группы контроллеров, руководящих отдельными агрегатами объекта (например, многосекционной печатной машиной, бумагообрабатывающей установкой, переплетно-обрабатывающей линией, прокатным станом и т.п.). Между отдельными подсистемами должен быть обеспеченная информационная связь. Поэтому обязательным элементом распределенной системы управления есть локальная сеть, которая объединяет определенные контроллеры в систему.

На такую сеть возлагаются обычно простые функции передачи сообщений за гарантированное время. Протяжность линий связи обычно не превышает десятков метров, размер сообщения – нескольких десятков байтов, а время доставки сообщений – в границах от 0.01 до 1 с. Типичными есть два режима информационного обмена в сети: *широковещательный,* если сообщение, которое присылается, назначается для всего остатка подсистем (микроконтроллеров сети), и *абонентский,* если сообщение назначается лишь одному МК. Обычно первым способом передаются разные информационные параметры, которые используются многими подсистемами. Это разрешает уменьшить загрузки сети за счет изъятия многочисленных передач одного и того же сообщения разным адресатам. Вторым способом обычно передаются команды управления от центрального устройства к исполнительным средствам или сообщения экстренного характера.

Наиболее широко распространены локальные сети двух структур: *кольцевые* и *моноканальные* (типа BITBUS). Последние есть более удобными для управляющих микросетей, так как допускают простое наращивание и модификацию системы. Кроме этого, в моноканальной сети время доставки сообщение не зависит от общего числа МК и они имеют высокую живучесть и надежность.

В сетях с единым моноканалом все МК связаны между собой одной общей (распределительной) линией связи (рис. 1). В линиях передачи данных обычно используется коаксиальный кабель или витая пара с резисторами согласования на их концах.

Известны разные методы доступа к распределительной линии (протоколы), которые разрешают осуществлять обмен данными между многими МК сети, то есть обеспечивать разделение канала связи между многими подсистемами.

Рассмотрим *интервально-маркерный метод* доступа к распределительному моноканалу, который разрешает устранить конфликты в канале и достаточно полно использовать пропускную способность канала. Суть метода состоит в следующем:

1. При нулевой загрузке в канале периодически появляется маркер, который генерируется одним из МК сети. Маркер содержит номер ведущего МК. Главное назначение ведущего – поддерживать синхронизм в сети за счет периодической выдачи маркера в канал.

2. Период генерации маркера состоит из определенного числа «окон», число которых равное числу МК в сети. Каждое окно имеет свой номер и принадлежит одной из подсистем.

3. В процессе захвата канала МК, которое желает выдать свой пакет (сообщение), должен дождаться появления маркера и отсчитать от него свое окно.

Если при этом его не опередят другие МК, то, дождавшись своего окна, подсистема может, не опасаясь конфликтов, начать передачу данных (рис. 14а).

4. После выдачи сообщения МК генерирует свой маркер и становится новым ведущим. Старый ведущий микросети, распознав, что моноканал захвачен, освобождается от этой роли.

5. Отсчет момента времени от маркера своего окна проводится по такому правилу. Продолжительность окна принимается равной времени передачи одного байта данных. Если ведущий имел номер *l*, то первое окно будет принадлежать МК с номером *l*+1, потом МК с номером *l*+2 и т.д. Время ожидания своего окна (Т) можно определить как T=t X, где t – время передачи одного байта, то есть продолжительность окна. Число Х определяется следующим образом:



где *k –* номер МК, который пробует захватить канал, *k* = 0÷(n-1); *l –* номер ведущего, *l* = 0÷(n-1); n – число МК в сети.

6. Если самый ведущему необходимо выдать сообщение, то он может захватить канал во время своего окна, то есть вместо генерации очередного маркера начать передавать свое сообщение.

Выдав маркер, ведущий МК запускает таймер на время t⋅(n-1), а если за это время ни единый МК не захватит канал, то весь цикл повторяется еще раз и т.д.

8. Любой МК принимает все байты, которые передаются по каналу. Для контроля пропадания маркера любой МК после приема каждого байта запускает таймер на задержку t⋅(n+1). Таким образом, пропадание маркера (а значит, и режима синхронизма микросети) фиксируется, если за время t⋅(n+1), не было передано ни одного сообщения.

9. При выявлении пропадания маркера для восстановления синхронизма в микросети любой МК выполняет следующие простые действия: выдерживает паузу продолжительностью t⋅(i+1), где *i –* собственный номер МК; если во время паузы снова не было принято ни одной информации, то данный МК становится ведущей и генерирует новый маркер.

Этой процедурой обеспечивается автоматическое восстановление работы микросети при отказе МК, которое есть в данный момент ведущим.

10. При интервально-маркерном методе удается избегнуть любых конфликтов в сети по следующим причинам:

* контроль пропадания маркера осуществляется постоянно всеми МК и он полностью синхронный, так как счетчики паузы корригируются приблизительно одновременно при приеме каждого байта. Все МК обнаружат пропадание маркера одновременно;
* одновременно начинается отсчет паузы t⋅(i+1) всеми МК;
* микроконтроллер с меньшим номером первым генерирует маркер и восстановит синхронизм в микросети.

Последовательный порт MCS-51 допускает передачу 9-битных кодов. Используя это, можно легко ввести признак маркера таким образом, что байт маркера будет отличаться от любого информационного байта. На рис. 2б изображена структура маркера; старший бит определяет признак маркера (для маркера – 1). Бит 7 используется для простейшего контроля за паритетом. Семибитное поле адреса разрешает иметь в системе до 127 подсистем с номерами от 0 до 126. Адреса 127 зарезервированная для широковещательной передачи.

Рекомендованный формат сообщения представленный на рис. 14в и предусматривает такие поля: А0, – адреса получателя; АВ – адреса отправителя; L – длина поля данных (0–255); CRC – байт контрольной суммы.

Можно определить четыре состояния, в которых будет находиться любой МК микросети.

*Приемник (R).* В этом состоянии МК прослушивает канал, принимает сообщение и выбирает из них необходимую ему информацию.

*Передатчик (W).* В этом состоянии МК, захватив канал, передает свое сообщение.

*Ведущий (Н).* МК – ведущий и поддерживает синхронизм в сети.

*Специальное состояние (RM).* МК реализует процедуру восстановления синхронизма в сети.

Граф состояний МК микросети представленный на рис 3.

Запрос на передачу сообщения формируется в МК прикладной программой управления объектом и обозначенный RQ. Задержки, которые реализуются таймером, имеют такое содержание:

ТМ1 – контроль исчезновения маркера, задержка равная τ⋅(n+1).

TМ2 – ожидание своего окна (τ⋅*х);*

ТМЗ – ожидание окончания периода сети (τ⋅*п);*

TМ4 – пауза перед выдачей маркера при восстановлении синхронизма, задержка равная τ⋅(i+1).

Для реализации подсистемы необходимы следующие ресурсы: УАПП, таймер, два уровня прерываний. Этими ресурсами владеет MCS51, что разрешает вести передачу и прием данных со скоростью до 375 кбит/с. Время передачи одного байта, обрамленного стартовыми и стоповыми битами (плюс 9-й разряд), составляет 58,7 мкс. Пропускная способность микросети при этом равна приблизительно 17 кбайт/с.

Микроконтроллер, который работает в составе распределенной системы управления на основе локальной микросети, должен, кроме прикладной программы управления, иметь еще программные средства доступа к моноканалу. Таким образом, МК должен работать в двухпрограммном режиме с распределением всех ресурсов между этими двумя сопрограммами. Понятно, что при этом должен быть реализован механизм взаимодействия между сетевой и прикладной программами. Чаще всего этот механизм реализуется путем присвоения сетевой программе более высокого приоритета.