**Миранян Сурен Артёмович**
**Санкт-Петербургский государственный политехнический университет**
**Факультет технической кибернетики**

# Системы контроля доступа.
# Руководство по лабораторным работам

**Санкт-Петербург 2012 г.**

# Аннотация

к руководству по лабораторным работам
“Системы контроля доступа”

В настоящем руководстве приводятся рекомендации по программированию современных систем контроля доступа, таких как считыватели магнитных карт, электронные таблетки (iButton), а также программирование некоторых устройств ввода вывода информации, применяемых с микропроцессорных системах контроля доступа. Рассматриваются вопросы программирования асинхронного последовательного интерфейса, графических жидкокристаллических дисплеев, а  также патентованный фирмой Intel способ организации матричной клавиатуры.

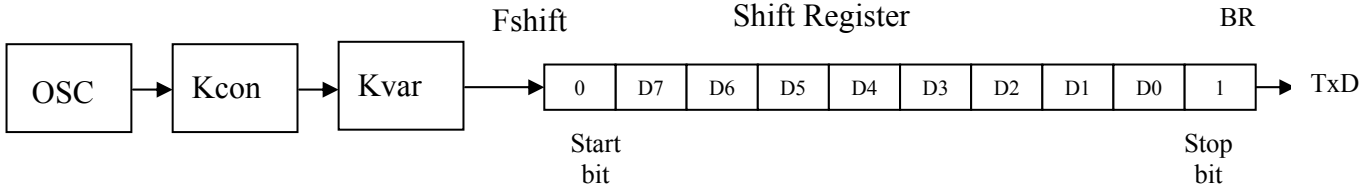
# ЛАБОРАТОРНАЯ РАБОТА 1

## Программирование порта RS232 процессоров MCS-51

**Цель работы-**приобретение навыков программирования UART интерфейса процессоров MCS-51 в режиме програмноуправляемого обмена и работы по прерываниям, при требуемых скоростях UART.

### 1. Установка скорости UART

Сигнал UART формируется сдвиговым регистром, на вход которого должна подаваться частота Fshift, задающая требуемую скорость BR UART. Поскольку один импульс этого сигнала Fshift выдвигает на линию TxD один бит данных, то частота сигнала Fshift в герцах должна быть равной требуемой скорости UART в бодах (бод=бит/сек.)




Сигнал Fshift формируется путём деления системной тактовой частоты OSC делителями частоты Kcon и Kvar с фиксированным и устанавливаемым коэффициентами деления таймеров CPU.

По требуемой скорости обмена BR, тактовой частоты OSC, определяется необходимый коэффициент деления счетчика

**Kvar = Fosc / (Kcon*BR)**

Для классического микропроцессора MCS 51 в качестве формирователя сигнала тактовой частоты сдвига может выбираться таймер 1 или 2. Структуры их приводятся на рисунках Рис.1 , Рис.2. При этом сигнал Ooverflow с выхода таймера 1 (Рис.1) подается в точку Timer1 Overflow на Рис.2. Из этих структур видно что

для T1 Kcon= 12*2*16. **Kvar = Fosc/(Kcon*BR)= Fosc/(**12*2*16**BR)**

для T2 Kcon= 2*16 **Kvar = Fosc/(Kcon*BR)= Fosc/(**2*16**BR)**

Счетчики с переменным коэффициентом деления представляют собой суммирующие счетчики с автоперезагрузкой, с предварительной установкой начального значения счета и вырабатывют один выходной импульс при переполнении. Этим же импульсом переполнения, по сигналу reload, в них перезагружается начальное значение счета. Счетчик T1 при

этом однобайтный (считает от 0 до 256), счетчик Т2- двухбайтовый (считает от 0 до 65536). Коэффициенты деления счетчиков определяются предустановленными в них значениями

для Т1 **Kvar=256-TH1.** **TH1=256- Fosc/(**12*2*16***BR).** **(1)**

для Т2 **Kvar= 65536-(RCAP2H, RCAP2L)** **(RCAP2H, RCAP2L)= 65536- Fosc/(**2*16***BR).** **(2)**

**Рис.1**

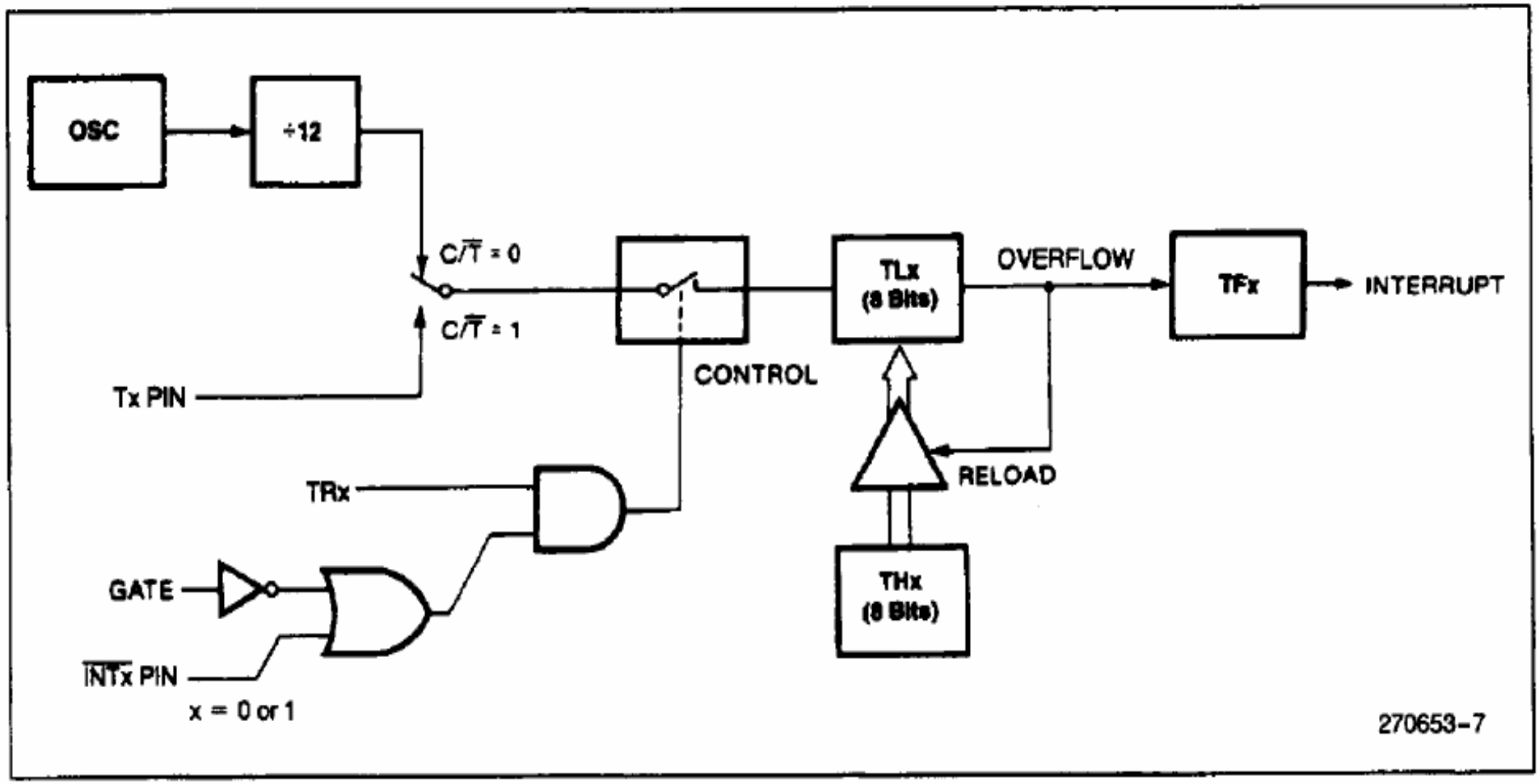



**Figure 10. Timer/Counter 1 Mode 2: 8-Bit Auto-Reload**

**Рис.2**

**Figure 4.** Timer 2 in Baud Rate Generator Mode

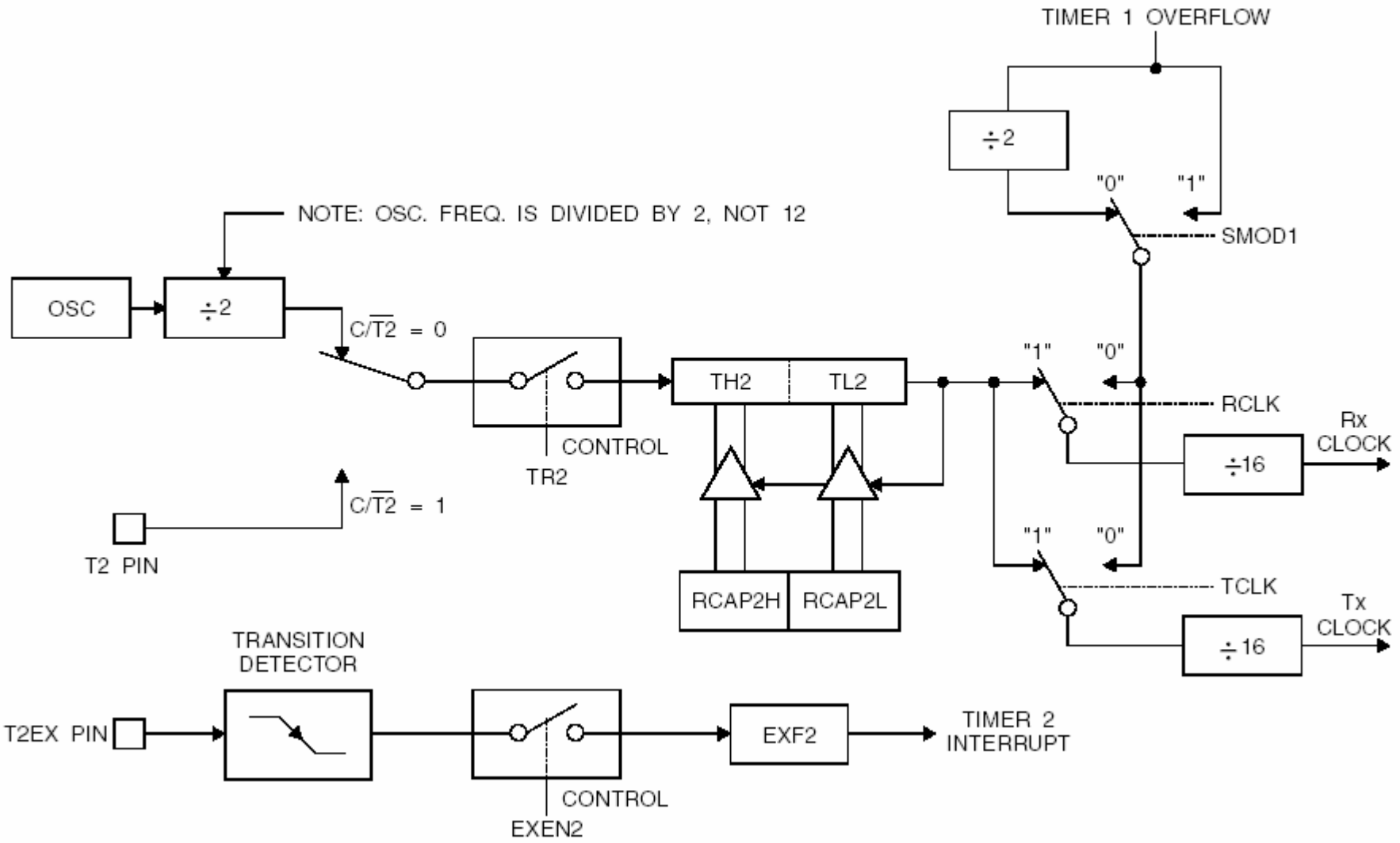


**Рис.3**

## TMOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE.

| GATE | C/$\overline{T}$ | M1 | M0 | GATE | C/$\overline{T}$ | M1 | M0 |
|---|---|---|---|---|---|---|---|
| TIMER 1 | | | | TIMER 0 | | | |

GATE When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).

C/$\overline{T}$ Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).

M1 Mode selector bit. (NOTE 1)

M0 Mode selector bit. (NOTE 1)

**NOTE 1:**

| M1 | M0 | Operating Mode | |
|---|---|---|---|
| 0 | 0 | 0 | 13-bit Timer (MCS-48 compatible) |
| 0 | 1 | 1 | 16-bit Timer/Counter |
| 1 | 0 | 2 | 8-bit Auto-Reload Timer/Counter |
| 1 | 1 | 3 | (Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits, TH0 is an 8-bit Timer and is controlled by Timer 1 control bits. |
| 1 | 1 | 3 | (Timer 1) Timer/Counter 1 stopped. |

**Для инициализации UART с таймером 1**, согласно структуре таймера (Рис.1) необходимо:

- В регистре TMOD установить режим ”8 бит с автоперезагрузкой ”
- Установить TR1=1 для подачи тактовой частоты на вход счетчика

- Установить начальное значение счета в регистрах TH1, TL1 согласно (1).
- В регистре SCON выбрать – асинхронный режим, передача 10 бит с программируемой скоростью обмена.

**Для инициализации UART с таймером 2**, согласно структуре таймера (Рис.2) необходимо:

- Установить TR2=1 для подачи тактовой частоты на вход счетчика
- RCLK, TCLK установить в 1
- Установить начальное значение счета в регистрах RCAP2H, RCAP2L согласно (2).
- В регистре SCON выбрать – асинхронный режим , передача 10 бит с программируемой скоростью обмена.

Проверка скорости UART выполняется в режиме отладчика в симуляторе uVision в окне Serial.

**2. Программирования UART в режиме програмноуправляемого обмена.**

- Написать программу вывода символа и строки с использованием функций putchar, putstr.

```
//вывод символа на RS
void putchar(char c)
{
SBUF = c;
while(!TI);
TI=0;
}
```

```
//вывод массива на RS
void putstr(unsigned char mass[])
{
unsigned char i=0;
while (mass[i]!=00)
      {putchar(mass[i]);
       i++;
       }
}
```

**2. Вывод символьной строки по прерываниям**

- Задается выводимый символьный массив, например mass[ ]=”Hello”;
- В программе main() в качестве основного цикл программы можно использовать пустой цикл “while(1);”.
- До этого пустого цикла разрешаются прерывание по UART ES=1и глобальное прерывное EA=1.
- Там же для старта процесса вывода массива искусственно устанавливается в единицу флаг прерывания передатчика UART - TI=1, вследствие чего вызывается обработчик прерывания.
- В обработчике сбрасывается флаг готовности прерывания TI=0;
- Проверяется значение текущего элемента массива mass[index]. Если это значение не равно нулю-признаку окончания массива, то оно записывается в регистр SBUF для вывода на UART.

# ЛАБОРАТОРНАЯ РАБОТА 2

## ЖИДКОКРИСТАЛИЧЕСКИЕ ГРАФИЧЕСКИЕ ДИСПЛЕИ

WG12864, PG12864 с контроллером S6B0108 фирм Winstar, Powertip

1. **Целью работы** является написание подпрограмм вывода на дисплей графической и символьной информации в соответствии с интерфейсом дисплея, описанным в презентации.
2. Для вывода на дисплей требуются функции вывода байта в регистр команд и регистр данных
   - void wr_comand(unsigned char com)
   - void wr_dat(unsigned char dat)

   при этом нужно установить соответствующие значения сигналов
   - CS1, CS2-выбор кристалла 1, 2 контроллера для работы в координатах X=0...63, или X=64...127.
   - Сигнал выбора регистра DI
   - Сигнал RD_WR направления передачи
   - Байт команды (данных)
   - Сигнал строба разрешения записи / чтения E.
   - Снимается сигнал выбора кристалла CS1, CS2=1.
3. **Функция** вывода точки с произвольными координатами X,Y на дисплейvoid put_dot( unsigned char X, Y)

В соответствии с описанием для этого необходимо выполнить 3 операции:

1. Установка адреса по оси X. Для этого выводится команда 01XXXXXX-где 01 это код операции, 6 бит XXXXXX это координата точки по оси X в диапазоне 0...63

   wr_comand(0x40+X);
2. Для задания координаты по оси Y выполняются 2 действия:

Устанавливается страница экранной памяти — для этого выводится команда -10111ppp, где

10111-код операции

ppp=Nстраницы= Y/8.

Таким образом в регистр команд записывается значение 0xB8 +Y/8.

wr_comand(0xB8 +Y/8);

3.Третьим действием в регистр данных записывается байт в котором бит соответствующий подсвечиваемой точке=1, остальные биты=0.Номер подсвечиваемого бита Nbit = Y% 8. Значение байта с таким битом Dbyte=$2^{Nbit}$

Для того чтобы не подключать библиотеку математических функций языка СИ, занимающей в памяти процессора много места,

этот байт Dbyte удобно получить табличным способом Dbyte=tab_n_poz [ Nbit] ;

где массив tab_n_poz[8]={1,2,4,8,16,32,64,128};

Таким образом в регистр данных записывается значение

tab_n_poz [ y % 8 ] .

wr_dat(tab_n_poz [ y % 8 ] ) .

**Функция рисования горизонтальной линии с** координатами начала линии X, Y и длинной L void Line_H(unsigned char X, Y, L)

Поскольку после записи в регистр данных происходит автоматический автоинкремент адреса X, то для рисования горизонтальной линии после вывода точки в начале линии, достаточно повторить операцию вывода байта в регистр данных

wr_dat(tab_n_poz [ y % 8 ] )

в цикле L раз. (L- длинна линии).

**Функция вода символа на дисплей из знакогенератора**

write_sim(unsigned char X, Y, 'sim')

Функция должна включать в себя 3 действия:

1. Установку адреса по X начала символа,
2. Установку страницы
3. Вывод в цикле 5-байт изображения символа из знакогенератора

```
font[256][5]= {
{0x00,0x00,0x00,0x00,0x00}, // 00
……..
{0x3E,0x51,0x49,0x45,0x3E}, // ‘0’
{0x00,0x42,0x7F,0x40,0x00}, // ‘1’
{….....} };
```

При этом первый индекс массива определяется ASCCI кодом выводимого символа и выбирается строка из таблицы задающая изображение выводимого символа. Вторым индексом массива является индекс номера цикла for(i=0; i<5;i++)

Для проверки программы на симуляторе Keil можно сделать точки останова после вывода в регистр данных и сравнить значение на порту шины данных с требуемыми значениями.

В начале main программы необходимо командами #define выделить сигналам дисплея

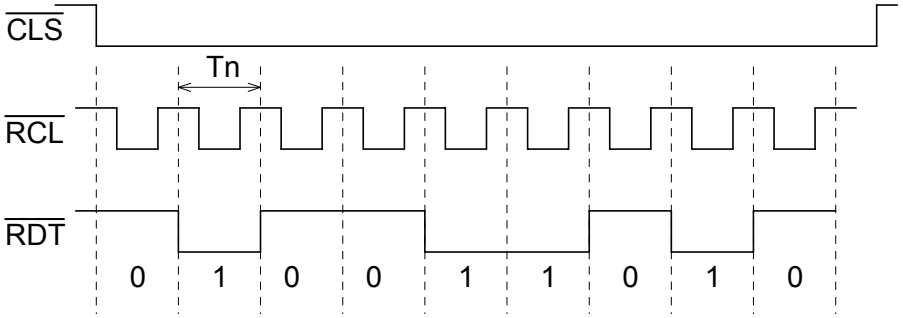
D0..7, D_I, R_WR, E, CS1, CS2/

какие либо разряды портов CPU.

# ЛАБОРАТОРНАЯ РАБОТА 3
# Считыватель магнитных карт на CPU

**Цель работы:** Создание программы чтения последовательного кода считывателя в буфер и вывода его на ЖКИ.

### Диаграммы работы считывателя




Сигнал присутствия карты в считывателе

Тактовый сигнал

Биты данных

Формат последовательных считываемых данных упакован следующим образом

| Nсим | | D0 | D1 | D2 | D3 | P |
|---|---|---|---|---|---|---|
| | Нули в начале 0…0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 старт символ 0xB (0x30+0xB=«;») | 1 | 0 | 1 | 1 | 1 |
| 2 . . . 38 | 37 символов ДАННЫХ<br>P1…P39-биты четности строк | | | | | |
| 39 | Стоп символ 0xF (0x30+0xF =‘?’) | 1 | 1 | 1 | 1 | 0 |
| 40 | биты паритета по столбцам | PC0 | PC1 | PC2 | PC3 | PP |
| | Нули в конце 0…0 | | | | | |

- В начале записи пишется какое то количество нулей
- Далее следует 1 (D0) стартового символа 1101
- В конце стартового символа также как и остальных символов следует бит паритета-сумма по модулю 2 предыдущих битов
- Далее следуют до 37 символов-чисел от 0 до 9
- Далее стоповый символ 1111=0x0F
- Последняя строка- биты паритета по столбцам

В первом варианте программы контроль бита паритета делать не нужно.

В начале программы сигналы CLS, RCL, RDT командой **#define** привязываем к разрядам порта процессора

Для упрощения алгоритма чтения данных удобно тактовый сигнал считывателя CLS подать на вход внешнего прерывания процессора, и в каждом цикле прерываний читать по одному биту, размещая его в соответствующем разряде соответствующего символа.

Поскольку в начале записи пишется какое то количество нулей, начинать преобразование последовательного кода нужно с первой единицы стартового символа. Для этого можно задать битовую переменную **start_on** и в самом начале обработчика присваивать ей значение считанного бита. Дальше нужно делать проверку этого бита, и пока он равен нулю, выходить из обработчика прерываний по команде **return**, не выполняя никаких действий.

Помимо переменной **start_on** необходимо задать переменные:

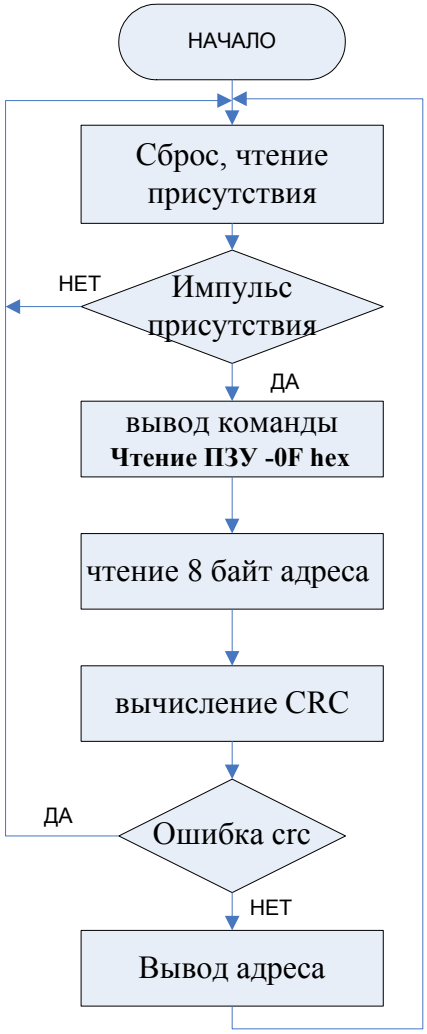
- **simbol_tmp**-байт в котором будет формироваться текущий символ
- **data_ array**[**simbol_nomber**] массив данных размерностью 37 байт
- **simbol_nomber-**номер текущего символа
- **bit_nomber-**номер текущего бита в символе
- **weight –** вес текущего разряда символа в соответствии с текущим номером бита в символе 1, 2, 4, 8. Преобразование последовательного кода в параллельный удобно вести складывая переменную **simbol_tmp с weight,** если считанный бит равен единице. При чтении нулевого бита **weight** устанавливается в единицу, в конце обработчика прерывания умножается на 2.Здесь же увеличивается на единицу **bit_nomber-**номер текущего бита.
- При достижении **bit_nomber=4** переменные **simbol_tmp, bit_nomber** обнуляются, **weight** устанавливается в 1, а преобразованное 4 битное число записывается в массив данных **data_ array**[**simbol_nomber**]
- При завершении формирования каждого слова (**bit_nomber=4)** оно проверяется на совпадение со стоповым символом 1111=0x0F. Если оно есть, то удобно установить битовую переменную- флаг окончания преобразования **finish=1.**
- В основной программе **main** необходимо в бесконечном цикле проверять этот флаг, и при его установке запретить прерывание CPU от считывателя и вывести массив на UART или дисплей.

Второй вариант программы с проверкой паритета по строкам нужно сделать на основании описанного алгоритма. Для этого:

- удобно ввести битовую переменную **h_parity** первоначально равную 0, и при чтении 0…3 бита суммировать его по модулю 2 с прочитанным битом.
- При **bit_nomber=4,** полученную контрольную сумму сравнить с битом паритета, прочитанным с карточки.
- Если эти значения не совпадают, устанавливается битовая переменная **error=1**, которая в бесконечном цикле в основной программе **main** проверяется, и при её установке выводится сообщение об ошибке.

Для сброса переменных перед очередным циклом чтения карточки удобно использовать факт перехода сигнала **CLS** из 1 в 0. Это также надо делать в программе **main.**

# ЛАБОРАТОРНАЯ РАБОТА 4




Алгоритм чтения электронной таблетки

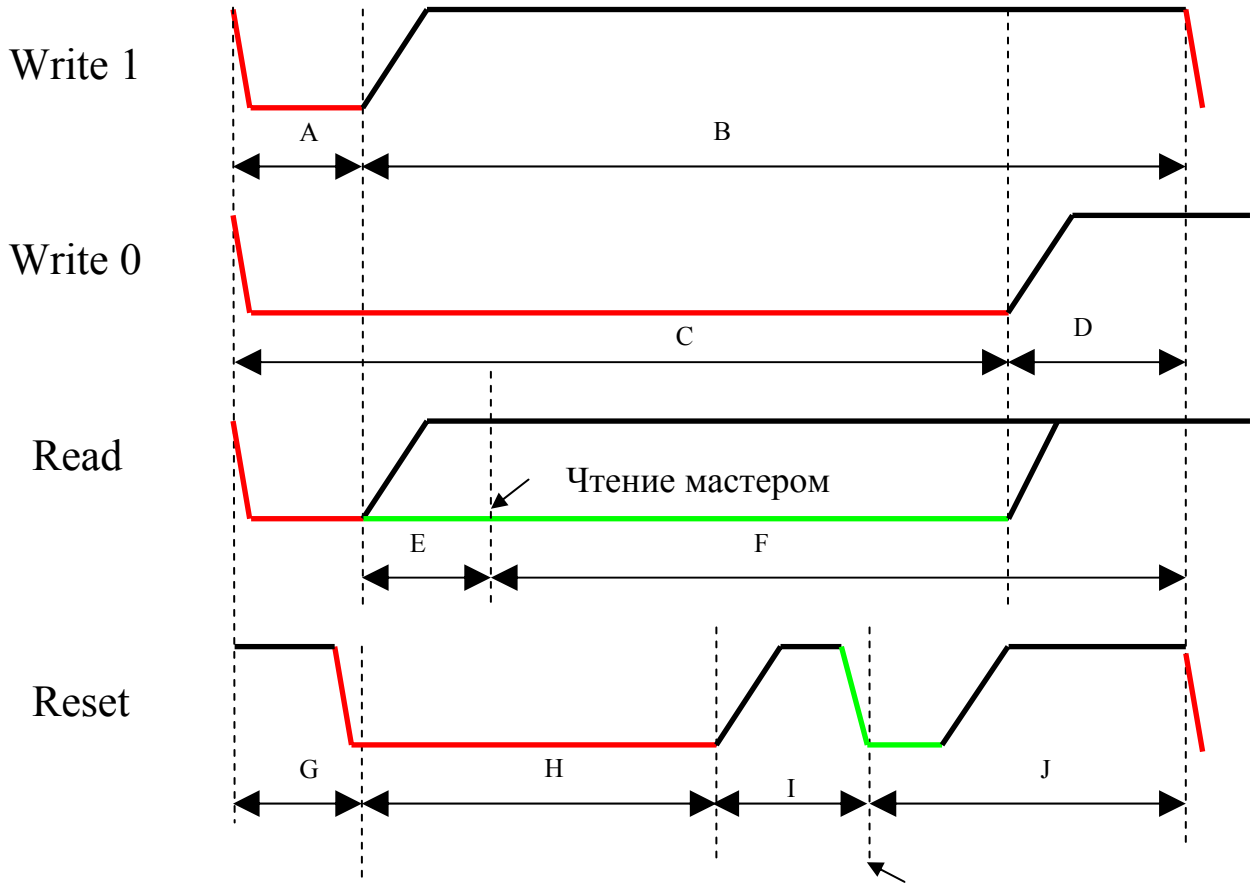
## Сетевой интерфейс 1-Wire MicroLAN.
## Алгоритм и программа чтения электронных таблеток iButton

**Целью работы** является написание и отладка программы чтения адреса электронной таблетки iButton DS1990.

В соответствии с описанием 1-проводного интерфейса в презентации алгоритм чтения приведен на рисунке.

- Вывод денных –8 байт начиная с младшего бита младшего байта.
- Формат данных при чтении с таблетки- 01, AD0,AD1...AD5,CRC
- Первый байт 01-номер серии прибора .AD0..5-разряды номера таблетки.
- На дисплей выводить в HEX формате 7....0 байт
- Алгоритм вычисления CRC
- crc=0;
- for(i=0; i< 7;i++) {crc=TAB_CRC[crc^dat[i]];}
- Таблица TAB_CRC[] записывается в память программ CPU и приведена ниже.
- Временные диаграммы для реализации операций протокола обмена приведены на рисунке.
- Значения временных интервалов (в микросекундах) для стандартной скорости обмена и высокой скорости приведены в таблице (колонки STD и HI)
- Алгоритмы для реализации функций записи, чтения и обнаружения присутствия приведены на рисунках.

Временные диаграммы для реализации операций протокола обмена




Значения временных интервалов

| Пар. | T µs | |
|---|---|---|
| | STD | HI |
| A | 6 | 1.5 |
| B | 64 | 7.5 |
| C | 60 | 7.5 |
| D | 10 | 2.5 |
| E | 9 | 0.75 |
| F | 55 | 7 |
| G | 0 | 2.5 |
| H | 480 | 70 |
| I | 70 | 8.5 |
| J | 410 | 40 |

Алгоритмы реализации функций записи, чтения и обнаружения присутствия.

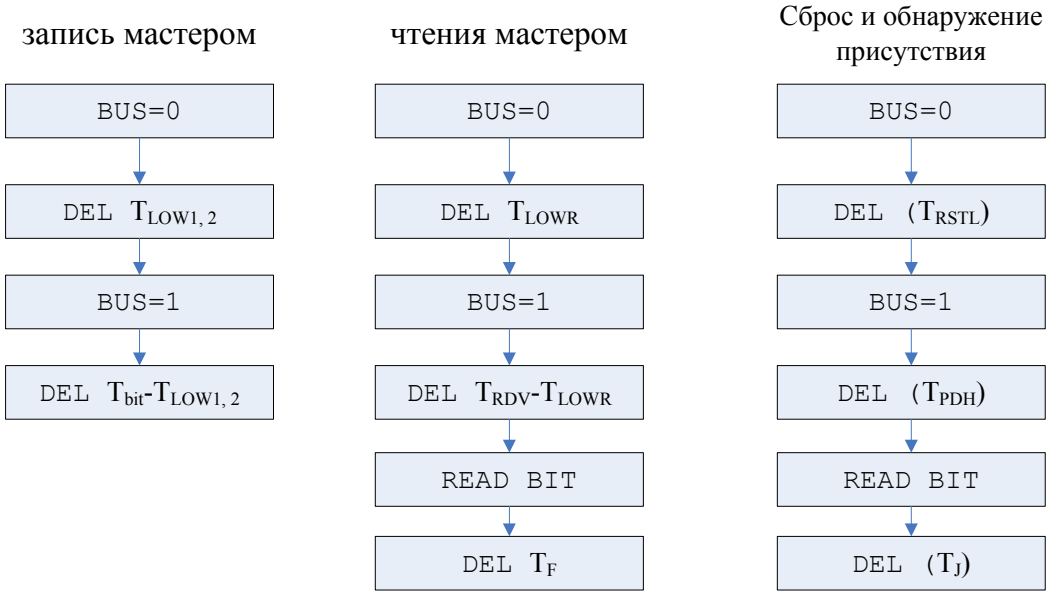



Таблица TAB_CRC для вычисления контрольной суммы. //PDF «1-WIRE BOOK p131»

```
unsigned char code    TAB_CRC[]={
  0,  94, 188, 226,  97,  63, 221, 131, 194, 156, 126,  32, 163, 253,  31,  65,
157, 195,  33, 127, 252, 162,  64,  30,  95,   1, 227, 189,  62,  96, 130, 220,
 35, 125, 159, 193,  66,  28, 254, 160, 225, 191,  93,   3, 128, 222,  60,  98,
190, 224,   2,  92, 223, 129,  99,  61, 124,  34, 192, 158,  29,  67, 161, 255,
 70,  24, 250, 164,  39, 121, 155, 197, 132, 218,  56, 102, 229, 187,  89,   7,
219, 133, 103,  57, 186, 228,   6,  88,  25,  71, 165, 251, 120,  38, 196, 154,
101,  59, 217, 135,   4,  90, 184, 230, 167, 249,  27,  69, 198, 152, 122,  36,
248, 166,  68,  26, 153, 199,  37, 123,  58, 100, 134, 216,  91,   5, 231, 185,
140, 210,  48, 110, 237, 179,  81,  15,  78,  16, 242, 172,  47, 113, 147, 205,
 17,  79, 173, 243, 112,  46, 204, 146, 211, 141,  11,  49, 178, 236,  14,  80,
175, 241,  19,  77, 206, 144, 114,  44, 109,  51, 209, 143,  12,  82, 176, 238,
 50, 108, 142, 208,  83,  13, 239, 177, 240, 174,  76,  18, 145, 207,  45, 115,
202, 148, 118,  40, 171, 245,  23, 073,   8,  86, 180, 234, 105,  55, 213, 139,
 87,   9, 235, 181,  54, 104, 138, 212, 149, 203,  41, 119, 244, 170,  72,  22,
233, 183,  85,  11, 136, 214,  52, 106,  43, 117, 151, 201,  74,  20, 246, 168,
116,  42, 200, 150,  21,  75, 169, 247, 182, 232,  10,  84, 215, 137, 107,  53};
```

# ЛАБОРАТОРНАЯ РАБОТА 5
## Интеловский алгоритм клавиатуры

**Цель работы:** Создание алгоритма и программы работы клавиатуры от фирмы Intel.

Схема подключения матрицы клавиатуры приведена на рисунке 1. Для определения номера активной клавиши Nkey (0…15), необходимо определить номер активной строки Nrow (1…4) и номер активной колонки Ncol (0…3).

$$Nkey = Ncol + (Nrow-1)*4$$

ASCCI код нажатой клавиши Cod_key вычисляется по таблице

$$Cod\_key = Tab\_key[Nkey]$$

где: Tab_key[Nkey]-таблица кодировки, для телефонной клавиатуры Рис.2,

Tab_key ={‘1’,’2’,’3’,’A’,’4’,’5’,’6’,’B’,’7’,’8’,’9’,’C’,’*’,’0’,’#’,’D’}

**Рис. 1**

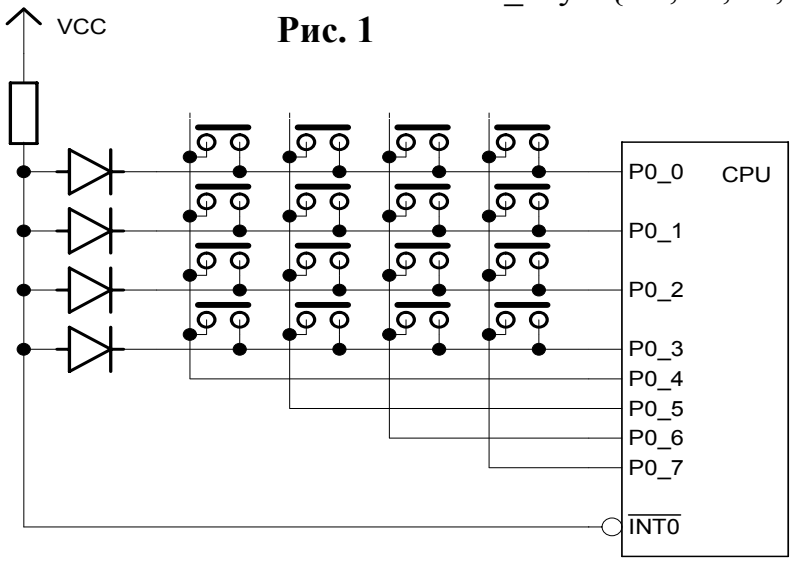
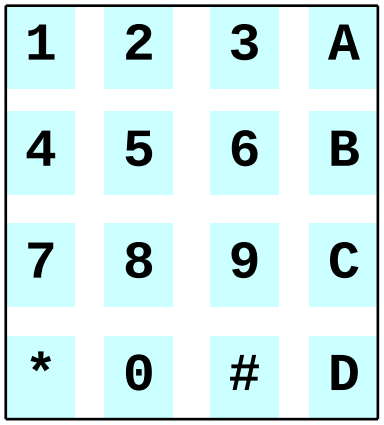



**Рис. 2**



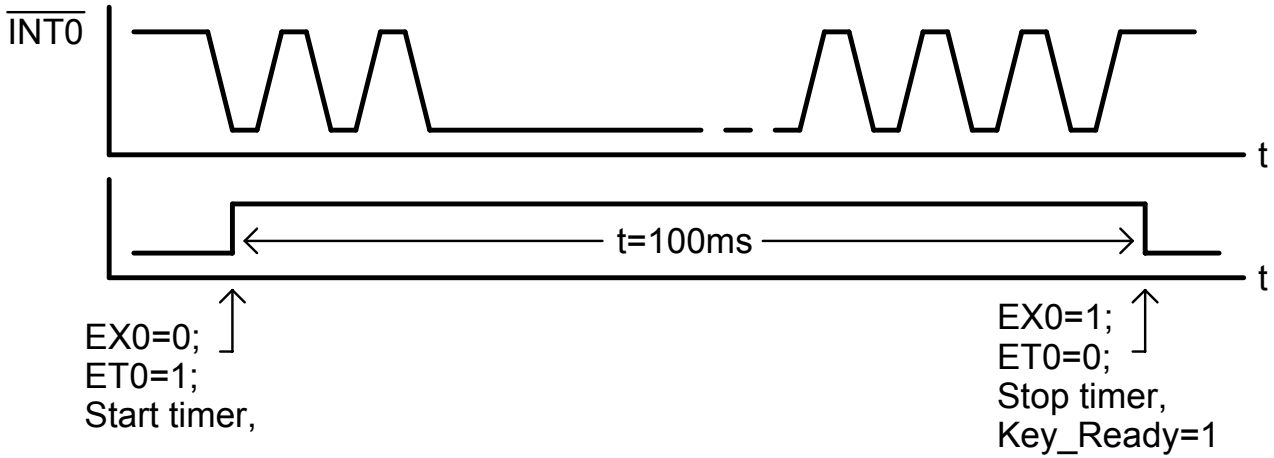

Алгоритм работы клавиатуры заключается в следующем:

- В программе main(), для инициализации работы, строки матрицы клавиатуры устанавливаются в 1, колонки – в 0

  P0=0x0F;
- Разрешается прерывание по входу INT0- EX0=1, разрешается глобальное прерывание EA=1;
- При нажатии на любую клавишу ноль с колонок подключается к строке, переводя ее в нулевое состояние.
- На выходе диодной схемы И, на входе INT0 CPU вырабатывается нулевой сигнал прерывания, вызывается обработчик прерывания.

- В обработчике прерывания по нулевому сигналу на одном из разрядов порта P0_0…P0_3 определяется номер активной строки.
- Далее в этом же цикле прерывания производится переинициализация порта P0. На строки матрицы клавиатуры подается 0, на колонки – 1.
- Нулевой сигнал со строк, через замкнутую клавишу попадает на столбец, обнуляя ее.
- По нулевому сигналу на одном из разрядов порта P0_4…P0_7 определяется номер активной колонки.
- По номеру строки и колонки вычисляется номер нажатой клавиши, и ее ASCCI код. Вводится битовая переменная Key_Ready=1, которая будет проверяться в основном цикле программы main() и выполняться соответствующие действия.
- Для устранения дребезга контактов Рис.3, необходимо использовать какой либо таймер CPU в режиме 16-и битного таймера счетчика, с прерыванием.
- При вхождении в прерывание по входу INT0
  - разрешается прерывание по таймеру
  - запускается таймер на 100 ms.
  - запрещаются прерывания по входу INT0 (EX0=0;),

**Рис.3**




Задержка в 100 ms определяется максимально возможной скоростью работы на клавиатуре ( не более 10 символов в секунду).

- По окончании времени 100 ms, таймер 0 входит в прерывание. В обработчике прерываний T0 производится
  - Стоп таймера,
  - EX0=1 - разрешение прерывания по клавиатуре
  - ET0=0 - запрет прерывания по таймеру
  - Key_Ready=1 – установка флага готовности клавиатуры.