**6 ЛАБОРАТОРНАЯ РАБОТА №6**

**МАТРИЧНАЯ НЕКОДИРУЮЩАЯ КЛАВИАТУРА**

**Цель работы**

Ознакомиться с функциональными возможностями и внутренней структурой отладочного стенда EV8031/AVR. Изучить внутреннюю организацию матричной некодирующей клавиатуры, научиться формировать коды нажатых клавиш и выполнять соответствующую обработку нажатий.

**6.1 Краткие теоретические сведения**

Клавишные или кнопочные устройства ввода информации в настоящее время являются самыми распространенными устройствами в компьютерах и цифровых устройствах. Клавиатуры компьютеров и мобильных телефонов, кнопочные поля телефонов, калькуляторов, пультов дистанционных устройств, панели управления в измерительных и бытовых приборах – далеко не полный перечень примеров использования клавиатур.

Различают два типа клавиатур: кодирующие и некодирующие. В клавиатурах первого типа при нажатии на клавишу схема контроллера клавиатуры формирует код, соответствующий этой клавише. К данному типу принадлежат компьютерные клавиатуры, в частности IBM PC AT. В цифровых устройствах наиболее часто используют простые и дешевые некодирующие клавиатуры, которые обычно представляют собой сканируемую матрицу кнопок.

Рассмотрим организацию матричной клавиатуры размерностью 5х3. Горизонтальные линии Х0 ... Х4 (ряды или строки) являются выходными, а вертикальные У0 ... У2 (колонки) являются входными (смотри рисунок 6.1). В неактивном состоянии на выходах Х0 ... Х4 присутствует уровень логической единицы, а входы У0 ... У2 обычно подключены через резисторы к питанию. Для этой цели, как правило, используются встроенные резисторы, которые имеют некоторые порты микроконтроллеров. При сканировании на горизонтальных линиях матрицы формируется сигнал «бегущего» нуля. Для каждой фазы «бегущего» нуля происходит считывание У0 ... У2. При замыкании какой-либо из клавиш матрицы на соответствующем выходе устанавливается нулевое значение. По результатам анализа кодов Х0 ... Х4 и У0 ... У2 происходит формирование кода нажатой клавиши.

В учебно-отладочном стенде EV8031/AVR матричная клавиатура реализована в виде 12 тактовых кнопок, сгруппированных в 3 столбца по 4 строки. Управление матричной клавиатурой осуществляется по линиям A0 ... A2 и KL0 ... KL3. Линии А0, А1, А2 – выводы столбцов 1, 2 и 3 соответственно (при счёте слева направо). Для того чтобы сформировать сигнал «бегущего» нуля для этих столбцов, необходимо по очереди обращаться для чтения к системному контроллеру учебно-отладочного стенда (т.е., к внешней памяти) по адресу 0x9006, 0x9005 и 0x9003 соответственно. Истинность данного утверждения проиллюстрирована ниже, на рисунке 6.2.

Поскольку столбцов матричной клавиатуры 3, то последние 3 бита каждого из адресов внешней памяти отвечают за состояние линий A2, A1 и A0, причём, большему номеру линии соответствует бит с большим номером. Именно поэтому адреса сформированы так, чтобы они содержали 0 только в одном из разрядов, т.е.: 0x6 = 0b0**110**, 0x5 = 0b0**101**, 0x3 = 0b0**011**. Таким образом, чтение содержимого внешней памяти по любому из указанных адресов позволяет получить байт данных, младшая тетрада которого содержит информацию о нажатии клавиш (состояние линий KL3 ... KL0). Если некоторый бит в указанной тетраде равен 0, то это означает, что соответствующая клавиша нажата.

**6.2 Порядок выполнения работы**

6.2.1 Запустить IDE AVR Studio 4.

6.2.2 Создать новый проект в IDE AVR Studio 4.

6.2.3 В появившемся окне написать программу на языке С или ассемблер с учётом варианта задания, который указан в таблице 6.1.

|  |  |
| --- | --- |
| № | Описание задания |
| 10 | Реализовать операцию умножения двух чисел. Вначале нажимается любая цифровая клавиша, номер которой будет первым числом. После чего нажимается вторая клавиша, номер которой будет вторым числом. По нажатию на клавишу “#->” осуществляется умножение операндов, а по нажатию “<-\*” – вывод результата. В процессе умножения отображать двоичный код каждого из операндов, а также результата на светодиодном индикаторе. |

6.2.4 Произвести компиляцию проекта.

6.2.5 При наличие сообщений об ошибках или предупреждениях вернуться к предыдущему пункту и внести необходимые исправления. В случае некорректной работы программы выполнить её отладку средствами меню Debug.

6.2.6 Проверить подключение USB-кабеля программатора к одноимённому разъёму системного блока.

6.2.7 Загрузить исполняемый файл проекта в микроконтроллер.

6.2.8 Визуально оценить правильность работы жидкокристаллического индикатора.

**6.3 Краткие теоретические сведения**

#define F\_CPU 7372800L //?????? ??????? ?????? (7,3728 ???)

#include <avr/io.h>

#include <avr/iom8515.h>

#include <util/delay.h>

#define keyboard\_col1\_scan 0x9006

#define keyboard\_col2\_scan 0x9005

#define keyboard\_col3\_scan 0x9003

#define stat\_7seg\_left 0xA000

#define stat\_7seg\_right 0xB000

#define stat\_7seg\_control 0xA004

#define LED\_LINE 0xA006

typedef unsigned char uchar;

volatile uchar op1 = 0;

volatile uchar op2 = 0;

volatile uchar \*left = (uchar\*)stat\_7seg\_left;

volatile uchar \*right = (uchar\*)stat\_7seg\_right;

volatile uchar \*sseg\_ctl = (uchar\*)stat\_7seg\_control;

volatile uchar \*led\_line = (uchar\*)LED\_LINE;

void set\_operands(uchar value){

op1 = op2;

op2 = value;

\*sseg\_ctl = 0b00000101;

\*left = op1;

\*right = op2;

\*leg\_line = op1 << 4 | op2;

}

int main(void) {

MCUCR= 1<<SRE | 1<<SRW10;

EMCUCR= 1<<SRW11;

ACSR= 1<<ACD;

volatile uchar temp;

volatile uchar \*control;

volatile uchar result = 0;

\*left= 0x00;

\*right= 0x00;

while(1) { //???? ???????????? ??????????

control= (unsigned char\*) keyboard\_col1\_scan;

temp= \*control;

temp= \*control;

temp= \*control;

temp= \*control;

if((temp & 0x01)== 0x00) { // (?.?. ?????? "1")

set\_operands(1);

} else if((temp & 0x02)== 0x00) { // (?.?. ?????? "4")

set\_operands(4);

} else if((temp & 0x04)== 0x00) { // (?.?. ?????? "7")

set\_operands(7);

} else if((temp & 0x08)== 0x00) { // (?.?. ?????? "<-\*")

\*sseg\_ctl = 0b00000011;

\*right = result / 10 << 4 | result % 10;

\*led\_line = result;

}

control= (unsigned char\*) keyboard\_col2\_scan;

temp= \*control;

temp= \*control;

temp= \*control;

temp= \*control;

if((temp & 0x01)== 0x00) { //(?.?. ?????? "2")

set\_operands(2);

} else if((temp & 0x02)== 0x00) { //(?.?. ?????? "5")

set\_operands(5);

} else if((temp & 0x04)== 0x00) { //(?.?. ?????? "8")

set\_operands(8);

} else if((temp & 0x08)== 0x00) { // (?.?. ?????? "0")

set\_operands(0);

}

control= (unsigned char\*) keyboard\_col3\_scan;

temp= \*control;

temp= \*control;

temp= \*control;

temp= \*control;

if((temp & 0x01)== 0x00) { (?.?. ?????? "3")

set\_operands(3);

} else if((temp & 0x02)== 0x00) { (?.?. ?????? "6")

set\_operands(6);

} else if((temp & 0x04)== 0x00) { (?.?. ?????? "9")

set\_operands(9);

} else if((temp & 0x08)== 0x00) { // (?.?. ?????? "#>")

result = op1\*op2;

}

\_delay\_ms(400);

}

return 0;

}

**6.4 Особенности IDE AVRStudio выявленные в ходе выполнения лабораторной работы**

В ходе лабораторной работы никаких новых особенностей IDE AVRStudio не было выявлено.

**Выводы**

В ходе лабораторной работы были изучены принципы получения информации с матричной некодирующей клавиатуры в учебно-отладочном стенде EV8031/AVR. Была написана программа, которая производит умножение чисел введенных с клавиатуры. Преимуществом некодирующей матричной клавиатуры является простота реализации. Недостатком является необходимость преобразования скан кода к значения клавиши.