/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

This program was created by the CodeWizardAVR V3.48b

Automatic Program Generator

© Copyright 1998-2022 Pavel Haiduc, HP InfoTech S.R.L.

http://www.hpinfotech.ro

Project :

Version :

Date : 13/05/2022

Author :

Company :

Comments:

Chip type : ATmega8535

Program type : Application

AVR Core Clock frequency: 1.000000 MHz

Memory model : Small

External RAM size : 0

Data Stack size : 128

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <mega8535.h>

#include <delay.h>

// Declare your global variables here

char modo = 4;

char columnas = 0x01;

char filas = 0;

char indice = 0, numero = 0;

char repetir = 0, renglon = 0;

int cambio\_caso = 0;

bit puede\_cambiar = 0;

char modoCero[] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

char modoUno[7][5] = {

{0x01, 0x01, 0x01, 0x01, 0x01},

{0x02, 0x02, 0x02, 0x02, 0x02},

{0x04, 0x04, 0x04, 0x04, 0x04},

{0x08, 0x08, 0x08, 0x08, 0x08},

{0x10, 0x10, 0x10, 0x10, 0x10},

{0x20, 0x20, 0x20, 0x20, 0x20},

{0x40, 0x40, 0x40, 0x40, 0x40}

};

char modoCuatro[10][5] = {

{0x41, 0x2E, 0x36, 0x3A, 0x41},

{0x3F, 0x3D, 0x00, 0x3F, 0x3F},

{0x3D, 0x1E, 0x2E, 0x36, 0x39},

{0x5D, 0x3E, 0x36, 0x36, 0x49},

{0x67, 0x6B, 0x6D, 0x00, 0x7F},

{0x58, 0x3A, 0x3A, 0x3A, 0x46},

{0x43, 0x35, 0x36, 0x36, 0x4F},

{0x7C, 0x7E, 0x0E, 0x76, 0x78},

{0x49, 0x36, 0x36, 0x36, 0x49},

{0x79, 0x36, 0x36, 0x56, 0x69}

};

char modoCinco[29][5] = {

{0x7f, 0x7f, 0x7f, 0x7f, 0x7f}, // -

{0x01, 0x76, 0x76, 0x76, 0x01}, // A

{0x00, 0x3f, 0x3f, 0x3f, 0x3f}, // L

{0x01, 0x76, 0x76, 0x76, 0x01}, // A

{0x00, 0x7b, 0x77, 0x6f, 0x00}, // N

{0x7f, 0x7f, 0x7f, 0x7f, 0x7f}, // -

{0x00, 0x7d, 0x7b, 0x7d, 0x00}, // M

{0x01, 0x76, 0x76, 0x76, 0x01}, // A

{0x00, 0x3f, 0x3f, 0x3f, 0x3f}, // L

{0x01, 0x76, 0x76, 0x76, 0x01}, // A

{0x41, 0x3e, 0x36, 0x36, 0x45}, // G

{0x41, 0x3e, 0x3e, 0x3e, 0x41}, // O

{0x00, 0x7b, 0x77, 0x6f, 0x00}, // N

{0x7f, 0x7f, 0x7f, 0x7f, 0x7f}, // -

{0x5f, 0x3f, 0x3f, 0x3f, 0x40}, // J

{0x41, 0x3e, 0x3e, 0x3e, 0x41}, // O

{0x00, 0x76, 0x66, 0x56, 0x39}, // R

{0x41, 0x3e, 0x36, 0x36, 0x45}, // G

{0x00, 0x36, 0x36, 0x36, 0x3e}, // E

{0x7f, 0x7f, 0x7f, 0x7f, 0x7f}, // -

{0x00, 0x7d, 0x7b, 0x7d, 0x00}, // M

{0x01, 0x76, 0x76, 0x76, 0x01}, // A

{0x00, 0x76, 0x66, 0x56, 0x39}, // R

{0x7e, 0x7e, 0x00, 0x7e, 0x7e}, // T

{0x7f, 0x3e, 0x00, 0x3e, 0x7f}, // I

{0x00, 0x7b, 0x77, 0x6f, 0x00}, // N

{0x00, 0x36, 0x36, 0x36, 0x3e}, // E

{0x1e, 0x2e, 0x36, 0x3a, 0x3c}, // Z

{0x7f, 0x7f, 0x7f, 0x7f, 0x7f} // -

};

void cambiar\_modo(){

cambio\_caso = 0;

indice = 0;

renglon = 0;

numero = 0;

repetir = 0;

columnas = 0x01;

modo++;

puede\_cambiar = 0;

if (modo == 6) modo = 0;

}

void main(void)

{

// Declare your local variables here

// Input/Output Ports initialization

// Port A initialization

// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In

DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1) | (0<<DDA0);

// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P

PORTA=(1<<PORTA7) | (1<<PORTA6) | (1<<PORTA5) | (1<<PORTA4) | (1<<PORTA3) | (1<<PORTA2) | (1<<PORTA1) | (1<<PORTA0);

// Port B initialization

// Function: Bit7=In Bit6=In Bit5=In Bit4=In Bit3=In Bit2=In Bit1=In Bit0=In

DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0);

// State: Bit7=P Bit6=P Bit5=P Bit4=P Bit3=P Bit2=P Bit1=P Bit0=P

PORTB=(1<<PORTB7) | (1<<PORTB6) | (1<<PORTB5) | (1<<PORTB4) | (1<<PORTB3) | (1<<PORTB2) | (1<<PORTB1) | (1<<PORTB0);

// Port C initialization

// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out

DDRC=(1<<DDC7) | (1<<DDC6) | (1<<DDC5) | (1<<DDC4) | (1<<DDC3) | (1<<DDC2) | (1<<DDC1) | (1<<DDC0);

// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0

PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);

// Port D initialization

// Function: Bit7=Out Bit6=Out Bit5=Out Bit4=Out Bit3=Out Bit2=Out Bit1=Out Bit0=Out

DDRD=(1<<DDD7) | (1<<DDD6) | (1<<DDD5) | (1<<DDD4) | (1<<DDD3) | (1<<DDD2) | (1<<DDD1) | (1<<DDD0);

// State: Bit7=0 Bit6=0 Bit5=0 Bit4=0 Bit3=0 Bit2=0 Bit1=0 Bit0=0

PORTD=(0<<PORTD7) | (0<<PORTD6) | (0<<PORTD5) | (0<<PORTD4) | (0<<PORTD3) | (0<<PORTD2) | (0<<PORTD1) | (0<<PORTD0);

// Timer/Counter 0 initialization

// Clock source: System Clock

// Clock value: Timer 0 Stopped

// Mode: Normal top=0xFF

// OC0 output: Disconnected

TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (0<<CS01) | (0<<CS00);

TCNT0=0x00;

OCR0=0x00;

// Timer/Counter 1 initialization

// Clock source: System Clock

// Clock value: Timer1 Stopped

// Mode: Normal top=0xFFFF

// OC1A output: Disconnected

// OC1B output: Disconnected

// Noise Canceler: Off

// Input Capture on Falling Edge

// Timer1 Overflow Interrupt: Off

// Input Capture Interrupt: Off

// Compare A Match Interrupt: Off

// Compare B Match Interrupt: Off

TCCR1A=(0<<COM1A1) | (0<<COM1A0) | (0<<COM1B1) | (0<<COM1B0) | (0<<WGM11) | (0<<WGM10);

TCCR1B=(0<<ICNC1) | (0<<ICES1) | (0<<WGM13) | (0<<WGM12) | (0<<CS12) | (0<<CS11) | (0<<CS10);

TCNT1H=0x00;

TCNT1L=0x00;

ICR1H=0x00;

ICR1L=0x00;

OCR1AH=0x00;

OCR1AL=0x00;

OCR1BH=0x00;

OCR1BL=0x00;

// Timer/Counter 2 initialization

// Clock source: System Clock

// Clock value: Timer2 Stopped

// Mode: Normal top=0xFF

// OC2 output: Disconnected

ASSR=0<<AS2;

TCCR2=(0<<WGM20) | (0<<COM21) | (0<<COM20) | (0<<WGM21) | (0<<CS22) | (0<<CS21) | (0<<CS20);

TCNT2=0x00;

OCR2=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization

TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (0<<TOIE0);

// External Interrupt(s) initialization

// INT0: Off

// INT1: Off

// INT2: Off

MCUCR=(0<<ISC11) | (0<<ISC10) | (0<<ISC01) | (0<<ISC00);

MCUCSR=(0<<ISC2);

// USART initialization

// USART disabled

UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (0<<RXEN) | (0<<TXEN) | (0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);

// Analog Comparator initialization

// Analog Comparator: Off

// The Analog Comparator's positive input is

// connected to the AIN0 pin

// The Analog Comparator's negative input is

// connected to the AIN1 pin

ACSR=(1<<ACD) | (0<<ACBG) | (0<<ACO) | (0<<ACI) | (0<<ACIE) | (0<<ACIC) | (0<<ACIS1) | (0<<ACIS0);

SFIOR=(0<<ACME);

// ADC initialization

// ADC disabled

ADCSRA=(0<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (0<<ADPS2) | (0<<ADPS1) | (0<<ADPS0);

// SPI initialization

// SPI disabled

SPCR=(0<<SPIE) | (0<<SPE) | (0<<DORD) | (0<<MSTR) | (0<<CPOL) | (0<<CPHA) | (0<<SPR1) | (0<<SPR0);

// TWI initialization

// TWI disabled

TWCR=(0<<TWEA) | (0<<TWSTA) | (0<<TWSTO) | (0<<TWEN) | (0<<TWIE);

while (1)

{

int i,j;

char temp;

switch (modo){

case 0:

filas = ~modoCero[indice];

delay\_ms(100);

break;

case 1:

filas = ~modoUno[renglon][indice];

repetir++;

if (repetir == 5){

renglon++;

repetir = 0;

}

if (renglon == 7) renglon = 0;

break;

case 2:

filas = ~modoUno[renglon][indice];

renglon++;

if (renglon == 7) renglon = 0;

break;

case 3:

filas = ~modoUno[renglon][indice];

delay\_ms(100);

repetir++;

if (repetir == 5){

renglon++;

repetir = 0;

}

if (renglon == 7) renglon = 0;

break;

case 4:

filas = modoCinco[numero][indice];

repetir++;

if (repetir == 60){

repetir = 0;

numero++;

}

break;

default:

filas = modoCuatro[numero][indice];

repetir++;

if (repetir == 60){

repetir = 0;

numero++;

}

if (numero == 10) {

puede\_cambiar = 1;

numero = 0;

}

}

// Contador de anillo

switch (columnas){

case 0x01:

columnas = 0x02;

break;

case 0x02:

columnas = 0x04;

break;

case 0x04:

columnas = 0x08;

break;

case 0x08:

columnas = 0x10;

break;

default:

columnas = 0x01;

}

// Indice

indice++;

if (indice == 5) {

indice = 0;

cambio\_caso++;

}

switch (modo){

case 0:

if (cambio\_caso == 4) cambiar\_modo();

break;

case 1:

if (cambio\_caso == 50) cambiar\_modo();

break;

case 2:

if (cambio\_caso == 50) cambiar\_modo();

break;

case 3:

if (cambio\_caso == 5) cambiar\_modo();

break;

case 4:

if (cambio\_caso == 340)cambiar\_modo();

break;

default:

if (puede\_cambiar) cambiar\_modo();

}

PORTC = columnas;

PORTD = filas;

delay\_ms(10);

}

}