Proposal Project

SMART CIRCUIT BASED ON
AT89S51-52

INTRODUCED BY

Dr. Assem Badr
The objectives

1. Design and implementing smart circuit based on AT89S51 with In-System Programming.
2. Recognize the Serial Peripheral Interface port (SPI) and its connection.
3. Recognize the in-circuit burner or In-System Programming (ISP).

The PIN assigned for the In-System-Programming (ISP) of AT89S51
The AT89S51 is a low-power, high-performance CMOS 8-bit microcontroller with 4K bytes of in-system programmable Flash memory. The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with in-system programmable Flash on a monolithic chip, the Atmel AT89S51 is a powerful microcontroller which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89S51 provides the following standard features: 4K bytes of Flash, 128 bytes of RAM, 32 I/O lines, Watchdog timer, two data pointers, two 16-bit timer/counters, a five-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89S51 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning.
The workflow for the hardware interfaces (hardware connections)

- Computer
- Arduino UNO-kit
  - 6 wires for SPI (GND, 5V, MISO, MOSI, SCK, RST)
- AT89S51-52
  - Ports connections (P0, P1, P2, P3)
- Hardware input/output
  - (Such as sensors, keypad, motors, 7-seg)
The interfacing between the Arduino-UNO and the microcontroller AT89S51 via the SPI port
Required electrical components and elements for the basic circuit

1) Microcontroller AT89S51 or AT89S52
2) Microcontroller socket (40 pins) or ZIF (zero insert force) socket.
3) Crystal element (from 8 MHz to 12 MHz)
4) 2 none polarize capacitors (from 30 PF to 80 PF – low voltage).
5) Polarize capacitor (from 10 µF to 15 µF – 16 to 35 V).
6) Resistance (from 10 KΩ to 15 KΩ – low watt).
7) Resistance (from 1 KΩ to 1.5 KΩ – low watt).
8) Diode (≤ 1 A).
9) Small Push button.
10) Printed Circuit Board (PCB) (recommend) or Pre-board (not Breadboard).
11) Regulator 7805 and battery 9V and its connector (recommend).

Required components and elements for the ISP (in-system-programming)

a) Arduino Uno kit
b) USB - Cable (model A to B)
c) 6 wires- connectors between UNO and AT89S51
   1. To connect the ground (GND) pin.
   2. To connect the bias from UNO (5V).
   3. To connect the master input slave output (MISO) pin.
   4. To connect the master output slave input (MOSI) pin.
   5. To connect the serial clock (SCK) pin.
   6. To connect the (RST) pin.

Required components and elements for the input/output

According to your selected laboratory experiment or other idea(s) that depend on them, choose your components such as stepper motor, keypad, sensor, LEDs, DIP-switches, LCD and so on.
**Median level experiments**

- Toggling lighting on 2 ports with counter (lecture 5)
- MCS-51 Interfacing With 4X4 Keypad

**High level experiments**

- Serial Transceiver Based On MCS-51

**Extra level**

A Great idea(s); such as AT89S51 interfacing with a communication module through SPI.
The workflow for the software interfaces (software interfacing)
/**
PROGRAMMING AN ATMEL AT89S51/52 USING ARDUINO
RELEASED AS IS WITHOUT WARRANTY
I AM NOT LIABLE FOR ANY DAMAGE DONE TO YOUR HARDWARE
THIS PROJECT IS FOR EDUCATIONAL PURPOSES ONLY
Credits to NICK PABLO for the Arduino Sketch
TIKTAK (C) 2014
**/

#define dummyData 0xAA
#define FDIY 75
#define NRDY 76

const int _MISO = 4;
const int _MOSI = 5;
const int _CLK = 3;
const int RST = 2;

/** Variable definition block */

byte data;
byte AL,AH; // 16-bit address
byte lockByte; // embed lock bits here
byte SigH,SigL; // Signature Bytes

Arduino Uno on COM9
/**
PROGRAMMING AN ATMEL AT89S51/52 USING ARDUINO
RELEASED AS IS WITHOUT WARRANTY
I AM NOT LIABLE FOR ANY DAMAGE DONE TO YOUR HARDWARE
THIS PROJECT IS FOR EDUCATIONAL PURPOSES ONLY
Credits to NICK PABLO for the Arduino Sketch
TIKTAK (C) 2014  **/

#define dummyData 0xAA
#define RDY 75
#define NRDY 76

const int _MISO = 4;
const int _MOSI = 5;
const int _CLK = 3;
const int RST = 2;

/* Variable definition block */

byte data;
byte AL,AH;  // 16-bit address
byte lockByte;  // embed lock bits here
byte SigH,SigL;  // Signature Bytes

void setup()
{
  pinMode(_MISO, INPUT);
pinMode(_MOSI, OUTPUT);
pinMode(_CLK, OUTPUT);
pinMode(RST, OUTPUT);
Serial.begin(115200); // depends on the setting of the host PC
}

void loop()
{
  while (!Serial.available());  // wait for character
  if (Serial.available() > 0)
switch (Serial.read())
{
  case 'p': Serial.write(progEnable());
    break;
  case 'r': readProgmem();
    Serial.write(data);
    break;
  case 'a': while (!Serial.available());
    AL = Serial.read();
    break;
  case 'A': while (!Serial.available());
    AH = Serial.read();
    break;
  case 'd': while (!Serial.available());
    data = Serial.read();
    break;
  case 'S': AH = 0;
    AL = 0;
    SigH = readSign();
    Serial.write(SigH);
    break;
  case 's': AH = 2;
    AL = 0;
    SigL = readSign();
    Serial.write(SigL);
    AH = 1;
    AL = 0;
    SigL = readSign();
    Serial.write(SigL);
    break; // read SigL
  case 'o': digitalWrite(RST, 1); break;
  case 'c': digitalWrite(RST, 0); break;
  case 'e': eraseChip();
    Serial.write(RDY);
    break;
case 'j': break;
    case 'w': writeProgmem();
              break;
  }
}

unsigned char SendSPI(unsigned char data)
{
    uint8_t retval = 0;
    uint8_t intData = data;
    int t;

    for (int ctr = 0; ctr < 7; ctr++)
    {
        if (intData & 0x80) digitalWrite(_MOSI,1);
        else digitalWrite(_MOSI,0);

        digitalWrite(_CLK,1);
        delayMicroseconds(1);

        t = digitalRead(_MISO);
        digitalWrite(_CLK,0);

        if (t) retval |= 1; else retval &= 0xFE;
        intData<<= 1;
        delayMicroseconds(1);
    }

    if (intData & 0x80) digitalWrite(_MOSI,1);
    else digitalWrite(_MOSI,0);

    digitalWrite(_CLK,1);
    delayMicroseconds(1);

    t = digitalRead(_MISO);
digitalWrite(_CLK,0);

if (t) retval |= 1;
else retval &= 0xFE;
return retval;

byte progEnable()
{
  SendSPI(0xAC);
  SendSPI(0x53);
  SendSPI(dummyData);

  return SendSPI(dummyData);
}

void eraseChip()
{
  SendSPI(0xAC);
  SendSPI(0x9F);
  SendSPI(dummyData);
  SendSPI(dummyData);
  delay(520);
}

void readProgmem()
{
  SendSPI(0x20);
  SendSPI(AH);
  SendSPI(AL);
  data = SendSPI(dummyData);
}

void writeProgmem()
{
  SendSPI(0x40);
  SendSPI(AH);
  SendSPI(AL);
  SendSPI(data);
}

void writeLockBits()
{ SendSPI(0xAC);
  SendSPI(lockByte);
  SendSPI(dummyData);
  SendSPI(dummyData); }

void readLockBits()
{ SendSPI(0x24);
  SendSPI(dummyData);
  SendSPI(dummyData);
  lockByte = SendSPI(dummyData); }

byte readSign()
{ SendSPI(0x28);
  SendSPI(AH);
  SendSPI(AL);
  return SendSPI(dummyData); }

The form of 8051 SPI programmer software