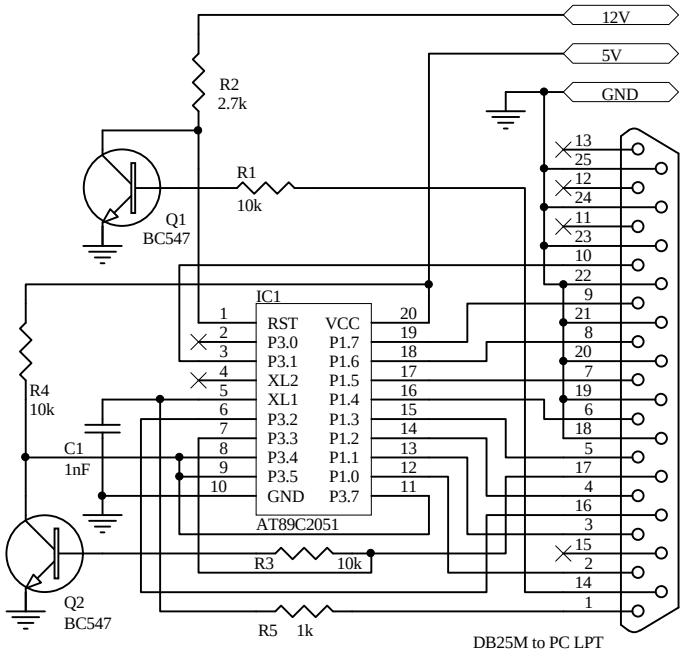
AT89C2051 Programmer



Description

BlowIT 2051 is simplest possible burner for AT89C2051 Flash Microcontroller. It is "as simple as possible" only designed for fast and dirty jobs, where you need to get going and fast. Only very few components are required. Functionality is minimal, only erase and program are implemented.

Power Supply

Basically +5 and +12 are available inside the PC on the floppy/HD power cable as example. Well use them at your own risk. A simple power supply can be buildt using 7812 and 7805.

Using BlowIT Software

Current release only supports binary files, use HEXBIN.EXE to convert from hex to binary. LPT1 to LPT3 can be used, LPT1 being the default. Program is been tested to run in Windows 95 DOS Box.

Correct Cable

Get a good DB25 extension cable and not too long one. All pins must be connected 1:1. I used a 5m long modem extension cable and it worked fine. When using a shorter cable then R5 and C1 may be obsolete, with a real long cable they where necessary to get reliable results.

About AT89C2051

AT89C2051 is a Intel'51 compatible Microcontroller from Atmel.

Package DIP20 or SO20
Program Memory 2k Bytes (Flash)
RAM 128 Bytes
Direct LED drive
12 or 24 MHz max clock
UART
Analog Comparator

Those are the main features.
What is unbelivable is the price: below 2USD

Software Tools for AT89C2051

Any 51 Software can be used
There are many free assemblers. The best are Metalink and Intel ones. BAS051 Basic compiler from W. Washington is not useable, as it places vars in extrernal memory. BASCO from Silicon Studio is a AT89C2051 comaptible Basic Compiler.

Additional Information

Datasheets for AT89C2051 can be downloaded from Atmel's WWW site
http://www.atmel.com

Updates and Bugfixes

BlowIT executable can be downloaded from Silicon Studio's WWW site:
http://sistudio.com

We are interested to hear if there are any problems using this hardware. Send us mail to:
comments@sistudio.com

Disclaimer

This hardware and program are tested on one single computer (486-DX66). It most likely works on most PC computers but we take no responsibility which or whatever. Use at your own risk.

Program listing

This is the first working version. It is not “cleaned up” or fully commented. The VTD unit is a responsible for time delays.

```
Program BlowIt2051;
{$D Copyright (C) 1996 Silicon Studio}
Uses  VTD;
var
  base: Word;
  i, j, k, l: Integer;
  mem: Array[0..2047] of byte;
  f: file;
Const
  inc_bit    = $01;
  vpp_bit    = $02;
  prog_bit   = $04; { not inverse! }
  erase_bit  = $08;
  prog_mode  = $0F;
  erase_mode = $07;
  idle       = $09; { Idle, RST = 0 }
label
  err_exit;
begin
   Writeln;
   Writeln('------ BlowIT (tm) 2051 ver 1.0 beta ------');
   Writeln('http://sistudio.com for schematics and latest release');
   Writeln;

   if ParamCount<1 then begin
      Writeln('usage: BI2051 filename.bin [n]');
      Writeln('usage: n=1,2,3 LPT number to use');
       Writeln;
       halt(1);
   end;
   base := memw[$40:8];
   if ParamStr(2) = '2' then begin
     base := memw[$40:10];
   end;
   if ParamStr(2) = '3' then begin
     base := memw[$40:12];
```

```
end;
If base = 0 then begin
  Writeln('Parallel port does not exist!');
  Halt(2);
end;
for i:=0 to 2047 do mem[i] := $FF;
assign(f, ParamStr(1));
reset(f,1);
blockread(f, mem, 2048,i);
close(f);
If IOResult <> 0 then begin
  Writeln('File not found!');
  halt(2); { File error }
end;

port[base] := $FF;
{ Erase }
port[base+2] := $01;
nms(10);
port[base+2] := $03;                        { RST = 12 }
nms(1);
port[base+2] := erase_mode;                 { RST = 12, PROG = H }
nms(1);
port[base+2] := erase_mode xor prog_bit;    { P3.2 = 0 }
nms(12);                                    { This is ERASE PULSE!}
port[base+2] := erase_mode;                 { P3.2 = 1 }
nms(12);
port[base+2] := erase_mode xor vpp_bit;     { RST = 0 }
nms(20);

{ Program ROM }
port[base+2] := idle;
nms(10);
port[base+2] := idle or vpp_bit;
nms(10);
port[base+2] := prog_mode;    { RST = 12, PROG = H }
nms(1);
Write('Blowing');
k := 0; l := 0;
for j := 0 to 2047 do
begin
   port[base] := mem[j];
```

```
    if mem[j] <> $FF then begin

    delay_mks(2);
    { PROG Pulse}
    EnterCriticalSection;
    port[base+2] := prog_mode xor prog_bit; { P3.2 = 0 }
    delay_mks(5);
    port[base+2] := prog_mode; { P3.2 = 1 }
    delay_mks(2);
    i := 0;
    while (port[base+1] and 64) = 0 do
    begin
      delay_mks(10);
      if i>200 then begin
         Writeln;
        Writeln('Error, never ready?');
         goto err_exit;
      end;
      if i > k then k := i;
      inc(i);
    end;
    LeaveCriticalSection;
    if l > 30 then begin
       Writeln;
       Writeln('Error, no device?');
       goto err_exit;
    end;
    if i = 0 then inc(l);
    delay_mks(2);
    end;
    port[base+2] := prog_mode xor inc_bit; { XT1 = 1 }
    delay_mks(5);
    port[base+2] := prog_mode; { XT1 = 1 }
    delay_mks(2);

    if (j mod 64) = 0 then Write('.');
  end;
  port[base] := $FF;  nms(1);
  port[base+2] := prog_mode xor vpp_bit; { P3.2 = 0!! }
  nms(1);
  port[base+2] := idle;
  port[base] := $00;
```

```
   Writeln;
   Writeln('We are finished.');
   Halt(0);

err_exit:
   LeaveCriticalSection;
   Halt(1);
end.
```