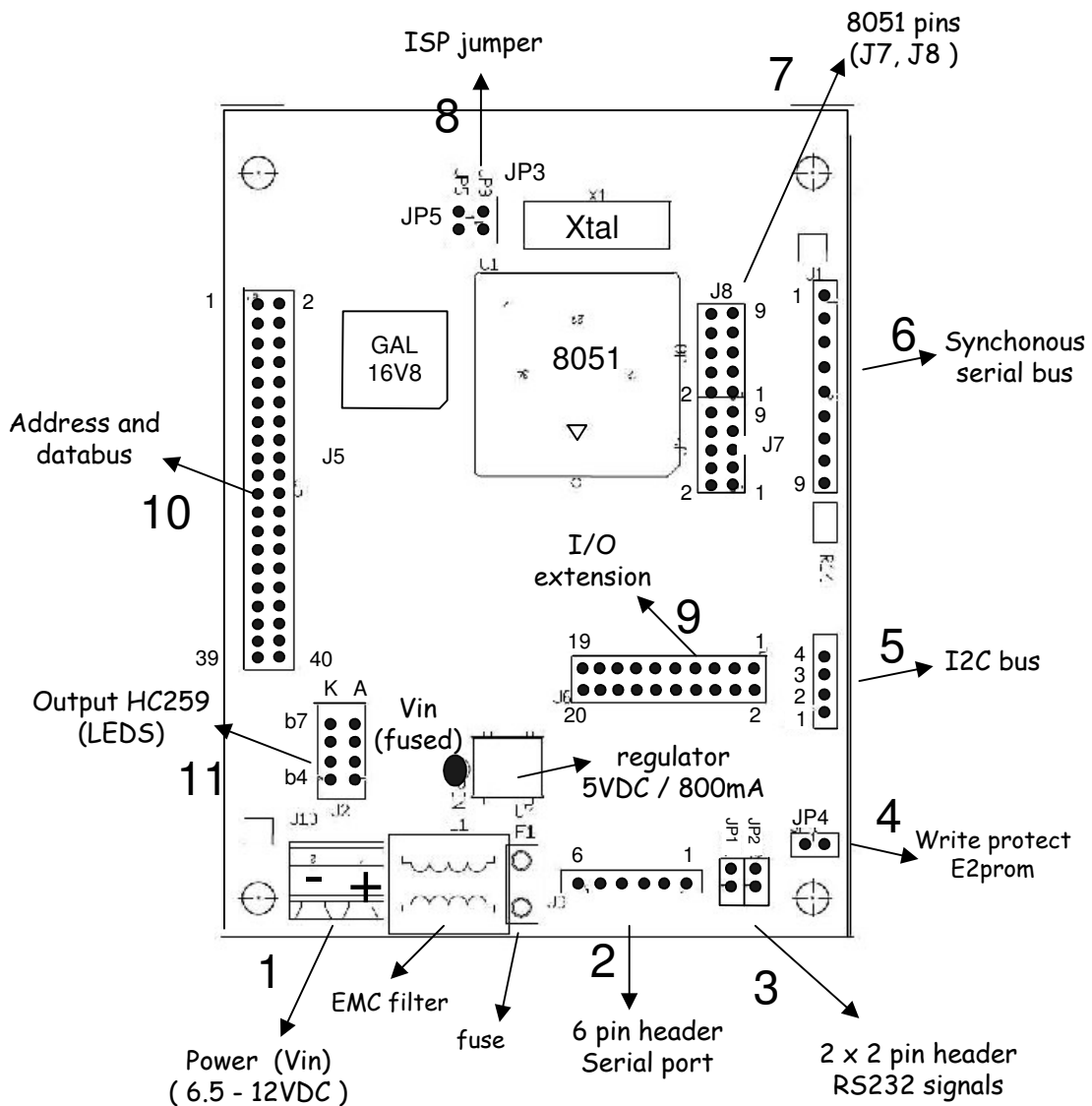


Universal microcontroller board with 8051



Board layout :



1 : Power input

nominal 7 - 9VDC, Max 12VDC

2 : Serial port connection :

from left (pin6) to right (pin1): GND, Txd, Rxd, P1.5(*), P1.4(*), 5VDC
(*): if jumper placed

3 : Jumpers

JP1 : P1.5 to serial port pin 3 (pins not mounted)
JP2 : P1.4 to serial port pin 2 (pins not mounted)

4 : Write protect E2prom

place jumper to protect (pins not mounted)

5 : I2C bus expansion (pin 1 = bottom)

pin 1 : VCC
pin 2 : SCL / P1.6
pin 3 : SDA / P1.7
pin 4 : GND

6 : Synchronous serial bus expansion (pin 1 = TOP)

This bus is used to interface existing digital and analog input- and outputmodule. The functionality below is according to this hardware interface. The user is free to use these outputs and single input for any purpose and can function to connect for example SPI devices.

(TOP) 1 : SDO (serial data output)
2 : N_OE (output enable, active low)
3 : CLK (clock output)
4 : Inhibit (output)
5 : GND
6 : SDI (serial data input)
7 : Shift / not load : high = shift enable, inverted pulse to load the
inputlevels (output)
8 : VCC
9 : Strobe : latch the shifted outputs, output

7 : 8051 pins + power

J7.1 : P1.0 / T2
J7.2 : P1.1 / T2X
J7.3 : P1.2
J7.4 : P1.3
J7.5 : P1.4
J7.6 : P1.5
J7.7 : P1.6
J7.8 : P1.7
J7.9 : GND
J7.10 : VCC

J8.1 : P3.0 / RxD
J8.2 : P3.1 / TxD
J8.3 : P3.2 / INT0
J8.4 : P3.3 / INT1
J8.5 : P3.4 / T0
J8.6 : P3.5 / T1
J8.7 : P3.6 / N_WR
J8.8 : P3.7 / N_RD
J8.9 : GND
J8.10 : VCC

8 : Jumper JP5, JP3

Place JP5 to program the P89C51Rx2 via ISP,
then reset the system and program with FlashMagic (or other application) via the serial port)
(JP3 is not needed)

9 : I/O extension pins on J6

Digital output register (byte addressable):

J6.1 : HC574, bit6

J6.2 : HC574, bit7

Digital inputs (byte addressable)

J6.3 : HC541, bit1

J6.4 : HC541, bit2

J6.5 : HC541, bit3

J6.6 : HC541, bit4

J6.7 : HC541, bit5

J6.8 : HC541, bit6

J6.9 : HC541, bit7

I/O selection (memory mapped)

J6.10 : N_IO_WR_SEL3

J6.11 : N_IO_WR_SEL4

J6.12 : N_IO_RD_SEL2

J6.13 : N_IO_RD_SEL3

J6.14 : N_IO_RD_SEL4

Digital outputs (bit addressable)

J6.15 : NEW_A15, HC259 bit0

J6.16 : NEW_A16, HC259 bit1

J6.17 : NEW_A17, HC259 bit2

J6.18 : NEW_A18, HC259 bit3

Power supply :

J6.19 : GND

J6.20 : VCC

10 : Address / data bus expansion (J5)

1 : N_Reset (output, active low)

2 : Reset (output, active high)

3 : N_RD / P3.7

4 : N_WR / P3.6

5 : N_PSEN

6 : ALE

7 : N_IO_RD

8 : N_IO_WR

9 : GND

10 : VCC

Latched lower address byte

23 : A0

24 : A1

25 : A2

26 : A3

27 : A4

28 : A5

29 : A6

30 : A7

11 : N_DEV1

12 : N_DEV2

13 : N_DEV3

14 : N_DEV4

31 : A08 / P2.0

32 : A09 / P2.1

33 : A10 / P2.2

34 : A11 / P2.3

35 : A12 / P2.4

36 : A13 / P2.5

37 : A14 / P2.6

38 : A15 / P2.7

15 : AD0 / P0.0

16 : AD1 / P0.1

17 : AD2 / P0.2

18 : AD3 / P0.3

19 : AD4 / P0.4

20 : AD5 / P0.5

21 : AD6 / P0.6

22 : AD7 / P0.7

39 : GND

40 : VCC

11 : Outputs to drive LEDS

Pinheaders to connect 4 LEDS, one on each pair. The LEDS are driven from VCC via a 1K resistor. 4 HC259 outputs drive 1 led each and need to be low to light it : i.e. pin 'b7' is controlled by HC259, bit 7. Connect the anode at side A, the kathode at side K.

Mapping of the memory and I/O

The board has no external flash memory and so needs a controller with internal flash. The board is designed around the Philips 89C51Rx2, but can also hold pin compatible types (like Dallas 89C4x0 series), although it is not guaranteed that these can be 'ISP' programmed.

The board has 32K RAM that is mapped :

0000H - 7FFFH : DATA, read / write
 8000H - EFFFH : DATA or CODE read / write

The RAM is mirrored at 8000H, i.e memory 0000H and 8000H points to the same physical location.

If the controller has maximum 32K flash on board, the RAM above 8000H can hold programcode. This can be usefull to run a monitor in lower FLASH and load the programcode from 8000H to debug.

I/O mapping for the selection outputs :

N_DEV1 : F400H - F7FFH, read / write in DATA or CODE
 N_DEV2 : F800H - F7FFH, read / write in DATA or CODE
 N_DEV3 : FC00H - FFFFH, read / write in DATA
 N_DEV4 : F400H - F7FFH, write in DATA

N_IO_RD : F000 - F3FFH, read in DATA
 N_IO_WR : F000 - 3FFFH, write in DATA

N_IO_RD_SEL1 : F000H - F0FFH, read from DATA, used for the HC541 inputbuffer
 N_IO_RD_SEL2 : F100H - F1FFH, read from DATA, free for expansions
 N_IO_RD_SEL3 : F200H - F2FFH, read from DATA, free for expansions
 N_IO_RD_SEL4 : F300H - F3FFH, read from DATA, free for expansions

N_IO_WR_SEL1 : F000H - F0FFH, write to DATA, used for the HC574 outputregister
 N_IO_WR_SEL2 : F100H - F1FFH, write to DATA, used for the HC259 outputbits
 N_IO_WR_SEL3 : F200H - F2FFH, write to DATA, free for expansions
 N_IO_WR_SEL4 : F300H - F3FFH, write to DATA, free for expansions

Below you find the equations used to create the decoder functionality in the GAL :

```

IO_ADDRESS <= A15 and A14 and A13 and A12;           -- 0F000H..0FFFFH
IO_SELECT <= IO_ADDRESS and N_PSEN;                   -- IO_ADDRESS in DATA segment (global IO-area)

RAM_SELECT <= ( not(IO_ADDRESS) and N_PSEN ) or ( not(IO_ADDRESS) and A15 and not(N_PSEN) ); --
                                                    [0000H..0EFFFH] in DATA segment, [0000h..0EFFFH] in code segment

N_RAM <= not( RAM_SELECT );
N_RAM_OE <= not( RAM_SELECT and ( not(N_RD) or not(N_PSEN) ) );

IO_READ <= not( N_RD OR not(IO_SELECT) or A10 or A11); -- 0F000H..0F3FFH
N_IO_RD <= not(IO_READ);

IO_WRITE <= not(N_WR OR not(IO_SELECT) or A10 or A11); -- 0F000H..0F3FFH
N_IO_WR <= not(IO_WRITE);

N_DEV1 <= not( IO_ADDRESS and not( A11 ) and A10 );    -- 0F400..0F7FFH read/write in data or code
N_DEV2 <= not( IO_ADDRESS and A11 and not(A10) );    -- 0F800..0FBFFH read/write in data or code
N_DEV3 <= not( IO_SELECT and A11 and A10 );          -- 0FC00..0FFFFH read/write in data
N_DEV4 <= not( IO_ADDRESS and not( A11 ) and A10 and not(N_WR) ); -- 0F400..0F7FFH write in data
  
```