MADLAB PICWORKS1

PICworks1 has the following features:

- suitable for small robotics & animatronics projects
- 2 x low-current outputs, 2 x high-current outputs, 2 x digital inputs
- 2 x LEDs and 2 x pushbuttons on-board
- drives 2 x dc motors (uni-directional) and 2 x servo motors
- motor speed and servo parameters adjustable in firmware
- analogue sensors (LDR, preset etc.) can be connected
- sample routines in documented source code
- requires access to PIC assembler and programmer
- in-circuit PIC programming
- supply voltage 4.8V - 12V dc

Construction

First fit and solder the resistors (R1 to R8) and trim their legs. Identify the resistors by the coloured stripes on the body. Next fit and solder the capacitors, paying attention to the polarity of the electrolytic capacitors C1 and C2 (negative is marked by a stripe on the side of the body, and also the shorter leg). The ceramic capacitor (C3) can be fitted either way around.

Then fit the transistors and the regulator. The symbols on the board indicate the orientation of the transistors and regulator (flat side of the component against the flat side of the symbol). Be careful not to mistake the regulator for a transistor.

Solder the LEDs (L1 and L2) putting the shorter leg into the hole with the line (the shorter leg is also marked by a flat on the rim of the body).

Next fit the chip socket IC1 (matching the notch in the socket against the notch in the symbol on the board). Care should be taken when soldering this component to avoid solder bridges between the pins. It is not recommended that the chip is soldered directly to the board.

Fit the pushbuttons (S1 and S2), and terminal blocks (CON1 to CON3). The wire-entry holes in the terminal blocks should all face outwards (towards the edge of the board).

Solder the header plug (CON4) to the board. The shorter pins are soldered to the board and the longer pins point out from the pcb. If in-circuit programming is not required the header plug can be omitted.

Solder the battery snap (BATTERY) to the board. Support holes are drilled on the board for the battery snap leads. Feed the leads up through the support holes from the track side of the board and then down the solder holes. Red is positive and black is negative.

Don't fit the chip until you have checked your construction. Check that all the components have been inserted correctly and that there are no dry joints and no solder bridges between pins. Then carefully bend the legs of the chip inwards a little with your fingers. Fit the chip into its socket matching the small notch in the chip to the notch in the socket.

The board can be powered by 4 AA or 6 AA cells. Primary (i.e. non-rechargeable, 1.5V) cells as well as secondary (rechargeable, 1.2V) ones can be used.

The four corner holes can be used to mount the pcb within a small case.

The factory firmware includes a power-on self-test. Connect a battery to the battery snap and both LEDs should flash twice. The state of the pushbuttons is then echoed to the low and high current outputs. Pressing S1 should cause L1 to light, and pressing S2 should cause L2 to light.

Connecting external LEDs

Two external LEDS can be connected in parallel with the on-board LEDs. Connect the first LED to LO1 (anode) & GND (cathode), and the second to LO2 (anode) & GND (cathode). The cathode of an LED is its shorter leg and is generally also indicated by a flat on the rim of its body. Resistors (typically 470 ohms) should be connected in series with the LEDs to limit the current.

The external (and also on-board) LEDs are lit by the instructions `bsf LO1` and `bsf LO2`, and turned off again by `bcf LO1` and `bcf LO2`.

Connecting external switches

External switches (read switches, microswitches etc.) can be connected in parallel with the on-board pushbuttons. Connect the first external switch to IN1 & GND, and the second to IN2 & GND. No external pull-up resistors are needed.

The state of the external switches (and also the on-board pushbuttons) can be checked by testing `IN1` and `IN2`. See the default program in the source code for an example.

Connecting dc motors

PICworks1 can drive two dc motors uni-directionally. Connect the first motor (MOTOR1) to HI1 & COM, and the second (MOTOR2) to HI2 & COM. 100n ceramic capacitors should be soldered across the tags of the motors to reduce noise.

Note that dc motors are driven at the supply voltage rather than regulated 5V. The Darlington transistors are capable of delivering a maximum of 500mA output current per motor.

The speed of each motor is set by writing to the variables `speed1` and `speed2`. The range of these variables is `0` (off) to `64` (fully on), and their values correspond to fractions (64^{th}'s) of the supply voltage. So, for example, a value of 32 is equivalent to supplying the motor with a voltage of 32/64, or half, the supply voltage. The motors are driven by a PWM signal of sufficiently high frequency that the pulsed voltage effectively appears as a constant voltage to the motor.

The constants `MOTOR1` and `MOTOR2` in the conditional assembly switches section of the source code should be set to `1` to enable the motors.

Be careful not to exceed the rated voltage for the motors used, otherwise excessive current consumption may crash the PIC microcontroller.

Connecting relays

Connect the relay coils to HI1 & COM and HI2 & COM. Diodes should be connected across the coils in the reverse direction to provide a path for the back emf (current normally flows from COM to ground through HI1 and HI2). The coils should be rated at the supply voltage.

The relays are energised by the instructions `bsf HI1` and `bsf HI2`, and turned off again by `bcf HI1` and `bcf HI2`.

Connecting servo motors

PICworks1 can control two servo motors. Connect the first servo (SERVO1) as follows: power (positive, usually red) to COM, ground (negative, usually black) to either GND, control (blue or white etc.) to LO1. Connect the second servo (SERVO2): power to COM, ground to either GND, control to LO2.

Note that servos are powered at the supply voltage rather than regulated 5V (but the control inputs are 5V).

Servo motors are controlled by means of a pulse-width modulated signal with a fixed duty cycle period. The length of the pulse determines the position of the servo. For a standard servo, pulses of 1.5ms centre it, pulses of 1.0ms move it the maximum 90° anti-clockwise, and pulses of 2.0ms move it the maximum 90° clockwise. Particular servos may have different maximum & minimum angles and control pulse ranges (many servos have a range of 0.5ms to 2.5ms). These parameters can be set by modifying the constants `SERVO_MIN` and `SERVO_MAX` in the source code (see below). Reducing the pulse length range usefully allows the movement of the servo to be constrained.

The position of each servo is set by writing to the variables `servo1` and `servo2`. The range of these variables is `0` (fully anti-clockwise) to `((SERVO_MAX-SERVO_MIN)/16)-1` (fully clockwise). The constants `SERVO1` and `SERVO2` in the conditional assembly switches section of the source code should be set to `1`.

Connecting analogue sensor

An analogue sensor such as an LDR or variable resistor (or other simple resistive sensor) can be connected to *PICworks1*. A 10 ohm resistor and a 100n polyester (or similar) capacitor are also required. Connect one lead of the sensor to COM, one lead of the capacitor to either GND, and one lead of the resistor to IN2. Connect the other leads of the three components all together. (If using a variable resistor, connect to the centre wiper and either end of the track.)

To enable analogue sensing, set the constant `ANALOGUE` in the conditional assembly switches section of the source code to `1`. The constant `DELAY` should be modified according to the sensor in use (see the source code for examples). The routine `sample_sensor` then returns an analogue sensor reading in the range `0` to `255`.

Connecting piezo speaker

Connect a piezo speaker to LO1 & GND, LO2 & GND, or LO1 & LO2. In the third case the piezo pins are driven in anti-phase and this results in an increase in loudness. A 100 ohm resistor should be connected in series with one of the piezo pins.

If the piezo is connected to LO1 then set `PIEZO1` in the conditional assembly switches section of the source code to `1`, if the piezo is connected to LO2 then set `PIEZO2` to `1`, and if it is connected to both set both constants to `1`. The routine `beep` then sounds a short beep.

The piezo is driven at a fixed frequency (~2kHz) via the interrupt service routine. For variable frequency operation the piezo should be driven directly (disabling interrupts to avoid glitches).

In-circuit programming

A 5-way header plug is included to allow *PICworks1* to be connected to a PIC programmer for in-circuit programming. You will need to make up a suitable cable to connect *PICworks1* to your programmer to use this facility. Five connections are needed - clock (CLK), data (DAT), programming voltage (VPP), supply voltage (5V), and ground (0V). These correspond to pins 6 (CLK), 7 (DAT), 4 (VPP), 1 (5V), and 8 (0V) on the PIC. Refer to the Microchip 12F629 documentation for more information.

If in-circuit programming is required then be careful not to drive LO1 or LO2 with too high a current which could lead to corruption during programming of the clock and data signals which share the same pins. These pins already drive the on-board LEDs so remove R6 and R7 if necessary.

Remember to disconnect the battery from *PICworks1* when programming the PIC.

Some PIC programmers don't have the drive capability to be able to program a PIC that is not inserted directly into the socket on the programmer. Whether in-circuit programming will be possible with a particular PIC programmer can be determined by trial and error.

Source code

The documented source code for the *PICworks1* firmware is downloadable from the MadLab website. This firmware is meant as a starting point for writing your own programs. It contains driver routines for dc motors, servo motors, piezos, and analogue sensors as well as useful utility routines for reading and writing the EEPROM memory and generating random numbers.

To use the supplied routines you will need access to a PIC assembler and programmer. A suitable programmer is PICSTART Plus from Microchip (http://www.microchip.com) but there are many other capable programmers on the market (it must be able to program 12F629's though).

The various supplied routines are enabled by means of conditional assembly switches. These are contained in a block near the start of the source code. To use a particular routine its switch must be enabled by setting its value to non-zero. Different devices share the same terminal blocks on *PICworks1* but are controlled in different ways and the firmware needs to know what it is connected to.

See http://www.madlab.org/kits/picworks1.html for further information.

Component List

Resistors

R1	1R (brown, black, gold, gold)
R2, R4	4R7 (yellow, purple, gold, gold)
R3, R5, R6, R7	1k (brown, black, red, gold)
R8	47k (yellow, purple, orange, gold)

Capacitors

C1	100u electrolytic
C2	10u electrolytic
C3	100n ceramic (brown, marked '104')

Semiconductors

TR1, TR2	MPSA13 Darlington transistors
REG	78L05 regulator (5V, 100mA)
L1	red LED
L2	green LED
IC1	8-pin socket + PIC12F629-I/P microcontroller (FE10)

Miscellaneous

S1, S2	miniature pushbuttons
CON1, CON2, CON3	3-way terminal blocks
CON4	5-way header plug
BATTERY	PP3 moulded battery snap

Battery box	4 x AA

PCB

Design and documentation © MadLab® 2006