MADLAB PICWORKS2

PICworks2 has the following features:

- drives 4 x LEDs (back-to-back in pairs)
- drives 2 x dc motors (bi-directional)
- or, drives 2 x servo motors
- or, drives 1 x stepper motor (bipolar or unipolar)
- analogue light sensor
- analogue preset control
- moving-coil speaker and 2-channel tune player
- keyswitch
- real-time control using PS/2 keyboard
- recording and playback of up to 120 events
- motor/servo/stepper parameters and speeds adjustable
- supply voltage 6V - 12V dc

Construction

First solder the two short pieces of wire to the pairs of holes marked LINK. Then fit and solder the resistors (R1 to R9) 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 C3, C6 and C7 (negative is marked by a stripe on the side of the body, and also the shorter leg). The polyester (C1) and ceramic (C2, C4 and C5) capacitors can be fitted either way around.

Then fit the transistors and the regulator. The symbols on the board indicate the orientation of the transistors (flat side of the component against the flat side of the symbol). The regulator is mounted upright with its metal side facing inside the board (towards the chips).

Solder the light sensor (LDR) to the board either way around. Be careful when soldering as excessive heat may melt the plastic.

Solder the keyboard socket (PS/2) and speaker (SPEAKER). Take care when soldering the PS/2 socket as the pins are very close together. The speaker has a marked polarity (+ and – signs on the underside) but in practice the polarity does not matter (in other words it can be soldered either way around).

Next fit the chip sockets IC1 and IC2 (matching the notch in the socket against the notch in the symbol on the board). Again care should be taken when soldering these components to avoid solder bridges between the pins. It is not recommended that the chips are soldered directly to the board.

Fit the crystal (XTAL), keyswitch (S1), and terminal blocks (CON1 to CON4). The wire-entry holes in the terminal blocks should all face outwards (towards the edge of the board).

Solder the preset (VR1) to the board and firmly push the spindle into the small the hole in the top.

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 chips until you have thoroughly 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 chips inwards a little with your fingers. Fit the chips into their sockets matching the small notch in the chip to the notch in the socket.

Rubber feet for the four corner holes are supplied. Alternatively these holes can be used to mount the pcb within a small case.

Insert 6 AA cells into the battery box, observing the correct polarity. The cells used should ideally be rechargeable NiMH or NiCd types, but if disposable cells are used they should be good quality alkaline ones. It is recommended that rechargeable NiMH cells are used.

The firmware includes a power-on self-test. Connect the battery box to the battery snap and a double beep should sound.

Disconnect the battery box, connect a PS/2 PC keyboard to the socket then re-connect the batteries. Keyboard options are provided to test the LDR light sensor and preset. Function key F11 echoes the light sensor to the keyboard LEDs. Waving your hand in front of the sensor should cause the LEDs to flicker. Function key F12 echoes the preset control to the keyboard LEDs. Rotating the spindle should cause the LEDs to come on and off.

The board can be powered by 4 AA cells rather than 6 cells, but if so they must be primary (i.e. non-rechargeable) types (1.5V) rather than secondary (rechargeable) ones (1.2V).

Connecting LEDs

Up to four LEDS can be connected as back-to-back pairs. The LEDs are multiplexed which means that only one of each pair of LEDs is illuminated at any one time, but the multiplexing frequency is sufficiently high for any flicker not to be apparent.

Connect the anode of the first LED (LED1) to LO1 and its cathode to LO2. Connect the second LED (LED2) anode to LO2 and cathode to LO1, the third (LED3) anode to LO3 and cathode to LO4, and the fourth (LED4) anode to LO4 and cathode to LO3. The cathode of an LED is its shorter leg and is generally also indicated by a flat on the rim of its body.

Press the digit keys **1**, **2**, **3** and **4** and check that all four LEDs are lit.

100 ohm resistors (R7 and R8) on the circuit board in series with the outputs LO1 and LO3 limit the (multiplexed) LED current to about 15mA (less for white or blue LEDs).

Resistors are not in series with the LO2 and LO4 outputs. These outputs can be driven as single-ended outputs but if doing so external resistors should be used to limit the current. Note that outputs LO2 and LO4 are still driven by a 50% duty cycle whatever is connected to them.

If LO1 or LO2 are used then the preset (VR1) should be turned fully clockwise.

LO3 and LO4 could be reconfigured as inputs (with weak pull-ups) but this is not supported by the supplied firmware. However the PIC source code is freely available online for user modifications (see below).

Connecting dc motors

PICworks2 can drive two dc motors bi-directionally (forward and reverse). Connect the first motor (MOTOR1) to HI1 & HI2, and the second (MOTOR2) to HI3 & HI4. 100n ceramic capacitors should be soldered across the tags of the motors to reduce noise.

Press **F1** then **ENTER** and check that MOTOR1 turns in both directions when the left and right arrow keys are pressed. Then press **shift F1** and **ENTER** and check that MOTOR2 turns in both directions when the up and down arrow keys are pressed.

Note that dc motors are driven at the supply voltage rather than regulated 5V. The H-bridge is capable of delivering a maximum of 600mA output current per motor.

The speeds of the two motors can be set independently using a PS/2 keyboard (see below). Speed control works by reducing the voltage applied to the motor, so motors rated at less than the supply voltage can be safely used. Be careful not to exceed the rated voltage for the motors used, otherwise excessive current consumption may crash the PIC microcontroller.

The hardware is capable of driving four uni-directional motors connected to HI1 & GND, HI2 & GND etc. but the supplied firmware doesn't support this. However the PIC source code is freely available online for user modifications (see below).

Note that *PICworks2* should be disconnected from the power when not being used as the H-bridge chip draws quite a large quiescent current of about 50mA even when not driving the motors, and the regulator is also a consumer of current.

Connecting servo motors

PICworks2 can control two servo motors. Connect the first servo (SERVO1) as follows: power (positive, usually red) to HI2, ground (negative, usually black) to GND, control (blue or white etc.) to HI1. Connect the second servo (SERVO2): power to HI4, ground to GND, control to HI3.

Press **F2** then **ENTER** and check that SERVO1 turns in both directions when the left and right arrow keys are pressed. Then press **shift F2** and **ENTER** and check that SERVO2 turns in both directions when the up and down arrow keys are pressed.

Note that servos are driven at the supply voltage rather than regulated 5V (this applies to the control input as well as the power inputs). The H-bridge is capable of delivering a maximum of 600mA output current per servo.

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. These parameters as well as the tracking speeds can be set independently for the two servos (see below).

Instead of two servos, one servo and one dc motor can be connected instead.

An additional two servos could be driven from HI2 and HI4 if the servos were powered externally, but this is not supported by the supplied firmware. However the PIC source code is freely available online for user modifications (see below).

Connecting stepper motor

A single bipolar stepper motor (STEPPER) can be used. Connect the first stepper coil to HI1 & HI2, and the second coil to HI4 & HI3.

Press **F5** then **ENTER** and check that STEPPER turns in both directions when the left and right arrow keys are pressed.

Note that the stepper is driven at the supply voltage rather than regulated 5V. The H-bridge is capable of delivering a maximum of 600mA output current per stepper coil.

The stepper tracking speed can be set using a keyboard (see below), and additionally one of three driving sequences can be selected:

Standard sequence: `1A -> 2A -> 1B -> 2B`
High-torque sequence: `1A+2A -> 2A+1B -> 1B+2B -> 2B+1A`
Half-step sequence: `1A+2A -> 2A -> 2A+1B -> 1B -> 1B+2B -> 2B -> 2B+1A -> 1A`
where `1` is the first coil and `2` the second, and `A` and `B` are the two connections to each coil (i.e. `1A` connected to HI1, `1B` to HI2, `2A` to HI4, and `2B` to HI3).

Note that the high-torque sequence consumes twice the current because both coils are energised at the same time, and the tracking speed of the half-step sequence is half that of the other two sequences.

A unipolar stepper can also be driven by grounding or not connecting the centre taps.

Analogue sensors

Two analogue sensors are provided on board, a light-sensitive LDR and a rotary preset control.

The light sensor can be used to trigger a recorded sequence of events when light falls on the sensor, when light stops falling on the sensor, or when the light level momentarily changes. The latter mode can be used as a simple form of motion sensor.

The threshold between light and dark can be set using the keyboard function key **F7** (see below).

The preset can be used as a speed control for the motors (see below).

The preset could be replaced with an external thermistor of similar resistance (using two of the three VR1 holes that aren't connected to each other by a track on the circuit board), or any other kind of simple resistive sensor. In this case the threshold for the sensor can be set using the keyboard function key **F9** (see below).

Note that LO1 and LO2 can't be connected to anything if the LDR or preset are used.

Source code

The documented source code for the *PICworks2* firmware is downloadable from the MadLab website and is ‘open source’. In other words you are encouraged to study its operation and modify it to suit your particular requirements. To do this you will need access to a PIC programmer. A suitable programmer is PICSTART Plus from Microchip but there are many other capable programmers on the market (it must be able to program 16F648A’s though).

See http://www.madlab.org/kits/picworks2.html for further information.

A Microsoft Windows application is provided on the MadLab website to convert a standard MIDI music file into a form suitable for *PICworks2*. To make use of this you will need a PIC programmer and software to re-assemble the source code. The MPLAB development system includes a suitable PIC assembler and is freely downloadable from the Microchip website (http://www.microchip.com).

Keyboard settings

F1 - set MOTOR1 maximum speed (1 to 64, higher = faster)
shift F1 - set MOTOR2 maximum speed (1 to 64, higher = faster)
F2 - set SERVO1 minimum pulse length (50 to 250, in increments of 10us)
shift F2 - set SERVO2 minimum pulse length (50 to 250, in increments of 10us)
F3 - set SERVO1 maximum pulse length (50 to 250, in increments of 10us)
shift F3 - set SERVO2 maximum pulse length (50 to 250, in increments of 10us)
F4 - set SERVO1 tracking speed (1 to 127, higher = faster)
shift F4 - set SERVO2 tracking speed (1 to 127, higher = faster)
F5 - set STEPPER maximum speed (1 to 127, higher = faster) (NB not linear)
F6,1 - standard stepper sequence
F6,2 - high-torque stepper sequence (higher power)
F6,3 - half-step stepper sequence (lower speed)
F7 - set light threshold (1 to 255) then echo to keyboard LEDs (all on or all off) (**ESC** to exit)
F8,1 - PRESET analogue sensor
F8,2 - motor/servo/stepper speed controlled by PRESET
F9 - set PRESET threshold (1 to 255) then echo to keyboard LEDs (all on or all off) (**ESC** to exit)
F11 - echo LDR sensor to keyboard LEDs (3 most significant bits) (**ESC** to exit)
F12 - echo PRESET to keyboard LEDs (3 most significant bits) (**ESC** to exit)

Settings are entered using the digit keys on the keyboard followed by the **ENTER** key. **ENTER** pressed by itself selects without altering a setting. So, for example, to enable SERVO1 without changing its pulse length, just press **F2** followed by **ENTER**. A double beep sounds when **ENTER** is pressed.

Note that you must select a dc motor, servo or stepper before it will work by pressing the corresponding function key and entering a setting or just pressing **ENTER**. These different devices share the same terminal blocks but are controlled in different ways and *PICworks2* needs to be told what it is connected to.

MOTOR1 and MOTOR2 speeds are specified in terms of fractions (64^{th}'s) of the supply voltage. So, for example, an entered 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. Be careful not to exceed the rated voltages for the motors in use.

Servo pulse lengths are specified in terms of 10us. So an entered value of 250 equals 2500us or 2.5ms. The default range is 1.0ms minimum to 2.0ms maximum, but many servos have a range of 0.5ms to 2.5ms. Reducing the pulse length range usefully allows the movement of the servo to be constrained.

The fastest servo tracking speed corresponds to 1/10 second for full travel (hard left to hard right) at nominal pulse limits (1ms => 2ms). Note that actual servos may not be able to track as fast as this.

The stepper tracking speed is in terms of milliseconds per step cycle. The fastest tracking speed is a minimum delay of 1ms between steps. Note that there is a physical limit to the maximum speed for every stepper motor. If the maximum for a particular stepper is exceeded then it will cease to rotate properly.

The stepper sequence is selected by pressing function key **F6** followed by the digit key **1**, **2** or **3** (followed by **ENTER**). The default sequence is the standard sequence which should be suitable for most situations.

The **F8,2** (**ENTER**) option allows the motor/servo/stepper speed to be continuously varied by rotating the preset control. **F8,1** (**ENTER**) disables this.

All settings are stored in non-volatile memory and retained after power is removed from the board.

Real-time control

1 - toggle LED1
2 - toggle LED2
3 - toggle LED3
4 - toggle LED4
RIGHT ARROW - toggle MOTOR1/SERVO1/STEPPER forward
LEFT ARROW - toggle MOTOR1/SERVO1/STEPPER reverse
UP ARROW - toggle MOTOR2/SERVO2/STEPPER forward
DOWN ARROW - toggle MOTOR2/SERVO2/STEPPER reverse
B - sound a beep
T - play tune (stops all motors)
ESC - restart

The arrow keys (cursor keys) on the keyboard are used to control the motors. Each time an arrow key is pressed the associated motor is toggled. That is to say if the motor was stationary it is started, and if it was running then it is stopped. It is not necessary to hold down an arrow key to keep a motor running.

A similar technique is used to toggle the LEDs on and off using the digit keys **1** to **4**.

Tune playing is mutually exclusive with motor control. In other words while a tune is playing all motors and outputs are disabled. Also the PS/2 keyboard is not polled while a tune is playing (this is necessary in order to maintain pitch accuracy while playing two notes simultaneously).

The default tune can be changed if you have access to a PIC programmer. An application downloadable from the MadLab website converts a standard MIDI file to the format expected by the *PICworks2* firmware (see above).

Recording events

shift R - record a sequence of events

S - stop recording
L - loop to start of sequence (and stop recording)
K,0 - wait for keyswitch off
K,1 - wait for keyswitch on
A,0 - wait for light off
A,1 - wait for light on
A,2 - wait for light change
V,0 - wait for preset off
V,1 - wait for preset on
V,2 - wait for preset change
ESC - abort

P or **ENTER** - play recorded sequence of events

The record feature allows the real-time motor and LED control keys above to be recorded and played back as a complete sequence. Press **shift R** to begin recording a sequence.

Sequences can be looped by pressing **L** at the end of the sequence, or ended by pressing **S** instead.

The keyswitch commands (**K** followed by **0** or **1**) allow simple user interaction. A sequence can be halted until the on-board keyswitch is pressed for example. The keyswitch could be replaced by an external switch (a reed relay triggered by the opening of a door for example) to allow more control.

The light threshold is set using **F7** (see above). The higher the value the less light is needed to trigger. The light change option (**A** followed by **2**) is useful as a simple proximity sensor. When an object passes over or near the light sensor then a sequence can be triggered.

Up to 120 events can be stored in the non-volatile memory of the PIC. An event in this context is a change of state, for example starting a motor or stopping it. The time between events can be from 0 to 16383 seconds, and times are recorded with a resolution of 1/16s.

Note that motor parameter and other setting changes are not stored in sequences. In other words if you change say the servo speed in the middle of recording a sequence, then the new speed will apply to the entire playback of the sequence.

If a PS/2 keyboard is not detected on power-up then *PICworks2* automatically plays any recorded sequence. Otherwise a recorded sequence is played back by pressing **P** or the **ENTER** key.

Component List

Resistors

R1, R2, R7, R8	100R (brown, black, brown, gold)
R3	10R (brown, black, black, gold)
R4, R5	1k (brown, black, red, gold)
R6	47k (yellow, purple, orange, gold)
R9	1R (brown, black, gold, gold)
VR1	47k preset + spindle

Capacitors

C1	22n polyester (yellow, square)
C2	100n ceramic (brown, marked '104')
C3	10u electrolytic
C4, C5	22p ceramic (small, marked '22')
C6, C7	100u electrolytic

Semiconductors

TR1, TR2	BC547B transistors
REG	L7805CV regulator (5V, 1A)
IC1	16-pin socket + L293D H-bridge driver
IC2	18-pin socket + PIC16F648A-I/P microcontroller (FF1X)

Miscellaneous

LINK	2 x wire links
XTAL	20MHz low-profile crystal
LDR	light dependent resistor
SPEAKER	enclosed pcb speaker
S1	keyswitch
PS/2	PS/2 socket
CON1, CON2	3-way terminal blocks
CON3, CON4	2-way terminal blocks
BATTERY	PP3 moulded battery snap
Battery box	6 x AA

PCB
4 x rubber feet