AXE119 PICAXE-14M Kit for iPod

Contents:

- 1 AXE117 Project board PCB
- 1 PICAXE-14M microcontroller
- 1 PodBreakout Kit
- 1 470k resistor
- 1 4.5V battery box

iPod not included

Description:

The PICAXE-14M project board provides a rapid development system for testing serial control of an iPod. It should function with any iPod which has a 30 pin dock connector.
Note that iPods use a voltage of 3.3V - do not exceed this voltage or permanent damage may occur!

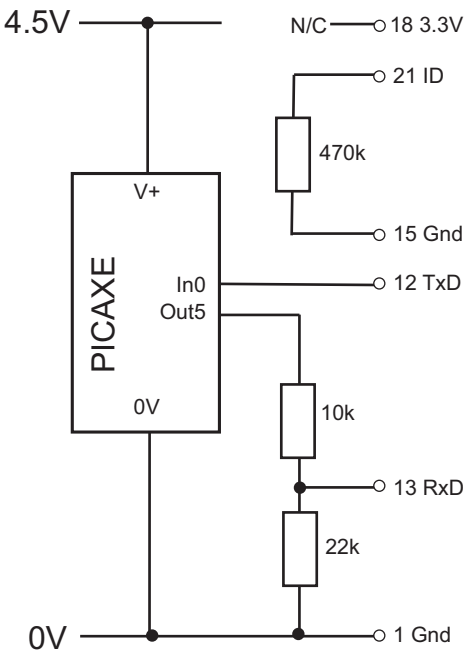
Instructions:

1. Solder the AXE117 kit together. The PCB contains an extra 10k/22k potential divider arrangement at the bottom of the board to allow 3.3V output on output 5, which must be used for the iPod output connection. This connection is marked O5 at the bottom of the board. The extra 10k and 22k resistor are fitted at the bottom of the board (directly above the WWW.PICAXE.CO.UK text). Note on version 1 boards the markings are accidentally incorrectly reversed = 22k must be fitted above the WWW. text and 10k above the .CO.UK
2. Insert a PICAXE-14M microcontroller (purchased separately). ONLY USE a 4.5V or 5V battery pack, not a 9V PP3 battery, as the power supply.
3. Connect the Podbreakout kit to the AXE117:

Pad 1	0V	0V pad at bottom of AXE117
Pad 12	input 0	0 pad directly beside leg 7 of PICAXE-14M
Pad 13	output 5	O5 pad at bottom of AXE117

4. Connect the Podbreakout pads 15 and 21 together with the 470k resistor.
5. Assemble the Podbreakout connector. Note the spring contacts are polarised and only fit one way around. Also note that you may need to file down the corner of the PCB to fit in the case - we have already advised the Podbreakout manufacturer of this design issue.

PICAXE to iPod Connection



Serial Protocol:

Details of the iPod serial protocol are widely available on the internet. Try searching 'Ipod Serial Protocol' on the Google search engine. The following program provides BASIC examples on how to control the iPod using the simple serial control mode, mode 2. More advanced control is available using the AiR mode.

Most third party controllers use a serial protocol of 19200,n,8,1. This is higher than possible with the PICAXE-14M. However as the iPod 'learns' the baud rate from the $FF $55 header on every transaction, in practice any baud rate can be used. These examples use 9600, which is achieved by running the 14M at double speed - 8MHz ('setfreq m8') and using the 't4800' baud rate.

Sample Program & Protocol Explanation:

```
; Serial baud rate - normally 19200,n,8,1 However iPod 'learns' baudrate from the header $FF $55
; Therefore most baudrates can be used - we will use 9600 by running 14M at double speed on 4800.
; Remember all pauses will now be half as long as normal!
; Serial Format
;   Header          $FF $55      always used, not included within checksum calculation
;   Length          $xx          = ? bytes    = number of bytes of mode + command + parameter
;   Mode            $xx          = 1 byte
;   Command         $xx $xx      = 2 bytes
;   Parameter       variable     = ? bytes    = 0 - 250 extra bytes
;   Checksum        $xx          = 1 byte     = $100 - ((L + M + C + P) & $FF)
; Modes
;     $00   mode switching
;     $01   voice recorder
;     $02   simple remote mode <<<< This is what we will use
;     $03   request mode
;     $04   Advanced Remore Mode (AiR)
; Mode switching commands
;     $01   $01   Switch to Voice Recorder Mode
;     $01   $02   Switch to Remote Mode
;     $01   $04   Switch to AiR Mode
;     $03         Get current mode status
;     $04   $xx   Get current mode number
; example, to switch to mode AiR
;    $FF $55 $03 $00 $01 $04 $F8  (checksum $F8 is calculated from $100 - (($03+$00+$01+$04) & $FF)
; For AiR commands information google 'iPod Serial Protocol'

; We will just use Mode 2 for now, simple iPod remote mode
; In this mode send the cmd then a 12ms delay and then send 'button released'
;     $00 $00            Button Released    $FF $55 $03 $02 $00 $00 $FB
;     $00 $01            Play /Pause        $FF $55 $03 $02 $00 $01 $FA
;     $00 $02            Vol+               $FF $55 $03 $02 $00 $02 $F9
;     $00 $04            Vol-               $FF $55 $03 $02 $00 $04 $F7
;     $00 $08            Skip >             $FF $55 $03 $02 $00 $08 $F3
;     $00 $10            Skip <             $FF $55 $03 $02 $00 $10 $EB
;     $00 $20            Next Album         $FF $55 $03 $02 $00 $20 $DB
;     $00 $40            Previous Album     $FF $55 $03 $02 $00 $40 $BB
;     $00 $80            Stop               $FF $55 $03 $02 $00 $80 $7B

; Additional commands found on internet - not tested, checksum not calculated yet,
; may not work with earlier model iPods
;     $00 $00 $01        Play               $FF $55 $04 $02 $00 $00 $01 $xx
;     $00 $00 $02        Pause              $FF $55 $04 $02 $00 $00 $02 $xx
;     $00 $00 $04        Toggle Mute        $FF $55 $04 $02 $00 $00 $04 $xx
;     $00 $00 $20        Next Playlist      $FF $55 $04 $02 $00 $00 $20 $xx
;     $00 $00 $40        Previous Playlist  $FF $55 $04 $02 $00 $00 $40 $xx
;     $00 $00 $80        Toggle Shuffle     $FF $55 $04 $02 $00 $00 $80 $xx
;     $00 $00 $00 $01    Toggle Repeat      $FF $55 $05 $02 $00 $00 $00 $01 $xx
;     $00 $00 $00 $04    iPod Off           $FF $55 $05 $02 $00 $00 $00 $04 $xx
;     $00 $00 $00  $08   iPod On            $FF $55 $05 $02 $00 $00 $00 $08 $xx
;     $00 $00 $00  $40   menu button        $FF $55 $05 $02 $00 $00 $00 $40 $xx
;     $00 $00 $00  $80   OK/Select          $FF $55 $05 $02 $00 $00 $00 $80 $xx
;     $00 $00 $00  $00 $01 Scroll up        $FF $55 $06 $02 $00 $00 $00 $00 $01 $xx
;     $00 $00 $00  $00 $02 Scroll down      $FF $55 $06 $02 $00 $00 $00 $00 $02 $xx
```

```
; Example 1 - Sample test program, to alternate between pause/play every 2 seconds
      #picaxe 14m
init:
      high 5                                          ' initialise output
      pause 10
      setfreq m8                                      ' double speed
main:
      serout 5,t4800,($FF,$55,$03,$02,$00,$01,$FA)    ' send play/pause cmd
      pause 24                                        ' wait 12 ms
      serout 5,t4800,($FF,$55,$03,$02,$00,$00,$FB)    ' send button up
      pause 4000                                      ' wait 2 seconds
      goto main
```

```
; Example 2 - Simple Controller Program
; uses inputs 1 to 4 to achieve different function
      #picaxe 14m
      symbol cmd = b0
      symbol sum = b1
init:
      high 5                              ; initialise output
      pause 10
      setfreq m8                          ; double speed
main:
      if pin1 = 1 then do_play            ; Input 1 - play / pause
      if pin2 = 1 then do_stop            ; Input 2 - stop
      if pin3 = 1 then do_skip1           ; Input 3 - skip >
      if pin4 = 1 then do_skip2           ; Input 4 - skip <
      goto main
do_play:
      cmd = $01
      sum = $FA
      goto btn_up
do_stop:
      cmd = $80
      sum = $7B
      goto btn_up
do_skip1:
      cmd = $08
      sum = $F3
      goto btn_up
do_skip2:
      cmd = $10
      sum = $EB
      ;goto btn_up
btn_up:
      serout 5,t4800,($FF,$55,$03,$02,$00,cmd,sum)    ' send command
      pause 24                                        ' wait 12 ms
      serout 5,t4800,($FF,$55,$03,$02,$00,$00,$FB)    ' send button up
      pause 2000                                      ' wait 1 second
      goto main
```