## PICAXE TRANSVERTER SEQUENCER


(/~vk4adc/web/../23cminfo/100_4383mid.JPG)

Sequencer mounted on the side wall of the 23cm transverter diecast box.
3.5mm socket at left is for interfacing to a serial port on a PC for re-programming.
The white trimpot sets the negative ALC voltage during transmit.
Small-ish black rectangular devices are the SPDT relays.

LHS is the L.O. box with tinplate side shields, heated and heat-shrinked crystal in black shroud,
foreground RHS is the corner of the EME72B transverter PCB.

The 23cm transverter project was under way and I knew that I was going to have to build up a sequencer - but what design would I use ??? I looked at the simple ones like the G3SEK dual relay version and then the various op-amp styles but I think I had this idea right at the start of the 23cm project that I would use a PICAXE as the active sequencing device. At one stage I briefly even contemplated buying a Minikits sequencer kit but figured that this is one bit I could readily build from scratch. Of course the other thing is that now I have a sequencer design, I can incorporate it into any future transverter project - eg 70cm, 13cm etc.

Which PICAXE ? Well the PICAXE-08M (the baby of the family) doesn't really have enough I/O pins and I happened to have some of the 18A series left over. Since it had a reasonable number of I/O's available, that was the way I went. I don't think they are available now having been superseded by the 18M & 18X but there are many other PICAXE versions that you can choose from, with different I/O pin counts and given that they range upwards from about $AUS5 per chip (eg an 08M in single unit quantities), this is a cheap but effective way to develop an accurate programmable time delay sequence. By the way, PICAXEs are available from MicroZed (http://microzed.com.au/) if you are here in Australia.

You may well be asking - what is a PICAXE ??? In real terms, it is a +5 volt-powered "Programmable Interface Controller" (PIC) chip preloaded with a variant of BASIC ("Beginner's All-purpose Symbolic Instruction Code") and looks just like a standard integrated circuit in 8, 14,18, 20, 28 or 40 pin DIP/DIL format. These devices don't require a special PIC programmer device/board (like many others) as there are 2 pins pre-defined for serial input & output. It requires just 2 resistors to make these pins "functionally-compatible" with a standard RS232 PC serial port. That means they are easy to use, debug and re-program. In my case I added a 3.5mm socket on the sequencer veroboard ( see above photo) and I can just plug in the programming lead, drop in a new/revised version of the software, check that it works as desired, disconnect the lead and leave it all alone. No shuffling of chips back from a programmer to the final PCB only to find a software "glitch" and having to swap it back again to re-program, it can all happen in-situ. No need to learn to program in assembly language either. In other words, a lot quicker and simpler than utilising a standard ( read that as dumb ! ) PIC for simple projects. PICAXEs are very flexible BUT you do have to learn to work within the limitations of the BASIC involved.

PICAXE Comparison : This table below shows a comparison between various PICAXE microcontrollers that might be encountered . Note that for this project, the most important column is the "Outputs" count. Only a few lines of program are required so the chip's Memory (lines) capability are largely irrelevant and the much larger devices are simply overkill. My suggestion is that anything from a "14M" (3rd line down in the table) downwards is ok.

!-PICAXE-18184013853L128-progNo-->!-PICAXE-2828802288464+256No--> !-PICAXE-40X40600329-178-203-7128+i2cYes-->!-PICAXE-28A28802288464+256Yes-->

| PICAXE Type | IC Size (pins) | Memory (lines) | I/O Pins | Outputs | Inputs | ADC | Data Memory | Polled Interrupt |
|---|---|---|---|---|---|---|---|---|
| PICAXE-08 | 8 | 40-110 | 5 | 1-4 | 1-4 | 1L | 128-prog | No |
| PICAXE-08M | 8 | 80-220 | 5 | 1-4 | 1-4 | 3 | 256-prog | Yes |
| PICAXE-14M | 14 | 80-220 | 13 | 5 | 6 | 2 | 256-prog | Yes |
| PICAXE-18A | 18 | 80-220 | 13 | 8 | 5 | 3 | 256 | Yes |
| | | | | | | | | |
| PICAXE-18M | 18 | 80-220 | 13 | 8 | 5 | 3 | 256-prog | Yes |
| PICAXE-18X | 18 | 600-1800 | 14 | 9 | 5 | 3 | 256+i2c | Yes |
| PICAXE-20M | 20 | 80-220 | 18 | 8 | 8 | 4 | 256-prog | Yes |
| PICAXE-20X2 | 20 | 1000-3200 | 18 | 1-17 | 1-17 | 0-9 | 256+i2c | Yes |
| PICAXE-28A | 28 | 80-220 | 20 | 8 | 8 | 4 | 64+256 | Yes |
| | | | | | | | | |
| | | | | | | | | |
| PICAXE-28X1 | 28 | 1000-3200 | 23 | 9-17 | 0-12 | 0-4 | 128+i2c | Yes |
| PICAXE-28X2 | 28 | 1000-3200 x4 | 23 | 0-20 | 0-20 | 0-9 | 256+i2c | Yes |
| | | | | | | | | |
| PICAXE-40X1 | 40 | 1000-3200 | 32 | 9-17 | 8-20 | 3-7 | 128+i2c | Yes |
| PICAXE-40X2 | 40 | 1000-3200 x4 | 32 | 0-26 | 0-26 | 0-11 | 256+i2c | Yes |

Some/most of these are now also available in SMD versions (as well as the old DIP/DIL) if you want to incorporate them with update-able firmware into a miniature project.

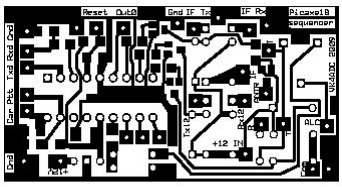
I used the free PICAXE programming editor { from http://www.rev-ed.co.uk/picaxe/software.htm (http://www.rev-ed.co.uk/picaxe/software.htm) , a 38MB download } with it's built-in simulator to debug the simple code I wrote and then set about building up the physical interface. I didn't go to the trouble of creating a PCB - I just used a piece of veroboard that would fit in the available space and built it on that. How you would build another one based on these concepts is up to you. Just one thing - always use a socket for the PICAXE. If perchance you need more output pins in sequence, all you need to do if follow the logic structure and add delays (pauses) and High or Low statements in the TurnTxOn and TurnTxOff sections for these new outputs - all as required. Don't forget to set the initial pin states high or low under "Main"....

The I/O (in this case) to connect to the external world is mainly through relays..... Why relays ??? They are versatile and make it easy to switch positive and negative voltages, RF signals and grounds. I also happened to have some small 12V SPDT style here. Don't think I didn't consider using saturated FETs as switches - and even NPN and PNP low/mid power transistors - but the relays were going to be more flexible in the long run, simpler, smaller ..... If you choose to use semiconductors for the DC switching, that is up to you. This web page is just an "ideas page" and the I/O technology used doesn't really affect the program flow.

{ As a matter of interest, I used a PICAXE plus a relay as a replacement controller for my 3-way fridge (when running on 12VDC) and it uses 2 x DS18B20 1-wire temperature sensors( they actually use 3 wires as you need to supply + volts and ground/common as well as the bi-directional data line) , one inside the cooler section, one on the chimney module, and it does a great job of controlling the fridge temperature. The control program is something like 30 lines of BASIC code and that includes a section to send the internal and external temperature readings via the serial port to a PC-based logging program [ that I specially wrote for this project in Delphi 5 ]. }

---

*POST-CREATION NOTE : I have actually done a single-sided PCB layout using mainly 0805 & 1206 surface mount parts (except for the PICAXE) on the bottom side and the whole sequencer including the miniature relays & ALC voltage adjustment trimpot fits on a board size of just 64mm x 34mm ! I have included extra connection pads so the extra output pins plus the third relay (normally for negative voltage control) contacts are available externally for flexibility and all that is required is to not put the negative voltage components on the PCB. The image below has lost a lot of quality/detail in the creation process but will give the general idea : DIL PICAXE at centre left, first 2 relays just to right of centre, trimpot above the 3rd relay at right. In practice, the through-tracks do not touch any adjacent pads and the larger square pads are for PCB pins to be inserted from the top side. The PTT input has optional pull-up or pull-down resistor positions too.*




The schematic has a number of pins designated and the following table gives the functionality :

| | |
|---|---|
| PTTin : | This pin is pulled high to +5V until a Ground PTT (active-Low) drops it to 0V (logical low) { Limits : 0V to +5V - do NOT apply a +12V PTT connection to this pin }<br>See comments below for changes if you have an active-High PTT function. |
| RFSense : | This pin is pulled low so that if it isn't used then it won't affect program flow. If a RF detector is fitted to the IF port and the DC is fed to thin pin then the T/R switching will be automatic. |
| TXD : | This is a RS232-compatible pin that goes to the TxD pin on the programming computer's serial port. |
| RXD : | This is a RS232-compatible pin that goes to the RxD pin on the programming computer's serial port. |

| | |
|---|---|
| ANTR : | This is a +12V-on-transmit output that feeds to the Minikits EME66 coax changeover relay PCB, noting that it requires +12V in. ( less than about 20mA ) The switched output voltage is in time-sync with the IF changeover relay (ie the IF port / RxIFout / TxIFin connection switching below). |
| +12V : | This is the incoming +12V supply to the PICAXE (via a 5V regulator), to the relay coils and to the RX +12V and TX +12V outputs. |
| TxIFin : | This is the connection to the Tx IF input on the transverter PCB - if necessary use an in-line attenuator to reduce the actual level at the transverter port. |
| RxIFout : | This is the connection to the Rx IF output on the transverter PCB - if necessary use an in-line attenuator |
| IF port : | This is the IF I/O port to connect the external transceiver to the transverter board. |
| TX12 : | This is the delayed +12V transmit power output pin, load about 500mA. |
| RX12 : | The is the partially-delayed +12V receiver power output pin. |
| ALCV : | This is a negative ALC control voltage that is fed back to the transceiver to reduce it's basic RF output power. In receive mode, the voltage is about -4.3VDC. In transmit mode, the voltage is set by the 50K trimpot to anywhere between 0V and -4.3V. In my case, I needed -3.7V to bring the 25W transceiver power back to about 10mW (+10dBm). |
| GND: | Ground pin , ie 12V common ground |

---

*Coding :* The basic PICAXE concept is that it sits in the receive condition and awaits a control signal on either the PTTin or RFSense pins. It then initiates a stepped delay sequence on the output pins. On removal of the PTT or RF sense voltage, the outputs step back at half-delay to the receive condition.

The sequence is Output 0 = high on receive, Outputs 1 through 3 are low. On transmit sequence commencement, Output 0 goes low instantly and then outputs 1 through 3 sequence to high at the selected intervals later. I chose not to use Output 0 to drive a relay and just start off with Output1. I could have just ignored Output0 in the coding - but it was going to be easier later if I wanted to come back later to revise the code in the light of a practical installation.

The line "symbol PTTon=0" is used to indicate that the PTT went to a logic zero for an active PTT (active-Low). If you have a PTT function that goes high on transmit (active-High), you will need to do 2 things - change this line so that it is "=1" and change the 5.6K pullup resistor so that it goes to ground instead of the +5V (i.e. now a pull-down).

Output 7 is toggled at about a 500Hz rate (1mS high, 1mS low) regardless of what else is occurring so acts as a charge pump source for a negative voltage output. The resulting voltage is then used to cut off the transmit RF section of the transceiver during receive and is switched by the last delay output to set the output to RF level acceptable to the transverter during transmit. This should limit tx power over-shoot as the tx PTT has already been activated at least 150mS before. If you don't require the negative voltage generation, simply don't use the components attached to that pin (13) and drop out all lines from the main coding that are "gosub NegGen", change the line "symbol delay0=48" to a value of 50 for a 50mS step period. That will also free up relay 3 for an additional sequencer output function.

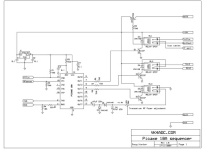
This is a simple project with simple coding. It didn't warrant trying to set up interrupts or anything too fancy - after all it is just a sequencer with non-critical delays. I used only 50mS delay between output pin sequences on going into transmit and about half that (25mS) on returning to receive mode. Those delays are adjustable by changing the symbol "delay0" and "delay1" and re-programming the PICAXE. Some sequencers use much larger delays but when I checked the specifications of both the main antenna changeover relay - and the ones used on this veroboard - all changeover operations were going to be complete within 5-7mS so a 50mS output pin step time seemed appropriate. If you want it to be 100mS per step then change the delay0 value to 98 (since the charge pump subroutine has 2mS delay in it).

Could you use another model PICAXE ??? Certainly - provided it has enough I/O pins for your project (see above) ! If you had an 18M or X series, it would be a drop-in replacement and only require minimal (if any) code changes (and those mainly relate to pre-setting the ADC mode on input 1). The PICAXE editor/programmer software with it's code-trapping would help you find out what you needed to do there.

Before you ask "Why mainly 5.6k resistors ?" This value was about what was needed for logic pullups at 5V and for establishing up to about 1mA of base current in the relay driver transistors - so they would really be saturated with only 20mA (or less) collector current. By all means substitute 4.7K or 6.8K as they really aren't overly critical. The 10K & 22K's in the RxD line are stipulated in the PICAXE documentation. The 0.1uF 50V monolithic caps are for regulator stability and noise considerations. Diodes are placed across all relay coils to kill off back-EMF's.

One advisory : my transceiver's PTT function is open circuit during receive and grounded on transmit - from separate relay contacts. If yours outputs +12V during receive and ground on transmit then you will destroy the PICAXE if you don't modify the PTTin input circuit. How you do it is up to you but the resistor R1 is a pull-up to +5V so you can't just use a super-simple in-line zener arrangement. The input PTTin pin MUST be dropped to 0V and cannot exceed +5V...

Ok, enough chatter. The schematic :


(/~vk4adc/web/images/UserFiles/Image/23cminfo/picaxe 18a sequencer.JPG)

Roll mouse over image to see larger..

If you build one of these up then please let me know.... I will be interested in what changes you make.
If you re-code for a different PICAXE version, email me the pin usage & updated BASIC code file so I can add it to this page.

The 18A code : as a text file (/~vk4adc/web/images/UserFiles/File/23cminfo/sequencer code.txt) that you can paste into the editor - or - as a PICAXE .bas file for the same editor. (/~vk4adc/web/images/UserFiles/File/23cminfo/18a sequencer.bas)

Sorry, the code below has lost it's formatting but you should be able to see the general program flow....

---

```
'BASIC picaxe 18a transverter sequencer
'VK4ADC 13/11/2009 at 11:17:24

; I/O port pin assigments
Symbol PttIn = pin0 ; Pin 17 - PTT input sense pin
Symbol RfIn = 1 ; Pin 18 - ADC input for RF sensing
symbol Seq1out = 0 ; Pin 6 - 12V on receive, 0V in Tx mode
symbol Seq2out = 1 ; Pin 7 - ant c/o relay; high = +12v out & relay operated
symbol Seq3out = 2 ; Pin 8 - 12v to tx low power stages eg tvtr PCB,1w tx amp, bias reg input on PA pcb, 0V in Rx mode
symbol Seq4out = 3 ; Pin 9 - used to control negative ALC voltage

symbol NegGenOp1 = 7 ; Pin 13 - used to generate negative ALC power reduction voltage

symbol PttOn = 0 ; 1 = high, 0 = low - can be reversed depending on PTT voltage sense

symbol delay0 = 48 ; delay ON period between sequence switches - 2ms { in mS }
symbol delay1 = delay0 / 2 ; delay OFF period half the above

symbol RFthreshold = 32



Main:

high Seq1out ; Rx power control
low Seq2out ; Tx op1
low Seq3out ; Tx op2
low Seq4out ; Tx op3
gosub NegGen

PttSense: if PttIn=PttOn then ;if ptt on then start tx sequence
b0 = 1 ; use register b0 as a flag indicating source of tx control 1 = PTT, 2 = RF sensing
goto TurnTxOn
endif



readadc RfIn,b1 ; Read RF sense voltage on Pin 17
if b1 >= RFthreshold then ; use RF sense voltage
b0 = 2 ; use register b0 as a flag indicating source of tx control : 2 = RF sensing
goto TurnTxOn
endif



gosub NegGen ; generate ALC cutoff voltage

goto PttSense ; loop to sense again

TurnTxOn:
low Seq1out ; turn off rx power output
gosub NegGen
pause delay0
high Seq2out ; turn on op1
gosub NegGen
pause delay0
high Seq3out ; turn on op2
```

```
gosub NegGen
pause delay0
high Seq4out ; turn on op3
gosub NegGen
pause delay0

ReSensePtt: if b0 = 1 then
if PttIn<>PttOn then TurnTxOff
endif

LoopRF: if b0 = 2 then
gosub NegGen
readadc RfIn,b1 ; read adc voltage from peak detector
if b1 >= RFthreshold then LoopRF ;loop until RF voltage drops below threshold
if b1 < RFthreshold then TurnTxOff ; if less than threshold then turn off tx
endif

gosub NegGen ; otherwise loop
goto ReSensePtt


TurnTxOff: b0 = 0 ; reset source flag back to zero
gosub NegGen
low Seq4out ; turn off op3
pause delay1
gosub NegGen
low Seq3out ; turn off op2
pause delay1
gosub NegGen
low Seq2out ; turn off op1
pause delay1
gosub NegGen
high Seq1out ; turn on rx power
pause delay1
gosub NegGen
goto pttsense ; return to main ptt sense loop



NegGen: low NegGenOp1 ; this subroutine toggles Pin 13 at 500Hz i.e. 1ms on, 1ms off
pause 1
high NegGenOp1
pause 1
return

---------------------------------------------------------------------------
```