A PICAXE-based Antenna Rotator Controller

17 January 2012

{ This article is not a how-to project but rather a how-I-did-it to show that it can be done provided enough ‘energy’ is attributed to it.}

Mouse-over images for a larger view...

Front view of the finished unit

Rear view

I had been toying with the idea of a PICAXE-based antenna controller for well over a year and even built up a veroboard for a DIL-style 40X chip and attached a 2-line LCD display to it way back then. The relays for rotate right (clockwise – CW) and rotate left (counterclockwise – CCW) were attached as well as one for a brake function. Originally I was going to replace the meter-based controller for my HAM-II rotator with this new unit but other things intervened and the project was left unfinished. I had even written the functions for right, left and brake in the PICAXE BASIC language code and had those working. The thing that brought it to the surface again was my desire to use a physical rotator unit (rather than the Armstrong method) for my field day jaunts.

I have had an old Channel Master TV rotator for 40-odd years and it has been unused and unloved for quite a few of those years. Even so, I suspected it would make an ideal rotator unit for field days if I could get the 50Hz energy to the motors from a 12VDC source. I finally achieved that by utilizing an old computer UPS – and that made continuing with the rotator project practical. The original lack of direction information was accommodated by adding a 3-turn 1K potentiometer inside the base of the rotator body and attached to the output shaft via a 1:1 mechanical coupling. The output shaft turns just one turn but that gives about a 1/3 of the resistance range across the pot’s travel – i.e. about 330 ohms – and as a voltage divider off a 5volt supply, that gives about 1.7v change or 340 A/D units for the 360 degrees rotation. That is nearly 1 A/D unit per degree and that provides a reasonable direction resolution. Given that the time taken for a full 360 degree rotation takes around 80 seconds (at 50Hz), there isn’t too much problem with overshooting the destination angle.

The 3-turn 1K pot was simply mounted on a L-shaped flange and coupled to the through-shaft via an old mechanical coupler. The 8-pin DIN socket is mounted at bottom right to provide the 6 connections now necessary.

A slightly different angle to provide a little more visual info.

I selected the PICAXE-28X1 for the task as I needed enough A/D inputs – at least 2, preferably more – plus enough output pins to drive an LCD display plus any and all relays, and any input switch functions I was going to need. The 28X1 actually has 4 A/D inputs that can be read in 10-bit mode hence giving a possible range of 0 to 1023 for a 5 volt DC range

supplied to the input pin. Using the potentiometer fitted as described above, that is quite adequate. Had I needed to, I could have run the rotator's pot from a higher regulated voltage (eg 10V from a 7810 series regulator) so that I had a larger voltage range to work with.

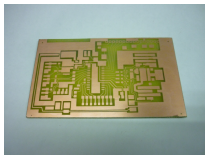
The display was to be a 4 line x 20 character LCD format type – if for no other reason than I already had a couple of these available after a purchase of LED and SMD products from China previously. This device can be operated in 4-bit mode thus reducing the number of signaling pins required of the PICAXE, but still requires 6 actual pins on the available output ports.

The next step was to lay out a PCB. I hadn't actually drawn out a schematic by this stage but I have done enough design work in both electronics and with these PICAXEs to visualize the arrangements, the parts needed (relays, transistors, diodes, pull-up or pull-down resistors and bypass capacitors etc..). The PICAXE I used was a SMD-style and the balance of the parts were SMD in either 0805 or 1206 formats, SOT23 for the relay driver transistors. All connections would be by 0.1" header pin strips. The other factor involved in the layout was that I wanted it to be essentially the same size as the physical LCD display so that the mounting screws for the display would double-function for this PCB too. By utilizing a 0.1" header strip, the connections to the LCD would become a wire-less process too, simply because the LCD uses 0.1" contact spacing. My cross-checking the layout occurs when I actually draw the schematic on paper based on my PCB layout – rather than the other way around.

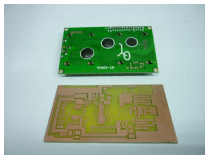
It was around this time that I became aware that a mate of mine had an old well-used Channel Master rotator too, and I was able to talk him into doing a swap for a complete Stolle rotator that had I bought at a Hamfest only a couple of months previously and cleaned up. After the swap, the newly-acquired unit underwent the same conversion : a 3 turn 1K pot plus a 8-pin DIN socket on the side – plus a coat of paint. That gave me the option of controlling two identical-series rotators from the one control unit. The multiple A/D inputs on the PICAXE made it possible to read the two pots in the two rotators simultaneously. The method of selecting which motor/rotator to drive was then to be taken care of by either selecting or de-selecting another 12V relay on the PCB.

I also wanted to be able to drive a relay external to the PCB to power-up the UPS – in fact the relay itself is mounted within the modified UPS (/~vk4adc/web/index.php/field-day-activities/75-fdhardware/156-fdantrot) itself. That might not require much PCB real estate but was another output pin and transistor relay switch required. Beyond that the CW/CCW control was simplified to either relay that was either operated or released but required a further relay that would switch the 24VAC to the motors off until required

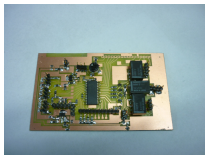
The outcome was a PCB that was as shown below, and was physically around 100mm long and 60mm wide.



The PCB just after etching and trimming to size

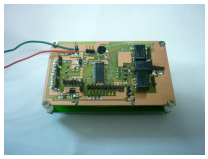


This image doesn't quite show that the PCB is almost identical in size to the LCD



With most of the components mounted. The 0.1" strip connector to the LCD has yet to be fitted.

The 3 x A/D inputs (ROT1, ROT2 and Goto pots) are at the LHS, the bottom left is the 78L05 regulator to feed just the external GPS. The 0.1" strip along the bottom centre is the whole of PortC, those being used as inputs have pull-down resistors fitted. The upper relay selects ROT1/2, the centre one is AC on/off and the lower one is the CW/CCW selection.



The PCB mounts directly onto the LCD display using 3mm tapped nylon spacers. The wire-wrap interwiring was easier to use for the 4 connections than going to a double-sided PCB !

Of course, the electronics is only a relatively minor part of the project. The heart of it is the code inside the PICAXE. I actually started afresh on writing the code, writing subroutines to check which switches were selected, reading the A/D values and converting the results to a degrees value between 0 and 360. These were then incorporated with code to display the output data to the LCD. The basic controller was born. That was just a matter of selecting relays, reading values and showing things on the LCD.

You will note from the photo that there is a “Goto” potentiometer with a switch below that simply says “Goto”. By setting the direction using the knob, the Goto direction shown on the LCD changes between 180 through 0 to 180. Pressing the momentary “Goto” switch starts the antenna rotation process. The most difficult part was to develop a truth table for the logic states to determine which way to rotate the shaft for a given start angle for a given destination angle. I sat out under the back patio for a couple of hours with numerous sheets of paper on which I wrote the letters CW and CCW against originating angles versus targets in other quadrants. (My wife was wondering whether it was possible to develop such a table of logic states – according to the muttering she overheard.) Finally I translated those scribbles into code and got the rotator moving in all relevant directions.

What complicated the code was that the True North on the physical rotator might not actually be TN in the final setup. I need to be able to offset TN in the code and the rotation direction had to take account of this. The Set TN switch on the front allows you to point the antennas at the actual TN, momentarily select the switch and it will then utilize this as TN until the power is turned off or the PICAXE reset. The ROT1 and ROT2 TN settings are separate. One might be set as TN with a 32 degree offset, the other might be at 354 degrees. The software doesn’t really care.. : it gets the new direction/angles right anyway.

That finished the “rotator” part of the software – but – I have a dislike of not making the maximum use of something like the PICAXE and the LCD display. I had in mind that I would attach a GPS receiver (a Garmin GPS-16LVS with serial RS232 & not USB) and sample the output strings to obtain the current time in UTC, plus the latitude and longitude then generate the Maidenhead Grid value. I have to admit that when I did the PCB layout that this was in the back of my mind so I intentionally did not use the HSERIN (pin 18) and HSEROUT pins (pin 17) on the PICAXE. These pins are supported by a set of separate serial routines to those used for loading up the devices with the firmware and for debugging. They allow for higher baud rates and, on receive, a partial background serial process to occur.

I had used these HSERIN & HSEROUT pins previously when I made my Icom auto-antenna switch so it was a matter of reworking the code to look at the 4800 baud NMEA strings and select just the $GPGGA one. That string contains the data I needed : time, lat, long, number of satellites being received, fix quality etc.. These values were saved in EEPROM within the device for later processing and display. I should mention that 28X1 PICAXEs (as well as other members of the family) have a number of numerical limitations :

a. whole numbers only, no decimals
b. no negative numbers are available
c. acceptable values in word variables are 0 to 65535, byte values of 0 to 255
d. a maximum of 13 word variables or 26 byte values can be used at any one time

At times it becomes necessary to save the byte or word values in EEPROM so as to not lose them while you run another section of code. It is slower than just RAM but at least the original or calculated values are available for later use.

It becomes tricky to convert latitude and longitude values such that you can subsequently multiply and divide and not run into negative values or those exceeding 65K !! It actually took me close on two days part-time of writing code that would work within the limitations of the PICAXE maths and still give the correct answer for the grid value. So many times I lost track of the magnitudes and resulted in overflows thus losing the effective values. Finally I was able to crack it with the result that if the serial GPS receiver is connected then it shows the current lat and long on line 2, the time in UTC on line 3 and the 6 character grid following that from the mid-point of line 3. If there isn’t a GPS present, it simply says “no GPS data” on the display for about 5 seconds before going to the normal rotator directions display. If the toggle switch is in the GPS Read position, the unit stays reading the lat, long and time and the calculated grid values. If the switch is in the centre position (off – the normal position for general operation), the current time is displayed in UTC as well as the rotator direction values. I based my conversion on the following Lat/Long to Grid Delphi conversion code :

{CALCULATE THE GRID LOCATOR}

GE := (Long_in_Decimals+ 180) / 20;

TE := (Lat_in_Decimals + 90) / 10;

G1 := trunc(GE);
G2 := trunc(TE);
G1ZZ := alphas[G1+1];
G2ZZ := alphas[G2+1];

GE := (GE - G1) * 10;
TE := (TE - g2) * 10;
G3 := trunc(GE) ;
G4 := trunc(TE);
G3ZZ := nums[G3+1];
G4ZZ := nums[G4+1];

G5 := trunc((GE - G3)*24);
G6 := trunc((TE - G4)*24);
G5ZZ := alphas[G5+1];
G6ZZ := alphas[G6+1];

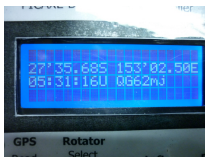
Maidenhead_Grid := G1ZZ+G2ZZ+G3ZZ+G4ZZ+G5ZZ+G6ZZ

Flicking the switch from ROT1 to ROT2 and using the Goto switch allows you to set both rotators to the same direction, one at a time. Eg. set the Goto pot to display 322 degrees, do a Goto with ROT1 active and it moves until it shows 322 degrees (+/- 1 degree). Flick the switch to ROT2 and then the Goto switch again and the second rotator will travel to the 322 degree point too. That makes it easy to set both sets of antennas to the same destination direction, a situation occurring relatively often during VHF/UHF field days as you skip one band to the next to the next while working the one station.. Having it all working from just one controller operating directly from the 12V supply makes for good power economy too.

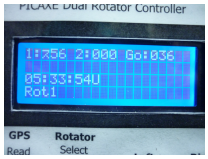
This is the device in the GPS Read mode (note the 2nd toggle switch from the left is up) and the unit is displaying my lat/long plus UTC time as read from the current $GPGGA data stream (The U is appended to remind me that it is in fact UTC.)

The grid locator is calculated from the lat/long info.

The rear view in a little more detail and without the connecting plugs messing up the view. The unmarked 3.5mm socket to the left of the Reset switch is the PICAXE programming/ debugging port. DIN sockets were used as they are probably one of the cheapest multi-pin connectors around and extremely useful for low current applications. The two Rotator connectors are 8-pin, the AC/UPS is a 4-pin, the GPS is a normal 6-pin mini-DIN.

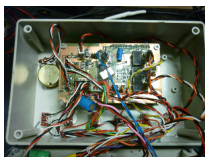


In GPS read mode.

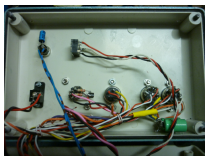


In rotator control mode.

The Rotator1 direction display has incorrect data as there was no actual rotator plugged in at the time, just the GPS receiver.



The inside view shows all of the wiring from the 0.1" header pins to the various sockets and switches.



The inside of the rear panel shows the connector end of the assembly. The green electrolytic visible at bottom right is actually two 105degree ones in series with steering diodes across each - this is in place of a larger bipolar motor style.

The final code uses the PICAXE operating at 16MHz using the external 16MHz crystal so that the A/D functions (etc) and 'Goto' functions are quite quick but actually switches down to 4MHz for the GPS routines because I could not get the serial coding running reliably at 16MHz. As soon as it exits the GPS routine, it switches back up to 16MHz.

My final source code is around 36KB in size and leaves about 200 bytes free in the 4096 byte size of the 28X1. That code (or excerpts) will not be made available to anyone – too much time and intellectual effort was put into writing the code.

My final comment is that "yes, it does indeed work". I took it along on the Summer 2012 VHF/UHF field day (a separate topic) (/~vk4adc/web/index.php/field-day-activities/76-2012fds/157-2012sfd) and once I got the modified UPS (/~vk4adc/web/index.php/field-day-activities/75-fdhardware/156-fdantrot) working (later found to have a loose wire/connection within some heatshrink), both rotators were doing their thing correctly. A subsequent malfunction on ROT1 was discovered to be due to the screws on the mechanical coupler not having been tightened sufficiently and the overall rotator direction display varied less than the normal 360 degrees. Resetting the rotator to its physical North position and then setting the pot back to 500 ohms before the screws were tightened fixed the problem.

I may have used Channel Master rotator heads in my project but any rotator that already has a direction pot inside (and preferably 24V AC motors) is a possible target for this type of project. That list includes the old Stolle I swapped, the CDE Ham-II's and many others. Just because I use the dual rotators doesn't mean that you can't use this type of design with just one unit. During the code development, I often had just one rotator plugged in.