EasyTrak, A PIC Based Rotor/Radio Controller Interface

by Steven R. Bible N7HPR (n-/hpr@tapr.org)

Abstract

EasyTrak is a rotor/radio controller interface based on the Microchip* PIC16F87x series PICmicro~ microcontroller. The goal was to design a rotor/radio interface that is compact, low-cost and easy to use. The PIC microcontroller contains many of the peripherals needed to design a rotor/radio controller interface: analog to digital converter @DC), timers, serial interface (USART), and individually programmable I/O pins. In addition, the PIC contains FLASH ROM for easy programming and upgrades, RAM for system variables, and EEPROM for storage of non-volatile configuration information. With a small amount of interface circuitry around the PIC microcontroller, the goals of compact and low-cost were obtained. Ease of use was obtained by designing E3asyTrak to interface to the most popular azimuth/elevation rotors and a serial RS-232 interface to communicate with virtually any computer.

EasyTrak is designed to easily interface to the Yaesu series of azimuth/elevation rotors, model numbers G5400B, G5600, and the newest G5500. These rotors have a computer interface built in with an g-pin DIN connector on the back of the rotor controller to control left, right, down, up and provide rotor position via a proportional O-5 VDC analog signal for azimuth and elevation. EasyTrak can be interface to other rotors, for example rotating HF antennas, but will require user modification of the rotor controller. Optional relays can be installed on EasyTrak to provide normally-open (NO) or normally-closed (NC) contacts for azimuth, elevation, and brake.

The computer interface is via a serial RS-232 connection. The rotor controller protocol is based on Chris Jackson's, G7UPN, EasyComm protocol. EasyComm is a simple ASCII character based protocol for controlling rotors and radios. The benefit was that a new protocol did not have to be created, it is easy for programmers to write for and interface to, and several programs already have EasyComm programmed in: WiSP, Nova, MacDoppler, MacAPRS and WinAPRS. The Mac and WinAPRS programs have the unique feature of tracking high and low altitude balloons in azimuth and elevation as well as other objects[2]. Mac and WinAPRS can also point HF beams using the DX Cluster feature or point to a moving object in azimuth only.

Introduction

This project started out several years ago with the goal of designing a low-cost rotor/radio controller interface. EasyTrak was designed primarily with satellite operation in mind. It had to interface easily to the most popular azimuth/elevation rotor system, the Yaesu G5400B/G560OB and G5500 series. The Yaesu rotor controllers have a built in computer interface which supplies a proportional O-5 V analog signal that corresponds to the rotor position (azimuth and elevation) and four command lines left, right, down, and up (when grounded, command the rotors in the respective directilon). EasyTrak also had to control the most popular satellite radios. However, EasyTrak is not limited to satellite operations only. It can easily be configured and interfaced to other rotor controllers but requires the user to modify the rotor controller.

Central to the EasyTrak design is the Microchip P1Cmicro.a The PIC was a natural choice with the plethora of integrated peripherals in a single integrated circuit package. Integrated peripherals reduce the

amount of external circuitry required and help keep costs down. The PICmicro chosen for the EasyTrak project was the PIC16F87x series. The F87x series has a maximum of 8 K FLASH ROM, 368 bytes of RAM, and 256 bytes of EEPROM. The flash ROM provides an easy method to program and upgrade the firmware (verses EPROM) and the EEPROM a means to store configuration values. Onboard peripherals of interest for designing a rotor controller include an eight channel lo-bit Analog to Digital Controller (ADC), one 16.bit and two 8-bit timers, and Universal Synchronous/Asynchronous Receiver Transmitter (USART) for serial communications to a host computer. The PlCmicro is the center of the hardware design, but a protocol was needed to control the rotors and radios.

It was important not to create another rotor controller protocol. There already exist several rotor controllers (commercial, kit, and homebrew), each with its own unique protocol. This complicates the software authors' job in keeping up with multiple protocols. What was needed was a protocol that was simple and provided the necessary basic commands.

EasyComm Protocol

The protocol chosen was Chris Jackson's, G7UPN, EasyComm[3] (thus the origin of selecting the name EasyTrak). Chris is the author of the popular digital satellite program WiSP (Windows Satellite Programs) and developed and implemented the EasyComm protocol for those who wish to design their own rotor and radio controllers. Given Chris' experience supporting a variety of rotor interfaces in WISP, the EasyComm protocol should contain the basic commands necessary to control rotors and radios. As a bonus, EasyComm is already implemented in several popular tracking programs such as WiSP, Nova, MacDoppler, and most recently, Mac and WinAPRS.

During the development of EasyTrak, it became evident that the EasyComm protocol was a good choice. It shielded the software author from knowledge of the rotor or radio. EasyTrak handled the translation from EasyComm command to rotor movement or radio control. For example, to move the azimuth rotor to 270 degrees, the following command is issued:

```
AZ270.0J
```

Where 'Z-J" can be a space, carriage return, or linefeed (the space allows multiple commands to be placed on one line). It is EasyTrak's responsibility to safely and accurately position the azimuth rotors to the commanded position. An example radio frequency command is:

```
UP145900000↵
```

This will command the uplink frequency to 145,900,OOO Hz (145.900 MHz). The software author does not need to know what radio model is commanded. EasyTrak handles all of the details. However, it became apparent that comiguration commands would be necessary to tell EasyTrak about the rotors and radios interfaced to it. Thus the decision was made to extend the basic EasyComm protocol with configuration commands.

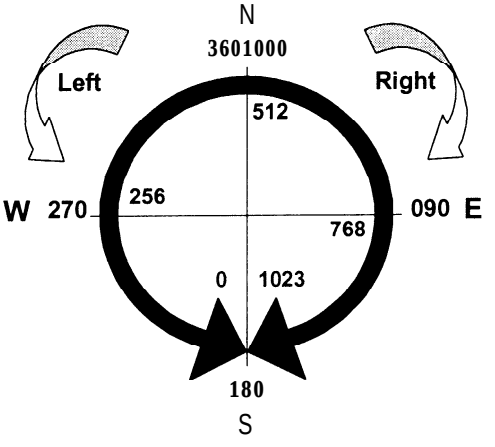
The goal of the extending EasyComm protocol for configuring EasyTrak required that the same basic command structure be followed (two letter command followed by a command value) and that it should not require the software author to implement the extended commands in order to configure EasyTrak. EasyTrak can be configured using a simple terminal program, and once configured, can be interfaced to any tracking program that supports the EasyComm protocol. The goal was to use the fewest commands

necessary to configuration EasyTrak. Attachment A summarizes the EasyComm 2 and extended commands supported by EasyTrak.

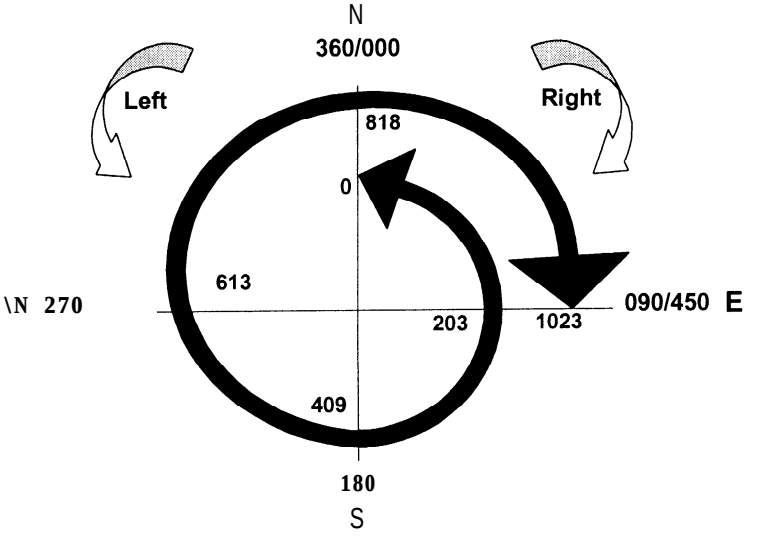
Rotor Configuration

The first rotor configuration command, RO, specifies the type of rotor combination. There are four choices: A = Azimuth only, E = Elevation only, C = AZ/EL Combination, or N = None (no rotor) (default setting). This command allows EasyTrak not to be tied to azimuth/elevation rotors only.

The second rotor configuration command, RS, specifies the physical stop of the rotor. There are two choices: S = South physical stop (such as the Yaesu G5400B/G5600B) or N = North physical stop (such as the Yaesu G5500). This command is necessary for EasyTrak to compute the physical position to the compass direction. The following diagrams explain why this is necessary:



The above diagram illustrates a south physical stop. The inner values represent the lo-bit ADC value that is read from the proportional O-5 V analog signal supplied by the rotor controller. The outer values represent the compass directions.



In contrast, the above diagram illustrates a north physical stop (with 90 degree overlap) of the Yaesu G5500 rotor. Again the inner value represents the approximate lo-bit ADC value read from the proportional O-5 V analog position signal supplied by the rotor controller. The outer values are the

compass directions. These two types of physical rotor configurations require a different mathematical conversion. Once EasyTrak has been configured with the rotor type and physical stop, the rotors must be calibrated.

Rotor Calibration

First and foremost, the rotors must be assembled and calibrated according to the manufactures instructions. Once this has been done, the rotors can be calibrated to EasyTrak. The first step is to set the maximum O-5 V analog position signal voltages (azimuth and/or elevation) coming from the rotor controller.

The CA command continuously reads and displays the ADC value for azimuth and elevation. This command precludes the need for a voltmeter. The display looks like:

```
AZ ADC = 127   EL ADC = 64
```

The numbers are the raw binary values (0.. 1023) read from the azimuth and elevation ADC channels. This number represents the actual physical position of the rotors. The user manually positions the azimuth rotor in the fully clockwise (right) position and adjusts the azimuth output voltage until the ADC value reads between 1020 and 1022. Do not set the voltage for a reading of 1023. That is the maximum value of the ADC and represents 5 V or greater.

To calibrate the elevation rotor, first determine if antenna-flip capability (0 to 180 degrees) or not (0 to 90 degrees) will be used. Manually position the elevation rotor to 180 or 90 degrees. For 180 degrees the above azimuth procedure applies. Manually position the rotor to 180 degrees and set the elevation ADC value between 1020 and 1022. For 90 degrees, manually position the rotor to 90 degrees and set the ADC value to 5 12 (half the maximum ADC value).

Calibration Limits

The second calibration step is to establish the limits of travel for the azimuth and elevation rotors. There are four calibration limit commands: Calibration limit Left (CL), Right (CR), Down (CD), and Up (CU). The rotor is manually positioned to the associated limit and the calibration limit command is entered with the rotor position in degrees. EasyTrak will record the ADC binary value associated with the position. If the calibration limit command is queried, the degrees and ADC value are displayed in the format CUd,n where d = degrees and n = ADC3 binary value. For example, if the calibration limit up is queried, the response is:

```
cu90,5 12
```

This example response displays the calibration limit up position is 90 degrees and the ADC binary value associated with it is 5 12.

The calibration limit commands provide an elegant method of calculating the linear travel of the rotors. The majority of rotors have 360 degrees of travel. However, the Yaesu G5 500 azimuth/elevation rotors present a unique challenge with 450 degrees of travel (an additional 90 degrees of overlap between 0 and 90 degrees). The azimuth rotor is manually positioned to 450 degrees and the command CR450 is entered.

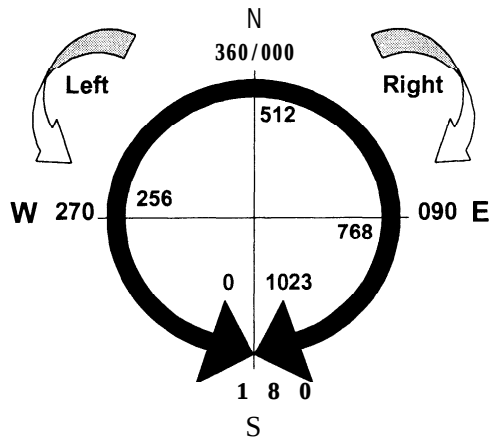
The calibration limit method also provides an elegant method of recording the minimum and maximum travel limits of the rotor. For example, if the rotor model is a Yaesu G5400B (which as a south physical stop) and the rotor installation has a physical obstruction between 180-270 degrees, a left most calibration limit can be entered. First, manually position the azimuth rotor to the desired left most limit, in this example, 270 degrees. Enter the command CL270. EasyTrak will not position the rotor any further left than 270 degrees to the physical stop at 180 degrees (to the left).

Design Implementation

The discussion up to now has been on the operation of EasyTrak. This section explains some of the technical details and design implementation of EasyTrak.

To position the rotors the commanded position is compared to the actual rotor position. If the rotor position is less than the commanded position, command the rotors to move right. If the rotor position is greater than the commanded position, command the rotors to move left. When rotor position is equal to commanded position command the rotors to stop. This is a simplified algorithm that in practice is a little more involved to implement. The rotors are positioned according to a proportional analog voltage that is digitized by the PIC ADC. This value has to be converted to and from degrees to be useful to tracking program.

The rotor indicates position with a proportional DC voltage. The Yaesu azimuth/elevation rotors use a linear O-5 VDC signal to indicate actual rotor position. 0 VDC indicates the left most position of the rotor and 5 VDC the rightmost. For example, the Yaesu G5400B rotors (which have a south physical stop) use 0 VDC to indicate 180 degrees (to the left) and 5 VDC 180 degrees (to the right). Recall the diagram:



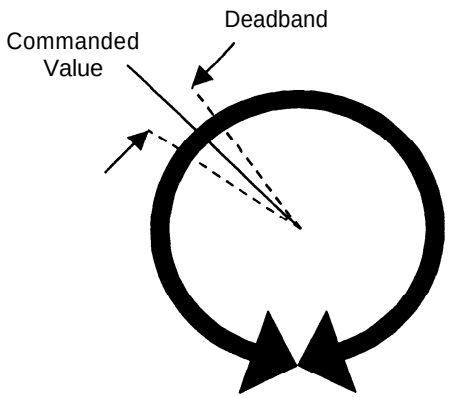
The PIC ADC converts O-5 VDC into a lo-bit binary value between 0 and 1023 decimal. Effectively the ADC divides 5 volts into 1023 units. The resolution is:

$$\frac{5 \text{ volts}}{1023 \text{ units}} = 4.9 \text{ millivolts per unit}$$

The range of O-5 VDC represents 360 degrees of rotation. The calculated volts per degree is:

$$\frac{5\ volts}{360\ \deg} = 13.8\ millivolts\ per\ \deg$$

The 13.8 millivolts per deg of rotation is a relatively small value. The rotor position voltage can be upset by noise (EM1 on the sensing line and in the ADC). Plus there are mechanical limitations (gear slop) of the rotor and when the rotors stop moving there is a coast down period. Therefore, a boundary value must be specified to avoid the controller from continuously commanding the rotors left and right (chatter). This boundary value is called the deadband and it is a value (in degrees) specified by the user when configuring EasyTrak. The following diagram illustrates:



For example, if the commanded rotor position is 3 15 degrees and the deadband value is 5 degrees, the actual rotor position can be in the range of 3 10 to 320 degrees. This range appears to be a large value, but in practice, the rotors fall close to the commanded value. The deadband value should be set to a value that reduces the amount of rotor movement. This is an important property that deadband provides.

In satellite tracking applications, the tracking software is continuously sending azimuth and elevation position commands. The deadband specifies how often the rotor position will change. If the value is too small, the rotors will move often causing increased wear and tear on the rotor motors. If the value is too large, antenna pointing will not be optimal. The largest factor affecting the choice of deadband value is the antenna beamwidth. The positioning error budget is a combination of factors that can add or cancel each other out. These factors include:

Antenna Beamwidth
Rotor Gearlash
Nonlinearities in the position potentiometer
ADC Error
Electrical Noise

PIC Programming

Now that the operational and technical aspects have been determined, the next step is to write the firmware routines for the PIC. The C programming language was chosen for the size and complexity of the EasyTrak project. The basic program structure uses an interrupt service routine to buffer the serial data from the host computer and an infinite main loop comparing the commanded to actual rotor position and issuing the appropriate move and stop commands.

The interrupt service routine (ISR) buffers the serial data from the host computer. The ISR allows the main loop to continuous check command to actual several times per second. When a complete command has been received, it is placed in a buffer and a received command flag is set.

The infinite main loop checks if the received command flag is set for a new antenna position. If one has been received, the ASCII command value is converted to binary and compared to see if it falls between the calibration limits. Then the command value (in degrees) is converted to the corresponding ADC binary. All comparisons are done in ADC binary values.

The antenna position routine first determines the low and highband ADC values from the command ADC and deadband values. The following code segment illustrates:

```
lowband = cmdAZadc - AzDeadbandAdc;

if (bit test(lowband, 15)          // lowband less than zero?
   lowband = 0;                    //  correct

highband = cmdAZadc + AzDeadbandAdc;

if (highband > 1023)               // greater than max?
   highband = 1023                 // correct
```

Once the low and highband values are determined, the rotor position is read:

```
antAZadc = readADC(AzChan);        // read antenna AZ ADC value
```

It is compared to the low and highband values to determine if the rotors should be commanded to move: Notice that the actual rotor position is compared to the deadband limits. If the actual rotor position falls inside the deadband limits, the rotors do not move. It is only when the actual rotor value fall outside the deadband limits that the rotors are commanded to move:

```
if (antAZadc c lowBand) {          // if antenna < lowband
   output_high(cmdRight);          // command right

} else if (antAZadc > highBand) { // else if antenna > highband
   output_highkmdleft);            // command left
›
```

Once the rotor is moving, it is desired that they stop in the center of the deadband. Therefore, the actual rotor position is compared to the command value:

```
if (antAZadc >= cmdAZadc) {        // if antenna >= cmd
   output_low(cmdRight) ;          // command stop
1

if (antAZadc <= cmdAZadc) {        // if antenna c= cmd
   output_low(cmdLeft);            // command stop
}
```

The above routine works for small antenna arrays and rotors that stop in a short amount of time. In practice, the Yaesu azimuth/elevation rotors stop very quickly. The above routine would have to be modified for larger antenna arrays and rotors that take a finite amount of time to coast to a stop.

The only difference between the azimuth and elevation routines is that azimuth needs to be corrected for north or south physical stop.

Radio Control

Radio control is not as complicated as rotor control. A frequency or mode command is received and sent according to the computer interface protocol of the radio. EasyTrak cannot change frequency bands on dual band radios. Very few (if any) have change band commands. Therefore, it is up to the operator to manually configure the radio prior to operation. Otherwise the radio will ignore the frequency command.

EasyTrak is configured by telling it the model radio used for the up and downlink. EasyTrak has two radio ports. This allows radio separates to be controlled for the up and downlink and it allows the control of dissimilar radio models. For example, a Kenwood TS711 can be configured to port 0 as the Mode J uplink radio (URTS711 ,O) and an ICOM IC475 can be configured to port 1 as the downlink radio (DRIC475,l). If a port is not specified in the radio configuration commands, it defaults to 0.

If the radio is a dual band model, the port value is the same for the Up and Downlink. For example, to configure an ICOM IC-821 dual band radio to port 0, the following commands would be entered:

```
URIC82 1 ,O
DRIC821,O
```

Conclusion

Design and development continues on the EasyTrak rotor/radio controller. Presently there are 8 beta units in test. Once testing is completed and finally design decisions are made, it is hoped to provide EasyTrak as a TAPR kit. Be cautioned, the information presented in this paper is subject to change as the design continues to develop. To follow EasyTrak developments see http://www.tapr.org/-n7hpr/easytrak/.

Acknowledgements

Many thanks to the EasyTrak development team who have contributed immensely to the EasyTrak design (in alphabetical order):

Don Agro, VE3VRW
Steve Dimse, K4HG
Dan Huton, VA3DH
John Koster, W9DDD
Lou McFadin, WSDID
Doug McISinney, KC3RL
Stacey Mill, W4SM
Keith Sproul, WU2Z
Paul Williamson, KB5MU

References

’ Microchip Technology, Inc. http://www.microchip.com

[2] Keith Sproul, WU22, and Mark Sproul, KB2IC1, *WinAPRShMacAPRS and .Automatic Rotor Tracking qf’A4oving Objects,* in this proceedings.

[3] The EasyComm 1 and 2 standard is documented in a text file “easycomm.txt” accompanying the WiSP satellite program available from http://www.amsat.org/amsat/ftpsoft.html#win-wisp

Trademarks and Copyrights