Azimuth & Elevation
Rotor Controller For LEO Satellites

A Program For The CRES Amateur Radio Club
August 15, 2007
Azimuth And Elevation Rotor Controller

- Actually it can be used for any satellite
- But my interest is the LEOs
  - There are a lot of them
  - A few of them still work!
Motivation

- I had been working a few of the LEOs with vertical antennas.
- I was interested in a better antenna system
- And I wanted a neat Microcontroller project!
- It seemed like a good idea at the time
Satellite QSOs Are Interesting!

- There are a lot of “things” involved in working the LEO satellites!
  - Computer screen
  - Keyboard
  - Mouse
  - Downlink frequency
  - Uplink frequency
  - Doppler effects
  - Code paddles or a microphone
  - Azimuth of the satellite
  - Elevation of the satellite
So Many things – So Little Time!

- The window for a QSO is often less than 8 minutes.
- If you can automate a few “things” your QSOs may be more enjoyable.
- This project is about automating the rotors for beam antennas.
My Approach To The Project

- Research the WEB for similar projects
- Evaluate what I might do that is different
- Keep it (relatively) cheap!
- Use an Atmega series controller
- Must have an LED display & flashing LEDs
A Few LEO Orbits
Satellites Vary In Both Size & Complexity

Japan’s FUJI Satellite

Suit Sat

N-Cube2
10x10x10 CM
1 LITER VOLUME
(University Projects)
What you Get With a U100 Rotor

www.normsrotorservice.com
Enough Introduction!

- What do these rotors look like?
The ability to put a pipe through the rotor body is fairly unique.
Anatomy Of A U100 Rotor #1
Anatomy Of A U100 Rotor #2
Anatomy Of A U100 Rotor #3

Physical stop tab

Pulser Cam
Anatomy Of A U100 Rotor #4

- Motor shaft Gear
- Motor Frame
- Pulsing contact
- Mechanical Stop
Anatomy Of A U100 Rotor #6
Anatomy Of A U100 Rotor #7
The Original U100 Rotor Schematic Diagram
deg/pulse = 360/# pulses counted

tics/deg = tics/pulse / deg/pulse

**High level strategy**
1. Absolute calibration at a pulse.
2. Interpolate between pulses to estimate position of rotor to a finer degree of resolution.
3. Time between pulses to detect problems.
4. Do an initial calibration to detect rotor characteristics.
Schematic For a Yaesu G800DXA Rotor – Potentiometer Type

Variable voltage DC motor

Potentiometer position feedback

Note: Control box not shown
A Case For Micro Controllers

Control box for the Yaesu rotor
Rotor Controls

2X16 BACKLIGHTED LCD

ON-OFF-ON

Indicates rotor pulse
A Look Under The Hood

- Controller Board
- Programming header
- Xfmr for controller board
- Fuse holder
- PWR cord connector
- Serial in from PC
- Serial out for debug
- Rotor power
- Rotor wires plug in here.
- Phasing caps/SS relay boards
Partial Schematic of the Rotor Controller System

- **Controller's power supply**
- **Front panel LED**
- **Azimuth Rotor**
- **Elevation Rotor**
- **Solid State Relays (opto isolated)**
- **15 VAC xfmr**
- **20 VDC**
- **Current source for Backlight on LCD**
- **Current source for Backlight on LCD**
- **VCC**
My Development Environment

Fedora Core 5 LINUX
With GNU tool chain

program flash memory //port

Debug data
Serial port

Rotor control cable
A Few Statistics

- **ATMEGA 16 Controller**
  - 16KBytes Flash memory
  - 512 Bytes of EEPROM
  - 1 K SRAM

- **Software Sizes**
  - Program 13394 Bytes
  - Data 262 Bytes – Initialized read only data
  - BSS 399 Bytes – initialized read/write data
  - Total 13995 Bytes

- 30 source files
  - All source is written in “C”

- GNU Tool Chain
Things Left Undone

- Need to get a complete schematic in electronic form
  - Scattered around in a notebook now
- Finish the front panel
  - Print another template and put plastic over it
- Need to paint the box
- Test with other Pulser rotors (AR-22)
  - Mainly for azimuth rotor use
  - Motor power requirements may not be compatible
- Adapt to “Potentiometer” type rotors
  - Made some accommodations but didn’t finish this
Closing Thoughts

- Most of my controller projects were easy
  - I had to work at this one!
- Pulsers are difficult
  - Many error conditions to consider
  - Most of the complexity of the code is due to this
- From a Software perspective the potentiometer types seem less complicated
  - Always know where the rotor is at all times
  - No interpolation required
  - No directional history needed
  - Less opportunity to get out of sync.
  - Tracking software may do most of the work for you
- A GREAT controller project!