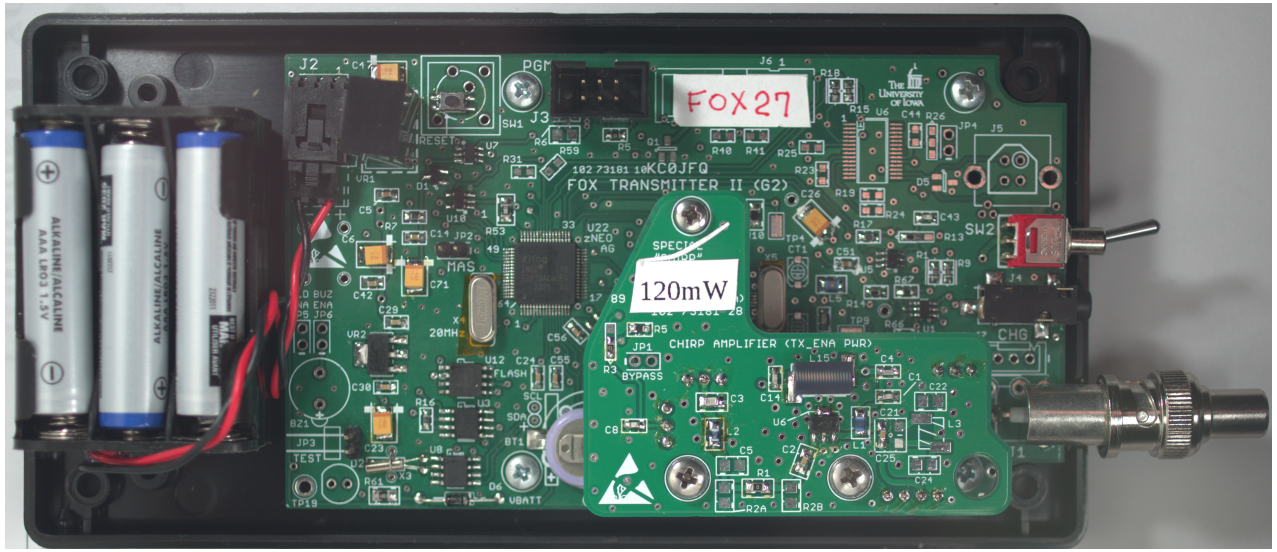


The ICARC FOX Transmitter System

High Function Transmitter System

KC0JFQ: William Robison

http://n952.ooguy.com/HamRDF/index.html
http://icarc.org/icarc_foxhunt.htm

File: fox˙present˙10.tex

April 9, 2025

Iowa City Amateur Radio Club

Outline

The FOX HUNT

Capabilities

Scheduling

Nominal Schedule

Nominal Sounding Schedule

Chirp Schedule

Stooges Schedule

Last Slide!

Iowa City Amateur Radio Club

I hide 'em, you hunt 'em!

This project started (A long time ago, in a galaxy far far away) to enable quick setup of a large hunt with minimal effort expended at the hunt venue.

Large *skulk* of **Fox Transmitters** that are simply switched on as they are set out for the hunt. A bumped power-switch **must not** be allowed to cause timing problems (or otherwise interfere with the hunt).

Feature-rich command language to define the *personality* of the **Fox Transmitters** as heard by the hunter.

Iowa City Amateur Radio Club

Capabilities

Review of the feature set

provided in this **Fox Transmitter System**

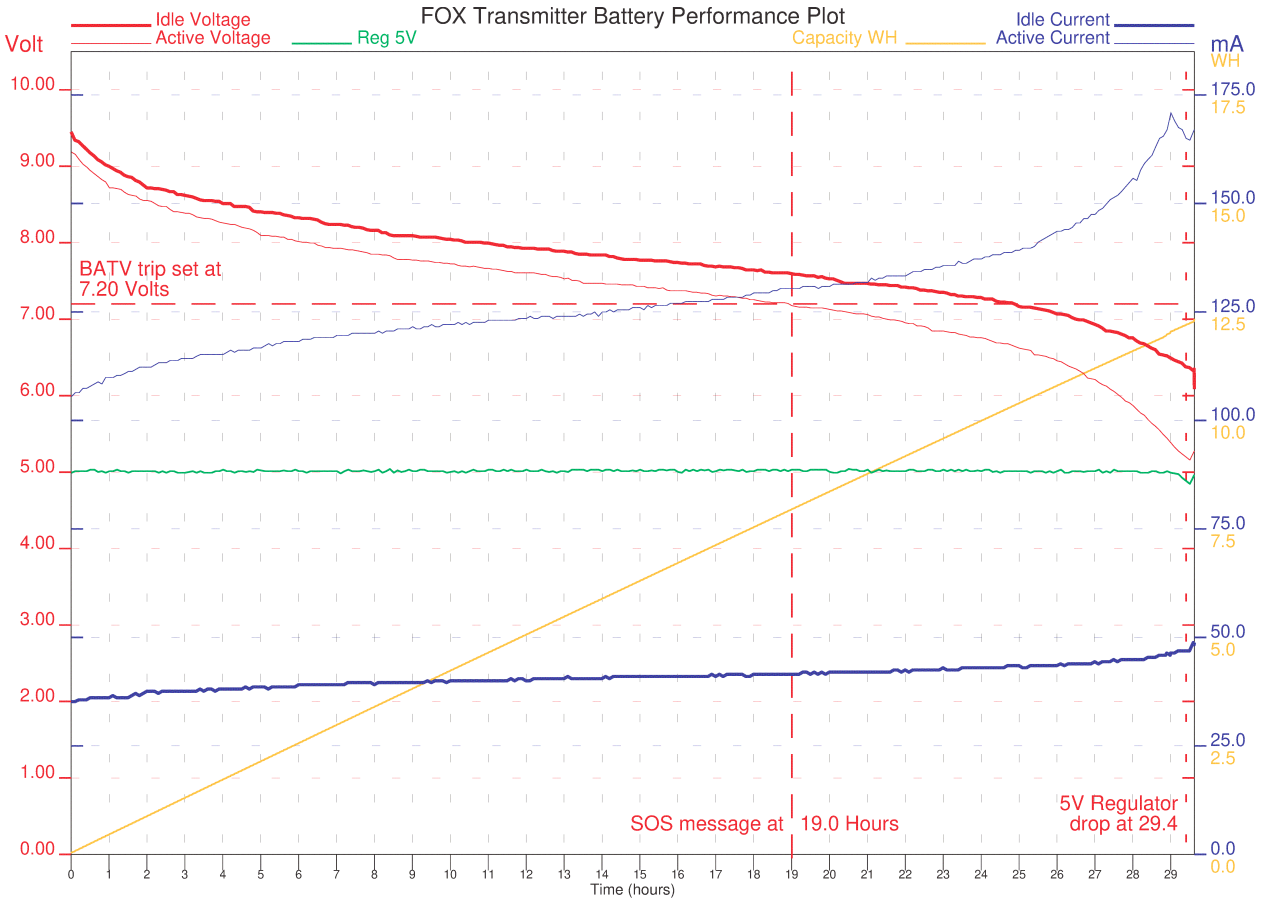
Self Contained Fox Transmitter

- **Highly Programmable** Uses a *Sequencing Control Language (plain text)*
- **Synchronous Operation** Uses a *TOY* clock (Time Of Year) for synchronization
- **Transmitter Controller** Will also control an external tranceiver
  - **PTT** Open Drain
  - **Audio** Tone and Voice, line level
  - **Serial Control** 3.3V CMOS levels
- **Battery Power** 7 Volts to 24 Volts
  - **6 AAA internal** Low cost alkaline cells
- **Low Power**
  - **Around 100 Milliwatt** Simple RF Amplifier
  - **Up to 1 Watt** RF Module SA818/DRA818
- **Simple Hunt Programming Hardware**
  - **Serial Cable** (USB to serial cable from FTDIchip)
  - Command Sequences and Audio Files

Iowa City Amateur Radio Club

Power

Running on a six-pack of AAA



V/I measured by the fox transmitter (**BATR** command)
Measured when idle and when transmitting
Logged as text file on host system
Plot utility on host system

Iowa City Amateur Radio Club

Scheduling

Review of the Scheduling Model

How we determine the **WHAT** and **WHEN**

Sequencing Language creates the FOX Transmitter *PERSONALITY*

implements the way the transmitter behaves

the way the transmitter sounds (i.e. what it sends over the air)

Scheduling (or how to conduct a conversation with yourself)

Transmitter uses TOY clock to keep track of time

Scheduler may have up to 10 *schedules* defined

These schedules are individually enabled

Scheduler uses *modular arithmetic* (simple division using the remainder as a result)

MODS S0=360,30 MODS S0=period,offset

period *in seconds* is the cycle time

so, 360 means we send message every six minutes

offset *in seconds* is offset into the period

The 30 here means we start 30 seconds into the 360 second period

Iowa City Amateur Radio Club

Time in the FOX Transmitter

Time Management Review

Remember the **%** is the remainder of a division operation.

System Time Nominally using the UNIX EPOCH, but think of seconds from midnight

- 32 bit integer that counts **system_seconds**.
- 8 bit integer that counts sub-seconds (**RTI** is 100 ticks/second).
- TOY clock loaded into SRAM by command (**TIME**) in the **INI=** script.
  - Time zone offset (**EPOC** command) is set after **TIME** command
- RTI interrupt updates **RTI** and **system_seconds**.
  - **RTI** runs 0 to 99 .
  - When **RTI** hits 100, increment seconds and set **RTI** back to ZERO.

Schedule Time (**MOD**ular **S**chedule)

- **period** 16 bit integer count in seconds
- **offset** 16 bit integer count in seconds **offset** must be less than **period**

Scheduler (software in the zNEO)

- **now = system_seconds % 86400** get the **now** variable down to within 1 day

Do the following test when **RTI** is zero.

- **if((now % period) == offset)** If true, it's our turn to run!

Iowa City Amateur Radio Club

Scheduling features

Several commands also use this scheduling model

That is to say they operate in a synchronous manner

WAIT Command

causes execution of the sequence to stall

WAIT 10 wait 10 seconds (non-synchronous)

WAIT 10,0 wait for *0==(now%10)* to occur (synchronous)

CHRP Command

Enters a *wildlife transmitter* emulation

Typical command: **CHRP 1.0, 5, 0.8, 10**

1st. arg: Audio tone frequency (1.0KHz)

2nd. arg: period (integer seconds)

3rd. arg: offset (integer seconds)

4th. arg: Audio tone duration (decimal seconds)

5th. arg: repeat count (integer)

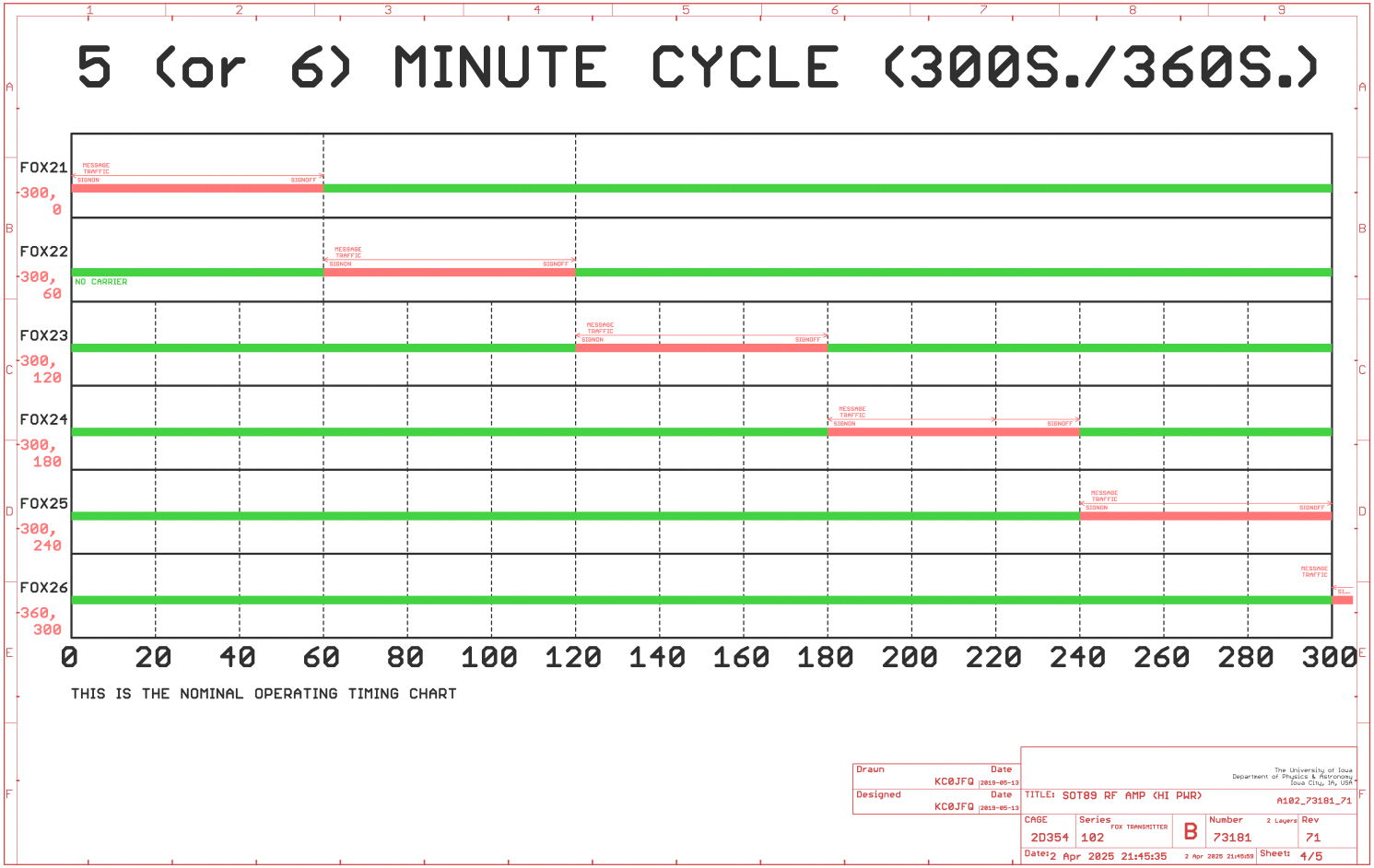
Synchronous commands provide the capability to do fine-grained synchronization.

Members of the hunt group not limited to waiting 300 out of 360 sedconds.

Iowa City Amateur Radio Club

Nominal Scheduling Chart (144.325 MHz)

Timing for the nominal schedule



RED bar indicates transmitter active

GREEN bar indicates transmitter idle

Iowa City Amateur Radio Club

Schedule S0=1

Listing 1: FOX2X_KC0JFQ_S0

```
#                                                                              111
#       Operating Schedules                                                    112
#       The power draw of the DRA818/SA818 may cause the                       113
#       "BATC" form of the report to take too long, so we                      114
#       ALWAYS use the vocal report.                                           115
#       ALSO We're making use of the <CALL> and <NAME> substitution            116
#              inside the fox transmitter!!!                                   117
#                                                                              118
esav S0=BATR                                                                   119
esav S0=CONF -AM                                                               120
esav S0=TONE 1.0                                                               121
esav S0=CWPM 30,-1,-1,-1,-1                                                    122
esav S0=BEGN                                                                   123
esav S0=TALK <CALL>                                                            124
esav S0=TALK <NAME>                                                            125
esav S0=WAIT 0.5                                                               126
#esav S0='batvc' V 7.2                                                         127
#esav S0=BATC EV 7.2                                                           128
#esav S0=WAIT 0.5                                                              129
```

Schedule S0=2

Listing 2: FOX2X_KC0JFQ_S0

```
#                                                                    130
#   Fill time so they have a chance of finding me                    131
#                                                                    132
esav S0=TONE 'tone'                                                  133
esav S0=CWPM 25,-1,-1,-1,-1                                          134
esav S0=WAIT 0.15                                                    135
#esav S0=CODE hi hi hi                                               136
esav S0=BATC EV 7.2                                                  137
esav S0=WAIT 0.5                                                     138
esav S0=CODE IOWA CITY                                               139
esav S0=CODE AMATEUR RADIO                                           140
esav S0=CODE CLUB FOXHUNT                                            141
esav S0=CODE F W KENT PARK                                           142
#esav S0=CODE MARCH 30 2025                                          143
```

Schedule S0=3

Listing 3: FOX2X_KC0JFQ_S0

```
#                                                                          144
#                                                                          145
# Prepare (kinda...) for Signoff                                           146
#    these extr REM- commands can be deleted (EZER)                        147
#    and replaced with more CODE commands to adjust time...                148
esav S0=REM-                                                               149
esav S0=REM-                                                               150
esav S0=REM-                                                               151
#                                                                          152
# Signoff                                                                  153
#                                                                          154
esav S0=BATR                                                               155
esav S0=TONE 1.0                                                           156
esav S0=CWPM 30,-1,-1,-1,-1                                                157
esav S0=DONE                                                               158
#                                                                          159
```

Initialization Fragments

Listing 4: FOX2X_KC0JFQ_INI

```
esav INI=TIME                                                    43
esav INI=WAIT 0.5                                                44
esav INI=TIME                                                    45
esav INI=EPOC -5.0                                               46
esav INI=NAME 'name'                                             47
esav INI=CALL 'call'                                             48
```

Callsign and Nickname defined in the initialization file that runs at power-on

The shell script that loads the fox transmitter replaces the *'name'* and *'call'* with **FOX-something** and **W0JV**.

```
esav INI=FREQ 144.150                                            55
```

ALL transmitters start out transmitting on 144.150MHz.

```
esav ANN=FREQ 'freq'                                             105
```

Change to operating frequency at end of signon message

INI Report

Initialization Report

```
sts01,00* ************************************************************                1
sts01,01* KC0JFQ  FOX Transmitter  V3.95                                                2
sts01,02*   Z16F2811FI20SG                                                             3
sts01,03* Tools Ver: 20230510  <ROM:26576 EROM:89918 Flash:116494>                      4
sts01,04* Flash Prog 04.26.03 MB85RS512T    FLASH_FRAM       Fujitsu   512K-bits        5
     2048-records CMD3
sts01,05* Flash WAVE 9D.7E.FF    25LD040    FLASH_PAGE          ISSI   4096K-bits       6
     62-seconds CMD4
sts01,06* ************************************************************                7
sts01,07* Run INI=                                                                      8
sts12,00* Handler_TIME (cmd_time.c*) Sys:00000000 TOY:FFFFFFFF   0.02 Sec               9
sts27,00* Handler_WAIT (cmd_message.c*) TIMER_delay_ticks(50);  0.49 Sec               10
sts12,00* Handler_TIME (cmd_time.c*) Sys:000F8994 TOY:000F8994 Dly:350mS  1018259.00   11
     Sec
sts15,00* Handler_EPOC (cmd_time.c*) 68400   0.01 Sec                                  12
sts17,01* Handler_NAME (cmd_code.c*) FOX17 (5 chars)  0.01 Sec                         13
sts16,01* Handler_CALL (cmd_code.c*) W0JV (4 chars)  0.01 Sec                          14
sts10,00* Handler_CONF (cmd_conf.c*) 0x1F0C43 SI5351  0.02 Sec                         15
sts10,00* Handler_CONF (cmd_conf.c*) 0x1F0C43 SI5351  0.01 Sec                         16
sts10,00* Handler_CONF (cmd_conf.c*) 0x1F0C43 SI5351  0.01 Sec                         17
sts21,57* Handler_FREQ (cmd_frequency.c*) 144.150=139E,EE47F,F4240                     18
sts21,01* Handler_FREQ (cmd_frequency.c*) 144.150 MHz slot=262  0.09 Sec               19
sts47,00* Handler_BATR (cmd_battery.c*) T=1018260.020 V=7.323[02EF] I=25.9[0035]       20
     5=0.605[003E]  State=T0   0.03 Sec
```

Iowa City Amateur Radio Club

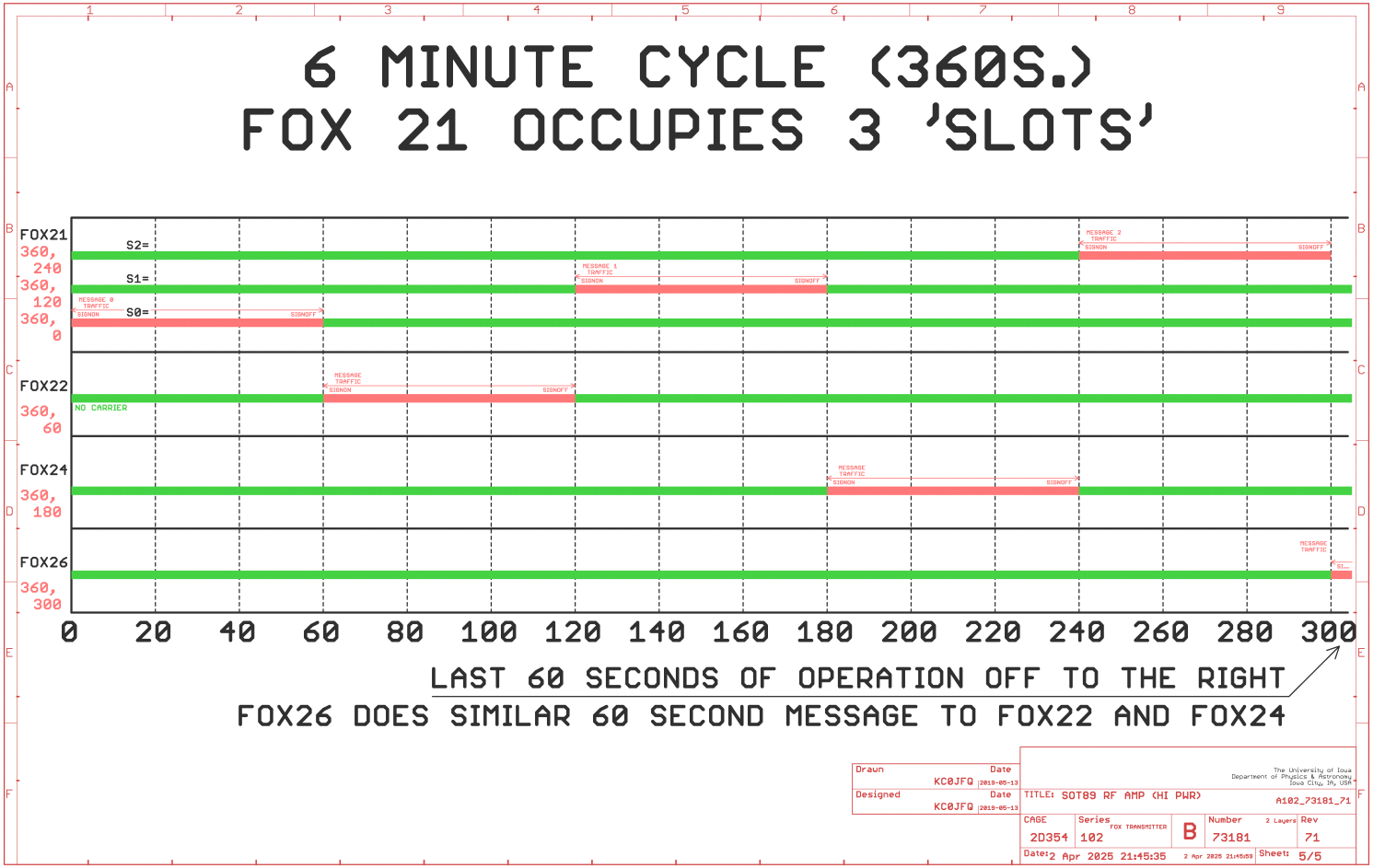
S0 Report

```
sts47,00* Handler_BATR (cmd_battery.c*) T=1018928.011 V=7.333[02F0] I=25.9[0035]          2
     5=0.624[0040]  State=T0   0.03 Sec
sts10,01* Handler_CONF (cmd_conf.c*) 0x1F0C43 SI5351  0.01 Sec                            3
sts19,01* Handler_TONE (cmd_code.c*) Audio 1.000KHz   0.01 Sec                            4
sts20,00* Handler_WPMR (cmd_code.c*) 30 (1 3 7 14)  0.01 Sec                              5
sts24,02* Handler_BEGN (cmd_message.c*)  144.150MHz                                       6
sts24,20* Handler_BEGN (cmd_message.c*) 14:02:08.190  "e.. CQ CQ CQ DE W0JV" BEGN         7
     8.47 Sec
sts26,00* Entry_AUDIO_ (cmd_voice.c*)  TALK=W0JV,56960                                    8
sts26,00* Handler_TALK (cmd_voice.c*)    1.54 Sec                                         9
sts26,00* Handler_TALK (cmd_voice.c*)    0.08 Sec                                         10
sts27,00* Handler_WAIT (cmd_message.c*) TIMER_delay_ticks(50);  0.49 Sec                  11
sts19,01* Handler_TONE (cmd_code.c*) Audio 1.000KHz   0.01 Sec                            12
sts20,00* Handler_WPMR (cmd_code.c*) 25 (1 3 7 14)  0.01 Sec                              13
sts27,00* Handler_WAIT (cmd_message.c*) TIMER_delay_ticks(15);  0.14 Sec                  14
sts30,23* Handler_BATC (cmd_battery.c*) BATC HI HI TTTTTTT EEE  6.78 Sec                  15
sts27,00* Handler_WAIT (cmd_message.c*) TIMER_delay_ticks(50);  0.49 Sec                  16
sts25,00* Handler_CODE (cmd_code.c*) "IOWA CITY "  OK  4.29 Sec                         17
sts25,00* Handler_CODE (cmd_code.c*) "AMATEUR RADIO "  OK  5.34 Sec                     18
sts25,00* Handler_CODE (cmd_code.c*) "CLUB FOXHUNT "  OK  6.20 Sec                      19
sts25,00* Handler_CODE (cmd_code.c*) "F W KENT PARK "  OK  5.44 Sec                     20
sts04,00* Handler_REM (cmd_help.c*)  0.00 Sec                                             21
sts04,00* Handler_REM (cmd_help.c*)  0.00 Sec                                             22
sts04,00* Handler_REM (cmd_help.c*)  0.01 Sec                                             23
sts47,00* Handler_BATR (cmd_battery.c*) T=1018967.056 V=7.333[02F0] I=27.3[0038]          24
     5=0.624[0040]  State=T3   0.02 Sec
sts19,01* Handler_TONE (cmd_code.c*) Audio 1.000KHz   0.01 Sec                            25
sts20,00* Handler_WPMR (cmd_code.c*) 30 (1 3 7 14)  0.02 Sec                              26
sts29,17* Handler_DONE (cmd_message.c*) 14:02:47.620  "DE W0JV SK SK SK "  45.710        27
     6.29 Sec
```

Iowa City Amateur Radio Club

Nominal Sounding Schedule

Hunt Group setup that sounds like we are using 6 transmitters



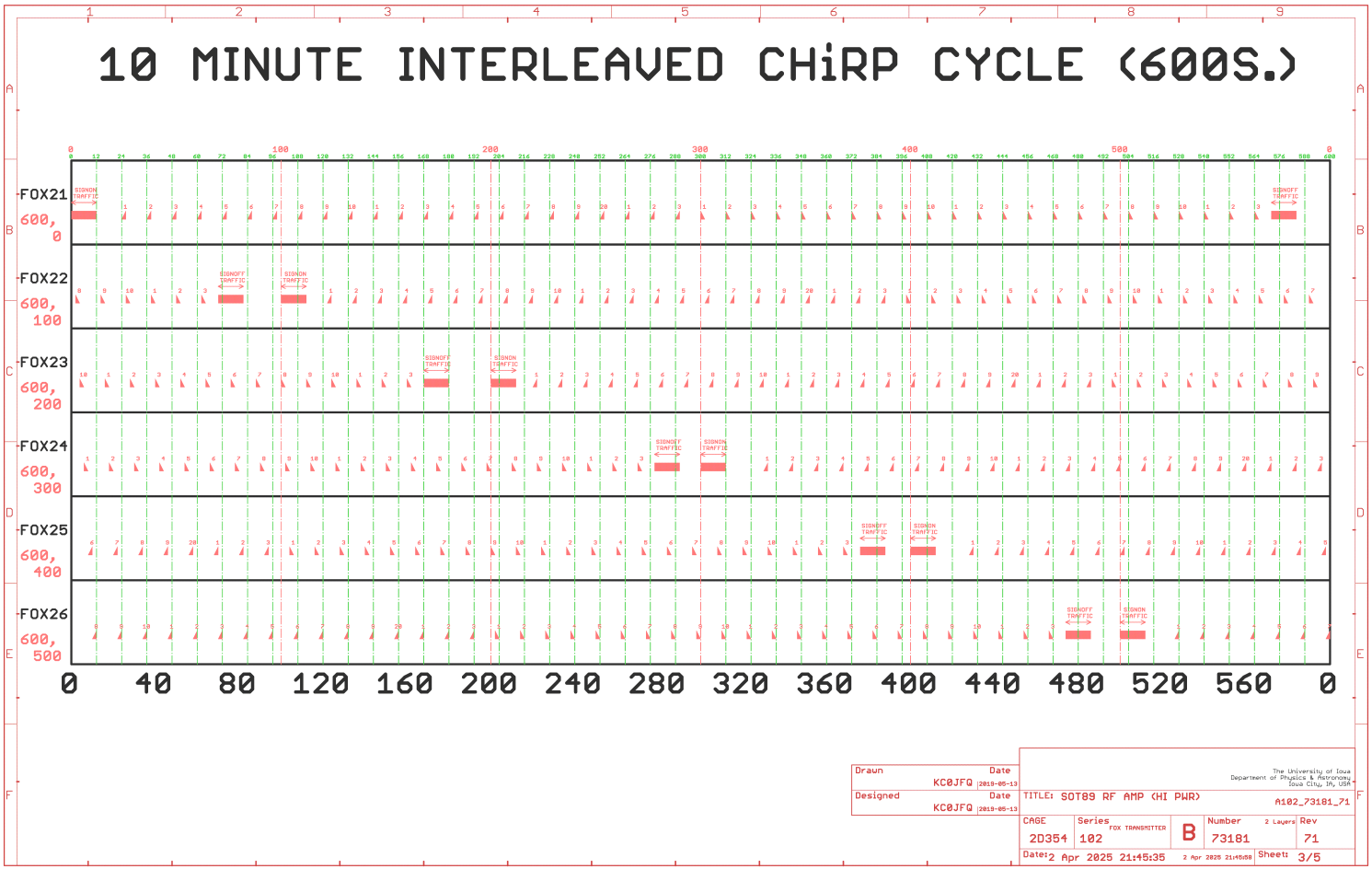
Some of this shows up on 144.250MHz

5 transmitters on 6 minute period, one does double duty

Iowa City Amateur Radio Club

CHiRP Scheduling Chart (144.225 MHz)

Hunt Group setup with the **CHRP** command



Iowa City Amateur Radio Club

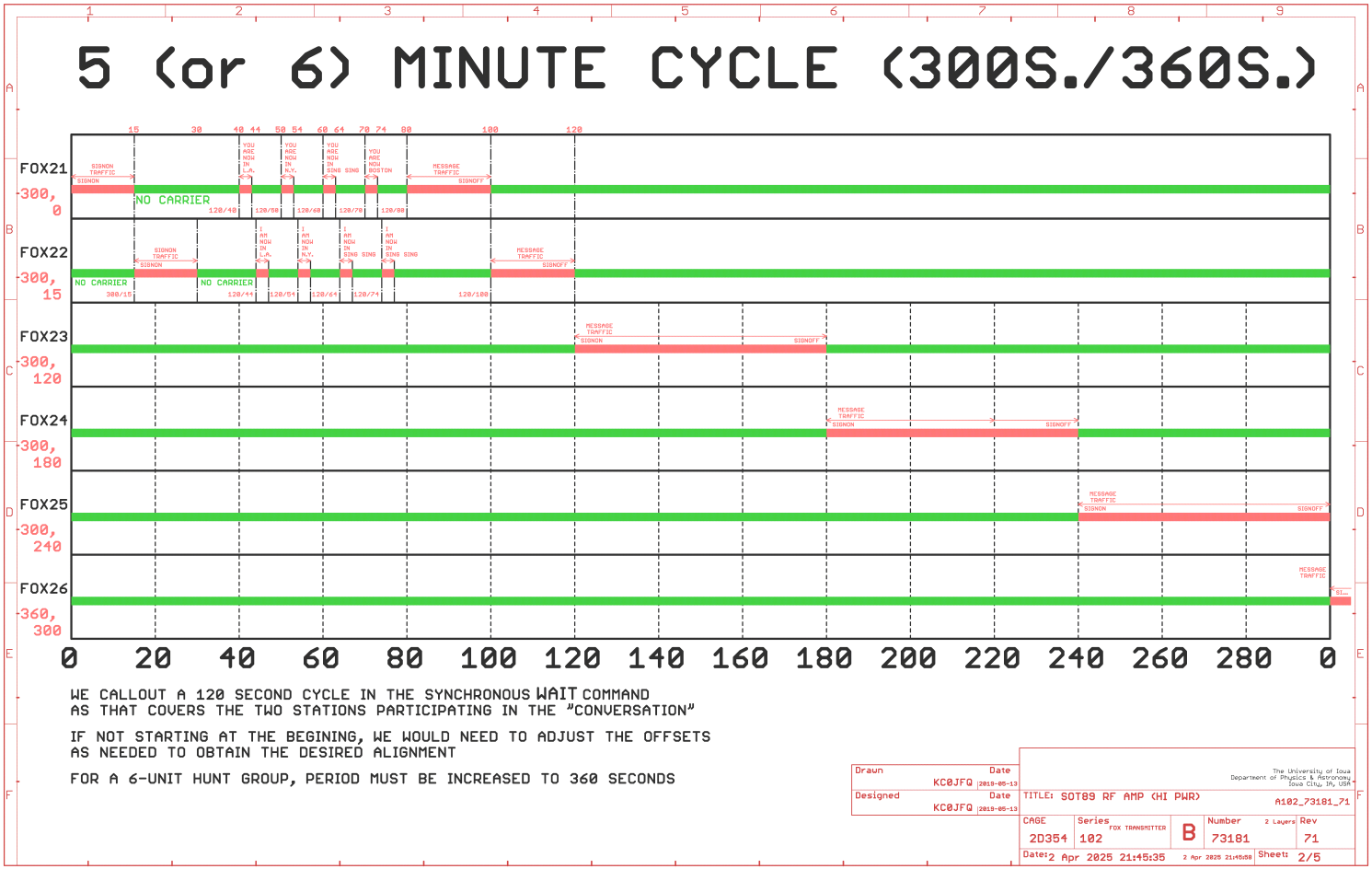
CHiRP S1

```
#                                                                          162
#  Wildlife signature (30 second period)                                   163
#       ALSO We're making use of the <CALL> and <NAME> substitution        164
#              inside the fox transmitter!!!                               165
esav S1=TONE 1.0                                                           166
esav S1=CWPM 30,-1,-1,-1,-1                                                167
esav S1=BEGN                                                               168
esav S1=TALK <CALL>                                                        169
esav S1=TALK <NAME>                                                        170
esav S1=CONF CW                                                            171
esav S1=CHRP 'chrpfrq' 'chrp1' 0.30 55                                     172
esav S1=CONF -CW                                                           173
esav S1=TONE 1.0                                                           174
esav S1=CWPM 30,-1,-1,-1,-1                                                175
esav S1=DONE SILENT                                                        176
#                                                                          177
```

Iowa City Amateur Radio Club

Stooges Scheduling Chart (144.250 MHz)

Two units play out a bit from the 3-Stooges



"You are now in Sing Sing"

Iowa City Amateur Radio Club

Stooges 8

```
esav S8=TONE 1.0                                    7
esav S8=CWPM 30,-1,-1,-1,-1                         8
esav S8=CONF -AM                                    9
esav S8=BEGN                                        10
esav S8=TALK <CALL>                                 11
esav S8=TALK <NAME>                                 12
esav S8=CONF CW                                     13
esav S8=CODE SING SING                              14
#                                                   15
esav S8=WAIT 'stooge'/40                            16
esav S8=TALK TS1_LA                                 17
esav S8=WAIT 'stooge'/50                            18
esav S8=TALK TS2_NY                                 19
esav S8=WAIT 'stooge'/60                            20
esav S8=TALK TS3_SING                               21
esav S8=WAIT 'stooge'/70                            22
esav S8=TALK TS4_BOSTON                             23
#                                                   24
esav S8=WAIT 'stooge'/80                            25
esav S8=CONF -AM                                    26
esav S8=TONE 1.2                                    27
esav S8=CODE BEEN LISTENIN TO                       28
esav S8=CODE THE THREE STOOGES                      29
#                                                   30
esav S8=TONE 1.0                                    31
esav S8=CWPM 30,-1,-1,-1,-1                         32
esav S8=DONE                                        33
```

Iowa City Amateur Radio Club

Stooges 9

```
esav S9=TONE 1.0                                    37
esav S9=CWPM 30,-1,-1,-1,-1                         38
esav S9=CONF -AM                                    39
esav S9=BEGN                                        40
esav S9=TALK <CALL>                                 41
esav S9=TALK <NAME>                                 42
esav S9=CONF CW                                     43
esav S9=CODE SING SING                              44
#                                                   45
esav S9=WAIT 'stooge'/44                            46
esav S9=TALK TS1R_LA                                47
esav S9=WAIT 'stooge'/54                            48
esav S9=TALK TS2R_NY                                49
esav S9=WAIT 'stooge'/64                            50
esav S9=TALK TS3R_SING                              51
esav S9=WAIT 'stooge'/74                            52
esav S9=TALK TS3R_SING                              53
#                                                   54
esav S9=WAIT 'stooge'/100                           55
esav S9=CONF -AM                                    56
esav S9=TONE 1.6                                    57
esav S9=CODE BEEN LISTENIN TO                       58
esav S9=CODE THE THREE STOOGES                      59
#                                                   60
esav S9=TONE 1.0                                    61
esav S9=CWPM 30,-1,-1,-1,-1                         62
esav S9=DONE                                        63
```

Iowa City Amateur Radio Club

Last Slide!

Radio Programs Active at the moment...

Assuming someone remembers to turn things on!

FOX27 .. FOX32 **S0** on 144.325

FOX2X_KC0JFQ.fox Standard from slide 9

This is what a more-or-less normal hunt sounds like

FOX21 .. FOX26 **S6** on 144.225

SX_CHIRP.fox CHiRP Schedule from slide 17

Interleaved operation. Effectively a 6 second period

Demands TOY clocks be within 100mS or so to work!

FOX33 .. FOX38 **S8/S9/S0** on 144.250

SX_STOOGE.fox Combination from from slides 16 and 19

Only 5 units operating with a 6-minute period

One unit transmits two messages!

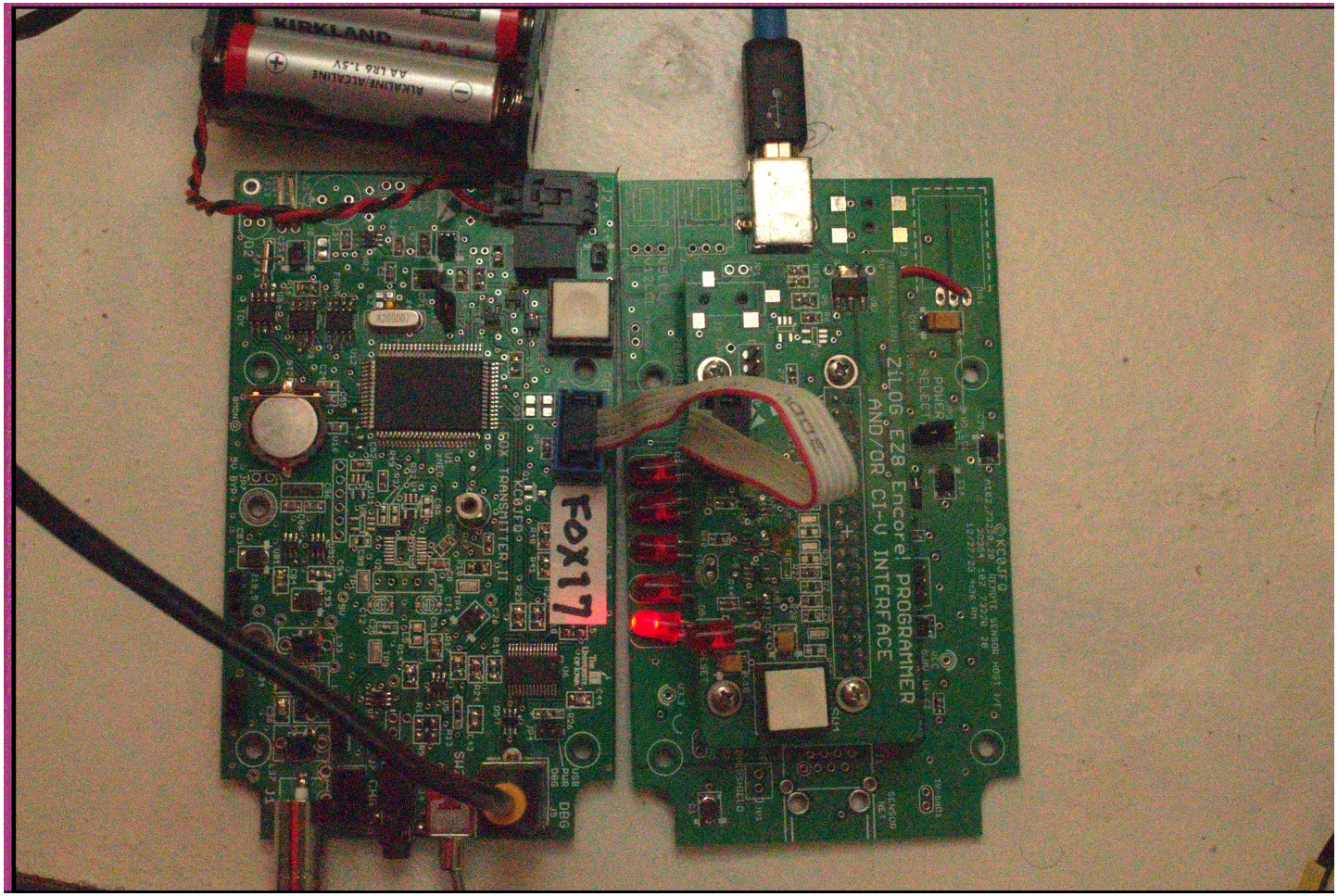
FOX20 **S0** on 144.285

FOX2X_KC0JFQ.fox from slide 17

Training Unit, almost continuous message traffic

Iowa City Amateur Radio Club

Software Prototype



Iowa City Amateur Radio Club

Listing 5: TOY Update Report

```
Decode Last Detail Record                                                    1
CMD: sort -t"," -k4,4 -k1,1 < foxlog.csv > sorted_foxlog.csv                 2
Report Todays Loads                                                          3
"FOX21": 2025-Apr-09T10:35:41,"V3.90","W0JV","FOX21",144.225,"S6",8.40-V,23-mA   4
      ,8.05-V,114-mA,-5Hrs
"FOX22": 2025-Apr-09T10:34:09,"V3.90","W0JV","FOX22",144.225,"S6",7.68-V,42-mA   5
      ,7.22-V,142-mA,-5Hrs
"FOX23": 2025-Apr-09T10:36:53,"V3.90","W0JV","FOX23",144.225,"S6",7.95-V,54-mA   6
      ,7.60-V,152-mA,-5Hrs
"FOX24": 2025-Apr-09T10:35:01,"V3.90","W0JV","FOX24",144.225,"S6",8.62-V,34-mA   7
      ,8.32-V,117-mA,-5Hrs
"FOX25": 2025-Apr-09T10:33:31,"V3.90","W0JV","FOX25",144.225,"S6",7.82-V,40-mA   8
      ,7.17-V,141-mA,-5Hrs,***Battery***
"FOX26": 2025-Apr-09T10:32:53,"V3.90","W0JV","FOX26",144.225,"S6",8.72-V,30-mA   9
      ,8.38-V,116-mA,-5Hrs


"FOX38": 2025-Apr-09T10:25:23,"V3.93","W0JV","FOX38",144.250,"S0",9.24-V,24-mA  20
      ,9.02-V,101-mA,-5Hrs
CMD: rm sorted_foxlog.csv                                                    21
UPDATE Labels: fox_label.csv                                                 22
CMD: mv fox_label.csv fox_label.csv-2025_Apr_09T10:36:53                     23
```

Iowa City Amateur Radio Club

Scary Notes for the Presenter

PAY NO ATTENTION TO THE MAN BEHIND THE CURTAIN!

Notes for the terminally forgeful

Here we go again!
-Dolly Parton

Reformat the PDF, expanding it to about 80% of the page:

```
gs -sDEVICE=pdfwrite \
        -dDEVICEWIDTHPOINTS=499 \
        -dDEVICEHEIGHTPOINTS=634 \
        -dCompatibilityLevel=1.4 \
        -dNOPAUSE \
        -dBATCH \
        -dPDFFitPage \
        -sOutputFile=fox_Print_10.pdf \
        fox_present_10.pdf
```

Reformat PDF file to PS file for printing

```
pdftops -paper letter \
        fox_Print_10.pdf \
        fox_Print_10.ps
```

Test Print

Verify Margins (for 3-hole punch)

Verify orientation (running from 3-ring binder???)

```
lp -d HP_LaserJet_M209 \
        -o portrait \
        -o media=letter \
        -o sides=two-sided-long-edge \
        -P 1,2 \
        fox_Print_10.ps
```

Print it with the presenter notes (at the end)

```
lp -d HP_LaserJet_M209 \
        -o portrait \
        -o media=letter \
        -o sides=two-sided-long-edge \
        fox_Print_10.ps
```

Before we begin

Please feel free to ask questions.

I'll try to give a good overview of what we have, how it came to be, and how it works.

If you've read ahead, and find yourself here, these are my **Presenters Notes**.

I will stop talking at the last slide, number 22, and take any questions about anything I seem to have missed.

Above, are my *cheat sheet* notes about how to render the presentation so that I can print it out and have these notes fill the page they are printed on.

This is all rendered using LaTeX to produce the PDF that is being presented. This rendering only fills up about 1/4 of the page when printed, and that's too small for me to read quickly! These commands render and print the presentation.

As you may infer from this, the entire **Fox Transmitter** project was built without touching a Windows machine, it's all built under Linux.

Some of the ZiLOG tools (i.e. the C compiler and linker) are Windows based, but the WINE emulator allows them to function as command-line tools under Linux. The software that runs the **Fox Transmitter** is assembled/compiled using a standard *Makefile*. Now to really antiquate myself, I use a text editor (JOE) with WordStar key-bindings and function (column mode is particularly useful!).

In the early days (like up until about February) I was using a ZiLOG Ethernet programmer to program the flash in the zNEO. A Fedoraq upgrade about then completely borked the interface to that device so it got pitched and replaced by the programmer shown in slide 23.

That programming board provides an open-collector half-duplex serial interface to the zNEO. ZiLOG documents the protocol details in the zNEO *Product Specification* that allowed for creating the programming software.

If you obtained this from the ICARC website, you should have noticed a link titled: **ICARC FOX zNEO System Users Manual** *(FOX_ICARC.pdf) FoxTx Bible*. That will give you access to the detailed System Users Manual that talks about the entire suite of hardware and software.

The **Fox Transmitter** is described in gory detail. Not only hardware, but also the software that resides in the zNEO program flash.

All of the utility programs used to prepare a fox hunt are also described in this manual.

I have tried to automate as much of the preperation as I can. I think I'm finally at the point where I run a shell script to update the **Fox Transmitter**s TOY clock and record the battery condition, all from the host system.

I plug each transmitter, in turn, into the host system while the setup script is running and the script updates the TOY clock and captures the battery voltage.

After that, all the labels and cards can be updated and printed that are needed for the hunt.

The day of the hunt, all I have to do is drop the transmitters in their hiding spots then sit back, relax, and watch the show unfold!

The FOX HUNT Slide 3

Quick summary of why these transmitters exist and what we expect out of them. This is, after all, the end result of a hobby project, probably taken to extremes:-)

It all started out with the early revisions, trying to come up with something that worked well and in sufficient numbers to run a basic hunt.

Initial target **was** 5 or 6 units. This was achieved with the 102-73161-25 board which is the third hardware revision.

A group of foxes is called a skulk. Other names that can also be used include troop, leash, and earth.

All setup occurs the night before. Typically all that is required is updating the time (stored in the TOY clock) and replacing a battery pack or two.

Sometimes the commands loaded through the serial port are updated or the audio file-system is updated.

Command loads are fast, typically taking a minute or two.

Audio file-system loads are huge and are, therefore, slooooooow. Painfully slow... Agonizingly slow...

Load format is a plain-old Intel HEX formatted text file. A more-or-less universal format that is static.

zNEO software occasionally updated but requires a device programmer. The zNEO programmer is a local design. It uses a USB UART, some buffers, and yet more software... The hardware to do this is shown on slide 23 at the end of the presentation.

The processor, a zNEO from ZiLOG has a relatively large program FLASH (at 128KB) and a tiny SRAM (at 4KB). The zNEO looks like a *Harvard Architecture* machine to the compiler, so code generation is relatively clean.

The big program FLASH allows room for a feature-rich command decoder. The software in this FLASH also provides the timing features to allow multiple units to share a common channel.

The operating software provides extreme flexibility, well as much as will fit in the FLASH.

We should be able to easily emulate any exiting commercial or hobby-level **Fox Transmitter**.

Capabilities Slide 4

Continuing with the for transmitter discussion from the previous slide.

Typical FOX HUNT is a hunt-group (or 5 or 6 transmitters) operating one-after-the-other (never at the same time).

Must identify regularly (FCC Rules), usually at the end of a message transmission. The rules say identify during the last transmission.

I wanted a *set and forget* type of operation, no hardware shenanigans at the hunt venue!
I specifically didn't want to have to turn everyone on at the same time (or connect batteries all at once!)
We will, therefore, employ a TOY clock to track and manage time

Just turn the damn thing on when convenient.
Insensitive to inadvertant power cycle!

Wanted a simple way to define a schedule and load it into the transmitter. Simple serial interface. Use a USB-to-serial integrated cable. Serial hardware in fox transmitter very simple. Access from a host is simple/easy.

We can control an external HT, the hardware and most of the software is there. The 14-pin connector provides connection to H.T.

So, I think I've provided all the *right stuff* in the hardware domain.

The software has continued to evolve into a feature-rich environment for controlling message transmission.

Power Slide 5

Battery monitor added to 102-73176 revision (Raspberry-PI) and carried forward to the 102-73181 boards so we can look at the current draw during transmit. Also provides a verbalization of the battery parameters during hunt setup. Just listen for battery voltage. it should announce above 7.2 volts.

Added the **BATR** battery report command to dump voltage and current. These commands are in the announce file that runs at power-on so we get a verbal report about the battery.

This traffic is visible through the serial port to help with automatically setting the time prior to a hunt and updating the CSV file that generates the transmitter check sheet.

We want to know that the battery will easily last through the entire hunt. So we measure and report it!

Fresh battery quickly drops to just below 9 volts and holds steady for many hours. Eventually, as the battery collapses, current spikes and the 5V regulator (green) starts to fall out of regulation.

zNEO has 3.3V linear regulator so continues to operate until the 5V regulator completely pukes when transmitting.

Moved away from 102-73176 design due to limited battery life. The 102-73176 board is a Raspberry-PI based system. The 102-73176 system was also prone to SD card corruption and suffered from a looooooooong startup (Linux startup :-( )

Last revision of the 102-73161 switched over to using a

The switch-mode device performs well, pulling the battery terminal voltage down to almost 5 volts before it can't regulate any longer.

Notes about the Battery Plot

The plot indicates sending an *SOS message* over the air when battery voltage drops below the trip level.

The **BATV** and the **BATC** commands both take the voltage trip level as an argument. Both send out **SOS** in code if the battery voltage falls below the trip point.

The hunt organizers can monitor this during the hunt if the BATV/BATC commands are present and have that trip voltage.

The plot indicates that there is six to eight hours of operating life in the battery at that trip voltage. The intent, then, is you can start the hunt as long as the battery message doesn't include an SOS when you set them.

If the battery voltage falls below the trip point during the hunt it will start sending **SOS** in code.

Why not Lithium Chemistry???

We probably can find packs that would fit in the space in the enlosusure. Need to be **at least** 2S (series), probably better if we get 3S packs. The regulators appear to work down to around 6 volts, so 2-series is the minimum. Using 3-series gives us a little more room to work.

We don't want to get the voltage so high that we can easily reverse a cell, as that would kill the pack outright.

Charging becomes an arduous task. We're using a stable of 3 hunt groups with 6 units in each. Charging 18 units before a hunt (ouch)! The AAA alkaline are cheap at Costco and on Amazon and will run four or five hunts. Battery changeout is, of course, considerably faster.

There is no restriction or requirement that all units use the same battery chemistry. The fox transmitter design simply doesn't care...

Lead Acid Chemistry ?

Yes, if that trips your trigger!

I see small units on DigiKey in the 10$+$ range.

The 12V nominal voltage may require updating the voltage monitor resistor divider. For 12V lead-acid chemistry R35 needs to change to 24.9K for older software or 26.1K for updated coefficients in newer software.

Coefficient tables in the schematic show the needed resistor values for various battery voltage ranges.

The coefficient tables in the software can be dumped using the **BATR I** command and compared with those in the schematic.

Other Power SOurces ?

Yes, but keep the voltage below 24 Volts.

Some of the regulators will tolerate up towards 36 volts, but keeping below 24 volts eliminates any stress problems introduced by parts selection.

Scheduling Slide 6

This has been discussed in earlier presentations

Typical FOX HUNT is one or more hunt-groups (of 5 or 6 transmitters per group) Each group operates on a unique frequency with all units in the group sharing **one** frequency. Units in the hunt group operate by transmitting one-after-the-other.

FCC rules dictate that the each transmitter must identify regularly and at the end of a *conversation*.
Both the **BEGN** and the **DONE** commands send our callsign in code.

Learn about modular arithmetic!

This description follows the convention used in the **C language** to invoke modular arithmetic operations.

TOY time is 32 bits but is truncated to time-of-day (i.e. seconds from midnight, local)

To figure out the **WHEN**, we specify a cycle (or repeat) period, and how far into the period as to the **WHEN**.

The period effectively counts from midnight and rolls over at the end of the period.

So the **WHEN** occurs when *offset==(system_time%period)*

The **(system_time%period)** calculation doesn't care if the fox time is larger than a day. Our fox transmitter doesn't care if time is truncated.

Time in the FOX Transmitter Slide 7

Review of how the fox transmitter keeps time straight!

Battery backed TOY clock to keep track of time when switched off. Copy time from TOY clock to system time using a **TIME** command.

Also we deal with local time zone using the **EPOC** command to define our offset from *ZULU* or *GMT*

Modular Arithmetic (Ignore Carry, Use remainder)

When we set the time, we are free to use any EPOCH we like, as long as it starts at midnight. The FOX Transmitter stores time in UT, probably using exactly the same time value as that used on the host system.

We have a mechanism in the fox transmitter to shift from UT to local time (the **EPOC** command). This only becomes necessary when using the **STAR** command to keep the Fox Transmitters quiet before the start of a hunt.

If your managing your numbers carefully, you may notice a potential problem mixing 16 bit and 32 bit fields in this time calculation.

In practice, it simply doesn't occur as we promote the 16 bit fields to zero-padded 32 bit fields to do the arithmetic.

We also run into a problem when the 32 bit time field rolls over in 2038. If it does cause a problem, I'll get a good chuckle out of it. In practice, however, you almos always load a truncated time into the Fox Transmitter, so the battery will die long before an overflow in the seconds field would cause a problem.

Although the fox transmitter software keeps time in a 32 bit integer, it doesn't use time beyond 1-day in the scheduling calculation.

now = system_seconds % 86400

So this is the calculation to arrive at seconds from midnight.
We do some magick to take the time-zone offset into account!
Keep period fitting into an hour and it always works.
(i.e. **3600 % period** results in zero).

Now, in the main loop, when seconds roll over
(same thing as when RTI == 0,
or perhaps when the current RTI is less than the previsous RTI)
we do this calculation for each active schedule.

if((now % period) == offset)

If we get a hit, then we run the commands in that schedule.
First one we get a hit on runs, other are ignored for now.
Scanned in order from S0 through to S9.

As long as the schedule isn't too bizarre, we can run across a day without problems. Don't use a period that doesn't fit into a day evenly if you want things to run perfectly across a day boundary.

Time Management

We do play some games, both in the host system, and in the fox transmitter to keep time error to a minimum.

Host system sends time when sub-seconds in the host system are zero. We assume the host isn't busy so it doesn't introduce unpredictable delays. Host always send out time message at xx:xx:x0.000.

Fox Transmitter executes the time set command immediately upon its reception. We assume the DS1672 zeros out its sub-second counter when saving the time data to the DS1672 counter. This **does seem to be the case**, although this behavior isn't explicitly stated in the Datasheet.

Fox Transmitter time set command polls the DS1672 until the LSB of seconds changes. This occurs, of course, when the sub-seconds count rolls over.

This should set the Fox Transmitter system time (this is the time the zNEO maintains) to within a very few tens of milliseconds of the time in the host system.
Since we use that one host to set all the transmitters, the entire set of transmitters are set to within that very few tens of milliseconds.

The DS1672 oscillators all drift over time.
After several weeks of sitting on my floor, patiently waiting for attention, they report time that is within several tens of seconds.

I set the time in all of them the night before a hunt. The battery condition is read and shows up on one of the check-sheets generated for the fox hunt. This setup, which is now automated, also grabs the current battery condition. Low battery voltage is reported so the battery pack can be changed so we don't have a transmitter go dead during the hunt.

Scheduling features Slide 8

Since the routines to deal with scheduling are already present in the code, we can extend the scheduling model to cover commands that deal with time.

There are but a few, well two right now.

WAIT

You may need to insert a short delay between message elements. Seperating senteces of code or putting a gap etween two voice utterances. That's accomplised by simply giving the **WAIT** command the desired time which is a decimal number.

The system ticks in 10mS intervals, so the requested delay is tweaks around into sme number of 10mS delays.

We can also specify the delay as a **period,offset** pair to cause synchronous scheduling.

This method treats the arguments as integers and applies the same scheduling strategy from the scheduler.

In this case, we would be using a far shorted period as the main scheduling cycle is set using the **MODS** command.

This fine-grained scheduling is provided to allow multiple transmitters to send message traffic that is intersperced down at the secondws level.

Two transmitters, for example, can carry on a coversation.

Try to hunt that type of operation.

CHRP

We can also emulate the operation of a wildlife tracker using this command. Here the tracker send a short RF *chirp* occasionally and shuts the RF section down between chirps to save power.

This mode of operation reduces the RF avaliable for direction finding.

In the Fox Transmitter, however, the RF synthesizer is not switched off, just the RF amplifier section. There is still a very small emission level that will be detected close to the transmitter.

The first **CHRP** argument can also specify and audio file. Here we just send that file rather than the tone burst when we reach a scheduling point.

You'll have to manually get the timing right, if the audio is too long, it will seem to schedult things *funny*.

The example audio is a tone burst that changes in frequency, and audio analogy to a real radar chirp. This audio sweep will confuse you when using a DTOA antenna switch!

Nominal Scheduling Chart Slide 9

Here we show a timing plot of who talks when.

This is the nominal case.

Not using anything fancy.

RED bars are transmitter active **window**

We build the message to fit in this window with about 5 seconds to spare.

GREEN bars are transmitter idle

Transmitter is (must be) idle in thjis window.

Trying to show how a 5 minute or a 6 minute cycle would look. Both appear on the plot.

For a 6 minute cycle, we have dropped of the last 60 seconds from the plot. Note, however, that FOX26 starts at second 300, as expected. It will run for 60 seconds, like all the others, and then we roll over from second 359 to second 0.

This is a filled out schedule chart of what we expect to happen with a nominal fox hunt group. Everyone in the group takes their turn and everything works out.

Nominal Schedule S0=1 Slide 10

Go through the commands quickly, these are used to implement the S0= schedule (duh!)

I'm jumping into the middle of things here, and will try to cover questions you'll, no doubt, have about details.

BATR

Battery Report when transmitter is idle.
That is when the transmitter is OFF

CONF -AM

Switch to normal FM operation
i.e. **not** AM

TONE 1.0

Set audio tone frequency to 1 KHz.

CWPM 30,-1,-1,-1,-1

Set code chirping rate to 30 Words per Minute using nominal spacings.

BEGN

Enable the RF path (turn of the RF amplifier)
This also send out: **CQ CQ CQ de** *call*

TALK

Verbalize the callsign and nickname
For the CW impaired among us

WAIT 0.5

Wait for 0.5 seconds

Nominal Schedule S0=2 Slide 11

TONE *tone*

Change the audio tone
the actual value will be set in the shell script that loads the fox transmitter.

CWPM 25,-1,-1,-1,-1

Change the code rate to make it sound different

BATC EV 7.2

Send out battery report in code (BAT**C**)

E tells it to encode the voltage as **T**s and **E**s. Count the **DAH**s for volts and the **DIT**s for tenths.

V Send the voltage channel

7.2 trip voltage. If the battery is below this voltage we'll send out **SOS** in the battery message.

WAIT 0.5

Sime delay, expressed in fractional seconds

CODE

This is the Morse traffic that will be sent.
Multiple line to fit the message text within the 32 byte FRAM record length.

Nominal Schedule S0=3 Slide 12

REM-

Comments record stuffed into FRAM during the loading process
These are simply spare records that can be overwritten later to change something.

BATR

Another battery report, this time when the RF subsystem is active.

We haven't shut the RF system off yet, so we get a report on the current draw by the RF subsystem.

TONE 1.0

Back to the3 same audio tone we used for the signon message

Make the signoff sound the same

CWPM 30,-1,-1,-1,-1

And we also return to the 30WPM rate we used with the signon message

DONE

Send the signoff message: **SK SK SK de** *call*

Switch off the synthesizer and remove power from the RF daughterboard.

INItilization Slide 13

I've skipped over initialization, so I'll jump back a bit here to illustrate how we define the callsign and nickname (and all things of that ilk) in only one place.

All the messsage sequences (**S0=** through **S9=**) use the <CALL>and <NAME>substitution.
That way the **S0=** sequence (and all the others) are identical for all fox transmitters. We don't maintain unique versions for each unit.

So then, the **INI=** file configures the identity of the fox transmitter (and the hardware configuration). We set our callsign and nickname **once** and then reference these saved names in the message sequences.

The **ANN=** (or *ANNOUNCE* file) is where the signon message lives. Signon reports callsign, nickname, battery state, and target operating frequency. That is, all the things I want to know as I set out the fox transmitter (with my H.T. tuned to 144.150 MHz).

Near the end of the **ANN=** file (after the message traffic has been sent) we switch to the target operating frequency.

With no more commands to execute, the transmitter system goes **idle** and waits for a scheduling point to occur.

INI Report Slide 14

This is what comes out of the serial port on the **Fox Transmitter** when it is first turned on.

This has been truncated!

TOY clock read (**TIME**) needs to run twice to get it right. **EPOC** sets up for CDT.

Define out nickname and callsign.

The **CONF** sets up the RF hardware, in this case the SI5351.

Announce frequency ste to 144.150 MHz.

Battery report, here when idle.

S0 Report Slide 15

This is what comes out of the serial port on the **Fox Transmitter** when the **S0=** sequence runs.

This has been truncated!

Battery report with RF amplifier OFF!

The **CONF** sets FM operation, Audio tone is 1KHz. Code rate is 30 WPM with standard spacings.

BEGN sends out a CQ announce.

The **BATC** command sends battery report in code. Count the DAH(letter T) to get 7 volts. Count the DIT(letter E) to get tenths of volts. BAttery is 7.3 volts.

The chatter away in code.

Battery report with RF amplifier ON! Now we have idel and active power measurement.

Change back to sounding like the **BEGN** message and send out signoff message.

Nominal Sounding Schedule Slide 16

I AM HORRIBLE!

Being somewhat lazy, all fox transmitters have all sound clips loaded (they're all the same).

Nickname utterances for FOX20..FOX39 are everywhere, the FLASH devices are HUGE.

Therefor, a unit is capable of lying about it's identity.

We could specify a schedule for FOX21 of **MODS S0=120,0** so FOX21 sounds the same each time it transmits. It would correctly announce its Nickname as FOX21.

There is, however, another way...

Consider an approach where we build 3 different schedules in FOX21. Each one sounding a little different. Different code settings, WPM rate and audio tone.

As a hunter, you get used to the sound (personality) of each unit.

The scheduler will happily and cleanly deal with multiple schedules if they don't overlap. In this example they don't.

To make life easier, the vocalization of the unit Nickname uses a file name taken from that given by the **NICK** command. This makes managing a large group of transmitters a bit easier.

In our devious setup here, we hardcode the Nickname differently for each schedule...

The **S0=** schedule announces as FOX21.
The **S1=** schedule announces as FOX23.
The **S2=** schedule announces as FOX25.

We also set the morse code generator different on each:

The **S0=** unit gets **CWPM 20** and **TONE 1.2**
The **S1=** unit gets **CWPM 25** and **TONE 1.5**
The **S2=** unit gets **CWPM 30** and **TONE 1.8**

As the hunt progresses, this one transmitteer masquerades as two other devices. The hunters are not informed of the masquerading.

Once the masquerading transmitter is found, you would expect the hunter to move on to locating the next transmitter, moving away from the unit that was just found.

After one minute of the other station transmitting, we're back to the masquerade transmitter.

This should give an idea of how the scheduling method can be used to implement some rather devious strategies.

CHiRP Scheduling Chart Slide 17

This gets complicated as we use the CHiRP command to get everyone talking at once.

Not talking on top of each-other, but transmitters switch for every chirp noise.

This is, in effect, the initial test of the synchronous feature in the CHiRP command. To see if fine-grained scheduling is possible, that we can load the TOY clock with millisecond precision.

Also notice we stretch this one out to 10 minutes.

This stretches our transmission out as much as the rules allow. We identify both at the begining and at the end of the message.

We then settle in to the interleaved chirp.

The ident message traffic is spaced out so that the fox transmitters don't send their ID traffic at the same time as other units.

We will, however, be sending the ID message traffic when other units are sending their chirps.

CHiRP S1 Slide 18

This proceeds similarly to the previous example until we configure for interrupted carrier operation.

CONF CW

Configure to enable RF power only when sending code or voice. In between we remove the power to the RF amp.

CHRP 'chrpfrq' 'chrp1' 0.30 55

Configure for synchronous CHiRP.

The audio frequency is supplied by the shell script. Each of the six transmitters are set to a different audio frequency.

All transmitters are set with the same period to effect synchronous operation. The operation is the same, but we operate with a 12 second period allowing each transmitter in the group 2 seconds to send out the chirp. The offset values are staggered by 2 seconds.

The shell script may supply either an audio frequency (i.e. a numeric argument) or a filename (non-numeric). The command decoder recgonizes the difference and will send out a audio tone burst, or an audio file.

The *'chrp1'* argument sets the schedule which consists of the period and offset (compatible with the substitution mechanism in the shell script and sequenct loader).

Here, we have loaded an audio waveform for the chirp to aurally emulates a RADAR chirp.

CONF -CW

Back to normal FM operation and end the message traffic as with the previous example.

Stooges Scheduling Chart Slide 19

Now, we will switch over to using the synchronouse feature of the **WAIT** command.

First-off, realize that the **MODS** command schedules exactly as it's told, it doesn't have smarts enough to see that we've started FOX21 and FOX22 with a 15 second offset, rather than (a more sane) 60 seconds.

FOX21 and FOX22 are given time to send their sign-on message before they start the bit where Shemp Hypnotises Moe. Shemp eventually moves Moe over to Sing Sing in New York. When Shamp tries to move on to Boston, Moe is stuck in Sing Sing.

For the youngsters, realize that Sing Sing is where the New York state prison is located, and in the short Moe picks up a straight-back chair, with vertical slats between the stiles, and replies "I am now in Sing Sing"

Shemp then calls out "You are now in Boston"

And Moe, of course, with the chair still in front of his face replies "I am now in Sing Sing".

All of us that grew up with *Dr. Max* on WMT (now KGAN) will find this hilarious...

FOX21 and FOX22 get done with the exchange, send their signoff messages, and then FOX23..FOX26 can send normally.

The **WAIT** command, using a 120 second period, achieves the required synchronization to conduct the conversation.

We do enable the mode that disables carrier during *dead air* as we do in the CHiRP example above. This allows the two transmitters to carry on the conversation without stomping on each other.

Stooges 8 Slide 20

We'll start with more-or-less normal signon at 30WPM

So send the signon: **CQ CQ CQ de** *call*

CONF CW
: Configure to enable RF power only when sending code or voice. In between we remove the power to the RF amp.

CODE SING SING
: Just a bit of testing to see that we're running the RF power switch as expected. current draw by the RF subsystem.

WAIT *stooge*/40
: Synchronous wait where the *stooge* field is filled in by the shell script.

  RF must be OFF during this wait! current draw by the RF subsystem.

TALK TS1_LA
: The voice utterance from Shamp: "You are now in Los Angeles"

  Moe follows, of course, responding from the other transmitter.

  We carry on with the sketch with Shemp then commanding Mode to New York, which Moe dutifully obeys.

The third time, Shemp moves Moe to Sing Sing in upsatee New York. Sing Sing being the location of the New York State Prison.

Moe answers this command by picking up a chair, a straight back chait with vertical stiles and dutifullt follows Shemp to Sing Sing

Shemp then tries to get Moe over to Boston.

Well, this being the **Three Stooges**, Moe is now stuck in Sing Sing...

WAIT *stooge*/80

This synchronous wait gets us toward the end of two minute section of the six minute cycle this 6 transmitter group uses.

The S9= schedule times this command 20 seconds later to get them both in.

TONE 1.2

Audio frequency 1.2KHz

We sound a bit different now current draw by the RF subsystem.

CODE

A bit more Morse to fill up the 15 second window.

TONE 1.0

Back to our signoff frequency of 1.0KHz

CWPM 30,-1,-1,-1,-1

And back to fast CW

DONE

Standard signoff message...

Stooges 9 Slide 21

This looks almost identical to the S8= schedule other than timing seldections.

The voice files are the complements to those in the S8= schedule. Where S8 is sending Shemps dialogue hypnotising Moe our S9= will be sendinf Moes response.

The schedules for S8 and S9 are staggered to get the signon message from each without them overlapping.

Once we've sent signon traffic the synchronous WAIT commands take over to get Moes response out right after Shemps voice command.

Once the short sketch has been re-enacted, these two transmitters fall back on the main schedule, which uses the same period as the remaining transmitters, such that they are quet until the start of the next 6 minute cycle.

Last Slide Slide 22

Show and Tell

If host remembers to turn on transmitters, we can all listen in on several active hunts. The stations should be using all of the scheduling magic described in slides 9, 16, 17, and 19

All stations report on 145.150MHz at power-on (i.e. when you switch on at the drop point) and then start sending their sign-on message traffic.

FOX27 .. FOX32

One hunt group is configured for a nominal hunt. Each stations is allocated about a minute to transmit its message. The group period (or cycle time) is six minutes.

TOY clock precision/accuracy requirements not particularly strict for this to work out as there is at least 5 to 10 secondws between messages.

FOX21 .. FOX26

Now lets drive the hunters absolutely nuts!

TOY clock precision/accuracy requirements here are strict. We need TOY clocks to stay synchronized a few hundred milliseconds for this to work.

The short chirps are each from a different unit. We send ONE CHiRP and wait for the six second synchronization point. The other units are staggered throughout our wait time.

FOX33 .. FOX38

Two units re-enact an old 3-stooges sketch.

TOY clock on Shemp and Moe needs sub-second precision/accuracy to properly synchronize in order to carry on the dialogue.

TOY clocks on the remaining units need the same precision/accuracy as FOX27..FOX32 require.

We're missing a transmitter in this group. FOX35 suffered a problem during assembly.

To operate as a six-unit hunt group, we task one of the working stations with sending two messages.

We can operate on a shortened schedule (i.e. 180 seconds rather than 360 seconds).

OR we can activate two different schedules. Running two at once means we do not need to send message traffic evenly spaced (i.e. eliminate the need to send every 180 seconds).

Running two schedules also means we will have the ability to run two different command sequences. This would allow us to masquerade and another unit in the unit identification utterance.

We may also have any CW traffic sound different; different audio tone frequency, different word rate. We could also alter the spacing used by the code generator (like increase the space bwteen characters or words). In effect trying to sound like a different fist.

FOX20

Training transmitter! On almost all the time.

Reduce the downtime between messages to a few seconds.

Use one of the low-power RF amplifiers to minimize battery use.

Software Prototype Slide 23

Now you get to look beneath the hood.

This is an early 102-73181-0 board. The RF synthesizer is an ICS307, which we can no longer obtain. Other than the RF synth, it is electrically identical to the later 102-73181-10 as far as the software is concerned.

This is using the hard-to-get zNEO 80 pin package. The 102-73181-10 boards switched to a 64 pin package. The 102-73181-10 also switch to the SI5351 synthesizer.

RF oscillator trim is achieved in a utility on the host system that generates the SI5351 frequency table. We simply measure the frequency error at 144.100MHz and then correct for the error when generating the external frequency table for the SI5351.

This particular board is populated with a replaceable coin cell for the TOY clock. Other units have a small coin cell simply soldered to the board. The TOY clock is above and to the left of the battery, next to the itty-bitty 32KHz watch crystal. *FOX17* label.

This is also before the switch to using a TTL serial cable. This board uses the same chip as the TTL serial cable but the chip is right there to the right of the

The 102-73181-10 board could be populated with the USB Serial Chip, but it requires opening the enclosure to access, so all subsequent units use the 3.5mm connector to access the serial port.

The low pass filter is visible lower left on the board when the RF amplifier isn't installed.

The board on the right is the zNEO programming device.

It is fashioned to be installed in the same Hammond enclosure, so it shares the same outline as the Fox Transmitter.

The main board has a USB serial device from the same family of devices used on the Fox Transmitter. The board started out life as an RS485 adapter.

The daughterboard has the buffers and connector to connect to the Fox Transmitter through a 6-pin header.

Blinky lights provide a nice light show when the zNEO is programmed.