```
#!/usr/local/bin/expectk
#
# $Id: tkd7,v 1.33 2002/11/24 01:51:47 jh Exp $
#
# Copyright 1999
#        Jonathan Hanna.  All rights reserved.
#
# Redistribution and use of this software with or without modification
# is permitted provided that redistributions retain the above copyright
# notice and the following disclaimer.
#
# No guarantee is expressed or implied.
#
#
# Email contact: jhanna@home.com, ve7yjh@qsl.net
# http://www.qsl.net/ve7yjh/tkd7/
#
# So far some functions have worked on:
#       FreeBSD 3.3, expectk 5.28.1
#
# To Do:
#       - Process APRS history properly, parsing looks wrong.
#       - Try outgoing APRS messages.
#       - TNC packet mode.
#       - Clean up hacks.
#       - Check input properly.
#       - Close spawn on error.
#       - Function to save all configuration.
#       - Read Kenwood configuration files.
#       - Port to other OSs.
#       - Add some statefull lockout on command sending, so that
#         replies can be matched against commands. At the moment
#         they are not, which work pretty well but breaks down for
#         things like AMSG and LIST, which are forced to erase old
#         data before new data is read.
#       - Rationalize mouse functions in scanner.
#       - Do real scanning, instead of using built in scan function.
#       - Find out why the radio stops responding occasionally.
#         Could be xon-xoff problems. Delays work reasonably well for now.
#
set tkd7_version {$Revision: 1.33 $}
set tkd7_version [string range $tkd7_version 11 \
        [expr [string length $tkd7_version] - 3]]

eval destroy [winfo child .]

#
# Program definitions
#
set d7_com_ports { /dev/cuaa0 /dev/cuaa1 }
set d7_window_title TKD7
set d7_icon_file kenwood.xbm

set d7_save_file d7.mem

#
# Window button for bringing up configuration
#
set d7_config_event <Button-2>
#
# Colours
#
set d7_on_indicator_colour green
set d7_off_indicator_colour gray
#
# Squelch
#
set d7_squelch_height   30
#
```

```
# Signal
#
set d7_signal_indicator_width   10
set d7_signal_indicator_height  40
#
# Scanner
#
set d7_scanner_graph_width 500
set d7_scanner_graph_height 50
set d7_scanner_graph_xmargin 50
set d7_scanner_graph_topmargin 20
set d7_scanner_graph_bottommargin 30
#
# off, radio, tnc, tnc_data
set d7_mode off
set d7_selected_mode off
#
set d7_spawn_id 0
#
# list of widgets that can only be used when their band
# is active
#
set d7_ab_widget_list(0) {}
set d7_ab_widget_list(1) {}

proc d7_add_to_ab_widget_list { l w } {
        global d7_ab_widget_list
        lappend d7_ab_widget_list($l) $w
}

proc d7_update_ab_widgets {} {
        global d7_ab_widget_list
        global d7_parameter

        foreach w $d7_ab_widget_list(0) {
                if {$d7_parameter(BC) == "0"} {
                        $w configure -state normal
                } else {
                        $w configure -state disabled
                }
        }
        foreach w $d7_ab_widget_list(1) {
                if {$d7_parameter(BC) == "1"} {
                        $w configure -state normal
                } else {
                        $w configure -state disabled
                }
        }
}

#
# Radio definitions
#
# constants
#
set d7_frequency_width          9
set d7_offset_width             5
set d7_memory_name_width        8
set d7_dtmf_memory_name_width   8
set d7_memory_name_pad          1
set d7_ARL_width                4
#
# translation tables
#
set d7_beep_codes(0)            Off
set d7_beep_codes(1)            Key
set d7_beep_codes(2)            Key+Data
set d7_beep_codes(3)            All
set d7_val_list(beep)           { 0 1 2 3 }
```

```
set d7_APO_codes(0)             Off
set d7_APO_codes(1)             "30 Minutes"
set d7_APO_codes(2)             "60 Minutes"
set d7_val_list(APO)            { 0 1 2 }

set d7_DTMF_delay_codes(0)      100
set d7_DTMF_delay_codes(1)      200
set d7_DTMF_delay_codes(2)      500
set d7_DTMF_delay_codes(3)      750
set d7_DTMF_delay_codes(4)      1000
set d7_DTMF_delay_codes(5)      1500
set d7_DTMF_delay_codes(6)      2000
set d7_val_list(DTMF_delay)             { 0 1 2 4 5 6 }

set d7_DCD_codes(0)             "Data"
set d7_DCD_codes(1)             "Both"
set d7_val_list(DCD)            { 0 1 }
set d7_val_width(DCD)           4

set d7_step_codes(0)            5
set d7_step_codes(1)            6.25
set d7_step_codes(2)            10
set d7_step_codes(3)            12.5
set d7_step_codes(4)            15
set d7_step_codes(5)            20
set d7_step_codes(6)            25
set d7_step_codes(7)            30
set d7_step_codes(8)            50
set d7_step_codes(9)            100
set d7_val_list(step)           { 0 1 2 3 4 5 6 7 8 9 }
set d7_val_width(step)          4

set d7_scan_resume_codes(0)     Time
set d7_scan_resume_codes(1)     Carrier
set d7_scan_resume_codes(2)     Seek
set d7_val_list(scan_resume)    { 0 1 2 }
set d7_val_width(scan_resume)   7

set d7_power_codes(0)           High
set d7_power_codes(2)           Low
set d7_power_codes(3)           ELow
set d7_val_list(power)          { 0 2 3 }
set d7_val_width(power)         4

set d7_position_comment_codes(0)        "Off Duty"
set d7_position_comment_codes(1)        "Enroute"
set d7_position_comment_codes(2)        "In Service"
set d7_position_comment_codes(3)        "Returning"
set d7_position_comment_codes(4)        "Committed"
set d7_position_comment_codes(5)        "Special"
set d7_position_comment_codes(6)        "Priority"
set d7_position_comment_codes(7)        "Emergency"
set d7_val_list(position_comment)       { 0 1 2 3 4 5 6 7 }
set d7_val_width(position_comment)      10

set d7_battery_saver_codes(0)           "Off"
set d7_battery_saver_codes(1)           "0.2 Sec"
set d7_battery_saver_codes(2)           "0.4 Sec"
set d7_battery_saver_codes(3)           "0.6 Sec"
set d7_battery_saver_codes(4)           "0.8 Sec"
set d7_battery_saver_codes(5)           "1 Sec"
set d7_battery_saver_codes(6)           "2 Secs"
set d7_battery_saver_codes(7)           "3 Secs"
set d7_battery_saver_codes(8)           "4 Secs"
set d7_battery_saver_codes(9)           "5 Secs"
set d7_val_list(battery_saver)          { 0 1 2 3 4 5 6 7 8 9 }

set d7_APRS_TXI_codes(0)                "0.5"
```

```
set d7_APRS_TXI_codes(1)                "1"
set d7_APRS_TXI_codes(2)                "2"
set d7_APRS_TXI_codes(3)                "3"
set d7_APRS_TXI_codes(4)                "5"
set d7_APRS_TXI_codes(5)                "10"
set d7_APRS_TXI_codes(6)                "20"
set d7_APRS_TXI_codes(7)                "30"
set d7_val_list(APRS_TXI)               { 0 1 2 3 4 5 6 7 }

set d7_APRS_category_codes(0)           Norm
set d7_APRS_category_codes(1)           WX
set d7_APRS_category_codes(2)           Move
set d7_APRS_category_codes(3)           Obj
set d7_APRS_category_codes(4)           Fix
set d7_APRS_category_codes(5)           GPSX
set d7_APRS_category_codes(6)           GPS
set d7_APRS_category_codes(7)           FixC
set d7_val_width(APRS_category)         4

set d7_GPS_codes(0)                     "Not Used"
set d7_GPS_codes(1)                     "NMEA"
set d7_val_list(GPS)                    { 0 1 }

set d7_offset_direction_codes(0)        " "
set d7_offset_direction_codes(1)        "+"
set d7_offset_direction_codes(2)        "-"
# D7E only
set d7_offset_direction_codes(3)        "-7.6 MHz"
set d7_val_list(offset_direction)       { 0 1 2 }

# Tone and CTSS
set d7_tone_codes(01)           67.0
set d7_tone_codes(03)           71.9
set d7_tone_codes(04)           74.4
set d7_tone_codes(05)           77.0
set d7_tone_codes(06)           79.7
set d7_tone_codes(07)           82.5
set d7_tone_codes(08)           85.4
set d7_tone_codes(09)           88.5
set d7_tone_codes(10)           91.5
set d7_tone_codes(11)           94.8
set d7_tone_codes(12)           97.4
set d7_tone_codes(13)           100.0
set d7_tone_codes(14)           103.5
set d7_tone_codes(15)           107.2
set d7_tone_codes(16)           110.9
set d7_tone_codes(17)           114.8
set d7_tone_codes(18)           118.8
set d7_tone_codes(19)           123.0
set d7_tone_codes(20)           127.3
set d7_tone_codes(21)           131.8
set d7_tone_codes(22)           136.5
set d7_tone_codes(23)           141.3
set d7_tone_codes(24)           146.2
set d7_tone_codes(25)           151.4
set d7_tone_codes(26)           156.7
set d7_tone_codes(27)           162.2
set d7_tone_codes(28)           167.9
set d7_tone_codes(29)           173.8
set d7_tone_codes(30)           179.9
set d7_tone_codes(31)           186.2
set d7_tone_codes(32)           192.8
set d7_tone_codes(33)           203.5
set d7_tone_codes(34)           210.7
set d7_tone_codes(35)           218.1
set d7_tone_codes(36)           225.7
set d7_tone_codes(37)           233.6
set d7_tone_codes(38)           241.8
set d7_tone_codes(39)           250.3
```

```
set d7_val_list(tone)           { 01    03 04 05 06 07 08 09 10
                                  11 12 13 14 15 16 17 18 19 20
                                  21 22 23 24 25 26 27 28 29 30
                                  31 32 33 34 35 36 37 38 39 }
set d7_val_width(tone)          5

set d7_beacon_method_codes(0)           Manual
set d7_beacon_method_codes(1)           PTT
set d7_beacon_method_codes(2)           Auto
set d7_val_list(beacon_method)          { 0 1 2 }

set d7_gps_codes(0)             "Not Used"
set d7_gps_codes(1)             "NMEA"
set d7_val_list(gps)            { 0 1 }

set d7_mem_mode_codes(0)        "VFO"
set d7_mem_mode_codes(2)        "MR"
set d7_mem_mode_codes(3)        "CALL"
set d7_val_list(mem_mode)       { 0 2 3 }
set d7_val_width(mem_mode)      4

set d7_colour_codes(0)          Black
set d7_colour_codes(1)          Blue
set d7_colour_codes(2)          Red
set d7_colour_codes(3)          Magenta
set d7_colour_codes(4)          Green
set d7_colour_codes(5)          Cyan
set d7_colour_codes(6)          Yellow
set d7_colour_codes(7)          White
set d7_val_list(colour)         { 0 1 2 3 4 5 6 7 }

#
# radio state
#

set d7_radio_simple_parameters {
        "AI"
        "AIP"
        "BAL"
        "LMP"
        "ARO"
        "ARL"
        "APO"
        "BC"
        "BCN"
        "BEP"
        "CH"
        "CNT"
        "DL"
        "DS"
        "DTB"
        "DTX"
        "DUP"
        "ELK"
        "GU"
        "ID"
        "LK"
        "MAC"
        "MD"
        "MES"
        "MNF"
        "MON"
        "MP"
        "MYC"
        "NSFT"
        "OS"
        "POSC"
        "PP"
        "PT"
```

```
        "RSC"
        "RSV"
        "SCC"
        "SCR"
        "SCT"
        "SKTN"
        "SMC"
        "SMSG"
        "SMY"
        "STAT"
        "SV"
        "TC"
        "TH"
        "TN"
        "TNC"
        "TSP"
        "TXH"
        "TXI"
        "TXN"
        "TXS"
        "UNIT"
        "UPR"
        "VCS"
}

foreach param $d7_radio_simple_parameters {
        set d7_parameter($param) 0
}

set d7_parameter(SKTN)          01

set d7_icon_mode                0
set d7_icon_data                0

set d7_beep                     $d7_beep_codes($d7_parameter(BEP))
set d7_APO_delay                $d7_APO_codes($d7_parameter(APO))
set d7_DTMF_delay_mSecs         $d7_DTMF_delay_codes($d7_parameter(PT))
set d7_scan_resume              $d7_scan_resume_codes($d7_parameter(SCR))
set d7_position_comment         $d7_position_comment_codes($d7_parameter(POSC))
set d7_battery_saver            $d7_battery_saver_codes($d7_parameter(SV))
set d7_APRS_TXI                 $d7_APRS_TXI_codes($d7_parameter(TXI))
set d7_beacon_method            $d7_beacon_method_codes($d7_parameter(DTX))
set d7_gps_unit                 $d7_GPS_codes($d7_parameter(GU))
set d7_dcd_sense                $d7_DCD_codes($d7_parameter(DS))
set d7_call_colour              $d7_colour_codes($d7_parameter(MAC))
set d7_sky_command_tone         $d7_tone_codes($d7_parameter(SKTN))
set d7_message_colour           $d7_colour_codes($d7_parameter(SMC))

set d7_transmitting             0

#
# Band specific
#
foreach band { 0 1 } {
        set d7_frequency($band)         0
        set d7_frequency_MHz($band)     000.00000
        set d7_offset_MHz($band)        00.00
        set d7_squelch($band)           0
        set d7_asc_selected($band)      0
        set d7_reverse($band)           0
        set d7_step($band)              0
        set d7_step_KHz($band)          $d7_step_codes($d7_step($band))
        set d7_power($band)             0
        set d7_power_text($band)        $d7_power_codes($d7_power($band))
        set d7_current_memory($band)    000
        set d7_tone_enable($band)       0
        set d7_tone_freq($band)         01
        set d7_tone_freq_name($band)    $d7_tone_codes($d7_tone_freq($band))
        set d7_ctcss_enable($band)      0
```

```
        set d7_ctcss_freq($band)        01
        set d7_ctcss_freq_name($band)   $d7_tone_codes($d7_ctcss_freq($band))
        set d7_ctcss_match($band)       0
        set d7_offset_direction($band)  $d7_offset_direction_codes($band)
        set d7_memmode($band)           0
        set d7_memmode_name($band)      $d7_mem_mode_codes($d7_memmode($band))
}

#
# VFO
#
set d7_vfo_list { 1 2 3 6 }
#
foreach vfo $d7_vfo_list {
        set d7_vfo($vfo,low)            000
        set d7_vfo($vfo,high)           000
        set d7_vfo($vfo,freq)           000.00000
        set d7_vfo($vfo,step)           0
        set d7_vfo($vfo,stepname)       $d7_step_codes($d7_vfo($vfo,step))
        set d7_vfo($vfo,reverse)        0
        set d7_vfo($vfo,tone)           0
        set d7_vfo($vfo,tonefreq)       01
        set d7_vfo($vfo,tonefreqname)   $d7_tone_codes($d7_vfo($vfo,tonefreq))
        set d7_vfo($vfo,ctcss)          0
        set d7_vfo($vfo,ctcssfreq)      01
        set d7_vfo($vfo,ctcssfreqname)  $d7_tone_codes($d7_vfo($vfo,ctcssfreq))
        set d7_vfo($vfo,offsetdirection)        0
        set d7_vfo($vfo,offsetdirectionname)    \
                $d7_offset_direction_codes($d7_vfo($vfo,offsetdirection))
        set d7_vfo($vfo,offset)                 00.00
        set d7_vfo($vfo,mode)                   FM
}

proc d7_Hz_to_MHz { f } {
        return [format %9.5f [expr $f / 1000000.0]]
}

proc d7_MHz_to_Hz { f } {
        if [catch {set s [expr $f * 1000000]}] {
                set s 0
        }
        if {[string first . $s] != -1} {
                set s [string range $s 0 [expr [string first . $s] - 1]]
        }
        return $s
}

#
# GUI utilities
#

proc d7_make_menu { t menuname variable command } {
        upvar #0 d7_${t}_codes c
        global d7_val_list

        menu $menuname -tearoff 0
        foreach x $d7_val_list($t) {
                $menuname add radiobutton \
                        -label $c($x) \
                        -variable $variable \
                        -value $x \
                        -command $command
        }
}

#
# DTMF
#
for { set x 0 } { $x < 10 } { incr x } {
```

```
        set d7_dtmf_memory($x,label)            $x
        set d7_dtmf_memory($x,name)             ""
        set d7_dtmf_memory($x,value)            ""
}

proc d7_dtmf_memory_set_name { num name } {
        global d7_dtmf_memory

        set d7_dtmf_memory($num,name) $name
        if {$name != ""} {
                set d7_dtmf_memory($num,label) $d7_dtmf_memory($num,name)
        } else {
                set d7_dtmf_memory($num,label) $num
        }
}

proc d7_dtmf_memory_read { num } {
        global d7_transmitting

        if {$d7_transmitting == "0"} {
                d7_send_string "DM 0$num"
                d7_send_string "DMN 0$num"
        }
}

#
# store dtmf memory setting in radio memory
#
proc d7_dtmf_memory_store { num } {
        global d7_dtmf_memory

        d7_send_string "DM 0$num,$d7_dtmf_memory($num,value)"
        d7_send_string "DMN 0$num,$d7_dtmf_memory($num,name)"
}

proc d7_make_dtmf {} {
        global d7_dtmf_memory_name_width
        global d7_dtmf_memory
        global d7_memory_name_pad

        set w [toplevel .dtmf]

        wm title $w "DTMF"

        frame $w.memories

        for {set x 0} {$x < 10} {incr x} {
                frame $w.memories.c$x

                if {$d7_dtmf_memory($x,name) == ""} {
                        set d7_dtmf_memory($x,label) $x
                } else {
                        set d7_dtmf_memory($x,label) $d7_dtmf_memory($x,name)
                }
                label $w.memories.c$x.label -width 2 -text $x
                # no use yet
                button $w.memories.c$x.send -text Send \
                        -command "if {\$d7_transmitting == \"1\"} {
                                        d7_send_string \"DM 0$x\"
                                }"
                entry $w.memories.c$x.name \
                        -width [expr $d7_dtmf_memory_name_width + \
                                        $d7_memory_name_pad] \
                        -textvariable d7_dtmf_memory($x,name)
                entry $w.memories.c$x.value \
                        -width 16 \
                        -textvariable d7_dtmf_memory($x,value)
                button $w.memories.c$x.read -text Read \
                        -command "d7_dtmf_memory_read $x"
```

```
                button $w.memories.c$x.set -text Set \
                        -command "d7_dtmf_memory_store $x"
                pack \
                        $w.memories.c$x.label \
                        $w.memories.c$x.name \
                        $w.memories.c$x.value \
                        $w.memories.c$x.read \
                        $w.memories.c$x.set \
                        $w.memories.c$x.send \
                        -side left -expand true -fil both
                pack \
                        $w.memories.c$x \
                        -expand true -fil both
        }

        pack \
                $w.memories \
                -expand true -fil both
        button $w.dismiss -text Dismiss -command "destroy $w"
        pack \
                $w.dismiss \
                -expand true -fil both
}

#
# Memory
#

proc d7_memory_get_name { num } {
        global d7_memory

        d7_send_string "MNA 0,$d7_memory($num,index)"
}

proc d7_memory_num_to_index { x } {
        if { $x == "CallA" } { return $x }
        if { $x == "CallB" } { return $x }
        if { $x < 10 } {
                set index 00$x
        } elseif { $x < 100 } {
                set index 0$x
        } elseif { $x < 200 } {
                set index $x
        } else {
                set n [expr ($x - 200) / 2]
                if { $x % 2 == 0 } {
                        set index L$n
                } else {
                        set index U$n
                }
        }
        return $index
}

proc d7_memory_index_to_num { x } {
        if {[regexp {^[0-9][0-9]*$} $x]} {
                scan $x %d n
        } elseif {[regexp {^L[0-9][0-9]*$} $x]} {
                scan [string range $x 1 end] %d n
                set n [expr $n * 2 + 200]
        } elseif {[regexp {^U[0-9][0-9]*$} $x]} {
                scan [string range $x 1 end] %d n
                set n [expr $n * 2 + 201]
        } else {
                set n 0
        }
        return $n
}
```

```
proc d7_init_mem { x } {
        set index [d7_memory_num_to_index $x]

        uplevel #0 "set d7_memory($x,index)             $index"
        uplevel #0 "set d7_memory($x,name)              \"\""
        uplevel #0 "set d7_memory($x,freq)              000.00000"
        uplevel #0 "set d7_memory($x,txfreq)            000.00000"
        uplevel #0 "set d7_memory($x,step)              0"
        uplevel #0 "set d7_memory($x,stepname)          \$d7_step_codes(0)"
        uplevel #0 "set d7_memory($x,txstep)            0"
        uplevel #0 "set d7_memory($x,txstepname)        \$d7_step_codes(0)"
        uplevel #0 "set d7_memory($x,reverse)           0"
        uplevel #0 "set d7_memory($x,tone)              0"
        uplevel #0 "set d7_memory($x,tonefreq)          01"
        uplevel #0 "set d7_memory($x,tonefreqname)      \$d7_tone_codes(01)"
        uplevel #0 "set d7_memory($x,ctcss)             0"
        uplevel #0 "set d7_memory($x,ctcssfreq)         01"
        uplevel #0 "set d7_memory($x,ctcssfreqname)     \$d7_tone_codes(01)"
        uplevel #0 "set d7_memory($x,offsetdirection)   0"
        uplevel #0 "set d7_memory($x,offsetdirectionname)
\$d7_offset_direction_codes(0)"
        uplevel #0 "set d7_memory($x,offset)            00.00"
        uplevel #0 "set d7_memory($x,mode)              FM"
        uplevel #0 "set d7_memory($x,lockout)           0"
}

for { set x 0 } { $x < 220 } { incr x } {
        d7_init_mem $x
}

foreach x { CallA CallB } {
        d7_init_mem $x
}

proc d7_memory_save_all_to_radio { } {
        for { set x 0 } { $x < 220 } { incr x } {
                d7_memory_store $x
        }
        foreach x { CallA CallB } {
                d7_memory_store $x
        }
}

proc d7_memory_set_bg { num colour } {
        if [winfo exists .memory] {
                set row [expr $num / 10]
                .memory.memories.row$row.o$num configure -bg $colour
        }
}

proc d7_memory_search { pat } {
        global d7_memory
        global d7_on_indicator_colour
        global d7_off_indicator_colour

        for { set x 0 } { $x < 220 } { incr x } {
                if [regexp -nocase -- "$pat" $d7_memory($x,name)] {
                        d7_memory_set_bg $x $d7_on_indicator_colour
                } elseif [regexp -- "$pat" $d7_memory($x,freq)] {
                        d7_memory_set_bg $x $d7_on_indicator_colour
                } elseif [regexp -- "$pat" $d7_memory($x,txfreq)] {
                        d7_memory_set_bg $x $d7_on_indicator_colour
                } else {
                        d7_memory_set_bg $x $d7_off_indicator_colour
                }
        }
}

proc d7_memory_save_all_to_file { f } {
```

```
        global d7_memory

        if [catch {set fd [open $f w+]} err] {
                tk_messageBox -message "$err" -type ok -icon error
                return
        }
        for { set x 0 } { $x < 220 } { incr x } {
                foreach f {
                                index name freq step reverse
                                tone tonefreq
                                ctcss ctcssfreq
                                offsetdirection offset
                                mode lockout
                                txfreq txstep } {
                        puts $fd "$x $f $d7_memory($x,$f)"
                }
        }
        foreach x { CallA CallB } {
                foreach f {
                                index freq step reverse
                                tone tonefreq
                                ctcss ctcssfreq
                                offsetdirection offset
                                mode lockout
                                txfreq txstep } {
                        puts $fd "$x $f $d7_memory($x,$f)"
                }
        }
        close $fd
}

proc d7_memory_read_all_from_radio { } {
        for { set x 0 } { $x < 220 } { incr x } {
                d7_memory_read $x
        }
        foreach x { CallA CallB } {
                d7_memory_read $x
        }
}

proc d7_memory_read_all_from_file { fn } {
        global d7_memory
        global d7_step_codes
        global d7_tone_codes
        global d7_offset_direction_codes

        if [catch {set fd [open $fn r]} err] {
                tk_messageBox -message "$err" -type ok -icon error
                return
        }
        set l 0
        while {1} {
                set num [gets $fd l]
                if {$num <= 0} {
                        break
                }
                if {[string length $l] == 0} {
                        continue
                }
                if {[string range $l 0 0] == "#"} {
                        continue
                }
                set e [string first " " $l]
                set x [string range $l 0 [expr $e - 1]]
                set l [string range $l [expr $e + 1] end]
                set e [string first " " $l]
                set f [string range $l 0 [expr $e - 1]]
                set l [string range $l [expr $e + 1] end]
                if {$f == "name"} {
```

```
                        d7_memory_set_name $x $l
                } else {
                        set d7_memory($x,$f) $l
                        if {$f == "step"} {
                                set d7_memory($x,stepname) $d7_step_codes($l)
                        } elseif {$f == "txstep"} {
                                set d7_memory($x,txstepname) $d7_step_codes($l)
                        } elseif {$f == "tonefreq"} {
                                set d7_memory($x,tonefreqname) $d7_tone_codes($l)
                        } elseif {$f == "ctcssfreq"} {
                                set d7_memory($x,ctcssfreqname) $d7_tone_codes($l)
                        } elseif {$f == "offsetdirection"} {
                                set d7_memory($x,offsetdirectionname) \
                                        $d7_offset_direction_codes($l)
                        }
                }
        }
        close $fd
}

proc d7_memory_widget { x } {
        global d7_memory
        global d7_memory_name_width
        global d7_memory_name_pad
        global d7_frequency_width
        global d7_offset_width
        global d7_val_width

        set w [toplevel .memory$x]

        wm title $w "Memory $d7_memory($x,index)"

        frame $w.data
        label $w.data.index -text $d7_memory($x,index)

        frame $w.data.d1
        if { ! ( $x == "CallA" || $x == "CallB" ) } {
                entry $w.data.d1.name -textvariable d7_memory($x,name) \
                        -width [expr $d7_memory_name_width + $d7_memory_name_pad]
        } else {
                frame $w.data.d1.name
        }
        entry $w.data.d1.freq -textvariable d7_memory($x,freq) \
                -width $d7_frequency_width

        label $w.data.d1.steplabel -text Step
        menubutton $w.data.d1.step \
                -width $d7_val_width(step) -anchor e \
                -textvariable d7_memory($x,stepname) \
                -menu $w.data.d1.step.menu
        d7_make_menu step \
                $w.data.d1.step.menu \
                d7_memory($x,step) \
                "set d7_memory($x,stepname) \$d7_step_codes(\$d7_memory($x,step))"
        label $w.data.d1.stepunit -text kHz

        checkbutton $w.data.d1.tone -text Tone -variable d7_memory($x,tone)
        menubutton $w.data.d1.tonefreq \
                -width $d7_val_width(tone) -anchor e \
                -textvariable d7_memory($x,tonefreqname) \
                -menu $w.data.d1.tonefreq.menu
        d7_make_menu tone \
                $w.data.d1.tonefreq.menu \
                d7_memory($x,tonefreq) \
                "set d7_memory($x,tonefreqname)
\$d7_tone_codes(\$d7_memory($x,tonefreq))"
        label $w.data.d1.toneunit -text Hz

        checkbutton $w.data.d1.ctcss -text CTCSS -variable d7_memory($x,ctcss)
```

```
        menubutton $w.data.d1.ctcssfreq \
                -width $d7_val_width(tone) -anchor e \
                -textvariable d7_memory($x,ctcssfreqname) \
                -menu $w.data.d1.ctcssfreq.menu
        d7_make_menu tone \
                $w.data.d1.ctcssfreq.menu \
                d7_memory($x,ctcssfreq) \
                "set d7_memory($x,ctcssfreqname)
\$d7_tone_codes(\$d7_memory($x,ctcssfreq))"
        label $w.data.d1.ctcssunit -text Hz

        menubutton $w.data.d1.mode -textvariable d7_memory($x,mode) \
                -menu $w.data.d1.mode.menu
        menu $w.data.d1.mode.menu -tearoff 0
        $w.data.d1.mode.menu add radiobutton -label FM \
                -variable d7_memory($x,mode) -value FM
        $w.data.d1.mode.menu add radiobutton -label AM \
                -variable d7_memory($x,mode) -value AM
        checkbutton $w.data.d1.lockout -text LockOut \
                -offvalue 0 -onvalue 1 \
                -variable d7_memory($x,lockout)
        pack \
                $w.data.d1.name \
                $w.data.d1.freq \
                $w.data.d1.steplabel \
                $w.data.d1.step \
                $w.data.d1.stepunit \
                $w.data.d1.tone \
                $w.data.d1.tonefreq \
                $w.data.d1.toneunit \
                $w.data.d1.ctcss \
                $w.data.d1.ctcssfreq \
                $w.data.d1.ctcssunit \
                $w.data.d1.mode \
                $w.data.d1.lockout \
                -side left

        frame $w.data.d2
        label $w.data.d2.offsetlabel -text Offset
        menubutton $w.data.d2.offsetdirection \
                -relief raised \
                -textvariable d7_memory($x,offsetdirectionname) \
                -menu $w.data.d2.offsetdirection.menu
        d7_make_menu offset_direction \
                $w.data.d2.offsetdirection.menu \
                d7_memory($x,offsetdirection) \
                "set d7_memory($x,offsetdirectionname)
\$d7_offset_direction_codes(\$d7_memory($x,offsetdirection))"
        entry $w.data.d2.offset -width $d7_offset_width \
                -textvariable d7_memory($x,offset)

        checkbutton $w.data.d2.reverse -text Reverse \
                -variable d7_memory($x,reverse)

        label $w.data.d2.label -text "or TX"
        entry $w.data.d2.freq -textvariable d7_memory($x,txfreq) \
                -width $d7_frequency_width
        label $w.data.d2.steplabel -text Step
        menubutton $w.data.d2.step -textvariable d7_memory($x,txstepname) \
                -menu $w.data.d2.step.menu
        d7_make_menu step \
                $w.data.d2.step.menu \
                d7_memory($x,txstep) \
                "set d7_memory($x,txstepname)
\$d7_step_codes(\$d7_memory($x,txstep))"
        label $w.data.d2.stepunit -text kHz

        pack \
                $w.data.d2.offsetlabel \
```

```
                $w.data.d2.offsetdirection \
                $w.data.d2.offset \
                $w.data.d2.reverse \
                $w.data.d2.label \
                $w.data.d2.freq \
                $w.data.d2.steplabel \
                $w.data.d2.step \
                $w.data.d2.stepunit \
                -side left

        pack \
                $w.data.index \
                -side left -expand true -fill both
        pack \
                $w.data.d1 \
                $w.data.d2 \
                -expand true -fill both

        frame $w.control
        button $w.control.read -text "Read" \
                -command "d7_memory_read $x"
        button $w.control.save -text "Save" \
                -command "d7_memory_store $x"
        button $w.control.load -text "Load Memory from Current" \
                -command "d7_memory_load $x"
        button $w.control.seta -text "Set A from Memory" \
                -command "d7_memory_recall 0 $x"
        button $w.control.setb -text "Set B from Memory" \
                -command "d7_memory_recall 1 $x"
        button $w.control.dismiss -text Dismiss \
                -command "destroy $w"
        pack \
                $w.control.read \
                $w.control.save \
                $w.control.load \
                $w.control.seta \
                $w.control.setb \
                $w.control.dismiss \
                -side left -expand true -fill both
        pack \
                $w.data \
                $w.control \
                -expand true -fill both
}

#
# load memory contents to current setting
#
proc d7_memory_recall { band num } {
        global d7_memory
        global d7_memmode

        # if in memory mode ?
        if { $d7_memmode($band) != "2"} {
                d7_send_string "VMC $band,2"
        } else {
        }
        d7_send_string "MC $band,$d7_memory($num,index)"
        d7_send_string "MSH"
}

proc d7_memory_set_name { num name } {
        global d7_memory

        set d7_memory($num,name) $name
        if [winfo exists .memory] {
                set row [expr $num / 10]
                if {$name == ""} {
                        set name [d7_memory_num_to_index $num]
```

```
                }
                .memory.memories.row$row.o$num configure -text $name
        }
}

proc d7_memory_read { num } {
        global d7_memory

        if {$num == "CallA"} {
                d7_send_string "CR 0,0"
                d7_send_string "CR 0,1"
        } elseif {$num == "CallB"} {
                d7_send_string "CR 1,0"
                d7_send_string "CR 1,1"
        } else {
                d7_send_string "MR 0,0,$d7_memory($num,index)"
                d7_send_string "MR 0,1,$d7_memory($num,index)"
                d7_send_string "MNA 0,$d7_memory($num,index)"
        }
}

#
# store memory setting in radio memory
#
proc d7_memory_store { num } {
        global d7_memory

        if {$d7_memory($num,mode) == "FM"} {
                set mode 0
        } else {
                set mode 1
        }
        if { $d7_memory($num,index) == "CallA" } {
                set band 0
                set name ""
                set command CW
        } elseif { $d7_memory($num,index) == "CallB" } {
                set band 1
                set name ""
                set command CW
        } else {
                set band 0
                set name ",$d7_memory($num,index)"
                set command MW
        }

        # second 0 is "split"
        d7_send_string \
                "$command $band,0$name,[d7_MHz_to_d7freq
$d7_memory($num,freq)],$d7_memory($num,step),$d7_memory($num,offsetdirection),$d7_mem
ory($num,reverse),$d7_memory($num,tone),$d7_memory($num,ctcss),,$d7_memory($num,tonef
req),,$d7_memory($num,ctcssfreq),[d7_MHz_to_d7offset
$d7_memory($num,offset)],$mode,$d7_memory($num,lockout)"
                d7_send_string "$command $band,1$name,[d7_MHz_to_d7freq
$d7_memory($num,txfreq)],$d7_memory($num,txstep)"
        #
        # The MW erases the name for some reason, so the MNA must be done after
        #
        if { ! (( $d7_memory($num,index) == "CallA" ) || ( $d7_memory($num,index) ==
"CallB" )) } {
                d7_send_string "MNA 0,$d7_memory($num,index),$d7_memory($num,name)"
        }
}

#
# load a memory from current values
#
proc d7_memory_load { num } {
        global d7_memory
```

```
        if {$d7_memory($num,index) == "CallA"} {
                if {$d7_parameter(BC) == "0"} {
                        d7_send_string "CIN"
                }
        } elseif {$d7_memory($num,index) == "CallB"} {
                if {$d7_parameter(BC) == "1"} {
                        d7_send_string "CIN"
                }
        } else {
                d7_send_string "MIN $d7_memory($num,index)"
        }
        d7_memory_read $num
}

proc d7_make_memory {} {
        global d7_memory_name_width
        global d7_memory_name_pad

        set w [toplevel .memory]

        wm title $w "Memory"

        frame $w.memories

        for {set x 0} {$x < 22} {incr x} {
                frame $w.memories.row$x
                for {set y 0} {$y < 10} {incr y} {
                        set num [expr $x * 10 + $y]

                        global d7_memory

                        if {$d7_memory($num,name) == ""} {
                                set name $d7_memory($num,index)
                        } else {
                                set name $d7_memory($num,name)
                        }
                        button $w.memories.row$x.o$num -text $name \
                                -width [expr $d7_memory_name_width + \
                                        $d7_memory_name_pad] \
                                -padx 0 -pady 0 \
                                -command "d7_memory_open $num"
                        button $w.memories.row$x.a$num -text A \
                                -width 1 -padx 0 -pady 0 \
                                -command "d7_memory_recall 0 $num"
                        button $w.memories.row$x.b$num -text B \
                                -width 1 -padx 0 -pady 0 \
                                -command "d7_memory_recall 1 $num"
                        pack \
                                $w.memories.row$x.o$num \
                                $w.memories.row$x.a$num \
                                $w.memories.row$x.b$num \
                                -side left -expand true -fill both
                }
                pack \
                        $w.memories.row$x \
                        -expand true -fill both
        }

        frame $w.call_channels
        button $w.call_channels.a -text "Call Channel A" \
                -command "d7_memory_open CallA"
        button $w.call_channels.b -text "Call Channel B" \
                -command "d7_memory_open CallB"
        pack \
                $w.call_channels.a \
                $w.call_channels.b \
                -side left -expand true -fill both
        pack \
```

```
                $w.memories \
                $w.call_channels \
                -expand true -fill both
        frame $w.file
        button $w.file.readallr -text "Read All from Radio" \
                -command d7_memory_read_all_from_radio
        button $w.file.saveallr -text "Save All to Radio" \
                -command d7_memory_save_all_to_radio
        entry $w.file.file -width 20 -textvariable d7_save_file
        button $w.file.saveallf -text "Save All to File" \
                -command {d7_memory_save_all_to_file $d7_save_file}
        button $w.file.readallf -text "Read All from File" \
                -command {d7_memory_read_all_from_file $d7_save_file}
        pack \
                $w.file.saveallr \
                $w.file.readallr \
                $w.file.file \
                $w.file.saveallf \
                $w.file.readallf \
                -side left -expand true -fill both
        pack $w.file \
                -expand true -fill both
        button $w.dismiss -text Dismiss -command "destroy $w"
        entry $w.pattern -width [expr $d7_memory_name_width + \
                                        $d7_memory_name_pad] \
                -textvariable d7_memory_search_pattern
        button $w.search -text "R.E. Search" \
                -command {d7_memory_search $d7_memory_search_pattern}
        pack \
                $w.pattern \
                $w.search \
                $w.dismiss \
                -side left -expand true -fill both
}

proc d7_memory_open { mem } {
        if ![winfo exists .memory$mem] {
                d7_memory_widget $mem
        } else {
                wm deiconify .memory$mem
                raise .memory$mem
        }
}

#
# VFO
#

proc d7_make_vfo {} {
        global d7_frequency_width
        global d7_offset_width
        global d7_val_width

        set w [toplevel .vfo]

        wm title $w "VFO"

        foreach vfo { 1 2 3 6 } {
                frame $w.vfo$vfo
                label $w.vfo$vfo.label -text $vfo
                frame $w.vfo$vfo.data
                frame $w.vfo$vfo.data.limits
                entry $w.vfo$vfo.data.limits.low -width 3 \
                        -textvariable d7_vfo($vfo,low)
                label $w.vfo$vfo.data.limits.to -text to
                entry $w.vfo$vfo.data.limits.high -width 3 \
                        -textvariable d7_vfo($vfo,high)
                button $w.vfo$vfo.data.limits.get -text Get \
                        -command "d7_send_string \"PV $vfo\""
```

```
                button $w.vfo$vfo.data.limits.set -text Set
                pack \
                        $w.vfo$vfo.data.limits.low \
                        $w.vfo$vfo.data.limits.to \
                        $w.vfo$vfo.data.limits.high \
                        $w.vfo$vfo.data.limits.get \
                        $w.vfo$vfo.data.limits.set \
                        -side left
                frame $w.vfo$vfo.data.current
                entry $w.vfo$vfo.data.current.freq \
                        -width $d7_frequency_width \
                        -textvariable d7_vfo($vfo,freq)
                label $w.vfo$vfo.data.current.steplabel -text Step
                menubutton $w.vfo$vfo.data.current.step \
                        -width $d7_val_width(step) -anchor e \
                        -textvariable d7_vfo($vfo,stepname) \
                        -menu $w.vfo$vfo.data.current.step.menu
                d7_make_menu step \
                        $w.vfo$vfo.data.current.step.menu \
                        d7_vfo($vfo,step) \
                        "set d7_vfo($vfo,stepname)
\$d7_step_codes(\$d7_vfo($vfo,step))"
                label $w.vfo$vfo.data.current.stepunit -text kHz

                checkbutton $w.vfo$vfo.data.current.reverse -text Reverse \
                        -variable d7_vfo($vfo,reverse)

                checkbutton $w.vfo$vfo.data.current.tone -text Tone \
                        -variable d7_vfo($vfo,tone)
                menubutton $w.vfo$vfo.data.current.tonefreq \
                        -width $d7_val_width(tone) -anchor e \
                        -textvariable d7_vfo($vfo,tonefreqname) \
                        -menu $w.vfo$vfo.data.current.tonefreq.menu
                d7_make_menu tone \
                        $w.vfo$vfo.data.current.tonefreq.menu \
                        d7_vfo($vfo,tonefreq) \
                        "set d7_vfo($vfo,tonefreqname)
\$d7_tone_codes(\$d7_vfo($vfo,tonefreq))"
                label $w.vfo$vfo.data.current.toneunit -text Hz

                checkbutton $w.vfo$vfo.data.current.ctcss -text CTCSS \
                        -onvalue 1 -offvalue 0 \
                        -variable d7_vfo($vfo,ctcss)
                menubutton $w.vfo$vfo.data.current.ctcssfreq \
                        -width $d7_val_width(tone) -anchor e \
                        -textvariable d7_vfo($vfo,ctcssfreqname) \
                        -menu $w.vfo$vfo.data.current.ctcssfreq.menu
                d7_make_menu tone \
                        $w.vfo$vfo.data.current.ctcssfreq.menu \
                        d7_vfo($vfo,ctcssfreq) \
                        "set d7_vfo($vfo,ctcssfreqname)
\$d7_tone_codes(\$d7_vfo($vfo,ctcssfreq))"
                label $w.vfo$vfo.data.current.ctcssunit -text Hz

                label $w.vfo$vfo.data.current.offsetlabel -text Offset
                menubutton $w.vfo$vfo.data.current.offsetdirection \
                        -relief raised \
                        -textvariable d7_vfo($vfo,offsetdirectionname) \
                        -menu $w.vfo$vfo.data.current.offsetdirection.menu
                d7_make_menu offset_direction \
                        $w.vfo$vfo.data.current.offsetdirection.menu \
                        d7_vfo($vfo,offsetdirection) \
                        "set d7_vfo($vfo,offsetdirectionname)
\$d7_offset_direction_codes(\$d7_vfo($vfo,offsetdirection))"
                entry $w.vfo$vfo.data.current.offset \
                        -width $d7_offset_width \
                        -textvariable d7_vfo($vfo,offset)

                menubutton $w.vfo$vfo.data.current.mode -textvariable
```

```
d7_vfo($vfo,mode) \
                        -menu $w.vfo$vfo.data.current.mode.menu
                menu $w.vfo$vfo.data.current.mode.menu -tearoff 0
                $w.vfo$vfo.data.current.mode.menu add radiobutton -label FM \
                        -variable d7_vfo($vfo,mode) -value FM
                $w.vfo$vfo.data.current.mode.menu add radiobutton -label AM \
                        -variable d7_vfo($vfo,mode) -value AM

                button $w.vfo$vfo.data.current.get -text Get \
                        -command "d7_send_string \"VR $vfo\""
                button $w.vfo$vfo.data.current.set -text Set
                pack \
                        $w.vfo$vfo.data.current.freq \
                        $w.vfo$vfo.data.current.steplabel \
                        $w.vfo$vfo.data.current.step \
                        $w.vfo$vfo.data.current.stepunit \
                        $w.vfo$vfo.data.current.reverse \
                        $w.vfo$vfo.data.current.tone \
                        $w.vfo$vfo.data.current.tonefreq \
                        $w.vfo$vfo.data.current.toneunit \
                        $w.vfo$vfo.data.current.ctcss \
                        $w.vfo$vfo.data.current.ctcssfreq \
                        $w.vfo$vfo.data.current.ctcssunit \
                        $w.vfo$vfo.data.current.offsetlabel \
                        $w.vfo$vfo.data.current.offsetdirection \
                        $w.vfo$vfo.data.current.offset \
                        $w.vfo$vfo.data.current.mode \
                        $w.vfo$vfo.data.current.get \
                        $w.vfo$vfo.data.current.set \
                        -side left
                pack \
                        $w.vfo$vfo.data.limits \
                        $w.vfo$vfo.data.current
                pack \
                        $w.vfo$vfo.label \
                        $w.vfo$vfo.data \
                        -side left
                pack $w.vfo$vfo
        }

        button $w.dismiss -text Dismiss -command "destroy $w"
        pack \
                $w.dismiss \
                -expand true
}

#
# APRS
#

set d7_symbol_name_width 10

set d7_APRS_symbols {
        {{/!}   {BB}    {Police}                {Police, Sheriff}}
        {{/"}   {BC}    {}                      {reserved (had been rain)}}
        {{/#}   {BD}    {DIGI}                  {DIGI (white center)}}
        {{/$}   {BE}    {PHONE}                 {}}
        {{/%}   {BF}    {DX CLUSTER}            {}}
        {{/&}   {BG}    {HF GATEway}            {}}
        {{/'}   {BH}    {S AIRCRAFT}            {Small AIRCRAFT (SSID = 7)}}
        {{/(}   {BI}    {CLOUDY}                {}}
        {{/)}   {BJ}    {}                      {avail (Mic moved to m)}}
        {{/*}   {BK}    {SnowMobile}            {}}
        {{/+}   {BL}    {Red Cross}             {}}
        {{/,}   {BM}    {Boy Scouts}            {}}
        {{/-}   {BN}    {House QTH (VHF)}       {}}
        {{/.}   {BO}    {X}                     {}}
        {{//}   {BP}    {Dot}                   {}}
        {{/0}   {P0}    {circle}                {Numeral circle}}
```

```
{{/1}   {P1}    {circle}                {Numeral circle}}
{{/2}   {P2}    {circle}                {Numeral circle}}
{{/3}   {P3}    {circle}                {Numeral circle}}
{{/4}   {P4}    {circle}                {Numeral circle}}
{{/5}   {P5}    {circle}                {Numeral circle}}
{{/6}   {P6}    {circle}                {Numeral circle}}
{{/7}   {P7}    {circle}                {Numeral circle}}
{{/8}   {P8}    {circle}                {Numeral circle}}
{{/9}   {P9}    {circle}                {Numeral circle}}
{{/:}   {MR}    {FIRE}                  {}}
{{/;}   {MS}    {Campground}            {}}
{{/<}   {MT}    {Motorcycle}            {Motorcycle (SSID = 10)}}
{{/=}   {MU}    {RAILROAD ENGINE}       {}}
{{/>}   {MV}    {CAR}                   {CAR (SSID = 9)}}
{{/?}   {MW}    {SERVER}                {SERVER for FIles}}
{{/@}   {MX}    {HC FUTURE}             {HC FUTURE predict (dot)}}
{{/A}   {PA}    {Aid Station}           {}}
{{/B}   {PB}    {BBS}                   {}}
{{/C}   {PC}    {Canoe}                 {}}
{{/D}   {PD}    {}                      {}}
{{/E}   {PE}    {EYEBALL}               {EYEBALL (Eye catcher)}}
{{/F}   {PF}    {}                      {}}
{{/G}   {PG}    {Grid}                  {Grid Square (6 digit)}}
{{/H}   {PH}    {HOTEL}                 {HOTEL (blue bed symbol)}}
{{/I}   {PI}    {Tcp-Ip}                {}}
{{/J}   {PJ}    {}                      {}}
{{/K}   {PK}    {School}                {}}
{{/L}   {PL}    {avail}                 {}}
{{/M}   {PM}    {MacAPRS}               {}}
{{/N}   {PN}    {NTS Station}           {}}
{{/O}   {PO}    {BALLOON}               {BALLOON (SSID = 11)}}
{{/P}   {PP}    {Police}                {}}
{{/Q}   {PQ}    {TBD}                   {}}
{{/R}   {PR}    {REC. VEHICLE}          {REC. VEHICLE (SSID = 13)}}
{{/S}   {PS}    {SHUTTLE}               {}}
{{/T}   {PT}    {SSTV}                  {}}
{{/U}   {PU}    {BUS}                   {BUS (SSID = 2)}}
{{/V}   {PV}    {ATV}                   {}}
{{/W}   {PW}    {NWX}                   {National WX Service Site}}
{{/X}   {PX}    {HELO}                  {HELO (SSID = 6)}}
{{/Y}   {PY}    {YACHT}                 {YACHT (sail) (SSID = 5)}}
{{/Z}   {PZ}    {WinAPRS}               {}}
{{/[}   {HS}    {Jogger}                {}}
{"/\\"  {HT}    {TRIANGLE}              {TRIANGLE (DF)}}
{{/]}   {HU}    {PBBS}                  {}}
{{/^}   {HV}    {L AIRCRAFT}            {LARGE AIRCRAFT}}
{{/_}   {HW}    {WEATHER Station}       {WEATHER Station (blue)}}
{{/`}   {HX}    {Dish Antenna}          {}}
{{/a}   {LA}    {AMBULANCE}             {AMBULANCE (SSID = 1)}}
{{/b}   {LB}    {BIKE}                  {BIKE (SSID = 4)}}
{{/c}   {LC}    {}                      {tbd}}
{{/d}   {LD}    {}                      {Dual Garage (Fire dept)}}
{{/e}   {LE}    {HORSE}                 {HORSE (equestrian)}}
{{/f}   {LF}    {FIRE TRUCK}            {FIRE TRUCK (SSID = 3)}}
{{/g}   {LG}    {Glider}                {}}
{{/h}   {LH}    {HOSPITAL}              {}}
{{/i}   {LI}    {IOTA}                  {IOTA (islands on the air)}}
{{/j}   {LJ}    {JEEP}                  {JEEP (SSID-12)}}
{{/k}   {LK}    {TRUCK}                 {TRUCK (SSID = 14)}}
{{/l}   {LL}    {}                      {avail}}
{{/m}   {LM}    {Mic-Repeater}          {}}
{{/n}   {LN}    {Node}                  {}}
{{/o}   {LO}    {EOC}                   {}}
{{/p}   {LP}    {ROVER}                 {ROVER (puppy)}}
{{/q}   {LQ}    {GRID SQ}               {GRID SQ shown above 128 m}}
{{/r}   {LR}    {ANTENNA}               {}}
{{/s}   {LS}    {SHIP}                  {SHIP (pwr boat) SSID-8)}}
{{/t}   {LT}    {TRUCK STOP}            {}}
{{/u}   {LU}    {TRUCK (18 wheeler)}    {}}
```

```
{{/v}   {LV}    {VAN}                   {VAN (SSID = 15)}}
{{/w}   {LW}    {WATER station}         {}}
{{/x}   {LX}    {xAPRS (Unix)}          {}}
{{/y}   {LY}    {YAGI @ QTH}            {}}
{{/z}   {LZ}    {}                      {}}
{"/\173"        {J1}    {}                      {}}
{{/|}   {J2}    {}                      {reserved (Stream Switch)}}
{{/\175}        {J3}    {}                      {}}
{{/~}   {J3}    {}                      {reserved (Stream Switch)}}
{{\!}   {OB}    {EMERGENCY (!)}         {}}
{{\"}   {OC}    {}                      {reserved}}
{{\#}   {OD}    {STAR}                  {NUMBERED STAR (green)}}
{{\$}   {OE}    {Bank}                  {Bank or ATM (green box)}}
{{\%}   {OF}    {}                      {}}
{{\&}   {OG}    {DIAMOND}               {NUMBERED DIAMOND}}
{{\'}   {OH}    {Crash site}            {}}
{{\(}   {OI}    {CLOUDY}                {}}
{{\)}   {OJ}    {}                      {}}
{{\*}   {OK}    {SNOW}                  {}}
{{\+}   {OL}    {Church}                {}}
{{\,}   {OM}    {Girl Scouts}           {}}
{{\-}   {ON}    {House (HF)}            {}}
{{\.}   {OO}    {}                      {Vicinity Ambiguous Plot (NEW)}}
{{\/}   {OP}    {}                      {}}
{{\0}   {A0}    {NUMBERED CIRCLE}       {}}
{{\1}   {A1}    {}                      {}}
{{\2}   {A2}    {}                      {}}
{{\3}   {A3}    {}                      {}}
{{\4}   {A4}    {}                      {}}
{{\5}   {A5}    {}                      {}}
{{\6}   {A6}    {}                      {}}
{{\7}   {A7}    {}                      {}}
{{\8}   {A8}    {}                      {}}
{{\9}   {A9}    {Gas}                   {Gas Station (blue pump)}}
{{\:}   {NR}    {Hail}                  {}}
{{\;}   {NS}    {Park}                  {Park/Picnic area}}
{{\<}   {NT}    {ADVISORY}              {}}
{{\=}   {NU}    {}                      {}}
{{\>}   {NV}    {NUMBERED CAR}          {}}
{{\?}   {NW}    {INFO}                  {INFO Kiosk (Blue box with ?)}}
{{\@}   {NX}    {HURICANE/Trop-Storm}   {}}
{{\A}   {AA}    {NUMBERED BOX}          {}}
{{\B}   {AB}    {Blowing Snow}          {}}
{{\C}   {AC}    {Coast Guard}           {}}
{{\D}   {AD}    {Drizzle}               {}}
{{\E}   {AE}    {Smoke}                 {}}
{{\F}   {AF}    {Freezing rain}         {}}
{{\G}   {AG}    {Snow Shower}           {}}
{{\H}   {AH}    {Haze}                  {}}
{{\I}   {AI}    {Rain Shower}           {}}
{{\J}   {AJ}    {Lightening}            {}}
{{\K}   {AK}    {Kenwood}               {}}
{{\L}   {AL}    {Lighthouse}            {}}
{{\M}   {AM}    {}                      {}}
{{\N}   {AN}    {Navigation Buoy}       {}}
{{\O}   {AO}    {}                      {}}
{{\P}   {AP}    {Parking}               {}}
{{\Q}   {AQ}    {QUAKE}                 {}}
{{\R}   {AR}    {Restaurant}            {}}
{{\S}   {AS}    {Satellite/Pacsat}      {}}
{{\T}   {AT}    {Thunderstorm}          {}}
{{\U}   {AU}    {SUNNY}                 {}}
{{\V}   {AV}    {VORTAC Nav Aid}        {}}
{{\W}   {AW}    {NWS site}              {NUMBERED NWS site (NWS options)}}
{{\X}   {AX}    {Pharmacy Rx}           {}}
{{\Y}   {AY}    {}                      {}}
{{\Z}   {AZ}    {}                      {}}
{{\[}   {DS}    {Wall Cloud}            {}}
{{\/}   {DT}    {}                      {}}
```

```
        {{\]}   {DU}    {}                      {}}
        {{\^}   {DV}    {Aircraft}              {NUMBERED Aircraft}}
        {{\_}   {DW}    {WX site}               {NUMBERED WX site (green digi)}}
        {{\`}   {DX}    {Rain}                  {}}
        {{\a}   {SA}    {ARRL ARES etc}         {}}
        {{\b}   {SB}    {Blowing Dust/Sand}     {}}
        {{\c}   {SC}    {Civil Defense}         {NUMBERED Civil Defense (RACES)}}
        {{\d}   {SD}    {DX}                    {DX spot by callsign}}
        {{\e}   {SE}    {Sleet}                 {}}
        {{\f}   {SF}    {Funnel Cloud}          {}}
        {{\g}   {SG}    {Gale Flags}            {}}
        {{\h}   {SH}    {HAM store}             {}}
        {{\i}   {SI}    {}                      {Indoor BOXn digipeater (w overlay)}}
        {{\j}   {SJ}    {WorkZone}              {WorkZone (Steam Shovel)}}
        {{\k}   {SK}    {}                      {}}
        {{\l}   {SL}    {}                      {Area Locations (box,circles,etc)}}
        {{\m}   {SM}    {}                      {Value Signpost (3 digit display)}}
        {{\n}   {SN}    {TRIANGLE}              {NUMBERED TRIANGLE}}
        {{\o}   {SO}    {small circle}          {}}
        {{\p}   {SP}    {Partly Cloudy}         {}}
        {{\q}   {SQ}    {}                      {}}
        {{\r}   {SR}    {Restrooms}             {}}
        {{\s}   {SS}    {SHIP}                  {NUMBERED SHIP/boat (top view)}}
        {{\t}   {ST}    {Tornado}               {}}
        {{\u}   {SU}    {TRUCK}                 {NUMBERED TRUCK}}
        {{\v}   {SV}    {Van}                   {NUMBERED Van}}
        {{\w}   {SW}    {Flooding}              {}}
        {{\x}   {SX}    {}                      {}}
        {{\y}   {SY}    {}                      {}}
        {{\z}   {SZ}    {}                      {}}
        {"\\\173"       {Q1}    {Fog}           {}}
        {{\|}   {Q2}    {}                      {}}
        {"\\\175"       {Q3}    {}              {}}
        {{\~}   {Q4}    {}                      {}}
}

proc d7_get_APRS_symbol { x } {
        global d7_APRS_symbols

        foreach e $d7_APRS_symbols {
                if {[lindex $e 0] == $x} {
                        if [string length [lindex $e 2]] {
                                set x [lindex $e 2]
                        }
                        break
                }
                if {[lindex $e 1] == $x} {
                        if [string length [lindex $e 2]] {
                                set x [lindex $e 2]
                        }
                        break
                }
        }
        return $x
}

#
# "read" number we are attempting to read
#
set d7_AMSG_read_number 01

proc d7_format_latitude { l } {
        if {[string length $l] != 8} {
                return "00'00.000N"
        }
        set s "[string range $l 0 1]'[string range $l 2 3].[string range $l 4 6]"
        if {[string range $l 7 7] == "0"} {
                return ${s}N
        } else {
```

```
                return ${s}S
        }
}

proc d7_format_longitude { l } {
        if {[string length $l] != 9} {
                return "000'00.000E"
        }
        set s "[string range $l 0 2]'[string range $l 3 4].[string range $l 5 7]"
        if {[string range $l 8 8] == "0"} {
                return ${s}E
        } else {
                return ${s}W
        }
}

proc d7_latlong_to_d7 { lat long } {
        if [regexp {^[0-9][0-9]'[0-9][0-9]\.[0-9][0-9][0-9][NSns]$} $lat] {
        } else {
                set lat "00'00.000N"
        }
        if [regexp {^[0-9][0-9][0-9]'[0-9][0-9]\.[0-9][0-9][0-9][EWew]$} $long] {
        } else {
                set long "000'00.000E"
        }
        set la [string range $lat 0 1][string range $lat 3 4][string range $lat 6 8]
        if [regexp {[Nn]} [string range $lat 9 9]] {
                set la ${la}0
        } else {
                set la ${la}1
        }
        set lo [string range $long 0 2][string range $long 4 5][string range $long 7
9]
        if [regexp {[Ee]} [string range $long 10 10]] {
                set lo ${lo}0
        } else {
                set lo ${lo}1
        }
        return ${la}${lo}
}

set d7_latitude [d7_format_latitude 00000000]
set d7_longitude [d7_format_longitude 000000000]

proc d7_ARL_num { n } {
        if [regexp {[0-9][0-9][0-9][0-9]} $n] {
                return $n
        } else {
                if {[scan $n %d x] != 1} {
                        return 0000
                } elseif {$x < 0} {
                        return 0000
                } elseif {$x > 9999} {
                        return 0000
                }
                return [format %04d $x]
        }
}

proc d7_APRS_list_init { list x } {
        upvar $list l

        set l($x,callsign)              ""
        set l($x,latitude)              0
        set l($x,longitude)             0
        set l($x,icon)                  0
        set l($x,position)              ""
        set l($x,category)              0
        set l($x,overlay)               0
```

```
        set l($x,status)                ""
        set l($x,information)           ""
}

set d7_APRS_list_list { 01 02 03 04 05 06 07 08 09 10
                        11 12 13 14 15 16 17 18 19 20
                        21 22 23 24 25 26 27 28 29 30
                        31 32 33 34 35 36 37 38 39 40 }
set d7_APRS_list_fields { callsign latitude longitude icon position \
                        category overlay status information }
foreach x $d7_APRS_list_list {
        d7_APRS_list_init d7_APRS_list $x
        d7_APRS_list_init d7_APRS_old_list $x
}

set d7_APRS_messages_list { 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 }
foreach x $d7_APRS_messages_list {
        set d7_APRS_messages($x,category)       ""
        set d7_APRS_messages($x,callsign)       ""
        set d7_APRS_messages($x,message)        ""
        set d7_APRS_messages($x,number)         ""
}

proc d7_APRS_list_cmp { aa i bb j } {
        upvar #0 $aa a
        upvar #0 $bb b
        global d7_APRS_list_fields

        foreach f $d7_APRS_list_fields {
                if {$a($i,$f) != $b($j,$f)} {
                        return 0
                }
        }
        return 1
}

proc d7_APRS_copy_list { aa bb } {
        upvar $aa a
        upvar $bb b
        global d7_APRS_list_list
        global d7_APRS_list_fields

        foreach n $d7_APRS_list_list {
                foreach f $d7_APRS_list_fields {
                        set b($n,$f) $a($n,$f)
                }
        }
}

proc d7_APRS_list_in { a i b } {
        global d7_APRS_list_list

        foreach n $d7_APRS_list_list {
                if [d7_APRS_list_cmp $a $i $b $n] {
                        return 1
                }
        }
        return 0
}

proc d7_make_aprs_messages {} {
        global d7_APRS_messages_list
        global d7_APRS_messages

        set w [toplevel .aprs_messages]

        wm title $w "APRS Messages"

        frame $w.data
```

```
        foreach x $d7_APRS_messages_list {
                frame $w.data.$x

                label $w.data.$x.index -text $x
                label $w.data.$x.category -width 5 \
                        -textvariable d7_APRS_messages($x,category)
                label $w.data.$x.callsign -width 9 \
                        -textvariable d7_APRS_messages($x,callsign)
                label $w.data.$x.message -width 50 \
                        -textvariable d7_APRS_messages($x,message)
                label $w.data.$x.number -width 7 \
                        -textvariable d7_APRS_messages($x,number)

                pack \
                        $w.data.$x.index \
                        $w.data.$x.category \
                        $w.data.$x.callsign \
                        $w.data.$x.message \
                        $w.data.$x.number \
                        -side left
                pack \
                        $w.data.$x
        }
        button $w.refresh -text Refresh \
                -command "foreach x \$d7_APRS_messages_list {
                        set d7_APRS_messages(\$x,category) \"\"
                        set d7_APRS_messages(\$x,callsign) \"\"
                        set d7_APRS_messages(\$x,message) \"\"
                        set d7_APRS_messages(\$x,number) \"\"
                        set d7_AMSG_read_number \$x
                        d7_send_string \"AMSG \$x\"
                        sleep 0.1
                }"
        button $w.dismiss -text Dismiss -command "destroy $w"
        pack \
                $w.data \
                $w.refresh \
                $w.dismiss \
                -expand true
}

proc d7_make_aprs_history {} {
        global d7_APRS_list_list
        global d7_APRS_list
        global d7_val_width
        global d7_symbol_name_width

        set w [toplevel .aprs_history]

        wm title $w "APRS Message History"

        frame $w.data
        foreach x $d7_APRS_list_list {
                frame $w.data.$x

                label $w.data.$x.index -pady 0 -text $x
                label $w.data.$x.callsign -pady 0 -width 12 \
                        -textvariable d7_APRS_list($x,callsign)
                label $w.data.$x.latitude -pady 0 -width 10 \
                        -textvariable d7_APRS_list($x,latitude)
                label $w.data.$x.longitude -pady 0 -width 11 \
                        -textvariable d7_APRS_list($x,longitude)
                label $w.data.$x.icon -pady 0 -width $d7_symbol_name_width \
                        -textvariable d7_APRS_list($x,icon)
                label $w.data.$x.position -pady 0 \
                        -width $d7_val_width(position_comment) \
                        -textvariable d7_APRS_list($x,position)
                label $w.data.$x.category -pady 0 -width
$d7_val_width(APRS_category) \
```

```
                        -textvariable d7_APRS_list($x,category)
                label $w.data.$x.overlay -pady 0 -width 3 \
                        -textvariable d7_APRS_list($x,overlay)
                label $w.data.$x.status -pady 0 -width 22 \
                        -textvariable d7_APRS_list($x,status)
                label $w.data.$x.information -pady 0 -width 24 \
                        -textvariable d7_APRS_list($x,information)

                pack \
                        $w.data.$x.index \
                        $w.data.$x.callsign \
                        $w.data.$x.latitude \
                        $w.data.$x.longitude \
                        $w.data.$x.icon \
                        $w.data.$x.position \
                        $w.data.$x.category \
                        $w.data.$x.overlay \
                        $w.data.$x.status \
                        $w.data.$x.information \
                        -side left

                pack $w.data.$x
        }

        button $w.refresh -text "Refresh" \
                -command "
                        d7_APRS_copy_list d7_APRS_list d7_APRS_old_list
                        foreach x \$d7_APRS_list_list {
                                d7_APRS_list_init d7_APRS_list \$x
                                d7_send_string \"LIST \$x\"
                                sleep 0.05
                        }"

        button $w.dismiss -text Dismiss -command "destroy $w"
        pack \
                $w.data \
                $w.refresh \
                $w.dismiss \
                -expand true
}

proc d7_make_aprs {} {
        global d7_ARL_width
        global d7_val_width

        set w [toplevel .aprs]

        wm title $w "APRS"

        frame $w.control
        frame $w.control.l1

        frame $w.control.l1.beacon
        checkbutton $w.control.l1.beacon.enable -text Beacon \
                -variable d7_parameter(BCN) \
                -command {d7_set_radio_parameter BCN $d7_parameter(BCN)} \
                -offvalue 0 -onvalue 1
        menubutton $w.control.l1.beacon.method \
                -textvariable d7_beacon_method \
                -menu $w.control.l1.beacon.method.menu
        d7_make_menu beacon_method \
                $w.control.l1.beacon.method.menu \
                d7_parameter(DTX) \
                {d7_set_radio_parameter DTX $d7_parameter(DTX)}
        pack \
                $w.control.l1.beacon.enable \
                $w.control.l1.beacon.method \
                -side left -expand false
```

```
frame $w.control.l1.interval
label $w.control.l1.interval.label -text "Transmit Interval"
menubutton $w.control.l1.interval.button \
        -textvariable d7_APRS_TXI \
        -menu $w.control.l1.interval.button.menu
d7_make_menu APRS_TXI \
        $w.control.l1.interval.button.menu \
        d7_parameter(TXI) \
        {d7_set_radio_parameter TXI $d7_parameter(TXI)}
label $w.control.l1.interval.units -text "Secs"
pack \
        $w.control.l1.interval.label \
        $w.control.l1.interval.button \
        $w.control.l1.interval.units \
        -side left -expand false

frame $w.control.l1.gps
label $w.control.l1.gps.label -text "GPS Unit"
menubutton $w.control.l1.gps.unit \
        -textvariable d7_gps_unit \
        -menu $w.control.l1.gps.unit.menu
d7_make_menu GPS \
        $w.control.l1.gps.unit.menu \
        d7_parameter(GU) \
        {d7_set_radio_parameter GU $d7_parameter(GU)}
pack \
        $w.control.l1.gps.label \
        $w.control.l1.gps.unit \
        -side left -expand true
pack \
        $w.control.l1.beacon \
        $w.control.l1.interval \
        $w.control.l1.gps \
        -side left -expand true -fill both

pack \
        $w.control.l1 \
        -expand true -fill both

frame $w.control.l2

frame $w.control.l2.position
label $w.control.l2.position.label -text "My Position"
entry $w.control.l2.position.latitude -width 10 \
        -textvariable d7_latitude
entry $w.control.l2.position.longitude -width 11 \
        -textvariable d7_longitude
button $w.control.l2.position.set -text Set \
        -command {
                d7_set_radio_parameter MP [d7_latlong_to_d7 \
                        $d7_latitude $d7_longitude]
        }
pack \
        $w.control.l2.position.label \
        $w.control.l2.position.latitude \
        $w.control.l2.position.longitude \
        $w.control.l2.position.set \
        -side left -expand true

frame $w.control.l2.posc
label $w.control.l2.posc.label -text "Position Comment"
menubutton $w.control.l2.posc.button \
        -menu $w.control.l2.posc.button.menu \
        -textvariable d7_position_comment
d7_make_menu position_comment \
        $w.control.l2.posc.button.menu \
        d7_parameter(POSC) \
        {d7_set_radio_parameter POSC $d7_parameter(POSC)}
pack  \
```

```
        $w.control.l2.posc.label \
        $w.control.l2.posc.button \
        -side left -expand false

pack \
        $w.control.l2.position \
        $w.control.l2.posc \
        -side left -expand true -fill both

pack \
        $w.control.l2 \
        -expand true -fill both

frame $w.control.l3

frame $w.control.l3.callsign
label $w.control.l3.callsign.label -text Callsign
entry $w.control.l3.callsign.entry -width 9 \
        -textvariable d7_parameter(MYC)
button $w.control.l3.callsign.set -text Set \
        -command {d7_set_radio_parameter MYC $d7_parameter(MYC)}
pack \
        $w.control.l3.callsign.label \
        $w.control.l3.callsign.entry \
        $w.control.l3.callsign.set \
        -side left -expand false

frame $w.control.l3.stat
label $w.control.l3.stat.label -text Status
entry $w.control.l3.stat.status \
        -textvariable d7_parameter(STAT)
button $w.control.l3.stat.set -text Set \
        -command {d7_set_radio_parameter STAT $d7_parameter(STAT)}
pack \
        $w.control.l3.stat.label \
        $w.control.l3.stat.status \
        $w.control.l3.stat.set \
        -side left -expand false

pack \
        $w.control.l3.callsign \
        $w.control.l3.stat \
        -side left -expand true -fill both
pack \
        $w.control.l3 \
        -expand true -fill both

frame $w.control.l4

frame $w.control.l4.pos_limit
label $w.control.l4.pos_limit.label -text "Pos Limit"
entry $w.control.l4.pos_limit.entry -width $d7_ARL_width \
        -textvariable d7_parameter(ARL)
label $w.control.l4.pos_limit.unit -text km
button $w.control.l4.pos_limit.set -text "Set" \
        -command {d7_set_radio_parameter ARL [d7_ARL_num $d7_parameter(ARL)]}
pack \
        $w.control.l4.pos_limit.label \
        $w.control.l4.pos_limit.entry \
        $w.control.l4.pos_limit.unit \
        $w.control.l4.pos_limit.set \
        -side left -expand false

frame $w.control.l4.unit
label $w.control.l4.unit.label -text Units
pack \
        $w.control.l4.unit.label \
        -side left -expand true
radiobutton $w.control.l4.unit.metric -text "Celsius/Kilometers" \
```

```
                -variable d7_parameter(UNIT) \
                -command {d7_set_radio_parameter UNIT $d7_parameter(UNIT)} \
                -value 1
        radiobutton $w.control.l4.unit.imperial -text "Fahrenheight/Miles" \
                -variable d7_parameter(UNIT) \
                -command {d7_set_radio_parameter UNIT $d7_parameter(UNIT)} \
                -value 0
        pack \
                $w.control.l4.unit.metric \
                $w.control.l4.unit.imperial \
                -side left -expand true

        pack \
                $w.control.l4.pos_limit \
                $w.control.l4.unit \
                -side left -expand true -fill both
        pack \
                $w.control.l4 \
                -expand true -fill both

        frame $w.control.l5

        frame $w.control.l5.icon
        label $w.control.l5.icon.label -text Icon
        entry $w.control.l5.icon.mode -width 1 \
                -textvariable d7_icon_mode
        entry $w.control.l5.icon.data -width 2 \
                -textvariable d7_icon_data
        button $w.control.l5.icon.set -text set \
                -command {d7_send_string "ICO $d7_icon_mode,$d7_icon_data"}
        pack \
                $w.control.l5.icon.label \
                $w.control.l5.icon.mode \
                $w.control.l5.icon.data \
                $w.control.l5.icon.set \
                -side left -expand false

        frame $w.control.l5.unprotocol
        label $w.control.l5.unprotocol.label -text Unprotocol
        entry $w.control.l5.unprotocol.entry -width 9 -textvariable d7_parameter(UPR)
        button $w.control.l5.unprotocol.set -text Set \
                -command {d7_set_radio_parameter UPR $d7_parameter(UPR)}
        pack \
                $w.control.l5.unprotocol.label \
                $w.control.l5.unprotocol.entry \
                $w.control.l5.unprotocol.set \
                -side left -expand false
        pack \
                $w.control.l5.icon \
                $w.control.l5.unprotocol \
                -side left -expand true -fill both
        pack \
                $w.control.l5 \
                -expand true -fill both

        frame $w.sendmessage
        label $w.sendmessage.label -text Message
        entry $w.sendmessage.callsign -width 9 \
                -textvariable d7_APRS_message_callsign
        entry $w.sendmessage.message -width 45 \
                -textvariable d7_APRS_message_message
        button $w.sendmessage.send -text Send \
                -command {d7_send_string "AMSG
00,$d7_APRS_message_callsign,$d7_APRS_message_message"}
        pack \
                $w.sendmessage.label \
                $w.sendmessage.callsign \
                $w.sendmessage.message \
                $w.sendmessage.send \
```

```
                -side left -expand false

        frame $w.control.l7

        frame $w.control.l7.path
        label $w.control.l7.path.label -text Path
        entry $w.control.l7.path.entry -textvariable d7_parameter(PP) -width 32
        button $w.control.l7.path.set -text Set \
                -command {d7_set_radio_parameter PP $d7_parameter(PP)}

        pack \
                $w.control.l7.path.label \
                $w.control.l7.path.entry \
                $w.control.l7.path.set \
                -side left -expand true


        pack \
                $w.control.l7.path \
                -side left -expand true
        pack \
                $w.control.l7 \
                -expand true -fill both

        frame $w.data

        button $w.data.messages -text Messages \
                -command "d7_function_window aprs_messages"
        button $w.data.history -text History \
                -command "d7_function_window aprs_history"

        pack \
                $w.data.messages \
                $w.data.history \
                -side left

        button $w.dismiss -text Dismiss -command "destroy $w"
        pack \
                $w.control \
                $w.sendmessage \
                $w.data \
                $w.dismiss \
                -expand true
}

#
# SSTV
#
proc d7_make_sstv {} {
        set w [toplevel .sstv]

        wm title $w "SSTV"

        frame $w.call_colour
        label $w.call_colour.label -text "My Call Colour"
        menubutton $w.call_colour.button \
                -textvariable d7_call_colour \
                -menu $w.call_colour.button.menu
        d7_make_menu colour \
                $w.call_colour.button.menu \
                d7_parameter(MAC) \
                {d7_set_radio_parameter MAC $d7_parameter(MAC)}
        pack \
                $w.call_colour.label \
                $w.call_colour.button \
                -side left
        frame $w.message_colour
        label $w.message_colour.label -text "Message Colour"
        menubutton $w.message_colour.button \
```

```
        -textvariable d7_message_colour \
        -menu $w.message_colour.button.menu
d7_make_menu colour \
        $w.message_colour.button.menu \
        d7_parameter(SMC) \
        {d7_set_radio_parameter SMC $d7_parameter(SMC)}
pack \
        $w.message_colour.label \
        $w.message_colour.button \
        -side left
frame $w.report
label $w.report.label -text "Signal Report"
entry $w.report.entry -textvariable d7_parameter(RSV)
button $w.report.set -text Set \
        -command {d7_set_radio_parameter RSV $d7_parameter(RSV)}
pack \
        $w.report.label \
        $w.report.entry \
        $w.report.set \
        -side left
frame $w.callsign
label $w.callsign.label -text "My Call Sign"
entry $w.callsign.entry -textvariable d7_parameter(SMY)
button $w.callsign.set -text Set \
        -command {d7_set_radio_parameter SMY $d7_parameter(SMY)}
pack \
        $w.callsign.label \
        $w.callsign.entry \
        $w.callsign.set \
        -side left
frame $w.ccallsign
label $w.ccallsign.label -text "Commander Call Sign"
entry $w.ccallsign.entry -textvariable d7_parameter(SCC)
button $w.ccallsign.set -text Set \
        -command {d7_set_radio_parameter SCC $d7_parameter(SCC)}
pack \
        $w.ccallsign.label \
        $w.ccallsign.entry \
        $w.ccallsign.set \
        -side left
frame $w.tcallsign
label $w.tcallsign.label -text "Transporter Call Sign"
entry $w.tcallsign.entry -textvariable d7_parameter(SCT)
button $w.tcallsign.set -text Set \
        -command {d7_set_radio_parameter SCT $d7_parameter(SCT)}
pack \
        $w.tcallsign.label \
        $w.tcallsign.entry \
        $w.tcallsign.set \
        -side left
frame $w.tone
label $w.tone.label -text Tone
menubutton $w.tone.button \
        -textvariable d7_sky_command_tone \
        -menu $w.tone.button.menu
d7_make_menu tone \
        $w.tone.button.menu \
        d7_parameter(SKTN) \
        {d7_set_radio_parameter SKTN $d7_parameter(SKTN)}

frame $w.message
label $w.message.label -text Message
entry $w.message.entry -textvariable d7_parameter(SMSG)
button $w.message.set -text Set \
        -command {d7_set_radio_parameter SMSG $d7_parameter(SMSG)}
pack \
        $w.message.label \
        $w.message.entry \
        $w.message.set \
```

```
                -side left
        checkbutton $w.shutter -text "VC Shutter" \
                -variable d7_parameter(VCS) \
                -offvalue 0 -onvalue 1 \
                -command {d7_set_radio_parameter VCS $d7_parameter(VCS)}
        button $w.dismiss -text Dismiss -command "destroy $w"
        pack \
                $w.call_colour \
                $w.message_colour \
                $w.report \
                $w.callsign \
                $w.ccallsign \
                $w.tcallsign \
                $w.message \
                $w.shutter \
                $w.dismiss \
                -expand true
}

#
# TNC
#

set d7_tnc_paramter_list {
        BEACON-inter
        BEACON-num
        BTEXT
        CHECK
        CMSG
        CMSGDISC
        CONOK
        DWAIT
        FIRMRNR
        FRACK
        GBAUD
        GPSTEXT
        HBAUD
        LOCATION-inter
        LOCATION-num
        LPATH
        LTEXT
        LTMHEAD
        LTMON
        MAXFRAME
        MYCALL
        NTSGRP
        NTSMRK
        NTSMSG
        PACLEN
        PACTIME-inter
        PACTIME-num
        PASSALL
        PERSIST
        PPERSIST
        RESPTIME
        RETRY
        SENDPAC
        SLOTTIME
        SOFTDCD
        TRIES
        TXDELAY
        TXUIFRAM
        UNPROTO
        XMITOK
}

foreach p $d7_tnc_paramter_list {
        set d7_tnc_paramter($p) 0
}
```

```
proc d7_tnc_checkbutton { w text parm short } {
        checkbutton $w -text $text \
                -variable d7_tnc_paramter($parm) \
                -onvalue ON \
                -offvalue OFF \
                -command "d7_send_string_tnc \"$short \$d7_tnc_paramter($parm)\""
}

proc d7_tnc_numvalue { w text parm short } {
        frame $w
        label $w.label -text $text
        entry $w.entry -width 3 \
                -textvariable d7_tnc_paramter($parm)
        pack \
                $w.label \
                $w.entry \
                -side left
}

proc d7_tnc_textentry { w text parm short } {
        frame $w
        label $w.label -text $text
        entry $w.entry \
                -textvariable d7_tnc_paramter($parm)
        pack \
                $w.label \
                $w.entry \
                -side left
}

proc d7_tnc_internum { w text parm short } {
        frame $w
        label $w.label -text $text
        menubutton $w.inter \
                -menu $w.inter.menu
        menu $w.inter.menu -tearoff 0
        foreach n { EVERY AFTER } {
                $w.inter.menu add radiobutton \
                        -label $n \
                        -variable d7_tnc_paramter(${parm}-inter) \
                        -value $n \
                        -command "d7_send_string_tnc \"$short $n
d7_tnc_paramter(${parm}-num\" ; d7_send_string_tnc $short"
        }
        entry $w.num -width 3 \
                -textvariable d7_tnc_paramter(${parm}-num)
        pack \
                $w.label \
                $w.inter \
                $w.num \
                -side left
}

proc d7_make_tnc {} {
        set w [toplevel .tnc]

        wm title $w "TNC"

        frame $w.v1
        frame $w.v1.basic
        label $w.v1.basic.label -text Basic
        frame $w.v1.basic.baud
        label $w.v1.basic.baud.label -text Baud
        menubutton $w.v1.basic.baud.menu -textvariable d7_tnc_paramter(HBAUD) \
                -menu $w.v1.basic.baud.menu.menu
        menu $w.v1.basic.baud.menu.menu -tearoff 0
        foreach n { 1200 9600 } {
                $w.v1.basic.baud.menu.menu add radiobutton \
```

```
                        -label $n \
                        -variable d7_tnc_paramter(HBAUD) \
                        -value $n \
                        -command "d7_send_string_tnc \"HB $n\" ; d7_send_string_tnc
HB"
        }
        pack \
                $w.v1.basic.baud.label \
                $w.v1.basic.baud.menu \
                -side left

        d7_tnc_checkbutton $w.v1.basic.xmitok "XMIT OK" XMITOK XMITOK
        d7_tnc_checkbutton $w.v1.basic.loop Loopback LOOP LOOP
        d7_tnc_numvalue $w.v1.basic.ppersist P-persist PPERSIST PP
        d7_tnc_numvalue $w.v1.basic.persist Persist PERSIST PE
        d7_tnc_numvalue $w.v1.basic.slottime Slottime SLOTTIME SL
        d7_tnc_numvalue $w.v1.basic.dwait Dwait DWAIT DW
        d7_tnc_numvalue $w.v1.basic.txdelay "TX Delay" TXDELAY TX
        d7_tnc_checkbutton $w.v1.basic.softdcd "Soft DCD" SOFTDCD SOFTDCD
        pack \
                $w.v1.basic.label \
                $w.v1.basic.baud \
                $w.v1.basic.xmitok \
                $w.v1.basic.ppersist \
                $w.v1.basic.persist \
                $w.v1.basic.slottime \
                $w.v1.basic.dwait \
                $w.v1.basic.txdelay \
                $w.v1.basic.softdcd \
                $w.v1.basic.loop

        frame $w.v1.rx
        label $w.v1.rx.label -text RX
        d7_tnc_checkbutton $w.v1.rx.passall "Pass All" PASSALL PASSA
        d7_tnc_numvalue $w.v1.rx.resptime "Response Time" RESPTIME RES
        d7_tnc_numvalue $w.v1.rx.frack "Frame Ack" FRACK FR
        d7_tnc_numvalue $w.v1.rx.retry Retries RETRY RE
        d7_tnc_numvalue $w.v1.rx.tries Tries TRIES TRI
        d7_tnc_numvalue $w.v1.rx.check Check CHECK CH
        d7_tnc_checkbutton $w.v1.rx.cmsg "Connection Message" CMSG CMS
        d7_tnc_checkbutton $w.v1.rx.cmsgd "Disconnection Message" CMSGDISC CMSGD
        d7_tnc_checkbutton $w.v1.rx.conok "Connection OK" CONOK CONO
        pack \
                $w.v1.rx.label \
                $w.v1.rx.passall \
                $w.v1.rx.resptime \
                $w.v1.rx.frack \
                $w.v1.rx.retry \
                $w.v1.rx.tries \
                $w.v1.rx.check \
                $w.v1.rx.cmsg \
                $w.v1.rx.cmsgd \
                $w.v1.rx.conok

        frame $w.v1.notcon
        d7_tnc_checkbutton $w.v1.notcon.txu "Unconnected TX" TXUIFRAM TXU
        d7_tnc_textentry $w.v1.notcon.unproto Unprotocol UNPROTO U
        d7_tnc_internum $w.v1.notcon.beacon "Send beacon" BEACON B
        d7_tnc_textentry $w.v1.notcon.btext "Beacon Text" BTEXT BT
        pack \
                $w.v1.notcon.txu \
                $w.v1.notcon.unproto \
                $w.v1.notcon.beacon \
                $w.v1.notcon.btext

        frame $w.v1.generate
        d7_tnc_textentry $w.v1.generate.sendpack "Send Char" SENDPAC SE
        d7_tnc_checkbutton $w.v1.generate.cr "Add CR to Packets" CR CR
        d7_tnc_numvalue $w.v1.generate.paclen "Packet Length" PACLEN P
```

```
        d7_tnc_internum $w.v1.generate.pactime Packtime PACTIME PACT
        d7_tnc_checkbutton $w.v1.generate.cpacktime "Packtime in Converse" CPACTIME
CP
        frame $w.v1.generate.maxframe
        label $w.v1.generate.maxframe.label -text "Max Frames"
        menubutton $w.v1.generate.maxframe.menu -textvariable
d7_tnc_paramter(MAXFRAME) \
                -menu $w.v1.generate.maxframe.menu.menu
        menu $w.v1.generate.maxframe.menu.menu -tearoff 0
        foreach n { 1 2 3 4 5 6 7 } {
                $w.v1.generate.maxframe.menu.menu add radiobutton \
                        -label $n \
                        -variable d7_tnc_paramter(MAXFRAME) \
                        -value $n \
                        -command "d7_send_string_tnc \"MAX $n\" ; d7_send_string_tnc
MAX"
        }
        pack \
                $w.v1.generate.maxframe.label \
                $w.v1.generate.maxframe.menu \
                -side left
        pack \
                $w.v1.generate.sendpack \
                $w.v1.generate.cr \
                $w.v1.generate.paclen \
                $w.v1.generate.pactime \
                $w.v1.generate.cpacktime \
                $w.v1.generate.maxframe

        frame $w.v1.monitor
        d7_tnc_checkbutton $w.v1.monitor.monitor "Monitor" MONITOR M
        d7_tnc_checkbutton $w.v1.monitor.mcom "Monitor All" MCOM MCOM
        d7_tnc_checkbutton $w.v1.monitor.mcon "Monitor While Connected" MCON MC
        d7_tnc_checkbutton $w.v1.monitor.mall "Monitor All Stations" MALL MA
        d7_tnc_checkbutton $w.v1.monitor.mrpt "Path in Header" MRPT MR
        d7_tnc_checkbutton $w.v1.monitor.trace "Detail" TRACE TRAC
        pack \
                $w.v1.monitor.monitor \
                $w.v1.monitor.mcom \
                $w.v1.monitor.mcon \
                $w.v1.monitor.mall \
                $w.v1.monitor.mrpt \
                $w.v1.monitor.trace

        frame $w.v1.gps
        label $w.v1.gps.label -text GPS
        frame $w.v1.gps.baud
        label $w.v1.gps.baud.label -text Baud
        menubutton $w.v1.gps.baud.menu -textvariable d7_tnc_paramter(GBAUD) \
                -menu $w.v1.gps.baud.menu.menu
        menu $w.v1.gps.baud.menu.menu -tearoff 0
        foreach n { 4800 9600 } {
                $w.v1.gps.baud.menu.menu add radiobutton \
                        -label $n \
                        -variable d7_tnc_paramter(GBAUD) \
                        -value $n \
                        -command "d7_send_string_tnc \"GB $n\" ; d7_send_string_tnc
GB"
        }
        pack \
                $w.v1.gps.baud.label \
                $w.v1.gps.baud.menu \
                -side left

        d7_tnc_textentry $w.v1.gps.lpath Path LPATH LPA
        d7_tnc_internum $w.v1.gps.location Location LOCATION LOC
        d7_tnc_textentry $w.v1.gps.ltext "Location Text" LTEXT LT
        d7_tnc_numvalue $w.v1.gps.ltmon "Monitor Interval" LTMON LTM
        d7_tnc_checkbutton $w.v1.gps.ltmhead "Add Header" LTMHEAD LTMH
```

```
        d7_tnc_textentry $w.v1.gps.gpstext "GPS Text Indicator" GPSTEXT GPST
        d7_tnc_textentry $w.v1.gps.ntsgrp "NTS Group Code" NTSGRP NTSGRP
        d7_tnc_numvalue $w.v1.gps.ntsmrk "NTS Icon" NTSMRK NTSMRK
        d7_tnc_textentry $w.v1.gps.ntsmsg "NTS Message" NTSMSG NTSMSG
        pack \
                $w.v1.gps.label \
                $w.v1.gps.baud \
                $w.v1.gps.lpath \
                $w.v1.gps.location \
                $w.v1.gps.ltext \
                $w.v1.gps.ltmon \
                $w.v1.gps.ltmhead \
                $w.v1.gps.gpstext \
                $w.v1.gps.ntsgrp \
                $w.v1.gps.ntsmrk \
                $w.v1.gps.ntsmsg

        pack \
                $w.v1.basic \
                $w.v1.rx \
                $w.v1.notcon \
                $w.v1.generate \
                $w.v1.monitor \
                $w.v1.gps \
                -side left -expand true -fill both

        frame $w.connect
        entry $w.connect.target
        button $w.connect.connect -text Connect
        button $w.connect.disconnect -text Disonnect
        pack \
                $w.connect.target \
                $w.connect.connect \
                $w.connect.disconnect \
                -side left -expand true -fill both

        frame $w.general
        button $w.general.restart -text Restart \
                -command "d7_send_string_tnc RESTART"
        button $w.general.reset -text Reset \
                -command "d7_send_string_tnc RESET"
        button $w.general.dismiss -text Dismiss -command "destroy $w"
        pack \
                $w.general.restart \
                $w.general.reset \
                $w.general.dismiss \
                -side left
        pack \
                $w.v1 \
                $w.connect \
                $w.general \
                -expand true
}

#
# Scanning
#

set d7_scan_list(0,low)         118000000
set d7_scan_list(0,high)        136000000
set d7_scan_list(1,low)         136000000
set d7_scan_list(1,high)        174000000
set d7_scan_list(2,low)         400000000
set d7_scan_list(2,high)        480000000

foreach x { 0 1 2 } {
        set d7_scan_list($x,lowMHz)     [d7_Hz_to_MHz $d7_scan_list(0,low)]
        set d7_scan_list($x,highMHz)    [d7_Hz_to_MHz $d7_scan_list(0,high)]
        set d7_scan_list($x,widget)     0
```

```
        set d7_scan_list($x,start)      0
        set d7_scan_list($x,end)        0
        set d7_scan_list($x,cursor)     0
        set d7_scan_list($x,cursorxpos) 0
}

set d7_scan_selected    000.00000

proc d7_set_freq { f } {
        global d7_parameter
        global d7_step

        set f [format %011d $f]
        d7_send_string "FQ $f,$d7_step($d7_parameter(BC))"
}

proc d7_graph_freq_pos { low high f width } {

        set sd [expr ( $high - $low ) / 1000]
        set fd [expr ( $f - $low ) / 1000]
        return [expr ( $fd * $width ) / $sd]
}

proc d7_graph_freq_line { n f } {
        global d7_scan_list
        global d7_scanner_graph_xmargin
        global d7_scanner_graph_height
        global d7_scanner_graph_topmargin
        global d7_scanner_graph_width

        set x [d7_graph_freq_pos $d7_scan_list($n,low) $d7_scan_list($n,high) $f
$d7_scanner_graph_width]
        set x [expr $x + $d7_scanner_graph_xmargin]

        $d7_scan_list($n,widget).scan create line \
                $x \
                [expr $d7_scanner_graph_topmargin + $d7_scanner_graph_height] \
                $x \
                [expr $d7_scanner_graph_topmargin + $d7_scanner_graph_height + 10]
        $d7_scan_list($n,widget).scan create text \
                $x \
                [expr $d7_scanner_graph_topmargin + $d7_scanner_graph_height + 20] \
                -text [format %6.2f [expr $f / 1000000.0]]
}

proc d7_graph_make_scale { n } {
        global d7_scan_list

        set d [expr $d7_scan_list($n,high) - $d7_scan_list($n,low)]
        set l [string length $d]
        set dd ""
        for {set i 2} {$i < $l} {incr i} {
                set dd 0$dd
        }
        set gd 1$dd
        if {$d / $gd > 70} {
                set gd [expr $gd * 10]
        }
        if {$d / $gd > 50} {
                set gd [expr $gd * 5]
        }
        if {$d / $gd > 20} {
                set gd [expr $gd * 2.5]
        }
        if {$d / $gd > 10} {
                set gd [expr $gd * 2]
        }
        set start [string range $d7_scan_list($n,low) 0 1][string range 000000000000
\
```

```
                3 [string length $d7_scan_list($n,low)]]
        for { set f $start} \
            { $f <= $d7_scan_list($n,high) } \
            {set f [expr $f + $gd] } {
                if {$f < $d7_scan_list($n,low)} continue
                d7_graph_freq_line $n $f
        }
}

proc d7_update_graph { w n f v } {
        global d7_scan_list
        global d7_on_indicator_colour
        global d7_scanner_graph_topmargin
        global d7_scanner_graph_height
        global d7_scanner_graph_width
        global d7_scanner_graph_xmargin

        set newpos [d7_graph_freq_pos $d7_scan_list($n,low) $d7_scan_list($n,high)
$f $d7_scanner_graph_width]
        set newpos [expr $newpos + $d7_scanner_graph_xmargin]
        set x [expr $newpos - $d7_scan_list($n,cursorxpos)]
        set d7_scan_list($n,cursorxpos) $newpos

        if {$x != 0 } {
                $d7_scan_list($n,widget).scan move $d7_scan_list($n,cursor) $x 0
        }
        $d7_scan_list($n,widget).scan delete t$f
        if {$v > 0} {
                set id [$d7_scan_list($n,widget).scan create line \
                        $newpos \
                        [expr $d7_scanner_graph_topmargin +
$d7_scanner_graph_height] \
                        $newpos \
                        [expr ($d7_scanner_graph_topmargin +
$d7_scanner_graph_height) - ( $v * 10 )] \
                        -fill $d7_on_indicator_colour]
                $d7_scan_list($n,widget).scan addtag t$f withtag $id
                $d7_scan_list($n,widget).scan bind t$f \
                        <ButtonPress-1> "d7_set_freq $f"
                $d7_scan_list($n,widget).scan bind t$f \
                        <ButtonPress-2> "set d7_scan_selected [d7_Hz_to_MHz $f]"
        }
}

proc d7_update_scan { freq val } {
        global d7_scan_list

        set f 0
        scan $freq %d f
        foreach x { 0 1 2 } {
                if {($f >= $d7_scan_list($x,low)) && ($f <= $d7_scan_list($x,high))}
{
                        if [winfo exists $d7_scan_list($x,widget)] {
                                d7_update_graph $d7_scan_list($x,widget) $x $f $val
                        }
                        break
                }
        }
}

proc d7_fill_scanner { n } {
        global d7_scan_list
        global d7_scanner_graph_width
        global d7_scanner_graph_xmargin
        global d7_scanner_graph_topmargin
        global d7_scanner_graph_height

        set w $d7_scan_list($n,widget)
```

```
        $w.scan create line \
                $d7_scanner_graph_xmargin \
                $d7_scanner_graph_topmargin \
                [expr $d7_scanner_graph_xmargin + $d7_scanner_graph_width] \
                $d7_scanner_graph_topmargin
        $w.scan create line \
                $d7_scanner_graph_xmargin \
                [expr $d7_scanner_graph_topmargin + $d7_scanner_graph_height] \
                [expr $d7_scanner_graph_xmargin + $d7_scanner_graph_width] \
                [expr $d7_scanner_graph_topmargin + $d7_scanner_graph_height]
        $w.scan create line \
                $d7_scanner_graph_xmargin \
                $d7_scanner_graph_topmargin \
                $d7_scanner_graph_xmargin \
                [expr $d7_scanner_graph_topmargin + $d7_scanner_graph_height]
        $w.scan create line \
                [expr $d7_scanner_graph_xmargin + $d7_scanner_graph_width] \
                $d7_scanner_graph_topmargin \
                [expr $d7_scanner_graph_xmargin + $d7_scanner_graph_width] \
                [expr $d7_scanner_graph_topmargin + $d7_scanner_graph_height]
        set d7_scan_list($n,cursor) [$w.scan create line \
                $d7_scanner_graph_xmargin \
                [expr $d7_scanner_graph_topmargin + $d7_scanner_graph_height] \
                $d7_scanner_graph_xmargin \
                [expr $d7_scanner_graph_topmargin + $d7_scanner_graph_height + 10] \
                -fill red]
        set d7_scan_list($n,cursorxpos) $d7_scanner_graph_xmargin
        d7_graph_make_scale $n
}

proc d7_make_scan {} {
        global d7_scan_list
        global d7_vfo_list
        global d7_scanner_graph_xmargin
        global d7_scanner_graph_topmargin
        global d7_scanner_graph_bottommargin
        global d7_scanner_graph_height
        global d7_scanner_graph_width
        global d7_frequency_width

        set w [toplevel .scan]

        wm title $w "Scan"

        set n 0

        canvas $w.scan \
                -height [expr $d7_scanner_graph_height + \
                        $d7_scanner_graph_topmargin + \
                        $d7_scanner_graph_bottommargin] \
                -width [expr 2 * $d7_scanner_graph_xmargin + \
                        $d7_scanner_graph_width]
        set d7_scan_list($n,widget) $w

        d7_fill_scanner $n

        pack $w.scan
        frame $w.freqs
        entry $w.freqs.low -width $d7_frequency_width \
                -textvariable d7_scan_list($n,lowMHz)
        bind $w.freqs.low <Key-Return> "
                set d7_scan_list($n,low) \[d7_MHz_to_Hz \$d7_scan_list($n,lowMHz)\]
                set d7_scan_list($n,lowMHz) \[d7_Hz_to_MHz \$d7_scan_list($n,low)\]
                $w.scan delete all
                d7_fill_scanner $n
        "
        label $w.freqs.select -width $d7_frequency_width \
                -textvariable d7_scan_selected
        entry $w.freqs.high -width $d7_frequency_width \
```

```
                 -textvariable d7_scan_list($n,highMHz)
        bind $w.freqs.high <Key-Return> "
                set d7_scan_list($n,high) \[d7_MHz_to_Hz \$d7_scan_list($n,highMHz)\]
                set d7_scan_list($n,highMHz) \[d7_Hz_to_MHz \$d7_scan_list($n,high)\]
                $w.scan delete all
                d7_fill_scanner $n
        "
        pack \
                $w.freqs.low \
                $w.freqs.select \
                $w.freqs.high \
                -side left -expand true -fill none
        pack $w.freqs -expand true -fill both

        frame $w.vfo
        foreach vfo $d7_vfo_list {
                button $w.vfo.$vfo -text "VFO $vfo Range" -command "
                        set d7_scan_list($n,low) \[expr \$d7_vfo($vfo,low) *
1000000\]
                        set d7_scan_list($n,high) \[expr (\$d7_vfo($vfo,high) + 1) *
1000000\]
                        set d7_scan_list($n,lowMHz) \[d7_Hz_to_MHz
\$d7_scan_list($n,low)\]
                        set d7_scan_list($n,highMHz) \[d7_Hz_to_MHz
\$d7_scan_list($n,high)\]
                        $w.scan delete all
                        d7_fill_scanner $n
                "

                pack $w.vfo.$vfo -side left -expand true -fill none
        }
        pack $w.vfo -expand true
        button $w.dismiss -text Dismiss -command "destroy $w"
        pack $w.dismiss -side left -expand true
}

#
# General Radio Funcions
#

proc d7_log { inout string } {
        if [winfo exists .log.log.text] {
                .log.log.text insert end "$string\n" $inout
                .log.log.text see end
        }
}

proc d7_log_trace { v e type } {
        upvar #0 $v var
        d7_log in "$var($e)"

        # Fix this ??
        # global $v
        # d7_log in "$v($e)"
}

#
# return 123.45678 format MHz from 00123456780 Hz
#
proc d7_d7freq_to_MHz { f } {
        return [string range $f 2 4].[string range $f 5 9]
}

#
# return 12.34 format MHz from 012340000 Hz
#
proc d7_d7offset_to_MHz { f } {
        return [string range $f 1 2].[string range $f 3 4]
}
```

```
#
# return 00123456780 Hz format form 123.45678 format MHz
#
proc d7_MHz_to_d7freq { m } {
        if [regexp {^[0-9][0-9]0-9]\.[0-9][0-9]0-9][0-9]0-9]$} $m] {
        } else {
                if {[scan $m %f f] != 1} {
                        set m 000.00000
                } else {
                        set m [format %09.5f $f]
                }
        }
        return 00[string range $m 0 2][string range $m 4 8]0
}

#
# return 012340000 Hz from 12.34 format MHz
#
proc d7_MHz_to_d7offset { m } {
        if [regexp {^[0-9]0-9]\.[0-9]0-9]$} $m] {
        } else {
                if {[scan $m %f f] != 1} {
                        set m 00.00
                } else {
                        set m [format %05.2f $f]
                }
        }
        return 0[string range $m 0 1][string range $m 3 4]0000
}

proc d7_switch_modes { newmode } {
        global d7_mode

        if {($d7_mode == "radio") && ($newmode == "tnc")} {
                d7_send_string "TC 0"
        } elseif {($d7_mode == "tnc") && ($newmode == "radio")} {
                d7_send_string_tnc "TC 1"
        }
}

proc d7_set_radio {} {
        global d7_mode
        global d7_spawn_id
        global spawn_id

        set d7_mode radio
        set d7_selected_mode radio

        trace variable expect_out(0,string) w d7_log_trace

        expect_background {
                -i $d7_spawn_id
                -re "AI (.)\r" {
                        set d7_parameter(AI) $expect_out(1,string)
                }
                -re "AIP (.)\r" {
                        set d7_parameter(AIP) $expect_out(1,string)
                }
                -re "AMSG (\[^\r,\]*),(\[^\r,\]*),(\[^\r,\]*),(\[^\r,\]*)\r" {
                        # hmm, what format is message number?
                        set i $d7_AMSG_read_number
                        set d7_APRS_messages($i,category) $expect_out(1,string)
                        set d7_APRS_messages($i,callsign) $expect_out(2,string)
                        set d7_APRS_messages($i,message) $expect_out(3,string)
                        set d7_APRS_messages($i,number) $expect_out(4,string)
                }
                -re "APO (.)\r" {
                        set d7_parameter(APO) $expect_out(1,string)
```

```
                        set d7_APO_delay $d7_APO_codes($d7_parameter(APO))
                }
                -re "APO (.),.\r" {
                        set d7_parameter(APO) $expect_out(1,string)
                        set d7_APO_delay $d7_APO_codes($d7_parameter(APO))
                }
                -re "ARO (.)\r" {
                        set d7_parameter(ARO) $expect_out(1,string)
                }
                -re "ARL (....)\r" {
                        set d7_parameter(ARL) $expect_out(1,string)
                }
                -re "ASC (.),(.)\r" {
                        set d7_asc_selected($expect_out(1,string))
$expect_out(2,string)
                        .tune.asc.$expect_out(1,string) configure -bg gray
                }
                -re "ASC (.),(.),(.)\r" {
                        set d7_asc_selected($expect_out(1,string))
$expect_out(2,string)
                        if {$expect_out(3,string) == "1"} {
                                .tune.asc.$expect_out(1,string) configure -bg
$d7_on_indicator_colour
                        } else {
                                .tune.asc.$expect_out(1,string) configure -bg
$d7_off_indicator_colour
                        }
                }
                -re "BAL (.)\r" {
                        set d7_parameter(BAL) $expect_out(1,string)
                }
                -re "BC (.)\r" {
                        set d7_parameter(BC) $expect_out(1,string)
                        d7_update_ab_widgets
                }
                -re "BCN (.)\r" {
                        set d7_parameter(BCN) $expect_out(1,string)
                }
                -re "BEL (.),(.)\r" {
                }
                -re "BEP (.)\r" {
                        set d7_parameter(BEP) $expect_out(1,string)
                        set d7_beep $d7_beep_codes($d7_parameter(BEP))
                }
                -re "BUF (.),(..........),(\[0-9\]),(.),(.),(.),(.),,(..),,
(..),(.........),(\[01\])\r" {
                        set d7_frequency($expect_out(1,string)) \
                                $expect_out(2,string)
                        set d7_frequency_MHz($expect_out(1,string)) \
                                [d7_d7freq_to_MHz $expect_out(2,string)]
                        set d7_step($expect_out(1,string)) $expect_out(3,string)
                        set d7_step_KHz($expect_out(1,string)) \
                                $d7_step_codes($expect_out(3,string))

                        set d7_offset_direction($expect_out(1,string)) \
                                $d7_offset_direction_codes($expect_out(4,string))
                        set d7_reverse($expect_out(1,string)) \
                                $expect_out(5,string)
                        set d7_tone_enable($d7_parameter(BC)) \
                                $expect_out(6,string)
                        set d7_parameter(CT) $expect_out(7,string)

                        set d7_tone_freq($d7_parameter(BC)) \
                                $expect_out(8,string)
                        set d7_tone_freq_name($d7_parameter(BC)) \
                                $d7_tone_codes($expect_out(8,string))

                        set d7_ctcss_freq($d7_parameter(BC)) $expect_out(9,string)
                        set d7_ctcss_freq_name($d7_parameter(BC)) \
```

```
                                $d7_tone_codes($expect_out(9,string))

                        set d7_parameter(OS) $expect_out(10,string)
                        set d7_offset_MHz($expect_out(1,string)) \
                                [d7_d7offset_to_MHz $expect_out(10,string)]

                        if {$expect_out(1,string) == "0"} {
                                if {$expect_out(11,string) == "0"} {
                                        .tune.bandopt.band_a.mod.fm configure -bg
$d7_on_indicator_colour
                                        .tune.bandopt.band_a.mod.am configure -bg
$d7_off_indicator_colour
                                } else {
                                        .tune.bandopt.band_a.mod.fm configure -bg
$d7_off_indicator_colour
                                        .tune.bandopt.band_a.mod.am configure -bg
$d7_on_indicator_colour
                                }
                        }
                        d7_update_scan $expect_out(2,string) 0
                }
                -re "BY (.),(.)\r" {
                        if {$expect_out(2,string) == "0"} {
                                .tune.squelch.$expect_out(1,string) configure -bg
$d7_off_indicator_colour
                        } else {
                                .tune.squelch.$expect_out(1,string) configure -bg
$d7_on_indicator_colour
                        }
                }
                -re "CH (.)\r" {
                        set d7_parameter(CH) $expect_out(1,string)
                }
                -re "CIN\r" {
                }
                -re "CR (.),1,(...........),(.)\r" {
                        if {$expect_out(1,string) == "0"} {
                                set x CallA
                        } else {
                                set x CallB
                        }
                        set d7_memory($x,txfreq) [d7_d7freq_to_MHz
$expect_out(2,string)]
                        set d7_memory($x,txstep) $expect_out(3,string)
                        set d7_memory($x,txstepname)
$d7_step_codes($expect_out(3,string))
                }
                -re "CR (.),(0),(...........),(.),(.),(.),(.),(.),,(..),,
(..),(.........)?,(.)\r" {
                        if {$expect_out(1,string) == "0"} {
                                set x CallA
                        } else {
                                set x CallB
                        }

                        set d7_memory($x,band)          $expect_out(1,string)
                        set d7_memory($x,split)         $expect_out(2,string)
                        set d7_memory($x,freq)          [d7_d7freq_to_MHz
$expect_out(3,string)]
                        set d7_memory($x,step)          $expect_out(4,string)
                        set d7_memory($x,stepname)
$d7_step_codes($d7_memory($x,step))
                        set d7_memory($x,reverse)               $expect_out(6,string)
                        set d7_memory($x,tone)          $expect_out(7,string)
                        set d7_memory($x,tonefreq)      $expect_out(9,string)
                        set d7_memory($x,tonefreqname)
$d7_tone_codes($d7_memory($x,tonefreq))
                        set d7_memory($x,ctcss)         $expect_out(8,string)
                        set d7_memory($x,ctcssfreq)     $expect_out(10,string)
```

```
                        set d7_memory($x,ctcssfreqname)
$d7_tone_codes($d7_memory($x,ctcssfreq))
                        set d7_memory($x,offsetdirection)       $expect_out(5,string)
                        set d7_memory($x,offsetdirectionname)
$d7_offset_direction_codes($d7_memory($x,offsetdirection))
                        set d7_memory($x,offset)                [d7_d7offset_to_MHz
$expect_out(11,string)]
                        if {$expect_out(11,string) == "0)" } {
                                set d7_memory($x,mode)  FM
                        } else {
                                set d7_memory($x,mode)  AM
                        }
                }
                -re "CNT (..)\r" {
                        set d7_parameter(CNT) $expect_out(1,string)
                }
                -re "CT (.)\r" {
                        set d7_ctcss_enable($d7_parameter(BC)) \
                                $expect_out(1,string)
                        if {$expect_out(1,string) == "0"} {
                                .tune.ctcss.$d7_parameter(BC).match configure -state
disabled
                        } else {
                                .tune.ctcss.$d7_parameter(BC).match configure -state
normal
                        }
                }
                -re "CTD (.),(.)\r" {
                        set d7_ctcss_match($expect_out(1,string)) \
                                $expect_out(2,string)
                        if {$d7_ctcss_match($expect_out(1,string)) == "0"} {
                                .tune.ctcss.$expect_out(1,string).match configure
-bg $d7_off_indicator_colour
                        } else {
                                .tune.ctcss.$expect_out(1,string).match configure
-bg $d7_on_indicator_colour
                        }
                }
                -re "CTN (..)\r" {
                        set d7_ctcss_freq($d7_parameter(BC)) $expect_out(1,string)
                        set d7_ctcss_freq_name($d7_parameter(BC)) \
                                $d7_tone_codes($expect_out(1,string))
                }
                -re "CW\r" {
                }
                -re "DL (.)\r" {
                        set d7_parameter(DL) $expect_out(1,string)
                        if {$d7_parameter(DL) == "0"} {
                                .misccontrol.cbcol.fullduplex configure -state
disabled
                        } else {
                                .misccontrol.cbcol.fullduplex configure -state normal
                        }
                }
                -re "DM 0(.),(\[0-9A-F\]*)\r" {
                        set d7_dtmf_memory($expect_out(1,string),value)
$expect_out(2,string)
                }
                -re "DMN 0(.),(\[^\r\]*)\r" {
                        d7_dtmf_memory_set_name $expect_out(1,string)
$expect_out(2,string)
                }
                -re "DS (.)\r" {
                        set d7_parameter(DS) $expect_out(1,string)
                        set d7_dcd_sense $d7_DCD_codes($d7_parameter(DS))
                }
                -re "DTB (.)\r" {
                        set d7_parameter(DTB) $expect_out(1,string)
                }
```

```
                -re "DTX (.)\r" {
                        set d7_parameter(DTX) $expect_out(1,string)
                        set d7_beacon_method
$d7_beacon_method_codes($expect_out(1,string))
                }
                -re "DUP (.)\r" {
                        set d7_parameter(DUP) $expect_out(1,string)
                }
                -re "ELK (.)\r" {
                        set d7_parameter(ELK) $expect_out(1,string)
                }
                -re "FQ ..........,(.)\r" {
                        set d7_frequency($d7_parameter(BC)) \
                                $expect_out(1,string)
                        set d7_frequency_MHz($d7_parameter(BC)) \
                                [d7_d7freq_to_MHz $expect_out(1,string)]
                        set d7_step($d7_parameter(BC)) $expect_out(2,string)
                        set d7_step_KHz($d7_parameter(BC)) \
                                $d7_step_codes($expect_out(2,string))
                }
                -re "GU (.)\r" {
                        set d7_parameter(GU) $expect_out(1,string)
                        set d7_gps_unit $d7_GPS_codes($d7_parameter(GU))
                }
                -re "ICO (.),(\[^\r\]*)\r" {
                        set d7_icon_mode $expect_out(1,string)
                        set d7_icon_data $expect_out(2,string)
                }
                -re "ID (\[^\r\]*)\r" {
                        set d7_parameter(ID) $expect_out(1,string)
                }
                -re "LIST (..),(\[^\r,\]*),(........)(.........),(\[A-Z/\\\].)?,
(.),(.),(\[^\r,\]*),(\[^\r,\]*),(\[^\r\]*)\r" {
                        set x $expect_out(1,string)

                        set d7_APRS_list($x,callsign)   $expect_out(2,string)
                        set d7_APRS_list($x,latitude) \
                                [d7_format_latitude $expect_out(3,string)]
                        set d7_APRS_list($x,longitude) \
                                [d7_format_longitude $expect_out(4,string)]
                        set d7_APRS_list($x,icon) \
                                [d7_get_APRS_symbol $expect_out(5,string)]
                        if {[string length $d7_APRS_list($x,icon)] >
$d7_symbol_name_width} {
                                set d7_APRS_list($x,icon) [string range \
                                        $d7_APRS_list($x,icon) \
                                        0 [expr $d7_symbol_name_width - 1]]
                        }
                        if {[regexp {^[0-7]$} $expect_out(6,string)]} {
                                set d7_APRS_list($x,position) \
                                $d7_position_comment_codes($expect_out(6,string))
                        } else {
                                set d7_APRS_list($x,position) \
                                        $expect_out(6,string)
                        }
                        if [catch {set d7_APRS_list($x,category)
$d7_APRS_category_codes($expect_out(7,string))}] {
                                set d7_APRS_list($x,category) $expect_out(7,string)
                        }
                        set d7_APRS_list($x,overlay)    $expect_out(8,string)
                        set d7_APRS_list($x,status)     $expect_out(9,string)
                        set d7_APRS_list($x,information) $expect_out(10,string)

                        if [d7_APRS_list_in d7_APRS_list $x d7_APRS_old_list] {
                                .aprs_history.data.$x.index configure -fg black
                        } else {
                                .aprs_history.data.$x.index configure -fg red
                        }
                }
```

```
                -re "LK (.)\r" {
                        set d7_parameter(LK) $expect_out(1,string)
                }
                -re "LMP (.)\r" {
                        set d7_parameter(LMP) $expect_out(1,string)
                }
                -re "MAC (.)\r" {
                        set d7_parameter(MAC) $expect_out(1,string)
                        set d7_call_colour $d7_colour_codes($d7_parameter(MAC))
                }
                -re "MC (.),(.)\r" {
                        # VMC seems to produce this
                        set d7_memmode($expect_out(1,string)) \
                                $expect_out(2,string)
                        set d7_memmode_name($expect_out(1,string)) \
                                $d7_mem_mode_codes($expect_out(2,string))
                }
                -re "MC (.),(\[^\r\]*)\r" {
                        set d7_current_memory($expect_out(1,string)) \
                                $expect_out(2,string)
                        .tune.memmode.$expect_out(1,string).memname \
                                configure -textvariable \
                                d7_memory([d7_memory_index_to_num
$expect_out(2,string)],name)
                }
                -re "MCL (.),(.)\r" {
                }
                -re "MES (\[^\r\]*)\r" {
                        set d7_parameter(MES) $expect_out(1,string)
                }
                -re "MD (.)\r" {
                        set d7_parameter(MD) $expect_out(1,string)
                        if {$expect_out(1,string) == "0"} {
                        } else {
                        }
                }
                -re "MNA (.),(...?),(\[^\r\]*)\r" {
                        set n [d7_memory_index_to_num $expect_out(2,string)]
                        d7_memory_set_name $n $expect_out(3,string)
                }
                -re "MIN\r" {
                }
                -re "MNF (.)\r" {
                        set d7_parameter(MNF) $expect_out(1,string)
                }
                -re "MON (.)\r" {
                        set d7_parameter(MON) $expect_out(1,string)
                }
                -re "MP (\[^\r\]*)\r" {
                        set d7_parameter(MP) $expect_out(1,string)

                        set d7_latitude [d7_format_latitude \
                                [string range $expect_out(1,string) 0 7]]
                        set d7_longitude [d7_format_longitude \
                                [string range $expect_out(1,string) 8 17]]
                }
                -re "MR 0,1,(\[^\r,\]*),(..........),(.)\r" {
                        set n [d7_memory_index_to_num $expect_out(1,string)]
                        set d7_memory($n,txfreq) [d7_d7freq_to_MHz
$expect_out(2,string)]
                        set d7_memory($n,txstep) $expect_out(3,string)
                        set d7_memory($n,txstepname)
$d7_step_codes($expect_out(3,string))
                }
                -re "MR 0,0,(\[^\r,\]*),(..........),(.),(.),(.),(.),(.),,(..),,
(..),(.........)?,(.),(.)\r" {
                        set n [d7_memory_index_to_num $expect_out(1,string)]
                        set d7_memory($n,freq)  [d7_d7freq_to_MHz
$expect_out(2,string)]
```

```
                        set d7_memory($n,step)  $expect_out(3,string)
                        set d7_memory($n,stepname)
$d7_step_codes($d7_memory($n,step))
                        set d7_memory($n,offsetdirection) $expect_out(4,string)
                        set d7_memory($n,offsetdirectionname) \
                                $d7_offset_direction_codes($expect_out(4,string))
                        set d7_memory($n,reverse)       $expect_out(5,string)
                        set d7_memory($n,tone)  $expect_out(6,string)
                        set d7_memory($n,ctcss) $expect_out(7,string)
                        set d7_memory($n,tonefreq)      $expect_out(8,string)
                        set d7_memory($n,tonefreqname)
$d7_tone_codes($d7_memory($n,tonefreq))
                        set d7_memory($n,ctcssfreq)     $expect_out(9,string)
                        set d7_memory($n,ctcssfreqname)
$d7_tone_codes($d7_memory($n,ctcssfreq))
                        set d7_memory($n,offset)        [d7_d7offset_to_MHz
$expect_out(10,string)]
                        if {$expect_out(11,string) == "0"} {
                                set d7_memory($n,mode) FM
                        } else {
                                set d7_memory($n,mode) AM
                        }
                        set d7_memory($n,lockout)       $expect_out(12,string)
                }
                -re "MYC (\[^\r\]*)\r" {
                        set d7_parameter(MYC) $expect_out(1,string)
                }
                -re "MSH\r" {
                }
                -re "MW\r" {
                }
                -re "N\r" {
                }
                -re "NSFT (.)\r" {
                        set d7_parameter(NSFT) $expect_out(1,string)
                }
                -re "OS (.........)\r" {
                        set d7_parameter(OS) $expect_out(1,string)
                        set d7_offset_MHz($d7_parameter(BC)) \
                                [d7_d7offset_to_MHz $expect_out(1,string)]
                }
                -re "PC (.),(.)\r" {
                        set d7_power($expect_out(1,string)) $expect_out(2,string)
                        set d7_power_text($expect_out(1,string)) \
                                $d7_power_codes($expect_out(2,string))
                }
                -re "POSC (.)\r" {
                        set d7_parameter(POSC) $expect_out(1,string)
                        set d7_position_comment
$d7_position_comment_codes($d7_parameter(POSC))
                }
                -re "PP (\[^\r\]*)\r" {
                        set d7_parameter(PP) $expect_out(1,string)
                }
                -re "PT (.)\r" {
                        set d7_parameter(PT) $expect_out(1,string)
                        set d7_DTMF_delay_mSecs
$d7_DTMF_delay_codes($d7_parameter(PT))
                }
                -re "PV (.),(.....),(.....)\r" {
                        set d7_vfo($expect_out(1,string),low) [string range
$expect_out(2,string) 2 4]
                        set d7_vfo($expect_out(1,string),high) [string range
$expect_out(3,string) 2 4]
                }
                -re "RBN (.)\r" {
                        if {$d7_parameter(BC) == 0} {
                                if {$expect_out(1,string) == 1} {
                                        .tune.bandopt.band_a.band.vhf \
```

```
                                                configure -bg
$d7_off_indicator_colour
                                        .tune.bandopt.band_a.band.air \
                                                configure -bg $d7_on_indicator_colour
                                } else {
                                        .tune.bandopt.band_a.band.vhf \
                                                configure -bg $d7_on_indicator_colour
                                        .tune.bandopt.band_a.band.air \
                                                configure -bg
$d7_off_indicator_colour
                                }
                        } else {
                                if {$expect_out(1,string) == 6} {
                                        .tune.bandopt.band_b.vhf \
                                                configure -bg
$d7_off_indicator_colour
                                        .tune.bandopt.band_b.uhf \
                                                configure -bg $d7_on_indicator_colour
                                } else {
                                        .tune.bandopt.band_b.vhf \
                                                configure -bg $d7_on_indicator_colour
                                        .tune.bandopt.band_b.uhf \
                                                configure -bg
$d7_off_indicator_colour
                                }
                        }
                        if {$expect_out(1,string) == 1} {
                                .tune.bandopt.band_a.mod.fm configure -state normal
                                .tune.bandopt.band_a.mod.am configure -state normal
                        } else {
                                .tune.bandopt.band_a.mod.fm configure -state disabled
                                .tune.bandopt.band_a.mod.am configure -state disabled
                        }
                }
                -re "REV (.)\r" {
                        set d7_reverse($d7_parameter(BC)) $expect_out(1,string)
                }
                -re "RSC (.)\r" {
                        set d7_parameter(RSC) $expect_out(1,string)
                }
                -re "RSV (.*)\r" {
                        set d7_parameter(RSV) $expect_out(1,string)
                }
                -re "RX\r" {
                        set d7_transmitting 0

                        .misccontrol.ctrlcol.ptt.ab.0 \
                                configure -bg $d7_off_indicator_colour
                        .misccontrol.ctrlcol.ptt.ab.1 \
                                configure -bg $d7_off_indicator_colour
                }
                -re "SC (.)\r" {
                        # 0-on 1-MHz scan
                }
                -re "SCC (.*)\r" {
                        set d7_parameter(SCC) $expect_out(1,string)
                }
                -re "SCR (.)\r" {
                        set d7_parameter(SCR) $expect_out(1,string)
                        set d7_scan_resume $d7_scan_resume_codes($d7_parameter(SCR))
                }
                -re "SCT (.*)\r" {
                        set d7_parameter(SCT) $expect_out(1,string)
                }
                -re "SFT (.)\r" {
                        set d7_offset_direction($d7_parameter(BC)) \
                                $d7_offset_direction_codes($d7_parameter(SFT))
                }
                -re "SKTN (..)\r" {
```

```
                        set d7_parameter(SKTN) $expect_out(1,string)
                }
                -re "SKAT (.*)\r" {
                        set d7_parameter(SKAT) $expect_out(1,string)
                }
                -re "SM (.),(..)\r" {
                        set tmpval [expr $d7_signal_indicator_height -
((($d7_signal_indicator_height * $expect_out(2,string)) / 5)]
                        .tune.signal.$expect_out(1,string).canvas \
                                coords \
                                d7_signal_$expect_out(1,string) \
                                0 $d7_signal_indicator_height \
                                $d7_signal_indicator_width $tmpval

                        d7_update_scan $d7_frequency($expect_out(1,string)) \
                                $expect_out(2,string)
                }
                -re "SMC (.)\r" {
                        set d7_parameter(SMC) $expect_out(1,string)
                        set d7_message_colour $d7_colour_codes($d7_parameter(SMC))
                }
                -re "SMSG (.*)\r" {
                        set d7_parameter(SMSG) $expect_out(1,string)
                }
                -re "SMY (.*)\r" {
                        set d7_parameter(SMY) $expect_out(1,string)
                }
                -re "SQ (.),(..)\r" {
                        set d7_squelch($expect_out(1,string)) $expect_out(2,string)
                }
                -re "ST (.)\r" {
                        set d7_step($d7_parameter(BC)) $expect_out(1,string)
                        set d7_step_KHz($d7_parameter(BC))
$d7_step_codes($expect_out(1,string))
                }
                -re "STAT (.*)\r" {
                        set d7_parameter(STAT) $expect_out(1,string)
                }
                -re "SV (.)\r" {
                        set d7_parameter(SV) $expect_out(1,string)
                        set d7_battery_saver
$d7_battery_saver_codes($d7_parameter(SV))
                }
                -re "TC (.)\r" {
                        set d7_parameter(TC) $expect_out(1,string)
                }
                -re "TS (.)\r" {
                        set d7_parameter(TC) $expect_out(1,string)
                }
                -re "TH (.)\r" {
                        set d7_parameter(TH) $expect_out(1,string)
                }
                -re "TN (..)\r" {
                        set d7_tone_freq($d7_parameter(BC)) \
                                $expect_out(1,string)
                        set d7_tone_freq_name($d7_parameter(BC)) \
                                $d7_tone_codes($expect_out(1,string))
                }
                -re "TNC (.)\r" {
                        set d7_parameter(TNC) $expect_out(1,string)
                }
                -re "TO (.)\r" {
                        set d7_tone_enable($d7_parameter(BC)) \
                                $expect_out(1,string)
                }
                -re "TSP (.)\r" {
                        set d7_parameter(TSP) $expect_out(1,string)
                }
                -re "TT\r" {
```

```
                }
                -re "TX (.)\r" {
                        set d7_transmitting 1
                        .misccontrol.ctrlcol.ptt.ab.$expect_out(1,string) \
                                configure -bg $d7_on_indicator_colour
                        .misccontrol.ctrlcol.ptt.ab.[expr ! $expect_out(1,string)] \
                                configure -bg $d7_off_indicator_colour
                }
                -re "TXH (.)\r" {
                        set d7_parameter(TXH) $expect_out(1,string)
                }
                -re "TXI (.)\r" {
                        set d7_parameter(TXI) $expect_out(1,string)
                        set d7_APRS_TXI $d7_APRS_TXI_codes($d7_parameter(TXI))
                }
                -re "TXN (.)\r" {
                        set d7_parameter(TXN) $expect_out(1,string)
                }
                -re "TXS (.)\r" {
                        set d7_parameter(TXS) $expect_out(1,string)
                }
                -re "UNIT (.)\r" {
                        set d7_parameter(UNIT) $expect_out(1,string)
                }
                -re "UP (..)\r" {
                }
                -re "UPR (\[^\r\]*)\r" {
                        set d7_parameter(UPR) $expect_out(1,string)
                }
                -re "VCS (.)\r" {
                        set d7_parameter(VCS) $expect_out(1,string)
                }
                -re "VMC (.),(.)\r" {
                        set d7_memmode($expect_out(1,string)) \
                                $expect_out(2,string)
                        set d7_memmode_name($expect_out(1,string)) \
                                $d7_mem_mode_codes($expect_out(2,string))
                }
                -re "VR (.),(...........),(.),(.),(.),(.),(.),,(..),,
(..),(.........),(.)\r" {
                        set vfo $expect_out(1,string)
                        set d7_vfo($vfo,freq) [d7_d7freq_to_MHz
$expect_out(2,string)]
                        set d7_vfo($vfo,step) $expect_out(3,string)
                        set d7_vfo($vfo,stepname) \
                                $d7_step_codes($expect_out(3,string))
                        set d7_vfo($vfo,offsetdirection) $expect_out(4,string)
                        set d7_vfo($vfo,offsetdirectionname) \
                                $d7_offset_direction_codes($expect_out(4,string))
                        set d7_vfo($vfo,reverse) $expect_out(5,string)
                        set d7_vfo($vfo,tone) $expect_out(6,string)
                        set d7_vfo($vfo,ctcss) $expect_out(7,string)
                        set d7_vfo($vfo,tonefreq) $expect_out(8,string)
                        set d7_vfo($vfo,tonefreqname) \
                                $d7_tone_codes($expect_out(8,string))
                        set d7_vfo($vfo,ctcssfreq) $expect_out(9,string)
                        set d7_vfo($vfo,ctcssfreqname) \
                                $d7_tone_codes($expect_out(9,string))
                        set d7_vfo($vfo,offset) [d7_d7offset_to_MHz
$expect_out(10,string)]
                        if {$expect_out(11,string) == "0)" } {
                                set d7_vfo($vfo,mode)   FM
                        } else {
                                set d7_vfo($vfo,mode)   AM
                        }
                }
                -re "VW (.)\r" {
                }
                -re "TASCO Radio Modem\r" {
```

```
                        d7_set_tnc
                }
                -re "\[^\r\]*\r" {
                }
                timeout {
                }
        }
        d7_get_radio_parameters
}

proc d7_set_tnc {} {
        global d7_mode
        global d7_spawn_id
        global spawn_id

        set d7_mode tnc
        set d7_selected_mode tnc

        trace variable expect_out(0,string) w d7_log_trace

        expect_background {
                -i $d7_spawn_id
                -re "AUTOLF +(ON|OFF)\r" {
                        set d7_tnc_paramter(AUTOLF) $expect_out(1,string)
                }
                -re "AWLEN +(.)\r" {
                }
                -re "BEACON +(.*) +(.*)\r" {
                        set d7_tnc_paramter(BEACON-inter) $expect_out(1,string)
                        set d7_tnc_paramter(BEACON-num) $expect_out(2,string)
                }
                -re "BTEXT +(.*)\r" {
                        set d7_tnc_paramter(BTEXT) $expect_out(1,string)
                }
                -re "CMSG +(ON|OFF)\r" {
                        set d7_tnc_paramter(CMSG) $expect_out(1,string)
                }
                -re "CMSGDISC +(ON|OFF)\r" {
                        set d7_tnc_paramter(CMSGDISC) $expect_out(1,string)
                }
                -re "CONOK +(ON|OFF)\r" {
                        set d7_tnc_paramter(CONOK) $expect_out(1,string)
                }
                -re "CPACTIME +(ON|OFF)\r" {
                        set d7_tnc_paramter(CPACTIME) $expect_out(1,string)
                }
                -re "CR +(ON|OFF)\r" {
                        set d7_tnc_paramter(CR) $expect_out(1,string)
                }
                -re "CTEXT +(.*)\r" {
                        # same as LTEXT
                        set d7_tnc_paramter(LTEXT) $expect_out(1,string)
                }
                -re "DWAIT +(\[0-9\]+)\r" {
                        set d7_tnc_paramter(DWAIT) $expect_out(1,string)
                }
                -re "ECHO +(ON|OFF)\r" {
                        set d7_tnc_paramter(ECHO) $expect_out(1,string)
                }
                -re "FIRMRNR +(ON|OFF)\r" {
                        set d7_tnc_paramter(FIRMRNR) $expect_out(1,string)
                }
                -re "FLOW +(ON|OFF)\r" {
                        set d7_tnc_paramter(FLOW) $expect_out(1,string)
                }
                -re "FRACK +(\[0-9\]+)\r" {
                        set d7_tnc_paramter(FRACK) $expect_out(1,string)
                }
                -re "GBAUD +(....)\r" {
```

```
        set d7_tnc_paramter(GBAUD) $expect_out(1,string)
}
-re "GPSTEXT +(.*)\r" {
        set d7_tnc_paramter(GPSTEXT) $expect_out(1,string)
}
-re "HBAUD +(....)\r" {
        set d7_tnc_paramter(HBAUD) $expect_out(1,string)
}
-re "LOCATION +(.*) +(\[0-9\]+)\r" {
        set d7_tnc_paramter(LOCATION-inter) $expect_out(1,string)
        set d7_tnc_paramter(LOCATION-num) $expect_out(2,string)
}
-re "LOC10X +(ON|OFF)\r" {
        set d7_tnc_paramter(LOC10X) $expect_out(1,string)
}
-re "LPATH +(.*)\r" {
        set d7_tnc_paramter(LPATH) $expect_out(1,string)
}
-re "LTEXT +(.*)\r" {
        set d7_tnc_paramter(LTEXT) $expect_out(1,string)
}
-re "LTMHEAD +(ON|OFF)\r" {
        set d7_tnc_paramter(LTMHEAD) $expect_out(1,string)
}
-re "LTMON +(\[0-9\]+)\r" {
        set d7_tnc_paramter(LTMON) $expect_out(1,string)
}
-re "MAXFRAME (.)\r" {
        set d7_tnc_paramter(MAXFRAME) $expect_out(1,string)
}
-re "MAXFRAME was (.)\r" {
}
-re "MALL +(ON|OFF)\r" {
        set d7_tnc_paramter(MALL) $expect_out(1,string)
}
-re "MCOM +(ON|OFF)\r" {
        set d7_tnc_paramter(MCOM) $expect_out(1,string)
}
-re "MCON +(ON|OFF)\r" {
        set d7_tnc_paramter(MCON) $expect_out(1,string)
}
-re "MONITOR +(ON|OFF)\r" {
        set d7_tnc_paramter(MONITOR) $expect_out(1,string)
}
-re "MRPT +(ON|OFF)\r" {
        set d7_tnc_mrpt $expect_out(1,string)
}
-re "MRPT +was (ON|OFF)\r" {
}
-re "MYCALL +(.*)\r" {
        set d7_tnc_paramter(MYCALL) $expect_out(1,string)
}
-re "NTSGRP +(\[0-9A-Z\]\[0-9A-Z\]\[0-9A-Z\])\r" {
        set d7_tnc_paramter(NTSGRP) $expect_out(1,string)
}
-re "NTSMRK +(\[0-9\]+)\r" {
        set d7_tnc_paramter(NTSMRK) $expect_out(1,string)
}
-re "NTSMSG +(.*)\r" {
        set d7_tnc_paramter(NTSMSG) $expect_out(1,string)
}
-re "PACLEN +(\[0-9\]+)\r" {
        set d7_tnc_paramter(PACLEN) $expect_out(1,string)
}
-re "PACTIME +(.*) +(\[0-9\]+)\r" {
        set d7_tnc_paramter(PACTIME-inter) $expect_out(1,string)
        set d7_tnc_paramter(PACTIME-num) $expect_out(2,string)
}
-re "PARITY +(.)\r" {
```

```
                        set d7_tnc_paramter(PARITY) $expect_out(1,string)
                }
                -re "PASSALL +(ON|OFF)\r" {
                        set d7_tnc_paramter(PASSALL) $expect_out(1,string)
                }
                -re "PERSIST +(\[0-9\]+)\r" {
                        set d7_tnc_paramter(PERSIST) $expect_out(1,string)
                }
                -re "PPERSIST +(ON|OFF)\r" {
                        set d7_tnc_paramter(PPERSIST) $expect_out(1,string)
                }
                -re "RESPTIME +(\[0-9\]+)\r" {
                        set d7_tnc_paramter(RESPTIME) $expect_out(1,string)
                }
                -re "RETRY +(\[0-9\]+)\r" {
                        set d7_tnc_paramter(RETRY) $expect_out(1,string)
                }
                -re "SENDPAC +(.*)\r" {
                        set d7_tnc_paramter(SENDPAC) $expect_out(1,string)
                }
                -re "SLOTTIME +(\[0-9\]+)\r" {
                        set d7_tnc_paramter(SLOTTIME) $expect_out(1,string)
                }
                -re "SOFTDCD +(ON|OFF)\r" {
                        set d7_tnc_paramter(SOFTDCD) $expect_out(1,string)
                }
                -re "TRACE +(ON|OFF)\r" {
                        set d7_tnc_paramter(TRACE) $expect_out(1,string)
                }
                -re "TRIES +(\[0-9\]+)\r" {
                }
                -re "TXDELAY +(\[0-9\]+)\r" {
                        set d7_tnc_paramter(TXDELAY) $expect_out(1,string)
                }
                -re "TXUIFRAM +(ON|OFF)\r" {
                        set d7_tnc_paramter(TXUIFRAM) $expect_out(1,string)
                }
                -re "UNPROTO +(.*)\r" {
                        set d7_tnc_paramter(UNPROTO) $expect_out(1,string)
                }
                -re "XFLOW +(ON|OFF)\r" {
                }
                -re "XMITOK +(ON|OFF)\r" {
                        set d7_tnc_paramter(XMITOK) $expect_out(1,string)
                }
                -re "cmd:TC 1" {
                        d7_set_radio
                }
                -re "TS 1\r" {
                        d7_set_radio
                }
                -re "\[^\r\]*\r" {
                }
                timeout {
                }
        }
        d7_send_string_tnc "DISP"
}

proc d7_set_squelch { band val } {
        d7_send_string "SQ $band,0$val"
}

proc d7_swtich_mode {} {
        global d7_mode
        global d7_selected_mode

        if {$d7_mode == "off"} {
                return
```

```
        }
        if {$d7_selected_mode == "radio"} {
                d7_set_radio
        } elseif {$d7_selected_mode == "tnc"} {
                d7_set_tnc
        }
}

proc d7_set_port { port } {
        global d7_mode
        global d7_spawn_id
        global tty_spawn_id
        global user_spawn_id

        if [catch {spawn -open [open $port RDWR ]} err] {
                set d7_mode off

                puts "Open of $port failed."
                return
        }
        set d7_spawn_id $spawn_id
        if [catch {stty raw ixon ixany -echo < $port} err] {
                set d7_mode off

                puts "stty of $port failed."
                return
        }
        d7_set_radio
}

proc d7_send_string_any_mode { param } {
        global d7_spawn_id
        global d7_mode

        if {$d7_mode == "off"} return

        exp_send -i $d7_spawn_id "$param\r"
        d7_log out "$param"
        sleep 0.05
}

proc d7_send_string { param } {
        global d7_mode

        if {$d7_mode != "radio"} return

        d7_send_string_any_mode $param
}

proc d7_send_string_tnc { param } {
        global d7_mode

        if {$d7_mode != "tnc"} return

        d7_send_string_any_mode $param
}

proc d7_send_cat_string { param1 param2 } {
        d7_send_string $param1$param2
}

proc d7_get_radio_parameter { param } {
        d7_send_string $param
}

proc d7_set_radio_parameter { param value } {
        d7_send_string "$param $value"
}
```

```
proc d7_get_radio_parameters {} {
        global d7_radio_simple_parameters

        foreach param $d7_radio_simple_parameters {
                d7_get_radio_parameter $param
        }
        foreach command { SQ ASC PC VMC MC BUF } {
                foreach band { 0 1 } {
                        d7_send_string "$command $band"
                }
        }
        foreach vfo { 1 2 3 6 } {
                d7_send_string "PV $vfo"
                d7_send_string "VR $vfo"
        }
        d7_send_string "ICO"
        d7_send_string "AI 1"
        foreach x { 0 1 2 3 4 5 6 7 8 9 } {
                d7_dtmf_memory_read $x
        }
}

proc d7_set_freq_MHz { m } {
        global d7_parameter
        global d7_step
        global d7_frequency

        if [catch {set s [format %09.5f $m]}] {
                return
        }
        set f [d7_MHz_to_d7freq $s]
        if {$f != $d7_frequency($d7_parameter(BC))} {
                d7_send_string "FQ $f,$d7_step($d7_parameter(BC))"
        }
}

proc d7_set_offset_MHz { m } {
        global d7_parameter

        if [catch {set s [format %05.2f $m]}] {
                return
        }
        set f [d7_MHz_to_d7offset $s]
        if {$f != $d7_parameter(OS)} {
                d7_send_string "OS $f"
        }
}

#
# Radio Port Logging
#
proc d7_hand_entry {} {
        set val [.log.entry.entry get]
        d7_send_string_any_mode $val
}

proc d7_make_log {} {

        set w [toplevel .log]

        wm title $w "Log"

        frame $w.log
        text $w.log.text -height 20 -width 60 \
                -yscrollcommand "$w.log.scroll set"
        .log.log.text tag configure in -foreground blue
        .log.log.text tag configure out -foreground red
        scrollbar $w.log.scroll -width 20 \
                -command "$w.log.text yview"
```

```
        pack \
                $w.log.text \
                -side left -expand true -fill both
        pack \
                $w.log.scroll \
                -side left -expand false -fill y
        frame $w.entry
        entry $w.entry.entry
        pack \
                $w.entry.entry \
                -side left -expand true -fill x
        button $w.entry.enter -text Enter -command "d7_hand_entry"
        pack \
                $w.entry.enter \
                -side left -expand false
        button $w.dismiss -text Dismiss -command "destroy $w"
        pack \
                $w.log \
                -expand true -fill both
        pack \
                $w.entry \
                $w.dismiss \
                -expand false -fill x
}

#
# Real work starts....
#
wm title . "$d7_window_title $tkd7_version"
if [file exists $d7_icon_file] {
        wm iconbitmap . @$d7_icon_file
}

# log_file -noappend log
log_user 0
trace variable expect_out(0,string) w d7_log_trace

# exp_internal 1

frame .modes
button .modes.quit -text Quit \
        -command exit
label .modes.mode \
        -textvariable d7_mode
radiobutton .modes.radio -text Radio \
        -variable d7_selected_mode \
        -value radio \
        -command d7_swtich_mode
radiobutton .modes.tnc -text TNC \
        -variable d7_selected_mode \
        -value tnc \
        -command d7_swtich_mode
pack \
        .modes.quit \
        .modes.mode \
        .modes.radio \
        .modes.tnc \
        -side left -expand 1 -fill both
# pack \
        .modes \
        -expand 1 -fill both

proc d7_function_window { win } {
        if ![winfo exists .$win] {
                d7_make_$win
        } else {
                wm deiconify .$win
                raise .$win
        }
```

```
}

frame .functions
button .functions.memory -text Memory \
        -command "d7_function_window memory"
button .functions.dtmf -text DTMF \
        -command "d7_function_window dtmf"
button .functions.aprs -text APRS \
        -command "d7_function_window aprs"
button .functions.vfo -text VFO \
        -command "d7_function_window vfo"
label .functions.ident -textvariable d7_parameter(ID)
menubutton .functions.mode -textvariable d7_mode \
        -menu .functions.mode.menu
menu .functions.mode.menu -tearoff 0
foreach mode { radio tnc } {
        .functions.mode.menu add command -label $mode \
                -command "d7_switch_modes $mode"
}
button .functions.sstv -text SSTV \
        -command "d7_function_window sstv"
button .functions.tnc -text TNC \
        -command "d7_function_window tnc"
button .functions.scan -text "Scan" \
        -command "d7_function_window scan"
button .functions.log -text "Log" \
        -command "d7_function_window log"
button .functions.quit -text "Quit" \
        -command "exit"
menubutton .functions.port -menu .functions.port.menu -text Port
menu .functions.port.menu -tearoff 0
foreach port $d7_com_ports {
        .functions.port.menu add command -label $port \
                -command "d7_set_port $port"
}
pack \
        .functions.port \
        .functions.memory \
        .functions.dtmf \
        .functions.aprs \
        .functions.vfo \
        -side left
pack \
        .functions.ident \
        .functions.mode \
        -side left -expand true
pack \
        .functions.quit \
        .functions.log \
        .functions.scan \
        .functions.sstv \
        .functions.tnc \
        -side right

pack \
        .functions \
        -expand true -fill both

frame .tune
scale .tune.balance -orient vertical -showvalue false -width 5 \
        -from 0 -to 4 \
        -variable d7_parameter(BAL) \
        -command "d7_set_radio_parameter BAL"
frame .tune.select
frame .tune.select.band_a
checkbutton .tune.select.band_a.band_a_selected -text A \
        -variable d7_parameter(BC) -onvalue 0 -offvalue 1 \
        -command {d7_set_radio_parameter BC $d7_parameter(BC)}
checkbutton .tune.select.band_a.data -text Data \
```

```
        -variable d7_parameter(DTB) -onvalue 0 -offvalue 1 \
        -command {d7_set_radio_parameter DTB $d7_parameter(DTB)}
pack \
        .tune.select.band_a.band_a_selected \
        .tune.select.band_a.data \
        -expand true -fill both
label .tune.select.label -text Band -padx 0 -pady 0
frame .tune.select.band_b
checkbutton .tune.select.band_b.band_b_selected -text B \
        -variable d7_parameter(BC) -onvalue 1 -offvalue 0 \
        -command {d7_set_radio_parameter BC $d7_parameter(BC)}
checkbutton .tune.select.band_b.data -text Data \
        -variable d7_parameter(DTB) -onvalue 1 -offvalue 0 \
        -command {d7_set_radio_parameter DTB $d7_parameter(DTB)}
pack \
        .tune.select.band_b.band_b_selected \
        .tune.select.band_b.data \
        -expand true -fill both
pack \
        .tune.select.band_a \
        .tune.select.label \
        .tune.select.band_b \
        -expand true -fill both

frame .tune.asc
foreach band { 0 1 } {
        checkbutton .tune.asc.$band \
                -variable d7_asc_selected($band) \
                -command "d7_set_radio_parameter ASC \"$band,\$d7_asc_selected($band)
\""
}
label .tune.asc.label -text ASC -padx 0 -pady 0
pack \
        .tune.asc.0 \
        .tune.asc.label \
        .tune.asc.1 \
        -expand true -fill both

frame .tune.squelch
foreach band { 0 1 } {

        scale .tune.squelch.$band -orient vertical -showvalue false \
                -width 5 \
                -length $d7_squelch_height \
                -sliderlength 10 \
                -from 5 -to 0 \
                -variable d7_squelch($band) \
                -command "d7_set_squelch $band"
}
checkbutton .tune.squelch.mon -padx 0 \
        -variable d7_parameter(MON) \
        -command {d7_set_radio_parameter MON $d7_parameter(MON)} \
        -onvalue 1
pack \
        .tune.squelch.0 \
        .tune.squelch.mon \
        .tune.squelch.1 \
        -expand true -fill both

frame .tune.power
foreach band { 0 1 } {
        menubutton .tune.power.$band -padx 0 -pady 0 \
                -width $d7_val_width(power) \
                -menu .tune.power.$band.menu \
                -textvariable d7_power_text($band)
        # the document does not say it but it seems only the current band
        # can be changed
        d7_add_to_ab_widget_list $band .tune.power.$band
        d7_make_menu power \
```

```
                .tune.power.$band.menu \
                d7_power($band) \
                "d7_send_string \"PC $band,\$d7_power($band)\""
}
label .tune.power.label -text PWR -padx 0 -pady 0
pack \
        .tune.power.0 \
        .tune.power.label \
        .tune.power.1 \
        -expand true -fill both

frame .tune.bandopt
frame .tune.bandopt.band_a
frame .tune.bandopt.band_a.band
button .tune.bandopt.band_a.band.vhf -text VHF -padx 0 -pady 0 \
        -command { d7_send_string "RBN 2" }
d7_add_to_ab_widget_list 0 .tune.bandopt.band_a.band.vhf
button .tune.bandopt.band_a.band.air -text AIR -padx 0 -pady 0 \
        -command { d7_send_string "RBN 1" }
d7_add_to_ab_widget_list 0 .tune.bandopt.band_a.band.air
pack \
        .tune.bandopt.band_a.band.vhf \
        .tune.bandopt.band_a.band.air \
        -side left -expand true -fill none
pack \
        .tune.bandopt.band_a.band \
        -side top -expand true -fill both
frame .tune.bandopt.band_a.mod
label .tune.bandopt.band_a.mod.label -text "118 MHz"
button .tune.bandopt.band_a.mod.fm -text FM -padx 0 -pady 0 \
        -command {d7_set_radio_parameter MD 0}
button .tune.bandopt.band_a.mod.am -text AM -padx 0 -pady 0 \
        -command {d7_set_radio_parameter MD 1}
pack \
        .tune.bandopt.band_a.mod.fm \
        .tune.bandopt.band_a.mod.am \
        -side left -expand true -fill none
pack \
        .tune.bandopt.band_a.mod \
        -side top -expand 1 -fill both
pack \
        .tune.bandopt.band_a \
        -expand true -fill both

frame .tune.bandopt.band_b
button .tune.bandopt.band_b.vhf -text VHF -padx 0 -pady 0 \
        -command { d7_send_string "RBN 3" }
d7_add_to_ab_widget_list 1 .tune.bandopt.band_b.vhf
button .tune.bandopt.band_b.uhf -text UHF -padx 0 -pady 0 \
        -command { d7_send_string "RBN 6" }
d7_add_to_ab_widget_list 1 .tune.bandopt.band_b.uhf
pack \
        .tune.bandopt.band_b.vhf \
        .tune.bandopt.band_b.uhf \
        -side left -expand true -fill none
pack \
        .tune.bandopt.band_b \
        -expand true -fill both

frame .tune.freq
frame .tune.freq.a
entry .tune.freq.a.band -fg blue -font {Courier 20 bold} \
        -width $d7_frequency_width \
        -textvariable d7_frequency_MHz(0)
bind .tune.freq.a.band <FocusOut> {
        d7_set_freq_MHz $d7_frequency_MHz(0)
        focus .tune.freq
}
bind .tune.freq.a.band <Key-Return> {
```

```
        d7_set_freq_MHz $d7_frequency_MHz(0)
}
d7_add_to_ab_widget_list 0 .tune.freq.a.band
pack \
        .tune.freq.a.band \
        -side left \
        -expand true -fill none

frame .tune.freq.updown
button .tune.freq.updown.up -text Up \
        -command {d7_send_string UP}
button .tune.freq.updown.down -text Down \
        -command {d7_send_string DW}
pack \
        .tune.freq.updown.up \
        .tune.freq.updown.down \
        -side left -expand true -fill both

frame .tune.freq.b
entry .tune.freq.b.band -fg blue -font {Courier 20 bold} \
        -width $d7_frequency_width \
        -textvariable d7_frequency_MHz(1)
bind .tune.freq.b.band <FocusOut> {
        d7_set_freq_MHz $d7_frequency_MHz(1)
        focus .tune.freq
}
bind .tune.freq.b.band <Key-Return> {
        d7_set_freq_MHz $d7_frequency_MHz(1)
}
d7_add_to_ab_widget_list 1 .tune.freq.b.band
pack \
        .tune.freq.b.band \
        -side left -expand true -fill none
pack \
        .tune.freq.a \
        .tune.freq.updown \
        .tune.freq.b \
        -expand true -fill both
frame .tune.stepoffset
foreach band { 0 1 } {
        frame .tune.stepoffset.$band
        frame .tune.stepoffset.$band.step
        menubutton .tune.stepoffset.$band.step.button \
                -width $d7_val_width(step) -anchor e \
                -textvariable d7_step_KHz($band) \
                -menu .tune.stepoffset.$band.step.button.stepsize
        d7_add_to_ab_widget_list $band .tune.stepoffset.$band.step.button
        d7_make_menu step \
                .tune.stepoffset.$band.step.button.stepsize \
                d7_step($band) \
                "d7_set_radio_parameter ST \$d7_step($band)"
        label .tune.stepoffset.$band.step.steplabel -text kHz
        pack \
                .tune.stepoffset.$band.step.button \
                .tune.stepoffset.$band.step.steplabel \
                -side left
        frame .tune.stepoffset.$band.offset
        menubutton .tune.stepoffset.$band.offset.direction \
                -relief raised -width 1 \
                -textvariable d7_offset_direction($band) \
                -menu .tune.stepoffset.$band.offset.direction.menu
        d7_add_to_ab_widget_list $band .tune.stepoffset.$band.offset.direction
        d7_make_menu offset_direction \
                .tune.stepoffset.$band.offset.direction.menu \
                d7_parameter(SFT) \
                {d7_set_radio_parameter SFT $d7_parameter(SFT)}

        entry .tune.stepoffset.$band.offset.offset -width $d7_offset_width \
                -textvariable d7_offset_MHz($band)
```

```
        d7_add_to_ab_widget_list $band .tune.stepoffset.$band.offset.offset
        bind .tune.stepoffset.$band.offset.offset <FocusOut> \
                "d7_set_offset_MHz \$d7_offset_MHz($band)
                focus .tune.stepoffset.$band.offset"
        bind .tune.stepoffset.$band.offset.offset <Key-Return> \
                "d7_set_offset_MHz \$d7_offset_MHz($band)"

        button .tune.stepoffset.$band.offset.set -text Set \
                -command "d7_send_string \"OS \[d7_MHz_to_d7offset
\$d7_offset_MHz($band)\]\""
        d7_add_to_ab_widget_list $band .tune.stepoffset.$band.offset.set

        pack \
                .tune.stepoffset.$band.offset.direction \
                .tune.stepoffset.$band.offset.offset \
                -side left
        pack \
                .tune.stepoffset.$band.step \
                .tune.stepoffset.$band.offset \
                -expand true -fill both
}
pack \
        .tune.stepoffset.0 \
        .tune.stepoffset.1 \
        -expand true -fill both
frame .tune.reverse
foreach band { 0 1 } {
        checkbutton .tune.reverse.$band \
                -padx 0 \
                -command "d7_set_radio_parameter REV \$d7_reverse($band)" \
                -variable d7_reverse($band) \
                -offvalue 0 -onvalue 1
        d7_add_to_ab_widget_list $band .tune.reverse.$band
}
label .tune.reverse.label -text Rev -padx 0 -pady 0
pack \
        .tune.reverse.0 \
        .tune.reverse.label \
        .tune.reverse.1 \
        -expand true -fill both
frame .tune.signal
foreach band { 0 1 } {
        frame .tune.signal.$band -bd 10
        canvas .tune.signal.$band.canvas \
                -bd 0 \
                -width $d7_signal_indicator_width \
                -height $d7_signal_indicator_height \
                -bd 0 -bg $d7_off_indicator_colour
        .tune.signal.$band.canvas create rectangle \
                0 $d7_signal_indicator_height \
                $d7_signal_indicator_width $d7_signal_indicator_height \
                -fill $d7_on_indicator_colour -tags d7_signal_$band
        bind .tune.signal.$band.canvas <Button-1> \
                "d7_send_string \"SM $band\""
        bind .tune.signal.$band <Button-1> \
                "d7_send_string \"SM $band\""
        pack \
                .tune.signal.$band.canvas \
                -expand true -fill none
}
pack \
        .tune.signal.0 \
        .tune.signal.1 \
        -expand true -fill both

frame .tune.memmode
foreach band { 0 1 } {
        frame .tune.memmode.$band
        frame .tune.memmode.$band.top
```

```
        menubutton .tune.memmode.$band.top.mode \
                -width $d7_val_width(mem_mode) \
                -padx 0 -pady 0 \
                -textvariable d7_memmode_name($band) \
                -menu .tune.memmode.$band.top.mode.menu
        d7_make_menu mem_mode \
                .tune.memmode.$band.top.mode.menu \
                d7_memmode($band) \
                "d7_send_string \"VMC $band,\$d7_memmode($band)\""
        label .tune.memmode.$band.top.memory \
                -textvariable d7_current_memory($band)
        pack \
                .tune.memmode.$band.top.mode \
                -side left
        pack \
                .tune.memmode.$band.top.memory \
                -side right
        button .tune.memmode.$band.memname \
                -padx 0 -pady 0 \
                -width [expr $d7_memory_name_width + $d7_memory_name_pad] \
                -textvariable d7_memory(0,name) \
                -command "if { \$d7_memmode($band) == \"2\" } { d7_send_string MSH }"
        bind .tune.memmode.$band.memname $d7_config_event \
                "d7_memory_open \[d7_memory_index_to_num
\$d7_current_memory($band)\]"
        d7_add_to_ab_widget_list $band .tune.memmode.$band.memname
        pack \
                .tune.memmode.$band.top \
                .tune.memmode.$band.memname \
                -expand true -fill both
}
pack \
        .tune.memmode.0 \
        .tune.memmode.1 \
        -expand true -fill both

frame .tune.tone
foreach band { 0 1 } {
        frame .tune.tone.$band
        checkbutton .tune.tone.$band.enable \
                -variable d7_tone_enable($band) \
                -offvalue 0 \
                -onvalue 1 \
                -command "d7_set_radio_parameter TO \$d7_tone_enable($band)"
        d7_add_to_ab_widget_list $band .tune.tone.$band.enable
        menubutton .tune.tone.$band.button \
                -width $d7_val_width(tone) -padx 0 -anchor e \
                -menu .tune.tone.$band.button.menu \
                -textvariable d7_tone_freq_name($band)
        d7_add_to_ab_widget_list $band .tune.tone.$band.button
        d7_make_menu tone \
                .tune.tone.$band.button.menu \
                d7_tone_freq($band) \
                "d7_set_radio_parameter TN \$d7_tone_freq($band)"
        label .tune.tone.$band.units -text "Hz"
        pack \
                .tune.tone.$band.enable \
                -side top
        pack \
                .tune.tone.$band.button \
                .tune.tone.$band.units \
                -side left
}
label .tune.tone.label -text "PL Tone" -padx 0 -pady 0
pack \
        .tune.tone.0 \
        .tune.tone.label \
        .tune.tone.1 \
        -expand true -fill both
```

```
frame .tune.ctcss
foreach band { 0 1 } {
        frame .tune.ctcss.$band
        frame .tune.ctcss.$band.ef
        checkbutton .tune.ctcss.$band.ef.enable \
                -variable d7_ctcss_enable($band) \
                -offvalue 0 \
                -onvalue 1 \
                -command "d7_set_radio_parameter CT \$d7_ctcss_enable($band)"
        d7_add_to_ab_widget_list $band .tune.ctcss.$band.ef.enable
        menubutton .tune.ctcss.$band.ef.freq \
                -width $d7_val_width(tone) -padx 0 -anchor e \
                -menu .tune.ctcss.$band.ef.freq.menu \
                -textvariable d7_ctcss_freq_name($band)
        d7_add_to_ab_widget_list $band .tune.ctcss.$band.ef.freq
        d7_make_menu tone \
                .tune.ctcss.$band.ef.freq.menu \
                d7_ctcss_freq($band) \
                "d7_set_radio_parameter CTN \$d7_ctcss_freq($band)"
        label .tune.ctcss.$band.ef.units -text "Hz"
        button .tune.ctcss.$band.match \
                -width 1 -padx 0 \
                -command "d7_send_string \"CTD $band\""
        pack \
                .tune.ctcss.$band.ef.enable \
                -side top
        pack \
                .tune.ctcss.$band.ef.freq \
                .tune.ctcss.$band.ef.units \
                -side left
        pack \
                .tune.ctcss.$band.ef \
                .tune.ctcss.$band.match \
                -side left -expand true -fill both
}
label .tune.ctcss.label -text CTCSS -padx 0 -pady 0
pack \
        .tune.ctcss.0 \
        .tune.ctcss.label \
        .tune.ctcss.1 \
        -expand true -fill both
pack \
        .tune.balance \
        .tune.select \
        .tune.asc \
        .tune.squelch \
        .tune.power \
        .tune.bandopt \
        .tune.freq \
        .tune.stepoffset \
        .tune.reverse \
        .tune.signal \
        .tune.memmode \
        .tune.tone \
        .tune.ctcss \
        -side left -expand 1 -fill both
pack \
        .tune \
        -expand 1 -fill both

frame .misccontrol
frame .misccontrol.cbcol
checkbutton .misccontrol.cbcol.dual -text Dual -anchor w \
        -command {d7_set_radio_parameter DL $d7_parameter(DL)} \
        -variable d7_parameter(DL) \
        -offvalue 0 -onvalue 1
checkbutton .misccontrol.cbcol.fullduplex -text "Full Duplex" -anchor w \
        -command {d7_set_radio_parameter DUP $d7_parameter(DUP)} \
```

```
        -variable d7_parameter(DUP) \
        -offvalue 0 -onvalue 1
checkbutton .misccontrol.cbcol.lamp -text Lamp -anchor w \
        -command {d7_set_radio_parameter LMP $d7_parameter(LMP)} \
        -variable d7_parameter(LMP) \
        -offvalue 0 -onvalue 1
checkbutton .misccontrol.cbcol.locked -text "Locked" -anchor w \
        -command {d7_set_radio_parameter LK $d7_parameter(LK)} \
        -variable d7_parameter(LK) \
        -offvalue 0 -onvalue 1
checkbutton .misccontrol.cbcol.txinhibit -text "TX Inhibit" -anchor w \
        -command {d7_set_radio_parameter TXS $d7_parameter(TXS)} \
        -variable d7_parameter(TXS) \
        -offvalue 0 -onvalue 1
checkbutton .misccontrol.cbcol.aip -text AIP -anchor w \
        -variable d7_parameter(AIP) \
        -command {d7_set_radio_parameter AIP $d7_parameter(AIP)} \
        -offvalue 0 -onvalue 1
pack \
        .misccontrol.cbcol.dual \
        .misccontrol.cbcol.fullduplex \
        .misccontrol.cbcol.lamp \
        .misccontrol.cbcol.locked  \
        .misccontrol.cbcol.txinhibit \
        .misccontrol.cbcol.aip \
        -expand true -fill both
frame .misccontrol.menucol

frame .misccontrol.menucol.apo
checkbutton .misccontrol.menucol.tnc -text TNC -anchor w \
        -variable d7_parameter(TNC) \
        -command {d7_set_radio_parameter TNC $d7_parameter(TNC)}
label .misccontrol.menucol.apo.label -text APO
checkbutton .misccontrol.menucol.noiseshift -text "Noise Shift" -anchor w \
        -variable d7_parameter(NSFT) \
        -command {d7_set_radio_parameter NSFT $d7_parameter(NSFT)}
menubutton .misccontrol.menucol.apo.menubutton -menu
.misccontrol.menucol.apo.menubutton.menu -textvariable d7_APO_delay
pack \
        .misccontrol.menucol.apo.label \
        .misccontrol.menucol.apo.menubutton \
        -side left -expand false -fill y
d7_make_menu APO \
        .misccontrol.menucol.apo.menubutton.menu \
        d7_parameter(APO) \
        {d7_set_radio_parameter APO $d7_parameter(APO)}
frame .misccontrol.menucol.beep
label .misccontrol.menucol.beep.label -text Beep
menubutton .misccontrol.menucol.beep.menubutton -textvariable d7_beep \
        -menu .misccontrol.menucol.beep.menubutton.menu
d7_make_menu beep \
        .misccontrol.menucol.beep.menubutton.menu \
        d7_parameter(BEP) \
        {d7_set_radio_parameter BEP $d7_parameter(BEP)}
pack \
        .misccontrol.menucol.beep.label \
        .misccontrol.menucol.beep.menubutton \
        -side left -expand false -fill both

frame .misccontrol.menucol.lcd
label .misccontrol.menucol.lcd.label -text "LCD Contrast"
scale .misccontrol.menucol.lcd.scale -orient horizontal -showvalue false \
        -width 10 \
        -length 60 \
        -sliderlength 10 \
        -from 1 -to 8 \
        -variable d7_parameter(CNT) \
        -command {d7_send_cat_string "CNT 0"}
pack \
```

```
        .misccontrol.menucol.lcd.label \
        .misccontrol.menucol.lcd.scale \
        -side left -expand false

frame .misccontrol.menucol.battery_saver
label .misccontrol.menucol.battery_saver.label -text "Battery Saver"
menubutton .misccontrol.menucol.battery_saver.button \
        -textvariable d7_battery_saver \
        -menu .misccontrol.menucol.battery_saver.button.menu
d7_make_menu battery_saver \
        .misccontrol.menucol.battery_saver.button.menu \
        d7_parameter(SV) \
        {d7_set_radio_parameter SV $d7_parameter(SV)}
pack \
        .misccontrol.menucol.battery_saver.label \
        .misccontrol.menucol.battery_saver.button \
        -side left
pack \
        .misccontrol.menucol.tnc \
        .misccontrol.menucol.noiseshift \
        .misccontrol.menucol.apo \
        .misccontrol.menucol.beep \
        .misccontrol.menucol.lcd \
        .misccontrol.menucol.battery_saver \
        -expand true -fill both

frame .misccontrol.dtmfcol
frame .misccontrol.dtmfcol.aro
checkbutton .misccontrol.dtmfcol.aro.on -text "Auto Repeater Offset" \
        -variable d7_parameter(ARO) -offvalue 0 -onvalue 1 \
        -command {d7_set_radio_parameter ARO $d7_parameter(ARO)}
pack \
        .misccontrol.dtmfcol.aro.on \
        -side left -expand false
pack \
        .misccontrol.dtmfcol.aro \
        -expand true -fill both

frame .misccontrol.dtmfcol.speed
label .misccontrol.dtmfcol.speed.label -text "DTMF Speed"
radiobutton .misccontrol.dtmfcol.speed.fast -text Fast \
        -value 0 \
        -variable d7_parameter(TSP) \
        -command {d7_set_radio_parameter TSP $d7_parameter(TSP)}
radiobutton .misccontrol.dtmfcol.speed.slow -text Slow \
        -value 1 \
        -variable d7_parameter(TSP) \
        -command {d7_set_radio_parameter TSP $d7_parameter(TSP)}
pack \
        .misccontrol.dtmfcol.speed.label \
        -side left
pack \
        .misccontrol.dtmfcol.speed.fast \
        .misccontrol.dtmfcol.speed.slow \
        -side left -expand false -fill both
frame .misccontrol.dtmfcol.pause
label .misccontrol.dtmfcol.pause.label -text "DTMF Pause"
menubutton .misccontrol.dtmfcol.pause.menubutton \
        -textvariable d7_DTMF_delay_mSecs \
        -menu .misccontrol.dtmfcol.pause.menubutton.menu
d7_make_menu DTMF_delay \
        .misccontrol.dtmfcol.pause.menubutton.menu \
        d7_parameter(PT) \
        {d7_set_radio_parameter PT $d7_parameter(PT)}
label .misccontrol.dtmfcol.pause.units -text "mSecs"
pack \
        .misccontrol.dtmfcol.pause.label \
        .misccontrol.dtmfcol.pause.menubutton \
        .misccontrol.dtmfcol.pause.units \
```

```
        -side left -expand false -fill none
pack \
        .misccontrol.dtmfcol.speed \
        .misccontrol.dtmfcol.pause \
        -expand true -fill both

frame .misccontrol.dtmfcol.dcd
label .misccontrol.dtmfcol.dcd.label -text "DCD Sense"
menubutton .misccontrol.dtmfcol.dcd.button \
        -textvariable d7_dcd_sense \
        -menu .misccontrol.dtmfcol.dcd.button.menu
d7_make_menu DCD \
        .misccontrol.dtmfcol.dcd.button.menu \
        d7_parameter(DS) \
        {d7_set_radio_parameter DS $d7_parameter(DS)}
pack \
        .misccontrol.dtmfcol.dcd.label \
        .misccontrol.dtmfcol.dcd.button \
        -side left
pack \
        .misccontrol.dtmfcol.dcd \
        -expand true -fill both

checkbutton .misccontrol.dtmfcol.chandisp -text "Channel Display" -anchor w \
        -variable d7_parameter(CH) \
        -command {d7_set_radio_parameter CH $d7_parameter(CH)} \
        -offvalue 0 -onvalue 1
pack \
        .misccontrol.dtmfcol.chandisp \
        -expand true -fill both

checkbutton .misccontrol.dtmfcol.autoinfo -text "Auto Information" -anchor w \
        -variable d7_parameter(AI) \
        -command {d7_set_radio_parameter AI $d7_parameter(AI)} \
        -offvalue 0 -onvalue 1

pack \
        .misccontrol.dtmfcol.autoinfo \
        -expand true -fill both

frame .misccontrol.ctrlcol
frame .misccontrol.ctrlcol.ptt
button .misccontrol.ctrlcol.ptt.on -text "PTT" \
        -command {d7_send_string "TX $d7_parameter(BC)"}
button .misccontrol.ctrlcol.ptt.off -text "RX" \
        -command {d7_send_string "RX"}
frame .misccontrol.ctrlcol.ptt.ab
label .misccontrol.ctrlcol.ptt.ab.0 -text A
label .misccontrol.ctrlcol.ptt.ab.1 -text B
pack \
        .misccontrol.ctrlcol.ptt.ab.0 \
        .misccontrol.ctrlcol.ptt.ab.1 \
        -expand true -fill both
pack \
        .misccontrol.ctrlcol.ptt.on \
        .misccontrol.ctrlcol.ptt.off \
        -side left -expand true -fill both
pack \
        .misccontrol.ctrlcol.ptt.ab \
        -side left -expand false -fill both
frame .misccontrol.ctrlcol.setcall
button .misccontrol.ctrlcol.setcall.set -text "Set Call Channel" \
        -command {d7_send_string "CIN"}
pack \
        .misccontrol.ctrlcol.setcall.set \
        -side left -expand true -fill both

frame .misccontrol.ctrlcol.scan
frame .misccontrol.ctrlcol.scan.startstop
```

```
button .misccontrol.ctrlcol.scan.startstop.start -text Start \
        -command {d7_send_string "SC 1"}
button .misccontrol.ctrlcol.scan.startstop.stop -text Stop \
        -command {d7_send_string "SC 0"}
pack \
        .misccontrol.ctrlcol.scan.startstop.start \
        .misccontrol.ctrlcol.scan.startstop.stop \
        -side left -expand true -fill both
label .misccontrol.ctrlcol.scan.label -text Scan
frame .misccontrol.ctrlcol.scan.mode
label .misccontrol.ctrlcol.scan.mode.label -text Mode
menubutton .misccontrol.ctrlcol.scan.mode.mode \
        -width $d7_val_width(scan_resume) \
        -textvariable d7_scan_resume \
        -menu .misccontrol.ctrlcol.scan.mode.mode.menu
d7_make_menu scan_resume \
        .misccontrol.ctrlcol.scan.mode.mode.menu \
        d7_parameter(SCR) \
        {d7_set_radio_parameter SCR $d7_parameter(SCR)}
pack \
        .misccontrol.ctrlcol.scan.mode.label \
        .misccontrol.ctrlcol.scan.mode.mode \
        -side left -expand false
pack \
        .misccontrol.ctrlcol.scan.label \
        .misccontrol.ctrlcol.scan.mode \
        .misccontrol.ctrlcol.scan.startstop \
        -side left -expand true -fill both
frame .misccontrol.ctrlcol.onmesg
label .misccontrol.ctrlcol.onmesg.label -text "On Message"
entry .misccontrol.ctrlcol.onmesg.entry -width 8 \
        -textvariable d7_parameter(MES)
button .misccontrol.ctrlcol.onmesg.set -text Set \
        -command {d7_set_radio_parameter MES $d7_parameter(MES)}
pack \
        .misccontrol.ctrlcol.onmesg.label \
        .misccontrol.ctrlcol.onmesg.entry \
        .misccontrol.ctrlcol.onmesg.set \
        -side left
pack \
        .misccontrol.ctrlcol.ptt \
        .misccontrol.ctrlcol.setcall \
        .misccontrol.ctrlcol.scan \
        .misccontrol.ctrlcol.onmesg \
        -expand true -fill both
pack \
        .misccontrol.cbcol \
        .misccontrol.menucol \
        .misccontrol.dtmfcol \
        .misccontrol.ctrlcol \
        -side left -expand true -fill both
pack .misccontrol -expand true -fill both
```