WA9XHN RTTYApp Program Scheduler and Editor

by
William Bytheway AA6ED

Introduction

Radio Teletype (RTTY) transmission is not a new medium. It is an old method that served numerous businesses, military and governmental agencies, news, press and information services, and other entities which had reason to transmit large volumes of information over long distances. A recent interest was expressed to establish a RTTY transmitter on the high frequency bands to provide this service to the community.

The Federal Communications Commission (FCC) has approved a request for a Special Temporary Authority (STA) to establish and operate a 1000-watt Radio Teletype (RTTY) Broadcast Station on 6994KHz and 13972KHz. The purpose of the proposed operation is to ascertain whether or not there is any significant interest on the part of those interested and active in RTTY operation to receive regularly scheduled transmissions of material not specifically amateur radio oriented. FCC rules prohibit such transmissions within the amateur bands. This STA assigned the callsign of WA9XHN.

Included in the application was data allowing use of Frequency Shift Keying (FSK) shifts of 170, 425, and 850 Hz., at BAUD rates of 45.45, 50.0, 56.9, and 74.2. The proposed transmissions would be clear text utilizing the ITA#2 5-level international Teletype code, more commonly referred to as BAUDOT or the Murray code.

The above set of requirements was used to develop an operations concept for a 24-hour a day, 7 days a week RTTY transmitter and scheduler maintainable by a single operator. Operations would be automated, text transmissions would be sheduled per the clock, and whole operation would be autonomous. Morse code (CW) identification would be used once per hour to allow for station identification.

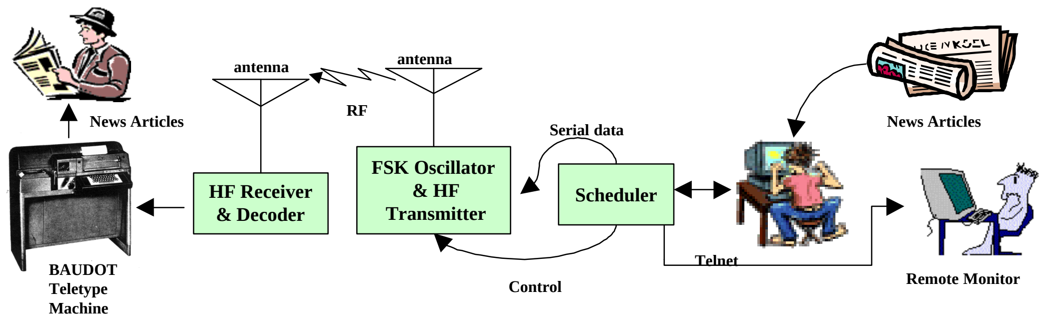


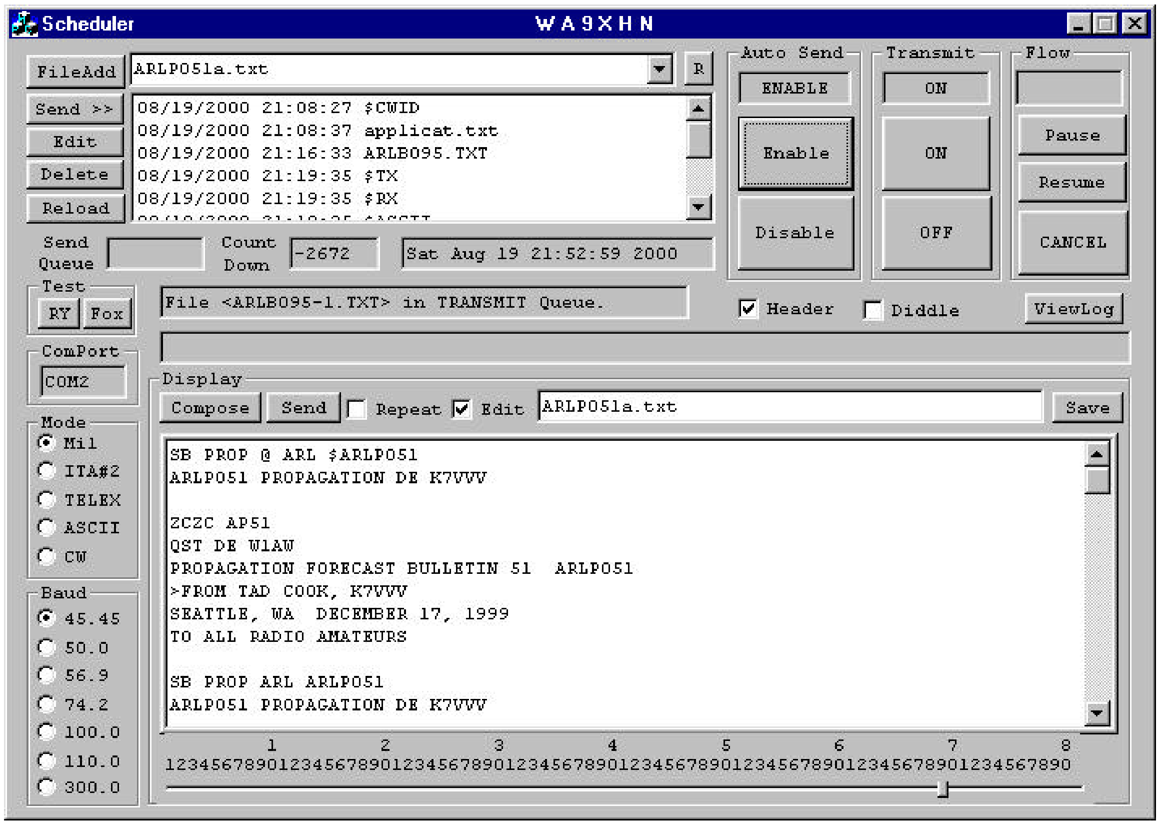
Figure 1: Elements of the WA9XHN Project

As shown in Figure 1, the project involves both a hardware and software solution to the problem. George Hutchison, amateur radio callsign W7KSJ, is providing work to assemble the Collins 1000-watt commercial broadcast transmitter and the FSK oscillator unit responsible for maintaining frequency tolerances as required by the FCC. The FSK oscillator unit will be responsible for providing a switchable (on/off) radio frequency (RF) signal to the transmitter that will be operating in a class-C configuration. This will provide transmitter control without having to switch high voltages to the Collins transmitter final amplifier tube. The antenna is a vertical designed for 360-degree coverage at a low angle of radiation.

The programming and scheduling of transmitted material thus became the software problem for William Bytheway, amateur radio callsign AA6ED. Software control would be used for controlling the transmitter ON/OFF functions, determining BAUD rates and scheduling of clear text broadcast material. The scheduling function would have to be relatively autonomous and pre-defined not only hours in advance of the current time, but also days or weeks.

Program Scheduler

In order to semi-automate the process of scheduling and transmitting data, the scheduler must have the capability to control the BAUD rates, transmitter on/off control and programming material. In addition, the scheduler would have the task of identifying the station call sign, WA9XHN. The target computer platform will be an Intel 80386 or later machine running Windows 95 or later operating system. The program scheduler will be written in Microsoft Visual C++ using the Microsoft Foundation Class (MFC) libraries, which will provide the appropriate graphical user interfaces (GUI) as shown in Figure 2.



ARLP051a.txt

```
08/19/2000 21:08:27 $CWID
08/19/2000 21:08:37 applicat.txt
08/19/2000 21:16:33 ARLB095.TXT
08/19/2000 21:19:35 $TX
08/19/2000 21:19:35 $RX
```

Count Down: -2672

Sat Aug 19 21:52:59 2000

File <ARLB095-1.TXT> in TRANSMIT Queue.

[✓ Header] Diddle

ComPort: COM2

Mode: [✓ Mil] ITA#2 TELEX ASCII CW

Baud: [✓ 45.45] 50.0 56.9 74.2 100.0 110.0 300.0

Repeat [✓ Edit] ARLP051a.txt

```
SB PROP @ ARL $ARLP051
ARLP051 PROPAGATION DE K7VVV

ZCZC AP51
QST DE W1AW
PROPAGATION FORECAST BULLETIN 51  ARLP051
>FROM TAD COOK, K7VVV
SEATTLE, WA  DECEMBER 17, 1999
TO ALL RADIO AMATEURS

SB PROP ARL ARLP051
ARLP051 PROPAGATION DE K7VVV
         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

Figure 2 : Graphical User Interface

The software scheduler contains several modules that are responsible for controlling its functions. Button controls as shown on Figure 2 shows basic human machine interfaces required to manage the scheduler.

1. **Send >> -** In the Send Queue Management group box, the next file to be sent is "Next Action is <$CWID>." to be sent at 21:08:27 hours. If AutoSend Enable is enabled, SendNext button will immediately promote this file to the transmitter, and bring up the next file to be transmitted. AutoSend must be enabled for this button to work. The SendNext button will send the next text file in the queue, if there are any other control commands in the queue, it will step through them and execute them. Control commands are executed and stepped through to set baud rate, mode and transmitter control.

2. **Edit** – Pressing this button invokes a new diaglog window that allows the user to edit the schedule and text files. The current transmission is suspended until the Edit dialog box is done. More information is provided later in this paper.

3. **FileAdd/Delete/Reload -** These functions were added so that the user could highlight a line and delete an entry in the Queue. If the Delete button were pressed, "$CWID" would be removed from the

Queue. If FileAdd were selected, the user could enter a new task using the "mm/dd/yyyy hh:mm:ss filename.txt" format. Long file names are supported, but no spaces are allowed in the name.

4. **RY** and **Fox Test -** The user can define a special file that contains the famous "RYRYRY" or "Quick Brown Fox…." transmit patterns, pressing this button turns on the transmitter and immediately sends this file. The file names are "ryryry.txt" and "thefox.txt" respectively.
5. **ComPort** – This display window shows the current hardware RS-232 communications port in use. The hardware addresses can be changed after runtime by scripting the following commands in the schedule. Examples of these commands are:
```
$COM1    // Comport = 0x3F8
$COM2    // Comport = 0x2F8
$COM3    // Comport = 0x3E8
$COM4    // Comport = 0x3E0
$COM5    // Comport = 0x2F0
$COM6    // Comport = 0x2E8
$COM7    // Comport = 0x2E0
$COM8    // Comport = 0x260
```
6. **Mode -** The primary mode will be the "Mil" or Military mode, or more commonly referred to as BAUDOT code. In addition, the International Telegraph Association ITA#2, TELEX and ASCII mode was provided so that additional testing can be performed. Pressing this button changes the mode for the next scheduled transmission, it does not change the existing transmission in the queue. All of the BAUDOT modes are sent in 5-bit code in upper case. The character set for BAUDOT code is a subset of the ASCII code.
7. **Baud -** The BAUD rates were specifically defined in the STA, with the addition of two more to be primarily used with ASCII. Pressing this button immediately changes the baud rate, even if while transmitting. Supported rates are 45.45, 50.0, 56.9, 74.2, 100.0,110.0 and 300.0 baud.
8. **AutoSend -** This function reads the scheduling file "schedule.lst" and will transmit this file when the time tag has expired. What you see in the list box just to the left is the current schedule. AutoSend must be enabled before the "SendNext" feature will work.
9. **Transmit (ON/OFF) -** This controls the transmitter. If these buttons don't work, then the software application is not capable of controlling the transmitter. You can press these buttons anytime, but if you are currently transmitting a message, data will be lost because it does not stop the data flow. This button is provided for test purposes only, i.e. tuning the transmitter.
10. **Flow Control Group -** While data is being transmitted, you are able to **Pause**, **Resume** or **Cancel** the transmission of this file. This button is provided so that the operator can Pause the transmission of data while the transmitter or antenna is being tuned. The cancel button not only cancels the current session, but it will also disable the AutoSend function. These buttons will not work if a transmission is not in progress.
11. **Count Down Status -** The countdown box displays the number of seconds until the next transmission. If several messages are in the queue with the same time or if the current file over-runs the start time of the next file, this value will go negative.
12. **SendQueue Status -** This box simply shows the number of characters in the current file.
13. **Preview Window -** A preview window has been provided so that the operator can review the text of any file in the send queue. The file size is limited to 32,000 bytes total in size. When a file is transferred to the transmit queue, it will automatically show up in this window.
14. **Compose Button -** This button reads in a pre-canned file named "Compose.txt" as a template. The purpose of this feature is so that the operator can compose a text message for immediate transmission. The text is shown in the edit window below the box. Checking the "Edit" box enables editing.
15. **Send Button** – The Send button sends whatever text is in the edit window below. Prior to sending the text, it first saves the text using the file name “EditDisplayTemp.txt”, then transmits it. This feature preserves the original input file.

16. **Repeat Check Box -** This allows the "RYRYRY" or "The Fox" forever until it is unchecked. It also sends anything in the Edit Window. These are pre-canned files named “ryryryry.txt” and “thefox.txt” and can be found in the Data directory.
17. **Edit Check Box -** The edit window is normally in the non-edit mode and if this box is checked the operator is allowed to edit the text displayed in the edit window. This check box is normally left unchecked.
18. **Save Button -** This button saves whatever text is in the edit window into the file name displayed in the window to the left of the button. The operator can change the name of the file to create versions of the original file. Caution should be used since the original file could be left unusable if the save button is accidentally pressed. At the bottom of the screen there is a slide bar that allows the user to change the right margin of the text. When the save button is pressed, these margin “carriage returns” are saved as soft carriage returns, and can be changed at a later date.
19. **Scroll Bar** – The scroll bar at the bottom of the display allows the user to edit the right margin of the current displayed document and save the changes. The saved changes are soft carriage returns and can be changed at a later time.
20. **ViewLog -** A log file is kept of all operator actions and time tagged using the computer clock. The main purpose of this log is to maintain an accurate log for FCC purposes and to aid in debugging of any software bugs that may occur. A separate log file is created for each calendar day, and entries are time tagged with hour:min:seconds as shown :

Example of Scheduler Log File

```
11:11:52 > CMRX : TRANSMITTER OFF
11:11:52 > CMDISABLE : SCHEDULER DISABLED
11:11:52 > CMDISABLE : SCHEDULER DISABLED
11:11:52 > SETUPWINDOW : COMPORT SET TO 0X2F8
11:11:52 > HANDLEMODEBOXMSG : BAUDOT MODE IS SET
11:11:52 > HANDLEBAUDBOXMSG : BAUD RATE IS SET TO 45.5
11:12:06 > CMENABLE : SCHEDULER ENABLED
```

Scheduler Editing

Editing the scheduler is very critical for the proper operation of this software application. Most combinations of bad inputs have been tested, but the basic design is such that on reading the file, only those events in the future will be scheduled (based on current computer clock). The scheduler is controlled by a file called "schedule.lst" that is maintained in the "data" directory. The user may want to further partition data into sub-directories, i.e. "data/subject1" and "data/subject2", just as long as Windows 95 file naming conventions are used without spaces in the file name.

The format for the "schedule.lst" contains a date field, time field and a filename field. If there is anything else on the line, the extra data is discarded. The application will try to interpret the date and time fields and compare them to the current computer clock. If the time is in the future, then the file is scheduled, otherwise it is discarded.

The format for the file is "mm/dd/yyyy hh:mm:ss filename.txt" where mm is the month, dd is the day, yyyy is the year, hh is the hour, mm is minutes, and ss is seconds. The filename may contain the path to the file from the current directory the application is running in. An example of this file is:

```
01/12/2000    19:08:00    applicat.txt
01/12/2000    18:56:30    ARLB094.txt
ARLP053.txt                                 // see text
01/12/2000    22:45:00    ARLB096.txt
01/12/2000    23:00:00    ARLB095.txt
01/12/2000    23:15:00    ARLP050.txt
```

In the example above, if there is a single entry on the line, it will assume the time for the previous entry plus the calculated run time for the file. The example will have an execution time of "01/12/2000

18:56:40" which will be displayed in the Scheduler list box. If the first entry in the list does not have a time field, the time used will be the current computer clock time plus 10 seconds.

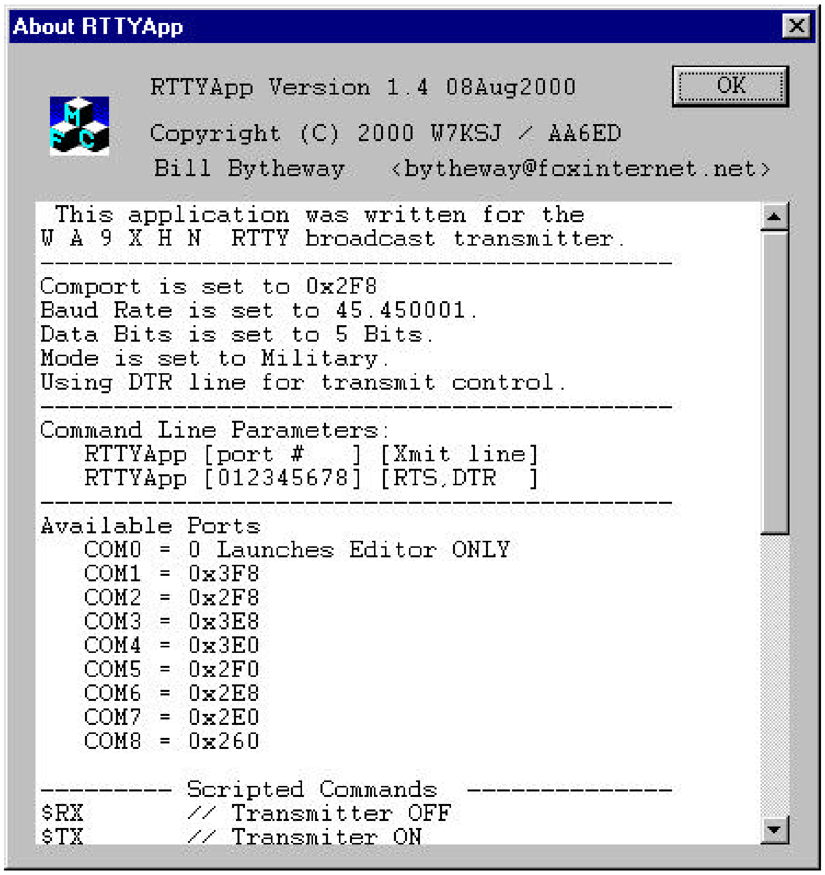
It's suggested that the operator use the RTTYApp Editor to edit and and manage the schedule. The send queue will show activities exactly as presented and will not resort them, so it is up to the operator to sort the inputs by transmit time.

The scheduler also has the ability to script in many Button functions to occur at the scheduled time. This feature will allow the scheduling of Baud rate and mode. The basic command is similar to scheduling of files as shown above with exception that a command is proceeded by a "$" dollarsign.

1. **Baud Rate -** The baud rate is controlled using a $B45, $B50, $B56, $B74, $B100, $B110 and $B300. Just remember that the only baud rates supported are the ones shown on the display. The command may be in either upper or lower case.
2. **Mode -** The five modes supported are the BAUDOT codes Mil (Military), ITA2, TELEX and ASCII. The command format is $MIL, $ITA#2, $TELEX, and $ASCII and may be in either upper case or lower case. The '$' sign is important, since it tells the scheduler that the following input is a command and not a filename. The $CW mode is a true CW mode and uses either RTS or DTR to control the transmitter on/off keying. The RTS/DTR control is set at initialization of the program.
3. **ComPort** – After runtime, the operator can change the current RS-232 port to a new port. The purpose of this feature is to allow the control of multiple transmitters, one at a time. Before changing port addresses, the software executes a receive command to turn off the transmitter. Valid commands are:
```
$COM1   // Comport = 0x3F8
$COM2   // Comport = 0x2F8
$COM3   // Comport = 0x3E8
$COM4   // Comport = 0x3E0
$COM5   // Comport = 0x2F0
$COM6   // Comport = 0x2E8
$COM7   // Comport = 0x2E0
$COM8   // Comport = 0x260
```
4. **Header Control** – Each RTTY transmission has a header and footer file transmitted before and after each file. Disabling this check box turns this feature off. This feature can be controlled by a command $HEADER or $NOHEADER.
5. **Diddle** – Common in modern RTTY transmissions is the desirement to send a letters (LTRS) character while no data is being sent. The receiving end can easily identify that the transmitter is a RTTY station just by the sound. This feature can be controlled by a command $DITTLE or $NODITTLE.
6. **Transmit Control -** Other commands are $TX and $RX for turning on/off the transmitter at prescheduled times. In addition, a $DISABLE command is available, but use it wisely. Once the "AutoSend" feature is disable, operator action is required to enable the scheduler. The $CWID function sends a “de WA9XHN WA9XHN” in RTTY frequency shift keyed CW at about 10 words per minute.
7. **Comments** – The user can store comments in the schedule plan with a “$--xxxx” command where the “xxxx” is the text of the message. No spaces are allowed, and the message length is limited to 32 characters.
8. **$CWID –** This command provides the sending of a morse code identification using FSK instead of the usual on-off keying usually used. The CWID is hardcoded to send “de WA9XHN WA9XHN”.

System Information

An AboutRTTYApp feature as shown in Figure 3 is accessible from the upper left hand title bar icon and describes the features and setting of this application.



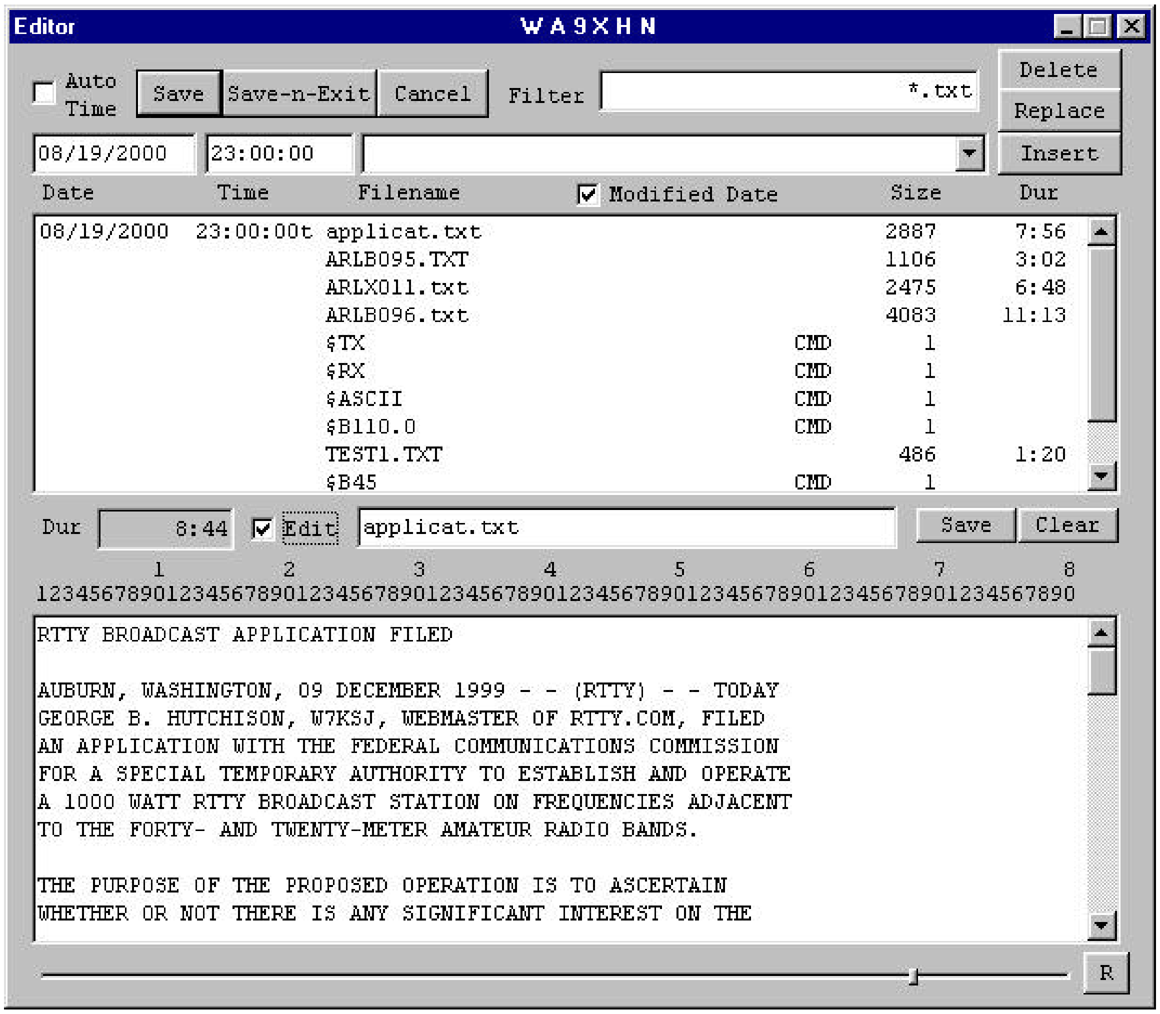
```
RTTYApp Version 1.4 08Aug2000

Copyright (C) 2000 W7KSJ / AA6ED
Bill Bytheway   <bytheway@foxinternet.net>

 This application was written for the
W A 9 X H N  RTTY broadcast transmitter.
-------------------------------------------
Comport is set to 0x2F8
Baud Rate is set to 45.450001.
Data Bits is set to 5 Bits.
Mode is set to Military.
Using DTR line for transmit control.
-------------------------------------------
Command Line Parameters:
   RTTYApp [port #    ] [Xmit line]
   RTTYApp [012345678] [RTS,DTR   ]
-------------------------------------------
Available Ports
   COM0 = 0 Launches Editor ONLY
   COM1 = 0x3F8
   COM2 = 0x2F8
   COM3 = 0x3E8
   COM4 = 0x3E0
   COM5 = 0x2F0
   COM6 = 0x2E8
   COM7 = 0x2E0
   COM8 = 0x260

---------- Scripted Commands  --------------
$RX        // Transmitter OFF
$TX        // Transmiter ON
```

Figure 3 About RTTYApp Information Window

This dialog box shows the current build version of the RTTYApp application. It also shows the current status and setting of many parameters in the application. It provides the user with information on command line parameters, communications port support and script commands.

RTTY Schedule Editor

An enhanced editor has been added to the RTTY Scheduler. The editor is invoked by pressing the "Edit" pushbutton on the main screen. The editor provides two basic functions, (1) allows editing of the schedule, and (2) allows editing of transmission text data. Figure 4 shows an example of the schedule page that will be discussed next.



```
Date        Time       Filename                              Size     Dur
08/19/2000  23:00:00t  applicat.txt                          2887    7:56
                       ARLB095.TXT                           1106    3:02
                       ARLX011.txt                           2475    6:48
                       ARLB096.txt                           4083   11:13
                       $TX                           CMD        1
                       $RX                           CMD        1
                       $ASCII                        CMD        1
                       $B110.0                       CMD        1
                       TEST1.TXT                              486    1:20
                       $B45                          CMD        1

Dur 8:44

         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
RTTY BROADCAST APPLICATION FILED

AUBURN, WASHINGTON, 09 DECEMBER 1999 - - (RTTY) - - TODAY
GEORGE B. HUTCHISON, W7KSJ, WEBMASTER OF RTTY.COM, FILED
AN APPLICATION WITH THE FEDERAL COMMUNICATIONS COMMISSION
FOR A SPECIAL TEMPORARY AUTHORITY TO ESTABLISH AND OPERATE
A 1000 WATT RTTY BROADCAST STATION ON FREQUENCIES ADJACENT
TO THE FORTY- AND TWENTY-METER AMATEUR RADIO BANDS.

THE PURPOSE OF THE PROPOSED OPERATION IS TO ASCERTAIN
WHETHER OR NOT THERE IS ANY SIGNIFICANT INTEREST ON THE
```

Figure 4 Schedule Editor

The top half of the scheduler allows the editing of the schedule. In this example, the data as shown is very similar to the schedule file "schedule.lst" that can be edited by hand. The 't' at the end of the time field indicated that this is a time tagged entry. All of the subsequent files transmission times are calculated at run time based on the baud rate. An explanation of the buttons is as follows:

1. **AutoTime** – Checking this box changes the display from the plain "schedule.lst" view to schedule with calculated times. These changes are only displayed, and only the times tagged with a 't' are saved in the new "schedule.lst" file.
2. **Save** – This saves the temporary edit schedule into the "schedule.lst" file without exiting the dialog box.
3. **Save-n-Exit** – Saves the tempory edit schedule into the "schedule.lst" file and returns to the Scheduler.
4. **Cancel** – Cancels all operations in the editor and returns to the Scheduler.

5. **Filter** – Changing this edit field allow the user to filter files that are displayed in the list combo box below.
6. **Modified Date** – Changes the order in which directory files are displayed in the drop-down combo box. If the box is checked, files are shown sorted by last-modified date and time. If the box is unchecked, files are sorted in alphabetical order.
7. **Delete** – Deletes a schedule entry in the list box
8. **Replace** – Replaces a schedule entry in the list box.
9. **Insert** – Inserts the new Date, Time, Filename entry ABOVE the highlighted entry in the list box. If no entry is highlighted, the entry is placed at the TOP of the list box.
10. **Date, Time, Filename Edit Windows** – The user can enter a date (mm/dd/yyyy) , time (hh:mm:ss) and short or long filename in this field. If the date and time are blank, the new entry is not tagged with a time and will execute based on the calculated run time of the previous entry. If there is no previous entry, then the time is the current time + 30 seconds.
11. **Filename** – The filename combo list box deserves some more discussion. The down arrow opens up a drop down list box that contains all of the data files available for transmission. Using the mouse, selecting a file brings it up to the top box and also provides a preview in the edit window at the bottom of this dialog. At the end of the list box are all of the commands (i.e. $RX, $TX,………$DISABLE) that the user can input in the schedule. The user can also manually type in an entry and press return. If the file is invalid, the edit window will say [PrintFile " Cannot open xxxx.txt input file.] and if you try to insert the file, the scheduler will show "????" next to the duration.

Figure 5 shows the schedule when the "AutoTime" checkbox is pressed. As you can see, times are calculated based on the size and duration of the executable file. Following discussion will describe the text editor. If a file is invalid, the size will show "????", and if an input is a command, the user will see a "CMD" in the description.

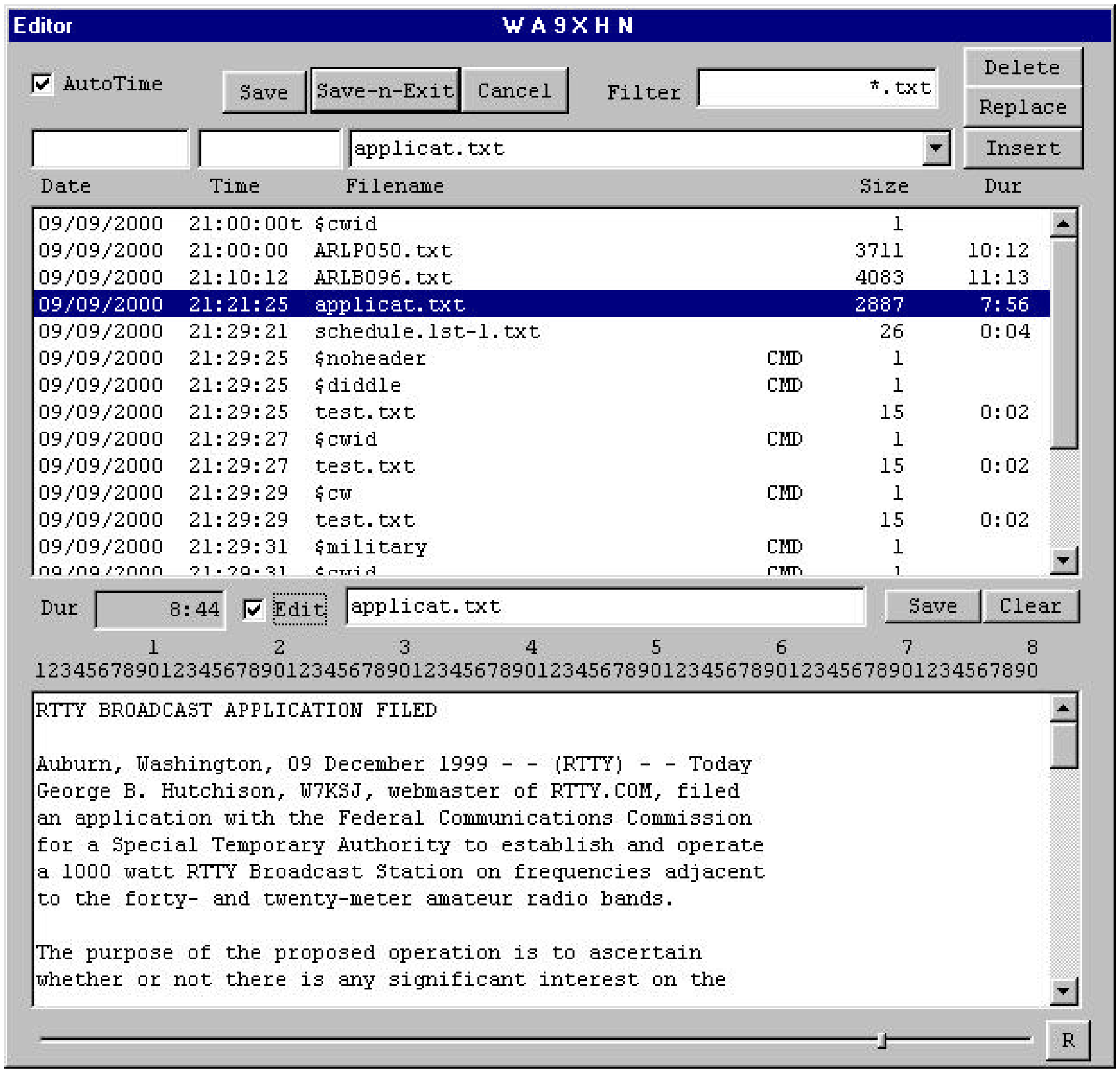
Schedule Text Editor

The editor shown on the bottom of Figure 5 shows the calculated file duration, and edit enable checkbox file input box, save and clear pushbuttons. When the filename box as described above selects a file, it is shown in the editor window and the background is grayed. To edit the window, check the edit check box and the background will change to white. At this time, if you move the slide bar at the bottom of the window, the text right margin will be adjusted based upon it's position. This feature is provided so that the user can customize the charters-per-line to anything they want (66 characters per line is recommended).

One note about using this editor is that Microsoft uses two versions of line formatting:

1. Hard carriage lines use a "carriage-return/line-feed" ('\r\n') combination for determining a new line. This combination is carried forward and cannot be edited out with replacing each entry.
2. Soft carriage lines are defined as a "carriage-return/carriage-return/line-feed" ('\r\r\n') combination. The RTTYApp Editor uses soft carriage return for editing the file. This allows the user to reformat the file later. Other editors may not understand soft carriage returns, so be carefull when editing files processed by the RTTYApp Editor. The transmit portion of the Scheduler does not use the carriage-return and only uses the line feeds. In this way, it can use the soft carriage returns and still maintain line control.

A refresh button is provided in the lower right corner to support heritage Microsoft VC++ version 5.0 and has been fixed on later versions of the release. The Slider bar provides it's own callbacks to the editor when the left mouse button is released.



```
09/09/2000  21:00:00t $cwid                                  1
09/09/2000  21:00:00  ARLP050.txt                         3711     10:12
09/09/2000  21:10:12  ARLB096.txt                         4083     11:13
09/09/2000  21:21:25  applicat.txt                        2887      7:56
09/09/2000  21:29:21  schedule.lst-1.txt                    26      0:04
09/09/2000  21:29:25  $noheader                   CMD        1
09/09/2000  21:29:25  $diddle                     CMD        1
09/09/2000  21:29:25  test.txt                              15      0:02
09/09/2000  21:29:27  $cwid                       CMD        1
09/09/2000  21:29:27  test.txt                              15      0:02
09/09/2000  21:29:29  $cw                         CMD        1
09/09/2000  21:29:29  test.txt                              15      0:02
09/09/2000  21:29:31  $military                   CMD        1
09/09/2000  21:29:31  $cwid                       CMD        1

Dur 8:44   Edit applicat.txt

         1         2         3         4         5         6         7         8
12345678901234567890123456789012345678901234567890123456789012345678901234567890

RTTY BROADCAST APPLICATION FILED

Auburn, Washington, 09 December 1999 - - (RTTY) - - Today
George B. Hutchison, W7KSJ, webmaster of RTTY.COM, filed
an application with the Federal Communications Commission
for a Special Temporary Authority to establish and operate
a 1000 watt RTTY Broadcast Station on frequencies adjacent
to the forty- and twenty-meter amateur radio bands.

The purpose of the proposed operation is to ascertain
whether or not there is any significant interest on the
```

Figure 5 Editor with AutoTime CheckBox Checked

ASCII - ITA#2 Protocol and Baud Rate Control

As shown in the previous section, the schedule.lst file controls all of the scheduling of files to be sent. In addition to being able to schedule files, there is an additional feature that allows the operator to schedule the protocol and baud rate desired for that part of the broadcast program.

A typical schedule will contain protocol mode and baud rate control as shown in the following example. Three files are scheduled to be transmitted, the first at 50.0 baud, ITA#2 mode, the second file at 110.0 baud in ASCII, and the third at 45.45 baud in ITA#2.

```
01/14/2000   11:30:20   $B50           // sets the baud rate to 45.45
01/14/2000   11:30:21   $ITA#2         // sets ITA#2 Baudot mode
01/14/2000   11:30:22   applicat.txt   // transmits the file
01/14/2000   11:55:30   $B110.0        // sets the baud rate to 110.0
01/14/2000   11:55:31   $ASCII         // sets ASCII mode
01/14/2000   11:55:32   ARLP050.txt    // transmits the file
01/14/2000   13:55:46   $b45           // sets the baud rate to 45.45, lower case is OK
01/14/2000   13:55:48   $ita#2         // sets  ITA#2 Baudot mode, lower case is OK
01/14/2000   12:55:49   ARLP051.txt    // transmits the file
```

The default mode is always ITA#2 and 45.45 baud when starting the application. While it is not expected that other modes will be used, nevertheless they are provided.

Two files should be edited containing call sign information, they are in the data directory and are named header.txt and footer.txt. These files are sent before and after each scheduled transmission.

RTTY Terminal Program

An additional RTTY Terminal program is provided with the RTTYApp Scheduler package, but will not be discussed in detail here. This is a basic RS-232 terminal program with the additional feature of ASCII to BAUDOT conversion. It was designed to interface with the HAL Communications Corporation ST-6 Terminal Unit that provides the interface between the computer, incoming BAUDOT Frequency Shift Keyed (FSK) tones and a Teletype machine. The HAL ST-6 operates similar to a telephone modem with additional features. Computer function keys "F1" and "F2" turn the transmitter OFF and ON using the RS-232 DTR line. The "F3" key is used to key the RTS line, which on my unit is used for Morse code (CW) identification. The software provides end of line wrapping, and responds to the "NNNN" sequence at the beginning of a new line to turn off the transmitter. It uses a single command line parameter for setting the RS-232 communications port number.

The RTTY Terminal Program is basically the same as the TCP/IP Babble Box program without the TCP/IP interface and with serial hardware interrupt driven access to the 16450/16550 Universal Asynchronous Receiver Transmitter (UART).

Metrics on Source Lines of Code

The following tables shows an estimated lines of code (LOC) for each application detailed in this paper.

Scheduler (Microsoft VC++)	**LOC**
RTTYApp.cpp	15
RTTYAppDlg.cpp	498
RTTYSchedule.cpp	500
RS232.cpp Com Port Driver	64
RTTYApp.h	8
RTTYAppDlg.h	56
Defines.h	38
Code.h	46
Resource.h	57
Comments	625

TCP/IP Telnet Server	**LOC**
Wsmtpsrv.c	844
RS232.c	72
Net.c	504
Net.h	68
Codes.h	211
Wsmtpsrv.h	109
Resource.h	14
Dialogs.h	78
Comments	371

TCP/IP Babble Box	**LOC**
Babble.cpp	938
Tcpip.c	234
Main.c	30
Comments	750

RTTYApp Command Line Parameters

As usual, there is always something left out in any document. The running of the RTTY scheduler allows for two modes of operations.

1. In the primary mode the communications ports are opened. Valid COM ports are COM1 through COM8 and valid ports are identified in the "About RTTYApp" menu. The default port is 2.
2. In the editor mode, only the Editor is invoked as in the case of COM0 or port number set to zero.

In addition, the transmit control line can use the RS-234 DTR or RTS line. The default is DTR if left blank. The complete command syntax is:

RTYApp [port # 0,1,2,3,4,5,6,7,8] [RTS,DTR]

Scheduler Design

As shown in Figure 6, news articles are organized into a schedule definition file that defines the date and start time for input text the file. The schedule is loaded and displayed in a list box for the operator to view and modify as required. When the schedule is enabled, the time for the next transmissions is checked against the computer clock, and when ready, is passed to the next function for transmitting. Depending on the MODE setting (Military, TELEX, ITA#2 or ASCII) the data is passed to the Transmit and RS-232 controls.

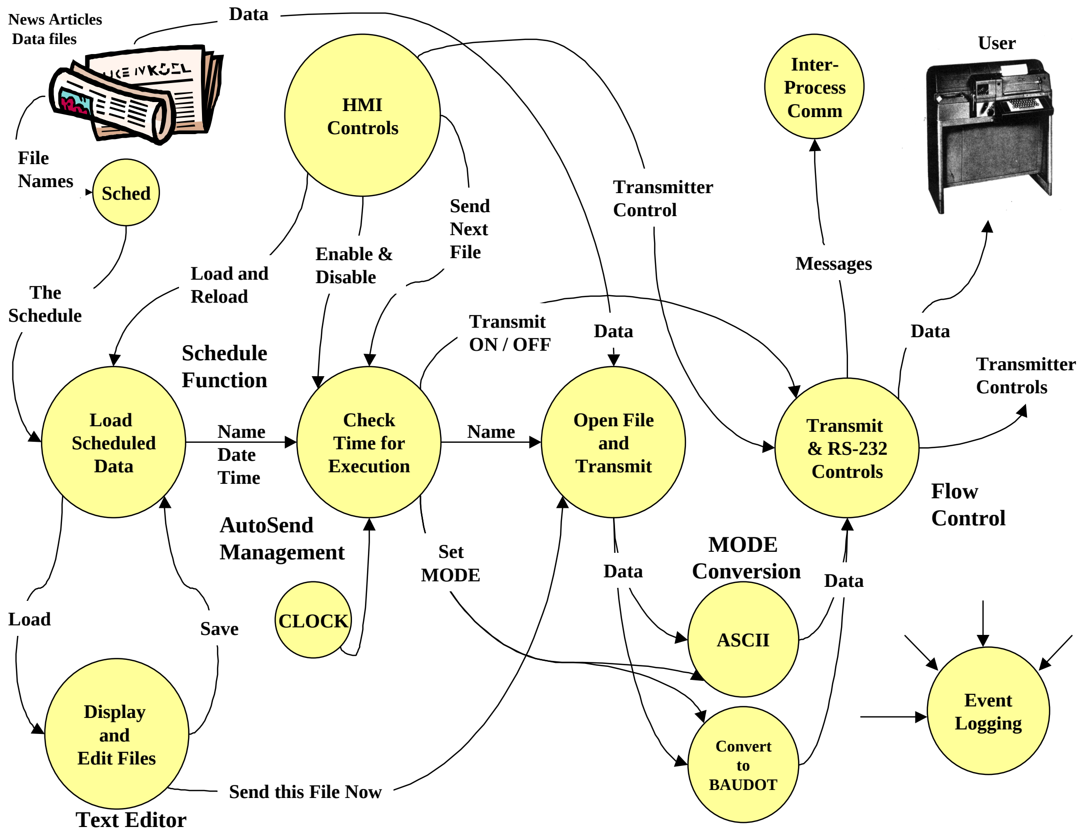


Figure 6: Scheduler Data Flow

In this version, Microsoft Visual C++ MFC provides the appropriate controls and messaging to send the appropriate message controls from the push buttons, check boxes, and edit windows. In addition, the Resource Editor was used to create the Human Machine Interfaces (HMI). Resources are data that define the visible portions of your Windows program. Resources provide a consistent user interface that makes it easy for users to switch from one Windows program to another.

In general, a Windows application's resources are separate from the program code, letting you make significant changes to the interface without even opening the file that contains your program code.

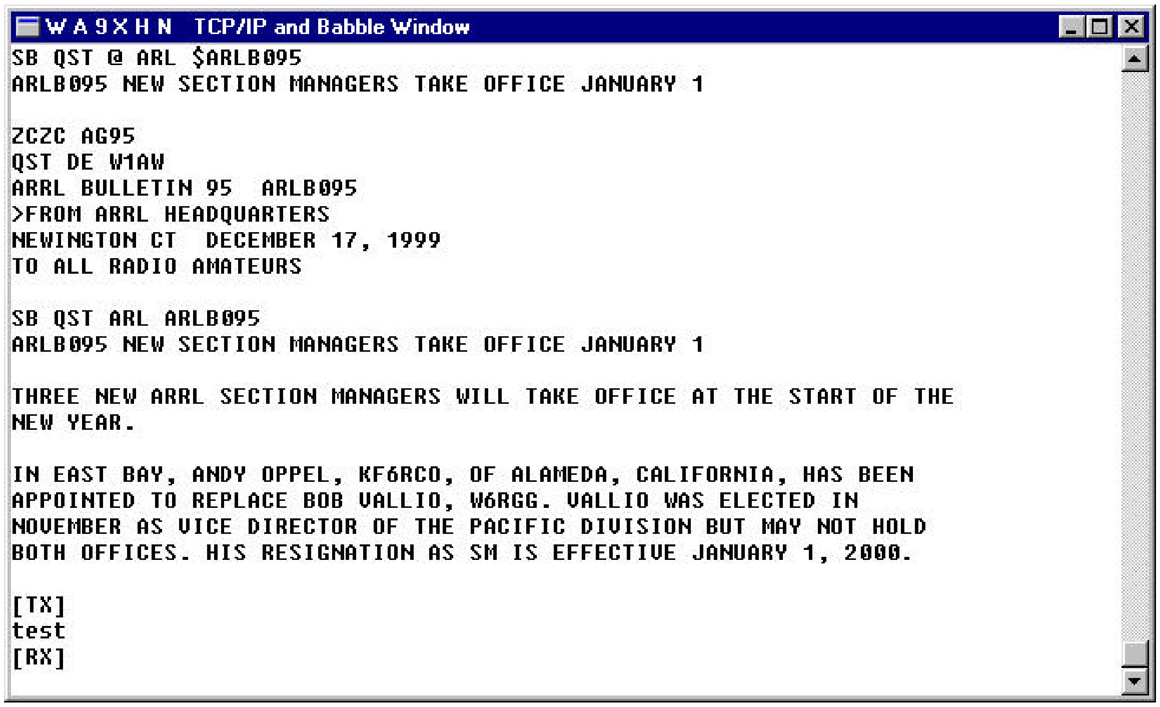
TCP/IP and Babble Box Application Tool

Another tool included with the scheduler is an application that provides two features, a TCP/IP interface and a Babble Box. This application as shown in Figure 7 can be started up after the scheduler is active and will receive up to 800 lines of any text that is being transmitted by the scheduler. In addition, it provides a TCP/IP Telnet interface (port 23) that allows anyone on the network to telnet into the application and monitor activity.

- The TCP/IP interface requires that the Windows 3.1 or Windows95 computer be equipped with an Ethernet card, TCP/IP Winsock and static IP address. The application acts as a server and only sends data out, it does not currently allow an external user the ability to control actions. The display will announce the IP address of a user when they connect, and announce when there is a disconnect. When a user Telnets into the Babble box, they are greeted with the message " Welcome to WA9XHN TCP/IP Listen Port".

- The Babble Box interface basically receives transmitted data from the scheduler. It also provides a transmit/receive capability using <F2> and <F1> respectively. The operator then can use the keyboard or paste to send data. The operator is locked out of sending data if the scheduler is transmitting a file.

The TCP/IP and Babble Box functions completely independently as a separate application from the scheduler and communicates with the scheduler using windows messages. It is important that the operator use a control-C to terminate the session and then close the window with the [X] on the title bar to allow for any open TCP/IP sockets to be closed properly.

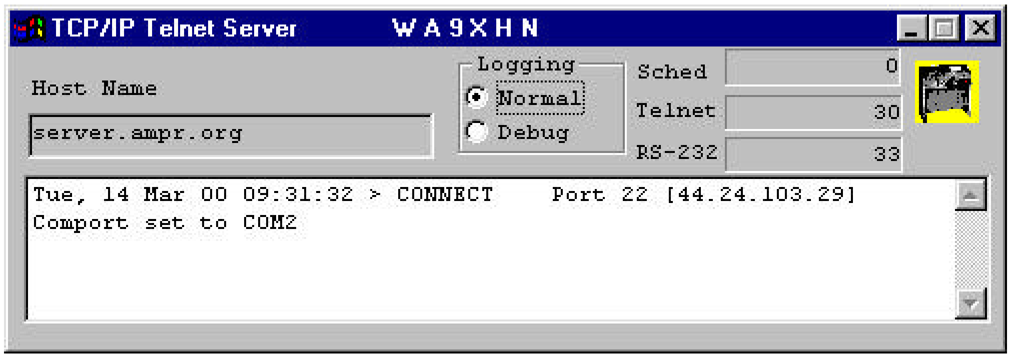


```
SB QST @ ARL $ARLB095
ARLB095 NEW SECTION MANAGERS TAKE OFFICE JANUARY 1

ZCZC AG95
QST DE W1AW
ARRL BULLETIN 95  ARLB095
>FROM ARRL HEADQUARTERS
NEWINGTON CT  DECEMBER 17, 1999
TO ALL RADIO AMATEURS

SB QST ARL ARLB095
ARLB095 NEW SECTION MANAGERS TAKE OFFICE JANUARY 1

THREE NEW ARRL SECTION MANAGERS WILL TAKE OFFICE AT THE START OF THE
NEW YEAR.

IN EAST BAY, ANDY OPPEL, KF6RCO, OF ALAMEDA, CALIFORNIA, HAS BEEN
APPOINTED TO REPLACE BOB VALLIO, W6RGG. VALLIO WAS ELECTED IN
NOVEMBER AS VICE DIRECTOR OF THE PACIFIC DIVISION BUT MAY NOT HOLD
BOTH OFFICES. HIS RESIGNATION AS SM IS EFFECTIVE JANUARY 1, 2000.

[TX]
test
[RX]
```

Figure 7 TCP/IP and Babble Window Example

TCP/IP Telnet Server Application Tool

The TCP/IP and Babble Box server only allowed a single Telnet session at a time to take place which limited the number ofusers that could be active at any one time. In the public domain, Ian Blenke wrote a Windows 3.1 SMTP Server (WSMTPSrv.C) that supported multiple connections. The re-use of his software allowed for the quick building of a Telnet server using Microsoft Visual C++ version 5.0 WIN32 application as shown in Figure 8. Unlike the Babble Box, this application supports bi-directional sending and receiving of data to all Telnet users and sending data to the RS-232 port.



Host Name: server.ampr.org
Logging: [✓ Normal] Debug
Sched: 0
Telnet: 30
RS-232: 33

```
Tue, 14 Mar 00 09:31:32 > CONNECT    Port 22 [44.24.103.29]
Comport set to COM2
```

Figure 8: TCP/IP Telnet Server

The SMTP code was striped out and replaced with the Baudot conversion utilities and RS-232 controls that allowed multiple users to control the transmitter using the "F1" and "F2" keys to turn OFF and ON the transmitter respectively. Three status windows show the number of bites received from the Scheduler via Windows messages, number of bytes received from Telnet connections and number of bytes sent to the RS-232 port using BAUDOT Military code. Logging allows for additional messages to be sent to the display. Command line parameters 1-5 allows for setting the RS-232 port, the default is COM2.

The Windows Sockets specification defines a network programming interface for Microsoft Windows which is based on the "socket" paradigm popularized in the Berkeley Software Distribution (BSD) from the University of California at Berkeley. It encompasses both familiar Berkeley socket style routines and a set of Windows-specific extensions designed to allow the programmer to take advantage of the message-driven nature of Windows.

Windows Sockets makes provisions for multithreaded Windows processes. A process contains one or more threads of execution. In the Microsoft Windows non-multithreaded world, a task corresponds to a process with a single thread. All references to threads in this document refer to actual "threads" in multithreaded Windows environments. In non-multithreaded environments (such as Windows 3.x), use of the term thread refers to a Windows process.

The Windows Sockets specification provides a number of extensions to the standard set of Berkeley Sockets routines. Principally, these extended APIs allow message-based, asynchronous access to network events. In Figure 9, during initialization the Microsoft extentions to created a multi-threaded environment whereas it could handle new incoming connections while returning to the listener process

The process also listens for Windows Message number 500 for incoming data from other applications designed to work with the WA9XHN Scheduler. This data is repeated to all open socket connections.

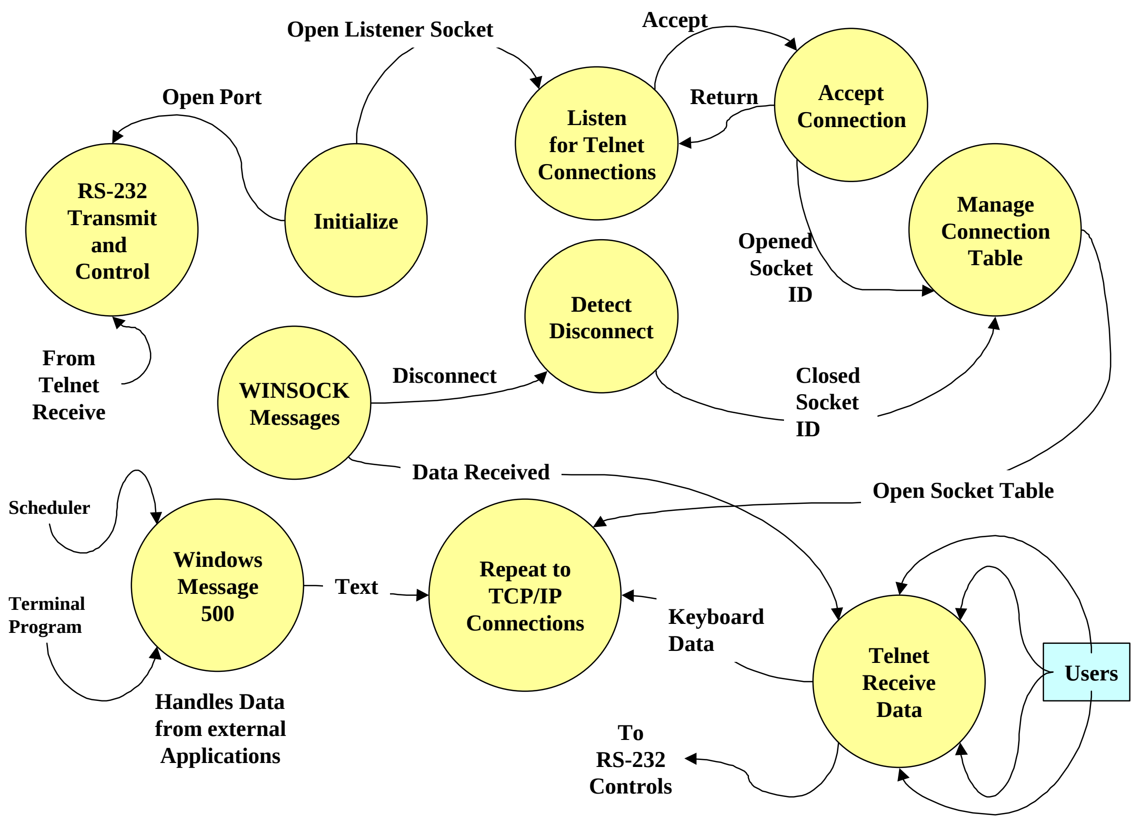


Figure 9: Data Flow Diagram for TCP/IP Telnet Server

RTTYApp Summary

This software development exercise was intended to demonstrate that a simple software application could be prototyped in a quick fashion in order to support an experiment. While the experiment's application may not been the most hi-tech one in recent history, it demonstrate a real-world problem that could be made easier by software.

The solution to the problem involved not only the management of the transmission of data, but the scheduling of the material. As requirements changed during the design and build phases, additional capabilities had to be incorporated to include multiple RS-232 ports for more than one transmitter. Monitoring of activities from workstations from remote locations via TCP/IP telnet sessions were added along with realtime changes to the schedule and a realtime terminal interface for keyboard access.

The original prototype was developed using Borland C++ and later ported over to Microsoft Visual C++. The software demonstrates an HMI interface using edit windows, list boxes, check boxes, radio controls and combo list boxes. Hardware interfaces include direct control of the 16450/16550 Universal Asyncronous Receive Transmit (UART) RS-232 and external control using I/O interfaces on these chips. Suppporting the BAUDOT protocol and Morse Code protocol were additional challenges by the user.

Bottom line, the product is in current use by the user in support of commercial broadcasts on the shortwave band.