This document describes the contents of the rig definition file “rig.xml”. A number of transceivers have rig defintion files written and tested which you may use. These are found in the xmls directory on the www.w1hkj.com web site. You will find subdirectories by manufacturer which contain files named by rig type, ie: TS-850.xml. If you create, test and verify the proper operation for a transceiver not yet posted please share that with others by sending it as an attachment to w1hkj@w1hkj.com and I will post it on the web site. You are encouraged to study the various rig definition files to learn more about how they are organized.

Comments are contained within the tag pair
<!--
-->
and may appear anywhere in the rig definition file

The entire rig definition must be contained within the tag pair
<RIGDEF>
</RIGDEF>

The text within the tag pair <RIG></RIG> specifies the transceiver to which this file applies, as in:
<RIG>Icom 746 PRO</RIG>

The text within the tag pair <PROGRAMMER></PROGRAMMER> is not used by the parser, but should as a minimum say who created and who tested the definition file, as in:
<PROGRAMMER>
Dave Freese W1HKJ
Tested by: W1HKJ, Dave
</PROGRAMMER>

The text within the tag pair <STATUS></STATUS> is not used by the parser, but should as a minimum state whether the definition file has been “Verifed”, is “Alpha”, what the Version and Date of creation or update, as in:
<STATUS>
Verified
Version: 1.0
Date: 2007 Jan 5
</STATUS>

The <TITLE></TITLE> tag pair contains the text which will be displayed on the window decoration bar, as in:
<TITLE>Rig Control - IC-746 PRO</TITLE>

The <PORT></PORT> tag pair contains multiple tag pair entries that specify the serial port and its configuration as used by fldig:

BAUD	normal baud rate that transceiver operates at for CAT
DEVICE	full path to the serial device
ECHO	set to true for CI-V type of interface in which a h/w loop back sends all outgoing data from Tx to Rx, or if the transceiver echos the data stream

RETRIES	# times that the computer tries to send a command sequence before failure
TIMEOUT	time in milliseconds between retries
WAIT	time to wait between successive transmissions
DTRINIT	inititialized state of the DTR pin, +/- 12 are valid entries. This is the voltage level that will be available at the DTR pin. For some CI-V type of interfaces this voltage may be used to power the serial converter
DTRPTT	logic true or false designating whether the DTR pin is used for PTT. PTT ON will toggle the DTRINIT state, ie: if +12 it goes to -12, if -12 it changes to +12.
RTSINIT	inititialized state of the RTS pin, same as for DTR
RTSPTT	specifies whether the RTS pin is used for PTT, same logic as with DTR.
RTSCTS	specifies whether hardware flow control is used with the radio. Many of the Kenwood rigs require RTS/CTS flow control. Set to true to enable this type of control
NOTE:	you cannot use both RTSCTS and RTSPTT. If you set both to true the RTSCTS flow control will have priority.

Example:

```
<PORT>
    <BAUD>19200</BAUD>
    <DEVICE>/dev/ttyS0</DEVICE>
    <ECHO>true</ECHO>
    <RETRIES>8</RETRIES>
    <TIMEOUT>20</TIMEOUT>
    <WAIT>0</WAIT>
    <DTRINIT>+12</DTRINIT>
    <DTRPTT>false</DTRPTT>
    <RTSINIT>-12</RTSINIT>
    <RTSPTT>false</RTSPTT>
    <RTSCTS>false</RTSCTS>
</PORT>
```

The transceiver modes are specified within the <MODES></MODES> tag pair. Each ntry or element associated with a mode has a symbol name (text) and a way of specifiy what the data transfer consists of. The data transfer might be a single byte, multiple bytes, or aa string

Example 1, for the Icom-746PRO

```
<MODES>
    <ELEMENT><SYMBOL>LSB</SYMBOL><BYTE>00</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>USB</SYMBOL><BYTE>01</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>AM</SYMBOL><BYTE>02</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>CW</SYMBOL><BYTE>03</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>RTTY</SYMBOL><BYTE>04</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>FM</SYMBOL><BYTE>05</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>CW-R</SYMBOL><BYTE>07</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>RTTY-R</SYMBOL><BYTE>08</BYTE></ELEMENT>
</MODES>
```

Example 2, for the Kenwood 850

```
<MODES>
```

```
	<ELEMENT><SYMBOL>LSB</SYMBOL><BYTE>31</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>USB</SYMBOL><BYTE>32</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>CW</SYMBOL><BYTE>33</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>FM</SYMBOL><BYTE>34</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>AM</SYMBOL><BYTE>35</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>FSK</SYMBOL><BYTE>36</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>CW-R</SYMBOL><BYTE>37</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>FSK-R</SYMBOL><BYTE>39</BYTE></ELEMENT>
</MODES>
```

Example 3, for the FT-100

```
<MODES>
	<ELEMENT><SYMBOL>LSB</SYMBOL><BYTE>00</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>USB</SYMBOL><BYTE>01</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>CW</SYMBOL><BYTE>02</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>CW-R</SYMBOL><BYTE>03</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>AM</SYMBOL><BYTE>04</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>DIG</SYMBOL><BYTE>05</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>FM</SYMBOL><BYTE>06</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>W-FM</SYMBOL><BYTE>07</BYTE></ELEMENT>
</MODES>
```

The modes which are supported by lower sideband in the transceiver are specified in the <LSBMODES></LSBMODES> tar pair. The string data for the lsb modes must match those given in the modes id specifier

For example in the Icom 746 Pro:

```
<LSBMODES>
	<STRING>LSB</STRING>
	<STRING>RTTY</STRING>
	<STRING>CW-R</STRING>
</LSBMODES>
```

If the transceiver data stream uses identically the same format for the bandwidth data then it is specified in the <BANDWIDTHS></BANDWIDTHS> tag pair

Example for the Icom 746 Pro:

```
<BANDWIDTHS>
	<ELEMENT><SYMBOL>50</SYMBOL><BYTE>00</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>100</SYMBOL><BYTE>01</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>150</SYMBOL><BYTE>02</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>200</SYMBOL><BYTE>03</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>250</SYMBOL><BYTE>04</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>300</SYMBOL><BYTE>05</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>350</SYMBOL><BYTE>06</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>400</SYMBOL><BYTE>07</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>450</SYMBOL><BYTE>08</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>500</SYMBOL><BYTE>09</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>600</SYMBOL><BYTE>10</BYTE></ELEMENT>
	<ELEMENT><SYMBOL>700</SYMBOL><BYTE>11</BYTE></ELEMENT>
```

```
    <ELEMENT><SYMBOL>800</SYMBOL><BYTE>12</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>900</SYMBOL><BYTE>13</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>1000</SYMBOL><BYTE>14</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>1100</SYMBOL><BYTE>15</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>1200</SYMBOL><BYTE>16</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>1300</SYMBOL><BYTE>17</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>1400</SYMBOL><BYTE>18</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>1500</SYMBOL><BYTE>19</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>1600</SYMBOL><BYTE>20</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>1700</SYMBOL><BYTE>21</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>1800</SYMBOL><BYTE>22</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>1900</SYMBOL><BYTE>23</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2000</SYMBOL><BYTE>24</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2100</SYMBOL><BYTE>25</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2200</SYMBOL><BYTE>26</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2300</SYMBOL><BYTE>27</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2400</SYMBOL><BYTE>28</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2500</SYMBOL><BYTE>29</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2600</SYMBOL><BYTE>30</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2700</SYMBOL><BYTE>31</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2800</SYMBOL><BYTE>32</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2900</SYMBOL><BYTE>33</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>3000</SYMBOL><BYTE>34</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>3100</SYMBOL><BYTE>35</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>3200</SYMBOL><BYTE>36</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>3300</SYMBOL><BYTE>37</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>3400</SYMBOL><BYTE>38</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>3500</SYMBOL><BYTE>39</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>3600</SYMBOL><BYTE>40</BYTE></ELEMENT>
</BANDWIDTHS>
```

If the bandwidth data stream is unique for send and receive data streams then they are specified separately with the <BW-CMD></BW-CMD> tag pair for data sent to the transceiver, and the <BW-REPLY></BW-REPLY> tag pair for data returned to the computer.

Example: FT-100:

```
<BW-CMD>
    <ELEMENT><SYMBOL>300</SYMBOL><BYTE>00</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>500</SYMBOL><BYTE>01</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2400</SYMBOL><BYTE>02</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>6000</SYMBOL><BYTE>03</BYTE></ELEMENT>
</BW-CMD>

<BW-REPLY>
    <ELEMENT><SYMBOL>300</SYMBOL><BYTE>03</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>500</SYMBOL><BYTE>02</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>2400</SYMBOL><BYTE>01</BYTE></ELEMENT>
    <ELEMENT><SYMBOL>6000</SYMBOL><BYTE>00</BYTE></ELEMENT>
</BW-REPLY>
```

Fldigi can parse and decode message returned from the transceiver that define 4 aspects of the transceiver operation:

OK – data accepted by the transceiver
BAD – data rejected by the transceiver
MODE – current operating mode of the transceiver
BW – current bandwidth setting of the transceiver
FREQ – frequency of the active vfo (might be either A or B for example).

These are all contained within multiple <REPLY></REPLY> tag pairs

This is an example of a fixed format message with no variable fields. It is the OK message sent back by the Icom-746 PRO:

```
<REPLY>
	<SYMBOL>OK</SYMBOL>
	<SIZE>6</SIZE>
	<BYTES>FE FE E0 66</BYTES>
	<BYTE>FB</BYTE>
	<BYTE>FD</BYTE>
</REPLY>
```

The <SYMBOL></SYMBOL> pair and the command defnition are mandatory
The <SIZE></SIZE> field is mandatory and specifies the number of bytes contained in this reply. The above definition could also have been coded as:

```
<REPLY>
	<SYMBOL>OK</SYMBOL>
	<SIZE>6</SIZE>
	<BYTES>FE FE E0 66 FB FD</BYTES>
</REPLY>
```

When the reply contains variable data it is specified in a contained tag pair <DATA></DATA>. This data field contains specifiers that describe the kind and size of the data. The <DTYPE></DTYPE> tag pair may be one of:

BINARY or
DECIMAL

This is an example for the reply to a mode query that is returned by the Icom-746 PRO:

```
<REPLY>
	<SYMBOL>MODE</SYMBOL>		specifies the response name
	<SIZE>8</SIZE>			8 bytes of data returned
	<BYTES>FE FE E0 66</BYTES>	4 bytes of preamble
	<BYTE>04</BYTE>			1 additional byte for preample
	<DATA>
		<DTYPE>BINARY</DTYPE>	binary data field of 1 byte
		<SIZE>1</SIZE>
	</DATA>
	<FILL>1</FILL>			a variable field (data) not used
```

<BYTE>FD</BYTE> 1 byte postamble
</REPLY>

Fldigi rigcat will check for both the preample and postamble to insure that a valid reply has been sent by the transceiver.

This is the reply definition for bandwidth sent by the FT100. The FT100 sends all responses back in a status message. The program must then extract the data from the correct data byte and bits contained with the status message.

```
<REPLY>
        <SYMBOL>BW</SYMBOL>                 specifies response name
        <SIZE>32</SIZE>                     number of bytes returned
        <FILL>5</FILL>                      ignore first 5 bytes
        <DATA>
              <DTYPE>BINARY</DTYPE>         binary value
              <SIZE>1</SIZE>                of 1 byte
              <SHIFT>4</SHIFT>              shift bits to right by 4
              <MASK>3</MASK>                mask with 0x03 for data value
        </DATA>
        <FILL>26</FILL>                     26 more bytes ignored
</REPLY>
```

Note that for this response there is no specified Preamble or Postamble and therefore no checks will be made for correct response other than the number of returned bytes.

The response to a request for frequency will be the most complicated as the transceiver may return the value as a string, a binary number, a binary coded decimal number, or a decimal number. It may further send the data LSB first or MSB first, which can be specified by the <REV></REV> boolean tag.

This is the response from the Icom-746PRO and for most of the Icom series of transceivers:

```
<REPLY>
       <SYMBOL>FREQ</SYMBOL>                specifies reponse type
       <SIZE>11</SIZE>                      11 bytes in response
       <BYTES>FE FE E0 66</BYTES>           4 byte fixed preamble
       <BYTE>03</BYTE>                      1 byte preamble specifies freq
       <DATA><DTYPE>BCD</DTYPE>             data is Binary Coded Decimal
             <SIZE> 9 </SIZE>               consisting of BCD digits (5 bytes)
             <MAX> 174000000 </MAX>         the max expected value
             <MIN> 30000 </MIN>             the min expected value
             <RESOL> 1 </RESOL>             the resolution of the smallest BCD
             <REV>true</REV>                sent in reverse order (smallest first)
        </DATA>
       <BYTE>FD</BYTE>                      1 byte postamble
</REPLY>
```

For the Yaesu FT-100 it is:

```
<!-- 4 byte binary value received in positions 1-4 (start with 0)
-->

<REPLY>
   <SYMBOL>FREQ</SYMBOL>                  specifies response
   <SIZE>32</SIZE>                        32 bytes long
   <FILL>1</FILL>                         ignore 1st byte
   <DATA>
      <DTYPE>BINARY</DTYPE>               data is binary
      <SIZE> 4 </SIZE>                    consisting of 4 bytes
      <MAX> 970000000 </MAX>              max expected value
      <MIN> 30000 </MIN>                  min expected value
      <RESOL> 1.25 </RESOL>               data resolution
      <REV>false</REV>                    data is most significant first
   </DATA>
   <FILL>27</FILL>                        27 trailing bytes ignored
</REPLY>
```

Some explanation is in order. The returned frequency is contained in the 2nd through the 5th bytes of the status message. It is a binary value with the most significant byte at position 2 and the least significant byte at position 5. The data has a resolution of 1.25 Hz per bit. Fldigi will scale the returned value by 1.25 (ie, multiply) for display and use within the program.

A third example that demonstrates the DECIMAL string response to a frequency request may be found for the Kenwood 850. The Kenwood data is also contained imbedded in a generic status response message. The format for the Kenwood status message is:

```
IFaaaaaaaaaaaXXXXXbbbbbcdXeefghjklmmX;
12345678901234567890123456789012345678
```

where:

- aaaaaaaaaaa => decimal value of vfo frequency
- bbbbb => rit/xit frequency
- c => rit off/on
- d => xit off/on
- e => memory channel
- f => tx/rx
- g => mode
- h => function
- j => scan off/on
- k => split off /on
- l => tone off /on
- m => tone number
- X => unused characters

All data in the Kenwood response if ASCII and can be intercepted by a program like MiniComm and visually interpreted.

```
<REPLY>
    <SYMBOL>FREQ</SYMBOL>              specifies reponse type
    <SIZE>38</SIZE>                    38 bytes in message
    <STRING>IF</STRING>                2 character preamble
    <DATA>
        <DTYPE>DECIMAL</DTYPE>         data is decimal
        <SIZE>11</SIZE>                11 bytes in length
        <MAX>99999999999</MAX>         with a max expected value
        <MIN>00001500000</MIN>         and a min expected value
        <RESOL>1</RESOL>               the resolution is 1 Hz
    </DATA>
    <FILL>24</FILL>                    24 bytes are ignored
    <STRING>;</STRING>                 1 character postamble
</REPLY>
```

Since the <REV></REV> tag is not specified the program will interpret the data string as the leading character is the most significant decimal value.

Commands sent from the program to the transceiver may be one of:

SETFREQ
GETFREQ
SETMODE
GETMODE
SETBW
GETBW
PTTON
PTTOFF

The SETMODE command for the Icom 746 PRO is:

```
<COMMAND>
    <SYMBOL>SETMODE</SYMBOL>          specifies the command type
    <SIZE>7</SIZE>                    7 bytes long
    <BYTES>FE FE 66 E0</BYTES>        4 fixed preamble bytes
    <BYTE>06</BYTE>                   1 preamble byte specifies mode
    <DATA>
        <DTYPE>BINARY</DTYPE>         1 binary data byte
        <SIZE>1</SIZE>                to be found in the <MODE>...
    </DATA>
    <BYTE>FD </BYTE>                  1 postamble byte
    <OK>OK</OK>                       expect an OK response or
    <BAD>BAD</BAD>                    a BAD response message returned
</COMMAND>
```

The same command for the FT100 is:

```
<COMMAND>
    <SYMBOL>SETMODE</SYMBOL>          specifies the command type
    <SIZE>5</SIZE>                    5 bytes long
    <BYTES>00 00 00</BYTES>           3 fixed bytes
    <DATA>
        <DTYPE>BINARY</DTYPE>         1 binary data byte
        <SIZE>1</SIZE>                to be found in <MODE> ...
    </DATA>
    <BYTE>0C</BYTE>                   1 byte specifying the mode command
</COMMAND>
```

The Kenwood 850 is simplier since it uses strings for all of its command parameters:

```
<COMMAND>
    <SYMBOL>SETMODE</SYMBOL>          specifies mode command
    <SIZE>4</SIZE>                    4 bytes long
    <STRING>MD</STRING>               MD mode command string
    <DATA>
        <DTYPE>BYTE</DTYPE>           1 data byte value given in <MODE>...
        <SIZE>1</SIZE>
    </DATA>
    <STRING>;</STRING>                semicolon terminator
</COMMAND>
```

The frequency command is complicated by the various schemes used by the manufacturers to specify the value of the frequency. All of the transceivers thus far tested have used BINARY, BCD or DECIMAL with some sequences requiring REV processing.

The command to the Icom 746 PRO to set the frequency (and in fact most of the Icom transceiver's) is nearly identical to it's response to a query for frequency. (Thank you Icom).

```
<COMMAND>
      <SYMBOL>SETFREQ</SYMBOL>          specifies set frequency command
      <SIZE>11</SIZE>                   11 bytes in length
      <BYTES>FE FE 66 E0</BYTES>        4 byte fixed preamble
      <BYTE>05</BYTE>                   1 byte in preamble for SETFREQ
      <DATA>
            <DTYPE>BCD</DTYPE>          Binary Coded Decimal value
            <SIZE> 9 </SIZE>            9 BCD digits (5 bytes) in length
            <MAX> 174000000 </MAX>      max expected value
            <MIN> 30000 </MIN>          min expected value
            <RESOL> 1 </RESOL>          1 Hz resolution
            <REV>true</REV>             in reverse order
      </DATA>
      <BYTE>FD </BYTE>                  1 byte postamble
      <OK>OK</OK>                       transceiver replies with OK or
      <BAD>BAD</BAD>                    BAD message
</COMMAND>
```

The FT-100 uses a different format for commanding a frequency than for the response. This is because all of the response's are contained within the status message. This is how the FT-100 frequency command is specified in the xml file.

```
<COMMAND>
      <SYMBOL>SETFREQ</SYMBOL>          specifies set frequency command
      <SIZE>5</SIZE>                    5 bytes long
      <DATA>
       <DTYPE>BCD</DTYPE>               data in Binary Coded Decimal
       <SIZE> 8 </SIZE>                 8 BCD digits in 4 bytes
       <MAX> 970000000 </MAX>           max value
       <MIN> 30000 </MIN>               min value
       <RESOL> 10 </RESOL>              least significant digit is 10 Hz
       <REV>true</REV>                  sent in reverse order
       </DATA>
      <BYTE>0A</BYTE>                   terminator byte for SETFREQ cmd
</COMMAND>
```

Note that the reply from the FT100 was also BCD, but with a resolution of 1.25 Hz. Go figure !

For most of the transceivers the PTTON and PTTOFF (if available) are the easiest to code into the xml format.

This is the Icom-746 Pro command string for PTTON

```
<COMMAND>
      <SYMBOL>PTTON</SYMBOL>          specifies push to talk command
      <SIZE>8</SIZE>                  8 bytes in length
      <BYTES>FE FE 66 E0</BYTES>      4 byte fixed postamble
      <BYTES>1C 00 01</BYTES>         3 bytes specifying PTT
      <BYTE>FD</BYTE>                 1 byte postamble
      <OK>OK</OK>                     rig can return either OK or
      <BAD>BAD</BAD>                  BAD message
</COMMAND>
```

This could also have been written as:

```
<COMMAND>
      <SYMBOL>PTTON</SYMBOL>          specifies push to talk ON command
      <SIZE>8</SIZE>                  8 bytes in length
      <BYTES>FE FE 66 E0 1C 00 01 FD</BYTES>
      <OK>OK</OK>                     rig can return either OK or
      <BAD>BAD</BAD>                  BAD message
</COMMAND>
```

The PTTON command for the FT100 is:

```
<COMMAND>
      <SYMBOL>PTTON</SYMBOL>
      <SIZE>5</SIZE>
      <BYTES>00 00 00 01 0F</BYTES>
</COMMAND>
```

and for the Kenwood 850:

```
<COMMAND>
      <SYMBOL>PTTON</SYMBOL>
      <SIZE>3</SIZE>
      <STRING>TX;</STRING>
</COMMAND>
```

If your transceiver does not support a CAT command for PTT then you can use a serial port control via RTS or DTR. You can also elect to use the serial port controls even if the transceiver supports the CAT PTT command. In that case simply delete the PTTON and PTTON xml specifier from your final rig.xml rig defintion file.