XML Gauge Programming for FS2004. Chapter 4. Radios

Version 1.0

By Nick Pike December, 2005

INDEX

XML is a text based programming language. Therefore, the code can be written in a standard text editor. There are applications available specifically written for XML, but I use a freeware text editor Notepad++. http://notepad-plus.sourceforge.net/uk/about.php If code is saved with a txt extension, rename with an xml extension. XML gauges usually consist of the xml file and bmp (bitmap) files, although gauges without bitmaps are quite common.

This tutorial provides a general introduction to radios and their functions.
Colours have been used to group relative information, or to allow the reader to easily find references in code or text.

1.0 Introduction and Radios section in the aircraft.cfg file

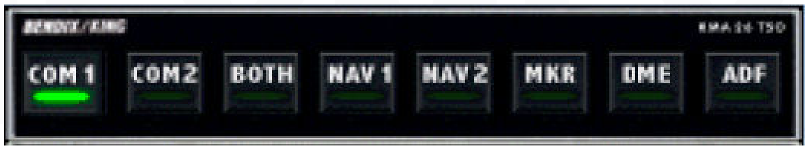
I’m not going to attempt to turn this tutorial into an instrument flying lesson as that’s a full subject all its own. This tutorial explains how to code gauges to produce the various types of radio equipment. So I’ll assume you know your NDB’s from your VOR’s, etc. A host of information is available on the internet for full explanations on radio equipment type and their use. A quick Google will reveal all.

Before you write any radio gauges, it is necessary to inform FS2004 what radio equipment is required and their functions. If this section is left out, FS assumes that all equipment is fitted and operational. If there are no radios in your (strange) aircraft, this section will be required with all values set to zero.
Look in the aircraft.cfg file for the section,

```
[Radios]
// Radio Type=available, standby frequency, has glide slope
Audio.1=1              <<Audio panel, buttons for Morse code ID, marker sounds, etc.
Com.1=1, 1             << Coms 1 radio
Com.2=1, 1             <<Coms 2 radio
Nav.1=1, 1, 1          <<Navigation 1 radio, ILS and the like
Nav.2=1, 1, 0          << Navigation 2 radio, ILS and the like
Adf.1=1, 1             << automatic direction finder (ADF1).
Adf.2=1                << automatic direction finder (ADF2).
Transponder.1=1, 1     <<Aircraft ID system
Marker.1=1             <<Airport boundary markers beacon, inner, middle and outer
```

This is a full list of radio equipment.The text after the ‘//’ gives a brief explanation. Each piece of equipment shown to the left of the ‘=’ sign has a ‘.1’ or ‘.2’ after type. There will only be one audio panel hence the ‘.1’. Com (communication) radios usually have two radios, hence ‘.1’ and ‘.2’, same with Nav radios, also ADF (although one is quite common) and one transponder and marker beacon. After the equals sign, the first number means 1=available and 0=unavailable, that is, is this equipment fitted. The second number also means 1=available and 0=unavailable for the radio to have the ability to set frequencies using a standby channel. The standby channel would be set to the required frequency and then switched to be the active frequency. FS2004 is very much geared to using this method for setting frequencies. However, it is possible to set active frequencies directly. If, for example, Com.1=1, 0 , the standby feature is disabled, and when setting radio frequencies, the active frequency is set directly. The Nav radios have a third number (has glide slope) and means if the radio can lock onto the ILS system for instrument landing. It is usual to only use Nav1 for this.
Note that Com2 can only exist so long as there is a Com1, and the same goes for Nav and Adf radios.

2.0 Audio Panel



This picture shows a default audio panel. These toggled push buttons switch various radio functions. Communication radios (transceivers), either COM1, COM2 or BOTH. Navigation Radios, either NAV1 or NAV2 and activates the navigation beacon Morse code identification.. MARKER airport beacon (Outer, Middle and Inner), activating a visual indication somewhere on the panel and the beacon sounds. DME (distance measuring equipment) activates the DME beacon Morse code identification. ADF (Automatic Direction Finder) activates the ADF beacon Morse code identification.

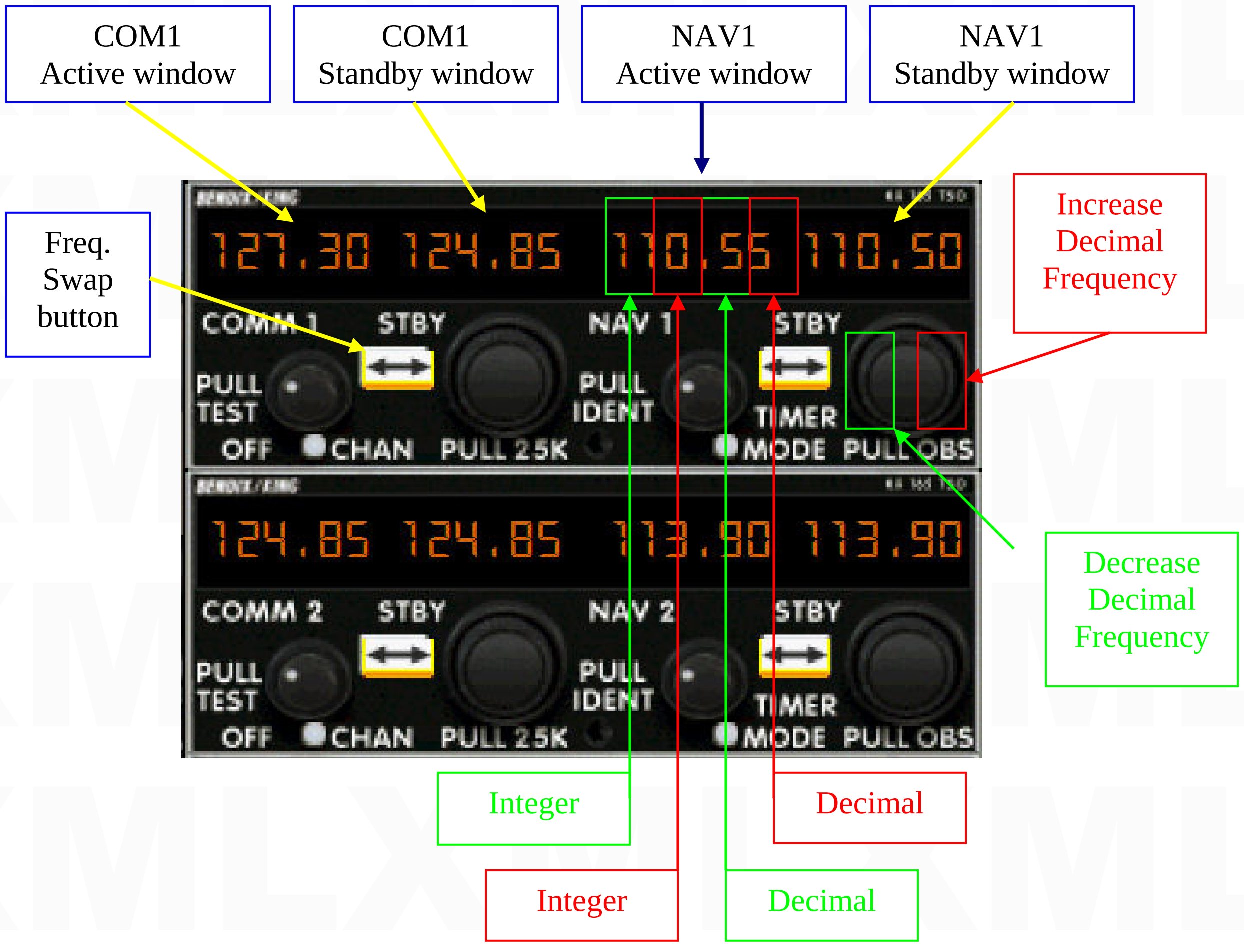
A complete code set for this equipment is as follows. Notes are written in RED text and are not part of the code. The buttons are shown to be active by adding an illuminated button overlaid on a general background bitmap.

```
<Gauge Name="Radio Audio Panel" Version="1.0">
<Image Name="Audio_Panel.bmp" Luminous="Yes"/>GENERAL BACKGROUND BITMAP

 <Element>
  <Visible>(A:Circuit general panel on, bool) (A:Circuit avionics on,bool) &amp;&amp;</Visible>
THIS LINE WILL ONLY ALLOW THE FOLLOWING ‘LIVE’ BUTTONS TO SHOW IF THE PANEL
ELECTRICAL CIRCUITS AND AVIONICS CIRCUITS ARE SWITCHED ON

     <Element>
     <Position X="6" Y="3"/>
      <Select>
         <Value>(A:COM1 TRANSMIT, bool)</Value>IS COM1 ACTIVE (BOOL=0 OR 1)
         <Case Value="1">
            <Image Name="com1_button.bmp" Bright="Yes"/>
         </Case>
      </Select>
   </Element>
   <Element>
     <Position X="36" Y="3"/>
      <Select>
         <Value>(A:COM2 TRANSMIT, bool)</Value>
         <Case Value="1">
            <Image Name="com2_button.bmp" Bright="Yes"/>
         </Case>
      </Select>
   </Element>
   <Element>
     <Position X="66" Y="3"/>
      <Select>
         <Value>(A:COM RECIEVE ALL, bool)</Value>ARE BOTH COM1 AND COM2 ACTIVE
         <Case Value="1">
            <Image Name="both_button.bmp" Bright="Yes"/>
```

```
        </Case>
      </Select>
    </Element>
    <Element>
      <Position X="96" Y="3"/>
      <Select>
        <Value>(A:NAV1 SOUND, bool)</Value>FOR NAV1 MORSE CODE ID SOUNDING
        <Case Value="1">
          <Image Name="nav1_button.bmp" Bright="Yes"/>
        </Case>
      </Select>
    </Element>
    <Element>
      <Position X="126" Y="3"/>
      <Select>
        <Value>(A:NAV2 SOUND, bool)</Value>
        <Case Value="1">
          <Image Name="nav2_button.bmp" Bright="Yes"/>
        </Case>
      </Select>
    </Element>
    <Element>
      <Position X="157" Y="3"/>
      <Select>
        <Value>(A:MARKER SOUND, bool)</Value>
        <Case Value="1">
          <Image Name="mkr_button.bmp" Bright="Yes"/>
        </Case>
      </Select>
    </Element>
    <Element>
      <Position X="186" Y="3"/>
      <Select>
        <Value>(A:DME SOUND, bool)</Value>
        <Case Value="1">
          <Image Name="dme_button.bmp" Bright="Yes"/>
        </Case>
      </Select>
    </Element>
    <Element>
      <Position X="217" Y="3"/>
      <Select>
        <Value>(A:ADF SOUND, bool)</Value>
        <Case Value="1">
          <Image Name="adf_button.bmp" Bright="Yes"/>
        </Case>
      </Select>
    </Element>
    </Element>
    <Mouse>
     <Tooltip>%Audio Control Panel (Transmit - %((A:COM1 TRANSMIT, BOOL))%{if}Com
  1%{else}%((A:COM2 TRANSMIT, BOOL))%{if}Com 2%{else}None%{end}%{end}, Receive -
  %((A:COM RECIEVE ALL, BOOL))%{if}BOTH%{else}%((A:COM1 TRANSMIT, BOOL))%{if}Com
  1%{else}%((A:COM2 TRANSMIT, BOOL))%{if}Com 2%{else}None%{end}%{end}%{end}, Marker
  - %((A:MARKER SOUND,bool))%{if}ON%{else}OFF%{end})</Tooltip>
  THIS TOOLTIP SHOWS WHAT SYSTEMS ARE ACTIVE
```

```
      <Area Left="0" Right="35" Top="0" Bottom="32">
          <Click Event="COM1_TRANSMIT_SELECT"/>TOGGLE COM1 ACTIVE
          <Cursor Type="Hand"/>
        </Area>
      <Area Left="36" Right="65" Top="0" Bottom="32">
          <Click Event="COM2_TRANSMIT_SELECT"/>
          <Cursor Type="Hand"/>
        </Area>
      <Area Left="66" Right="95" Top="0" Bottom="32">
          <Click Event="COM_RECEIVE_ALL_TOGGLE"/>
          <Cursor Type="Hand"/>
        </Area>
      <Area Left="96" Right="125" Top="0" Bottom="32">
          <Click Event="RADIO_VOR1_IDENT_TOGGLE"/>
          <Cursor Type="Hand"/>
        </Area>
      <Area Left="126" Right="155" Top="0" Bottom="32">
          <Click Event="RADIO_VOR2_IDENT_TOGGLE"/>
          <Cursor Type="Hand"/>
        </Area>
      <Area Left="156" Right="185" Top="0" Bottom="32">
          <Click Event="MARKER_SOUND_TOGGLE"/>
          <Cursor Type="Hand"/>
        </Area>
      <Area Left="186" Right="215" Top="0" Bottom="32">
          <Click Event="RADIO_DME1_IDENT_TOGGLE"/>
          <Cursor Type="Hand"/>
        </Area>
IT IS POSSIBLE TO HAVE A SECOND DME SYSTEM WHICH WOULD USE
          <Click Event="RADIO_DME2_IDENT_TOGGLE"/> AS AN ADDITIONAL CLICK AREA.
TO OPERATE THE 'LIVE' BUTTON, A TOGGLED VARIABLE SUCH AS (G:Var1) ! (>G:Var1)
WOULD BE ADDED TO THE CLICK EVENT AS THERE IS NO 'DME2 SOUND' VARIABLE


      <Area Left="216" Right="250" Top="0" Bottom="32">
          <Click Event="RADIO_ADF_IDENT_TOGGLE"/>
          <Cursor Type="Hand"/>
        </Area>
    </Mouse>
<copyright>For FS2004 (c)2003 Nick Pike</copyright>
</Gauge>
```

3.0 COM1, COM2, NAV1 and NAV2



This picture shows a default graphic. Because the standby feature is activated in the aircraft.cfg file (see section 2.0), when the frequency is changed, the STBY frequency is set. Clicking the STBY button reverses the frequency between the standby and active windows. The frequency is set by clicking on the knobs areas, one of which is shown in the above graphic. This is split into 2 click areas, one area for value increase and one for decrease. The decimal value increments up or down, and the whole (integer) values change at either end of the decimal range. The Com and Nav radios are covered together as the code to change the frequencies is similar to both.

A complete code set for this equipment is as follows. Notes are written in RED text and are not part of the code. Note, the code layout does not relate exactly to the above graphic, although the graphic is a good example of radio layout. To save space, the code is for COM1 and NAV1 only. For COM2 and NAV2 use COM2 ACTIVE FREQUENCY for example (but also see notes at the end of the code).

```
<Gauge Name="com1nav1 radio" Version="1.0">
  <Image Name="com1_nav1.bmp" Luminous="Yes"/>
GENERAL BACKGROUND IMAGE
    <Element>
     <Visible>(A:Circuit general panel on, bool) (A:Circuit avionics on,bool)
&amp;&amp;</Visible>NUMBERS WILL SHOW IF THE ELECTRICAL CIRCUITS
AND AVIONICS CIRCUITS ARE ON
         <Element>
           <Position X="4" Y="15"/>
           <Text X="58" Y="15" Bright="Yes" Length="6" Font="Arial" Color="White"
Adjust="Center" VerticalAdjust="Center" Multiline="No" Fixed="No">
             <String>%((A:COM1 ACTIVE FREQUENCY, Megahertz))%!6.2f!</String>
           </Text>
         </Element>
         <Element>
           <Position X="64" Y="15"/>
           <Text X="58" Y="15" Bright="Yes" Length="6" Font="Arial" Color="White"
Adjust="Center" VerticalAdjust="Center" Multiline="No" Fixed="No">
             <String>%((A:COM1 STANDBY FREQUENCY,
Megahertz))%!6.2f!</String>
           </Text>
         </Element>
         <Element>
           <Position X="126" Y="15"/>
           <Text X="58" Y="15" Bright="Yes" Length="6" Font="Arial" Color="White"
Adjust="Center" VerticalAdjust="Center" Multiline="No" Fixed="No">
             <String>%((A:NAV1 ACTIVE FREQUENCY, Megahertz))%!6.2f!</String>
           </Text>
         </Element>
         <Element>
           <Position X="186" Y="15"/>
           <Text X="58" Y="15" Bright="Yes" Length="6" Font="Arial" Color="White"
Adjust="Center" VerticalAdjust="Center" Multiline="No" Fixed="No">
             <String>%((A:NAV1 STANDBY FREQUENCY,
Megahertz))%!6.2f!</String>
           </Text>
         </Element>
    </Element>
  <Mouse>
     <Area Left="0" Right="125">
     <Tooltip ID="TOOLTIPTEXT_COMM1_FREQ_ACTIVE_STANDBY"/>
       </Area>
     <Area Left="125" Right="250">
     <Tooltip ID="TOOLTIPTEXT_NAV1_FREQ_ACTIVE_STANDBY"/>
       </Area>
TOOLTIPS WILL SHOW THE FREQUENCY VALUES IN THE TOOLTIP
```

```
     <Area Left="52" Right="72" Top="36" Bottom="50">
          <Click Event="COM_STBY_RADIO_SWAP"/>SWAP BUTTON
          <Cursor Type="Hand"/>
        </Area>
     <Area Left="72" Right="88" Top="36" Bottom="50">
           <Click Event="COM_RADIO_FRACT_INC" Repeat="Yes"/>THE 'FRACT'
INSTRUCTION CHANGES THE FRACTIONAL PART OF THE FREQUENCY,
THAT IS, THE DECIMAL PART
           <Cursor Type="UpArrow"/>
         </Area>
      <Area Left="88" Right="103" Top="36" Bottom="50">
           <Click Event="COM_RADIO_ FRACT _DEC" Repeat="Yes"/>
           <Cursor Type="DownArrow"/>
         </Area>
           <Area Left="175" Right="196" Top="36" Bottom="50">
          <Click Event="NAV1_RADIO_SWAP"/>
          <Cursor Type="Hand"/>
        </Area>
     <Area Left="196" Right="211" Top="36" Bottom="50">
           <Click Event="NAV1_RADIO_ FRACT _INC" Repeat="Yes"/>
           <Cursor Type="UpArrow"/>
         </Area>
      <Area Left="211" Right="226" Top="36" Bottom="50">
           <Click Event="NAV1_RADIO_ FRACT _DEC" Repeat="Yes"/>
           <Cursor Type="DownArrow"/>
         </Area>
      </Mouse>
</Gauge>
```

Notes. Navigation 1 and 2 have 1 and 2 suffixes, i.e., NAV1 and NAV2. Communication 1 and 2 have COM and COM2. The swap instructions are COM_STBY_RADIO_SWAP and NAV1_RADIO_SWAP.
To speed thing up, the integer values can be incremented separately to the decimal values. This means introducing a second pair of clickable areas. So there would now be 2 areas per increase or decrease in frequency, one for integers and one for decimals.
For example,

```
     <Area Left="72" Right="88" Top="36" Bottom="50">
           <Click Event="COM_RADIO_WHOLE_INC" Repeat="Yes"/>
           <Cursor Type="UpArrow"/>
         </Area>
      <Area Left="88" Right="103" Top="36" Bottom="50">
           <Click Event="COM_RADIO_WHOLE_DEC" Repeat="Yes"/>
           <Cursor Type="DownArrow"/>
         </Area>
```

Just to confuse things a little more, on the radio that the pictures have been taken from, the integers and decimals can be changed by clicking on the frequency widows, but again, this can be achieved by having the clickable paired areas. In this case, the areas lie on a horizontal line.
All the click (Key) event variables can be found in 1) Key Events (inputs)

4.0 ADF (Automatic Direction Finder)

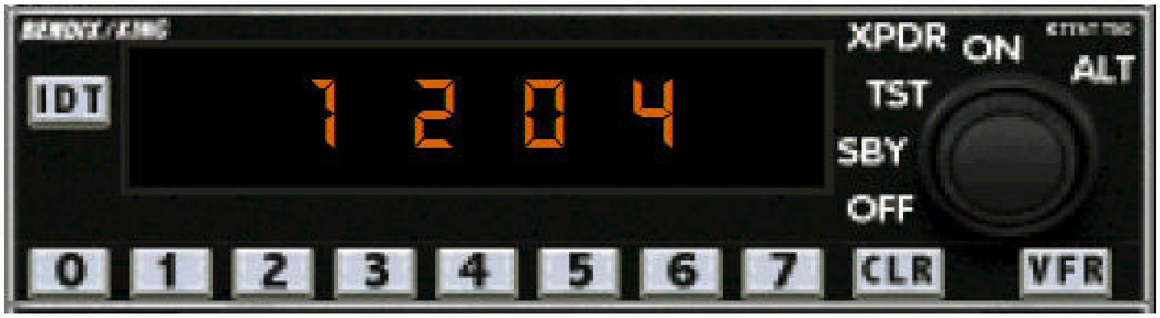
The frequency is changed in the same manner as the nav and com radios in the previous section. The only difference with this particular example is that each value can be changed by clicking on the upper part of the digit to increase and lower part to decrease. A complete code set for this equipment is as follows. Notes are written in RED text and are not part of the code. Note, the code layout does not relate exactly to the above graphic, although the graphic is a good example of radio layout. This code shows the click areas for changing each number. The knob areas as with previous examples would only change the decimal values, and it would take an inordinate amount of time to reach the desired frequency.
Note that the 1000 values do not have a separate click area. The 100 values are changed, and the 1000 value changes automatically (being either 0 or 1).

```
<Gauge Name=ADF Radio" Version="1.0">
  <Image Name="ADF_Radio_Background.bmp"/>
    <Element>
      <Visible>(A:Circuit general panel on, bool) (A:Circuit avionics on,bool)
&amp;&amp;</Visible>NUMBERS WILL SHOW IF THE ELECTRICAL CIRCUITS
AND AVIONICS CIRCUITS ARE ON
  <Element>
      <Position X="75" Y="8"/>
      <Text X="120" Y="20" Length="6" Fixed="No" Font="Arial"  Color="#FF0000"
Luminous="Yes" Adjust="Center" VerticalAdjust="Center">
         <String>%((A:Adf1 active frequency, KHz))%!05.1f!</String>
      </Text>
  </Element>
 </Element>
  <Mouse>
```

```
      <Area Left="75" Width="120">SHOWS FREQUENCY OVER DIGITS
        <Tooltip ID="TOOLTIPTEXT_ADF1_FREQ"/>
         </Area>
      <Area Left="195" Width="120">SHOWS BEARING OVER RIGHT GAUGE
AREA
         <Tooltip ID=" TOOLTIPTEXT_ADF1_BEARING_TO_STATION"/>
         </Area>
       <Area Left="100" Top="8" Width="20" Height="10" ">ACTIVE AREA TOP OF
DIGIT
            <Cursor Type="UpArrow"/>
            <Click Repeat="Yes" Event="ADF_100_INC"/>
         </Area>
          <Area Left="100" Top="18" Width="20" Height="10" "> ACTIVE AREA
BOTTOM OF DIGIT
            <Cursor Type="DownArrow"/>
            <Click Repeat="Yes" Event="ADF_100_DEC"/>
         </Area>
       <Area Left="120" Top="8" Width="20" Height="10" ">
            <Cursor Type="UpArrow"/>
            <Click Repeat="Yes" Event="ADF_10_INC"/>
         </Area>
        <Area Left="120" Top="18" Width="20" Height="10" ">
            <Cursor Type="DownArrow"/>
            <Click Repeat="Yes" Event="ADF_10_DEC"/>
         </Area>
       <Area Left="140" Top="8" Width="20" Height="10" ">
            <Cursor Type="UpArrow"/>
            <Click Repeat="Yes" Event="ADF_1_INC"/>
         </Area>
       <Area Left="140" Top="18" Width="20" Height="10" ">
            <Cursor Type="DownArrow"/>
            <Click Repeat="Yes" Event="ADF_1_DEC"/>
         </Area>
       <Area Left="170" Top="8" Width="20" Height="10" ">
            <Cursor Type="UpArrow"/>
            <Click Repeat="Yes" Event="ADF_FRACT_INC_CARRY"/>
         </Area>
       <Area Left="170" Top="18" Width="20" Height="10" ">
            <Cursor Type="DownArrow"/>
            <Click Repeat="Yes" Event="ADF_FRACT_DEC_CARRY"/>
         </Area>
   </Mouse>
</Gauge>
```

The code can be added for ADF2 radios. See: 1) Key Events (inputs) for variables that may be subtly different to those for ADF. For ADF2 decimal values, you need to use ADF2_RADIO_TENTHS_INC and ADF2_RADIO_TENTHS_DEC (FS bug?).
The use of DEC_CARRY causes the units digit to change automatically when changing the decimal value. Incidentally, some ADF radios do not have the decimal value.
There seems to be a lot of variables in the Key Events list that are not required for anything, but many of them can be used for specialized radio designs.

5.0 XPDR (Transponder)

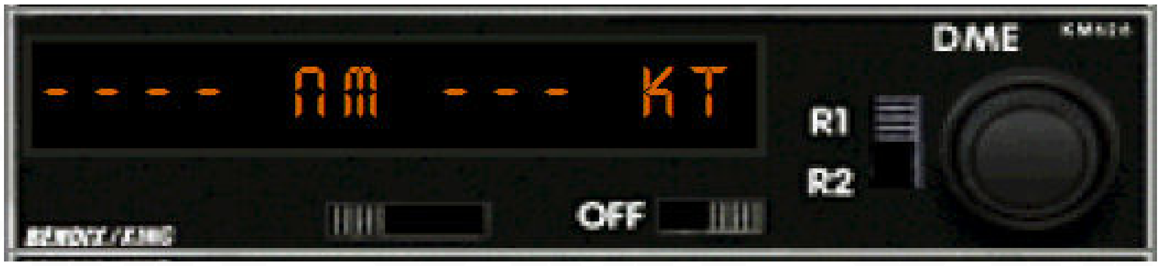


This radio transmits an identification signal. ATC ask it be set to a certain frequency, and then the aircraft can be identified on radar.
The frequency is changed with this particular example by clicking on the upper part of the digit to increase and lower part to decrease.
A complete code set for this equipment is as follows. The knob areas typically would not be utilised as there are no decimal values. Notes are written in RED text and are not part of the code. Note, the code layout does not relate exactly to the above graphic, although the graphic is a good example of radio layout. This code shows the click areas for changing each number.

```
<Gauge Name=XPDR Radio" Version="1.0">
   <Image Name="XPDR_Radio_Background.bmp"/>
     <Element>
      <Visible>(A:Circuit general panel on, bool) (A:Circuit avionics on,bool)
&amp;&amp;</Visible>NUMBERS WILL SHOW IF THE ELECTRICAL CIRCUITS
AND AVIONICS CIRCUITS ARE ON
  <Element>
      <Position X="60" Y="20"/>
      <Text X="80" Y="20" Bright="Yes" Length="4" Font="Quartz" FontWeight="100"
Color="Red" Adjust="Center" VerticalAdjust="Center" Multiline="No" Fixed="No">
         <String>%((A:TRANSPONDER1 CODE, number))%!4d!</String>
      </Text>
   </Element>
</Element>
```

```
    <Mouse>

      <Area Left="0" Top="0" Width="150" Height="60">
        <Tooltip ID="TOOLTIPTEXT_TRANSPONDER_ID"/>
      </Area>

      <Area Left="60" Top="20" Width="20" Height="10">
        <Cursor Type="UpArrow"/>
        <Click Event="XPNDR_1000_INC" Repeat="Yes"/>
      </Area>
      <Area Left="60" Top="30" Width="20" Height="10">
        <Cursor Type="DownArrow"/>
        <Click Event="XPNDR_1000_DEC" Repeat="Yes"/>
      </Area>
      <Area Left="80" Top="20" Width="20" Height="10">
        <Cursor Type="UpArrow"/>
        <Click Event="XPNDR_100_INC" Repeat="Yes"/>
      </Area>
      <Area Left="80" Top="30" Width="20" Height="10">
        <Cursor Type="DownArrow"/>
        <Click Event="XPNDR_100_DEC" Repeat="Yes"/>
      </Area>
      <Area Left="100" Top="20" Width="20" Height="10">
        <Cursor Type="UpArrow"/>
        <Click Event="XPNDR_10_INC" Repeat="Yes"/>
      </Area>
      <Area Left="100" Top="30" Width="20" Height="10">
        <Cursor Type="DownArrow"/>
        <Click Event="XPNDR_10_DEC" Repeat="Yes"/>
      </Area>
      <Area Left="120" Top="20" Width="20" Height="10">
        <Cursor Type="UpArrow"/>
        <Click Event="XPNDR_1_INC" Repeat="Yes"/>
      </Area>
      <Area Left="120" Top="30" Width="20" Height="10">
        <Cursor Type="DownArrow"/>
        <Click Event="XPNDR_1_DEC" Repeat="Yes"/>
      </Area>
    </Mouse>
  </Gauge>
```

6.0 DME (Distance Measuring Equipment)



When the navigation radio is tuned, the airport beacon will relay the distance to the apron and the aircraft's speed. The only clickable area on this example is the R1/ R2 switch (NAV1/ NAV2). The hyphens will show values when tuned.

```
<Gauge Name=DME Radio" Version="1.0">
  <Image Name="DME_Radio_Background.bmp"/>
    <Element>
     <Visible>(A:Circuit general panel on, bool) (A:Circuit avionics on,bool)
&amp;&amp;</Visible>NUMBERS WILL SHOW IF THE ELECTRICAL CIRCUITS
AND AVIONICS CIRCUITS ARE ON
  <Element>
     <Position X="60" Y="20"/>

     <Text X="130" Y="20" Length="12" Fixed="Yes" Font="Quartz" Adjust="Center"
VerticalAdjust="Center" Color="White" Bright="Yes">

           <String>%((G:Var1))%{if}%((A:NAV2 DME, nmiles) s0 (A:NAV2
DMESPEED, knots) s1)%{else}%((A:NAV1 DME, nmiles) s0 (A:NAV1 DMESPEED,
knots) s1)%{end}%(l0 0 &gt;=)%{if}%(l0 9999 min d 99
&lt;=)%{if}%!04.1f!%{else}%!4d!%{end} %(l1 9999 min)%!04d! %(l0 l1 / 60 * 99
min)%!02d! min%{else}---- ---- --%{end})"</String>
```

If you don't wish to follow the code explanation, then skip the next part in coloured text and just copy the code over. It gives a nice output for a radio using 2 DME beacons, with dashes shown if the radio is not tuned in or the signal is not yet being received.

This line of code produces distance and speed, but these values can also be used to show the time to landing (distance /speed), and the time value in minutes is added to this code with
(l0 l1 / 60 * 99 min)%!02d! min, so the graphic would have to modified to accommodate this.
The (G:Var1) switches between NAV1 and NAV2. If switched, (G:Var1) equals 1, then, (A:NAV2 DME, nmiles) s0 (A:NAV2 DMESPEED, knots) s1) is active and the two NAV2 values (distance and speed) are stored in s0 and s1 to be used later in the line of code.

If (G:Var1) is not switched and equals zero, the {else} condition applies and then, (A:NAV1 DME, nmiles) s0 (A:NAV1 DMESPEED, knots) s1) is active and stores NAV1 values in s0 and s1.
When an ‘if’ condition is applied as {if}, then that section must end in {end}.
(l0 0 &gt;=)%{if}%(l0 9999 min d 99 &lt;=) %{if}%!04.1f!% {else}%!4d!% checks if NAV2 DME or NAV1 DME is greater than or equal to zero. If so the DME is checked to a maximum of 9999, the d ensures the value is placed on the top of the stack, and then compared to 99. If less than or equal to 99, the value is printed with !04.1f!. If greater than 99, the value is printed with !4d!. This is formatting according to the accuracy required. Above 99, no decimal places are required, but below this, a decimal is introduced as the distance becomes more critical.
%(l1 9999 min)%!04d! This looks at the speed, checks to a maximum of 9999 and prints without a decimal.
%(l0 l1 / 60 * 99 min)%!02d! min% takes the distance and divides by the speed to give the time which is nmiles/knots which mathematically is nmiles x hour/nmile. The nmiles cancel out and a time of hours remains. The hours are multiplied by 60 to convert to minutes and then printed with !02d!.
{else}---- ---- -- The ‘if’ for this starts back at (l0 0 &gt;=)%{if}, so if nmiles is less than zero (probably no signal), the dashes are displayed.
%{if}%!04.1f!%{else}%!4d!%{end} is a nested ‘if’ statement.
I hope you’re following this as there will be questions later.

The rest of the code is as follows, but I’ll repeat the <String> code line for ease of reading and possible copying,

```
        <String>%((G:Var1))%{if}%((A:NAV2 DME, nmiles) s0 (A:NAV2
DMESPEED, knots) s1)%{else}%((A:NAV1 DME, nmiles) s0 (A:NAV1 DMESPEED,
knots) s1)%{end}%(l0 0 &gt;=)%{if}%(l0 9999 min d 99
&lt;=)%{if}%!04.1f!%{else}%!4d!%{end} %(l1 9999 min)%!04d! %(l0 l1 / 60 * 99
min)%!02d! min%{else}---- ---- --%{end})"</String>
    </Text>
  </Element>
</Element>
  <Mouse>
    <Tooltip ID="TOOLTIPTEXT_DME_NAV_NM_KT_MIN"
MetricID="TOOLTIPTEXT_DME_NAV_KM_MS_MIN"/>
    <Area Left="140" Top="0" Width="110" Height="45">
      <Tooltip ID="TOOLTIPTEXT_DME_N1_N2_SELECT"/>
      <Cursor Type="Hand"/>
      <Click>(G:Var1) ! (&gt;G:Var1)</Click>
    </Area>
  </Mouse>
<copyright>(c)2003 Nick Pike</copyright>
</Gauge>
```

7.0 Using the SET feature

Using the SET feature, for example, (&gt;K:NAV1_RADIO_SET).
Example: There are radios in the real world where a set of frequencies can be entered and stored. If there are a number of waypoints in a flight plan, the frequencies can be entered and stored, and recalled when required. This means all the hard work can be conducted before the flight, and take the load off the pilot whose primary job is to actually fly the aircraft.
Note that whilst making gauges, the following feature may crop-up in other applications. Incidentally, the following method is only suitable for NAV and COM frequencies. It can also be used for ADF but read the relevant section below.

A frequency can be stored in an L:Variable, say,
(A:NAV1 active frequency, MHz) (&gt;L:NAV1store1,MHz) then set the radio to another frequency and press a second store button and store the frequency thus,
(A:NAV1 active frequency, MHz) (&gt;L:NAV1store2,MHz) and so on.

An associated recall button can be pressed to set the active frequency to say (L:NAV1store2,MHz) using (L:NAV1store2,MHz) (&gt;K:NAV1_RADIO_SET) right?
WRONG!
A frequency can be stored in an L:Variable, but the L:Variable stores it actually as a number with no units. Where I have shown (L:NAV1store2,MHz), this could have been (L:NAV1store2,number), but it is handy to use the same units as an aide memoir.
So basically, (A:NAV1 active frequency, MHz) (&gt;L:NAV1store1,MHz) saves as a number.
(L:NAV1store2,MHz) (&gt;K:NAV1_RADIO_SET) will not set the radio with a decimal number. The number has to be conditioned before it can be used to set a radio frequency. It's all to do with decimal to binary conversions, but there's no need to get bogged down in the complex mathematics. Basically, the following line of code does this conversion and is known as the Horner scheme (this all originated from Arne Bartels).

```
(L:memorizedfrequencyindecimal,Megahertz) 100 * near 10000 % d 10 % r 10 / int d 10
% r 10 / int d 10 % r 10 / int 16 * + 16 * + 16 * + (&gt;K:NAV1_RADIO_SET)
```

This can be used in <Value>, <String> or <Click> code lines.
The basic layout of this procedure for the mathematically minded is,

```
(frequency value)
100 * near 10000 %
d 10 % r 10 / int
d 10 % r 10 / int
d 10 % r 10 / int
16 * +
16 * +
16 * +
(&gt;...FREQUENCY_SET)
```

ADF. This procedure can also be used with ADF radios but there is a complication because of the low and high ranges. In this case the code is,

```
(L: memorizedfrequencyindecimal,Kilohertz) s0 1000 &lt; if{ l0 10 * near 10000 % d 10
% r 10 / int d 10 % r 10 / int d 10 % r 10 / int 16 * + 16 * + 16 * +
(&gt;K:ADF_LOWRANGE_SET) } els{ l0 10 * near 10000 % d 10 % r 10 / int d 10 % r
10 / int d 10 % r 10 / int 16 * + 16 * + 16 * + (&gt;K:ADF_HIGHRANGE_SET) }
```

If you look at the two Horner schemes, they are the same code, but it’s all to do with the starting value and whether it’s less than 1000 or whether it’s equal to or greater than 1000. This can actually be ‘simplified’ to the following,

```
(L: memorizedfrequencyindecimal,Kilohertz) d 10 * 10000 % int d 10 % r 10 / int d 10 %
r 10 / int d 10 % r 10 / int 16 * + 16 * + 16 * + r 1000 &lt; if{
(&gt;K:ADF_LOWRANGE_SET) } els{ (&gt;K:ADF_HIGHRANGE_SET) }
```

Transponder. Again, if this value is memorized, or each digit were to be set with its own click spot, so the resulting value is a decimal, this too requires conversion before setting back into the transponder. This uses a different type on conversion as follows.

```
(L: memorizednumberindecimal,number) s0 10 % l0 10 / int 10 % 16 * + l0 100 / int 10
% 256 * + l0 1000 / int 4096 * + (&gt;K:XPNDR_SET)
```

In effect, the decimal value is split up into its individual digits, and each one is dealt with in turn, hence the use of s0 and l0.

8.0 Final discussion

Well, that’s all for now. There is more to radios than covered in this tutorial, but for the beginner, the above should give a good grounding and enable the construction of working gauges. I have also learned a great deal by regularly visiting the forums at Avsim.com, and I suggest you do the same.