```
// This sketch will decode ICOM CI-V frequency information
// and switch antenna's according to preset (memorized)
// values. RX and TX antenna's can be different.

// Press and hold TXsim button and select antenna with left or right button.
// Press and hold left and right button to reset EEPROM

// Instead of antennas, bandfilters band be selected


//////////////////////////////////////////////////////////
//                                                      //
// Firmware by ON7EQ November 2011                      //
//                                                      //
// modified for i2c LCD display, 6 antenna switching    //
// and ICOM IC-7300 for 70MHz  (4m-Band)                //
// by Ronny Mang DM2RM in January 2016                  //
//                                                      //
// With thx to Jeff Smith VE1ZAC                        //
// from which some code was inspired.                   //
//                                                      //
//////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////
//                    IMPORTANT                            //
//                                                         //
```

```
//    PLEASE COMPILE WITH ARDUINO VERION 0022 ! ! ! !        //
//                                                           //
///////////////////////////////////////////////////////////////



// include EEPROM write - required to memorize antenna / band config.
#include <EEPROM.h>

// Serial NewSoftSerial is required, as standard Serial is conflicting with LCD
#include<NewSoftSerial.h>

// Wire is required for i2c LCD Display
#include <Wire.h>

// Use pins 2 and 3 to talk to the CAT. 2 is the RX pin, 0 is the TX pin
// this is a dummy pin, as no TX is performed
// Connect the RX pin to the CAT output through a 4k7 resistor.

// IMPORTANT : select on the rig in the menu
//       CAT 'TRANSCEIVE' (ON)
// to implement broadcast of displayed frequency!

NewSoftSerial mySerial =  NewSoftSerial(2, 0);
```

```
// Pin Variables
#define  A1Pin      (12)   // Pin for  Antenna 1
#define  A2Pin      (11)   // Pin for  Antenna 2
#define  A3Pin      (10)   // Pin for  Antenna 3
#define  A4Pin      (9)    // Pin for  Antenna 4
#define  A5Pin      (8)    // Pin for  Antenna 5
#define  A6Pin      (7)    // Pin for  Antenna 6

#define  LeftPin    (A0)    // Pin for 'left scroll select'  - +5v with 4k7 pullup, GND when depress
#define  RightPin   (A1)    // Pin for 'right scroll select' - idem
#define  TXsimPin   (A2)    // Pin for 'TX simulator'        - idem
#define  PTTPin     (A3)    // PTT input pin. 0 = TX, 1 = RX

#define  TonePin    (1)    // Pin for  Beeper out




// Define band edges, some out of band allowed for antenna SWR curves

#define  High4       (72000)
#define  Low4        (69000)
#define  High6       (53999)
#define  Low6        (49000)
#define  High10      (30000)
#define  Low10       (27000)
```

```
#define  High12     (26999)
#define  Low12      (24000)
#define  High15     (23999)
#define  Low15      (20000)
#define  High17     (19999)
#define  Low17      (17000)
#define  High20     (16999)
#define  Low20      (13000)
#define  High30     (12999)
#define  Low30      (10000)
#define  High40     (9999)
#define  Low40      (6000)
#define  High80     (5999)
#define  Low80      (2500)
#define  High160    (2499)
#define  Low160     (100)

//=========================================================

// LiquidCrystal_I2C is required for i2c LCD Display
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16,2);

//=========================================================
```

```
int buffget[10] ;                              // the receive buffer

int unsigned long MHZ = 0;
int unsigned long KHZ = 0;
int unsigned long HZ = 0;
int unsigned long QRG = 0;
int unsigned long QRGcomp = 0;

int unsigned memotime = (1000);                // delay (ms) before selected antenna is memorized. M

byte BAND =(0);                                 // the actual band we are on
byte oldBAND = (0);

byte RXantenna = (1);                          // the default RX antenna for the band
byte TXantenna = (1);                          // the default RX antenna for the band

byte TXstatus = (0);                           // 0 = RX mode, 1 = TX mode
byte oldTXstatus = (0);                        // 0 = RX mode, 1 = TX mode

byte buttonpressed = (0);                      // 0 = no button pressed
byte simbuttonpressed = (0);

// LCD specific characters 'C' for CAT activity indication

byte cat [8] = {
  B00010,
```

```
  B11111,
  B00010,
  B00000,
  B00000,
  B01000,
  B11111,
  B01000,
};

// Antenna indicator Style 1

byte memory [8] = {
  B00000,
  B00000,
  B11011,
  B10101,
  B10001,
  B00000,
  B00000,
};

byte ant1 [8] = {
  B11111,
  B11011,
  B10011,
  B11011,
```

```
  B11011,
  B11011,
  B11011,
  B10001,

};

byte ant2 [8] = {
  B10001,
  B01110,
  B11110,
  B11101,
  B11011,
  B10111,
  B00000,
  B11111,
};

byte ant3 [8] = {
  B11111,
  B00000,
  B11101,
  B11011,
  B11101,
  B11110,
  B01110,
```

```
  B10001,
};

byte ant4 [8] = {
  B11101,
  B11001,
  B10101,
  B01101,
  B00000,
  B11101,
  B11101,
  B11111,
};

byte ant5 [8] = {
  B11111,
  B00000,
  B01111,
  B00001,
  B11110,
  B11110,
  B01110,
  B10001,
};

byte ant6 [8] = {
```

```
  B11001,
  B10111,
  B01111,
  B00001,
  B01110,
  B01110,
  B10001,
  B11111,
};


// some control variables

int i;

int incoming;
int unsigned long DisplayTime = 0;                    // timer display refresh
int unsigned long ButtonTime = 0;                     // Timer button depressed

byte memorize = (0);                                  // if = 1, them position to be memorzied

////////////// This loop run once on start up  ////////////////

void setup() {
```

```
  //set up i2c LCD display with backlight

  lcd.init();                                          // LCD initializing
  lcd.backlight();                                     // LCD backlight


  //set up output pins antenna control & beeper

  pinMode(A1Pin, OUTPUT);
  pinMode(A2Pin, OUTPUT);
  pinMode(A3Pin, OUTPUT);
  pinMode(A4Pin, OUTPUT);
  pinMode(A5Pin, OUTPUT);
  pinMode(A6Pin, OUTPUT);
  pinMode(TonePin, OUTPUT);

  digitalWrite(A1Pin,0) ;
  digitalWrite(A2Pin,0) ;
  digitalWrite(A3Pin,0) ;
  digitalWrite(A4Pin,0) ;
  digitalWrite(A5Pin,0) ;
  digitalWrite(A6Pin,0) ;
  digitalWrite(TonePin,0) ;

// create LCD characters for antenna, cat and memory
  lcd.createChar(0, cat);
```

```
lcd.createChar(1, memory);
lcd.createChar(2, ant1);
lcd.createChar(3, ant2);
lcd.createChar(4, ant3);
lcd.createChar(5, ant4);
lcd.createChar(6, ant5);
lcd.createChar(7, ant6);

// set up the LCD's number of columns and rows:
lcd.begin(16, 2);
lcd.clear();

// Print a message to the LCD.
lcd.setCursor(0, 0);
lcd.print(" ICOM CI-V CAT");
lcd.setCursor(0, 1);
lcd.print(" Antenna Switch");
delay (3000);
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(" Firmware V1.27");
lcd.setCursor(0, 1);
lcd.print(" by DM2RM 02/16");
delay (2000);
lcd.clear();
lcd.print("<select antenna>");
```

```
 lcd.setCursor(0, 1);
 lcd.print(" ...and good DX");
 delay (3000);
 lcd.clear();

 // RESET if SCROLL left & right buttons depressed at startup !

   if ((analogRead(LeftPin)<512) and (analogRead(RightPin)<512)) Reset ();

// Startup Beep

  tone(TonePin,1200);
   delay(150);
   tone(TonePin,1600);
   delay (150);
   tone(TonePin,2000);
   delay (150);
   noTone (TonePin);

 delay (1000);


 // print screen template
 lcd.clear();

 lcd.setCursor(0, 0);
```

```
  lcd.print("QRG --.----- MHz  ");

  lcd.setCursor(0, 1);
  lcd.print("Bnd ");

  clearindicator ();
  RXindicator ();

  mySerial.begin(9600);  // CAT running at 9k6

}



//////////////////////MAIN LOOP/////////////////////////////////

void loop() {


listen:

  delay(1);                          // reduced from 50 !


next:
```

```
// detect if TX simulation switch depressed

if ((analogRead(TXsimPin)<512) and (simbuttonpressed ==0) and (TXstatus == 0)) {    // TX simulation

  simbuttonpressed = 1;
  lcd.setCursor(0, 1);
  lcd.print("TX ");
  clearindicator ();       // always to do, before RX and TX change
  TXindicator ();          // We are displaying TX condition
  }

if ((analogRead(TXsimPin)>512) and (simbuttonpressed ==1) ) {                        // TX simulation
  simbuttonpressed = 0;
  lcd.setCursor(0, 1);
  if (BAND != 0) lcd.print(BAND,DEC);
  if (BAND == 0) lcd.print("Man");
  clearindicator ();       // always to do, before RX and TX change
  RXindicator ();          // We are displaying RX condition
  }

// detect if PTT low (Transmit status)

if ((analogRead(PTTPin)<128) and (TXstatus == 0)) {                                  // We go into TX

  TXstatus = 1;
  lcd.setCursor(0, 1);
```

```
  lcd.print("TX ");
  clearindicator ();       // always to do, before RX and TX change
  TXindicator ();          // We are displaying & setting TX condition
  }

if ((analogRead(PTTPin)>256) and (TXstatus ==1 )) {   // We go into RX with some hysterisis
  TXstatus = 0;
  lcd.setCursor(0, 1);
  if (BAND != 0) lcd.print(BAND,DEC);
  if (BAND == 0) lcd.print("Man");
  clearindicator ();       // always to do, before RX and TX change
  RXindicator ();          // We are displaying & setting RX condition
  }

/* DEBUG  : test update cycle
    tone(TonePin,2000);
    delay (5);
    noTone (TonePin);
*/



if (TXstatus == 0) ReadButtons ();  // no reprogramming when really in TX mode !

if ((millis() - DisplayTime) > 350) {  // refresh CAT indicator
            lcd.setCursor(15, 0);
```

```
            lcd.print(" ");
                }

  if (mySerial.available() > 0) {
    incoming = mySerial.read();
    if (incoming == 254) {           // 1st byte is  an FE look for an FE to start
      goto start;
    }

    goto next;
  }


start:

  buffget[0] = 0;

  // delay(1);
  for ( i=0;i<10;i++) {               // get next 10 bytes
    if(mySerial.available() > 0) {
      buffget[i]=mySerial.read();     // load buffget with next 10 characters
       delay(2);                      //delay 1 ms if true, time for buffer fill
    }
  }
```

```
delay (10);                                   // delay is required to process buffer, do not remov

  if (buffget[0] == 254) {                    // again FE , as 2nd character ?

    goto next1;
  }                                           // yes, now do '00' test
  goto listen;                                // wrong array, get another


next1:

  if ((buffget[3] == 0) or (buffget[3] == 3) or  (buffget[3] == 5)     ){  // check to see if 4th ch
    goto next2;
  }                                           // detected, goto next2

/*
  if ((buffget[3] == 28) and (buffget[4] == 0)) TXstatus = 0;       // RX mode detect, if set by CA
  if ((buffget[3] == 28) and (buffget[4] == 1)) TXstatus = 1;       // TX mode detect, if set by CA
*/

  goto listen;                                // wrong array, get another


next2:
```

```
// Check for last byte

  if(buffget[9] == 253 ){                    // look for FD at end of array
    goto frequency;
  }                                          // if FD detected, goto frequency
  goto listen;                               // wrong command



frequency:                                   // we have frequency array on hand

    MHZ = (buffget[7]);

    MHZ = MHZ - (((MHZ/16) * 6));            // Transform bytes ICOM CAT
    if (MHZ >= 100)  goto listen;            // wrong byte

    KHZ = buffget[6];
    KHZ = KHZ - (((KHZ/16) * 6));            // Transform bytes ICOM CAT
    if (KHZ >= 100)  goto listen;            // wrong byte

    HZ = buffget[5];
    HZ = HZ - (((HZ/16) * 6));               // Transform bytes ICOM CAT
    if (HZ >= 100)  goto listen;             // wrong byte

  QRG = ((MHZ * 10000) + (KHZ * 100) + (HZ * 1)); // QRG variable stores frequency in MMkkkH  format
```

```
 // Print frequency

  lcd.setCursor(4, 0);
  if (QRG < 100000) lcd.print(" ");
  lcd.print(QRG/10000,DEC);
  lcd.print(".");
  if (QRG%10000 < 1000 )  lcd.print("0");
  if (QRG%10000 < 100 )   lcd.print("0");
  if (QRG%10000 < 10 )    lcd.print("0");
  lcd.print(QRG%10000,DEC);
  lcd.print(" MHz");
  lcd.write(0);                              // print the CAT activity indicator

// set display timer
  DisplayTime = millis();


// Which band ?

 QRGcomp = QRG / 10;
 BAND = 0;                 // default band = will generate error
 if ((QRGcomp < High4)   and (QRGcomp > Low4))     BAND = 4;
 if ((QRGcomp < High6)   and (QRGcomp > Low6))     BAND = 6;
 if ((QRGcomp < High10)  and (QRGcomp > Low10))    BAND = 10;
 if ((QRGcomp < High12)  and (QRGcomp > Low12))    BAND = 12;
 if ((QRGcomp < High15)  and (QRGcomp > Low15))    BAND = 15;
```

```
 if ((QRGcomp < High17)  and (QRGcomp > Low17))    BAND = 17;
 if ((QRGcomp < High20)  and (QRGcomp > Low20))    BAND = 20;
 if ((QRGcomp < High30)  and (QRGcomp > Low30))    BAND = 30;
 if ((QRGcomp < High40)  and (QRGcomp > Low40))    BAND = 40;
 if ((QRGcomp < High80)  and (QRGcomp > Low80))    BAND = 80;
 if ((QRGcomp < High160) and (QRGcomp > Low160))   BAND = 160;
 if (BAND == 0) {
   clearindicator ();                  // We are not on standard band, clear antenna's
      }



if (memorize == 1)   goto finish;  // antenna position must be memorized, do not read what is now in

 // Check band changed

if (BAND == oldBAND) goto finish;  // no bandchange

 ///// Band is changed ! /////

oldBAND = BAND;

 // now process antenna change
```

```
 // read value from EEPROM

  if (BAND == 0) {
        lcd.setCursor(0, 1);
        lcd.print("   ");
        goto finish;     // no need to read
        }

// display band
        lcd.setCursor(0, 1);
        lcd.print("   ");
        lcd.setCursor(0, 1);
        if (BAND != 0) lcd.print(BAND,DEC);
       clearindicator ();

  ReadEEPROMvalues ();

 //Show in display

   clearindicator ();       // always to do, before RX and TX change
   RXindicator ();          // We are assuming in RX mode while changing band



// Music Maestro
```

```
    tone(TonePin,1600);
    delay(150);
    tone(TonePin,1200);
    delay (150);
    noTone (TonePin);


   goto finish;
// end process antenna change //


finish:

  delay(0);

  goto listen;

}

////////////////SUBROUTINES//////////////////

void clearindicator () {  // Clear all antenna
  lcd.setCursor(3, 1);
  lcd.print(" 1 2 3 4 5 6");
  digitalWrite(A1Pin,0) ;
  digitalWrite(A2Pin,0) ;
```

```
  digitalWrite(A3Pin,0) ;
  digitalWrite(A4Pin,0) ;
  digitalWrite(A5Pin,0) ;
  digitalWrite(A6Pin,0) ;
}
////////////////////////////////////////////////////////////////////

void RXindicator () {    // Process selected RX antenna
  if (RXantenna == 1) {
      digitalWrite(A1Pin,1) ;
      lcd.setCursor(4, 1);
      lcd.write(2);

    }
if (RXantenna == 2) {
      digitalWrite(A2Pin,1) ;
      lcd.setCursor(6, 1);
      lcd.write(3);

     }
if (RXantenna == 3) {
      digitalWrite(A3Pin,1) ;
      lcd.setCursor(8, 1);
      lcd.write(4);

    }
```

```
if (RXantenna == 4) {
      digitalWrite(A4Pin,1) ;
      lcd.setCursor(10, 1);
      lcd.write(5);

    }
if (RXantenna == 5) {
      digitalWrite(A5Pin,1) ;
      lcd.setCursor(12, 1);
      lcd.write(6);

    }
if (RXantenna == 6) {
      digitalWrite(A6Pin,1) ;
      lcd.setCursor(14, 1);
      lcd.write(7);

    }
/* / Check if TX = RX ant
lcd.setCursor(15, 1);
lcd.print(" ");
lcd.setCursor(15, 1);
if (TXantenna = RXantenna) lcd.print("=");  // same RX as TX antenna
*/

}
```

```
////////////////////////////////////////////////////////////////////

void TXindicator () {    // Process selected TX antenna
  if (TXantenna == 1) {
      digitalWrite(A1Pin,1) ;
      lcd.setCursor(4, 1);
      lcd.write(2);

    }
if (TXantenna == 2) {
      digitalWrite(A2Pin,1) ;
      lcd.setCursor(6, 1);
      lcd.write(3);

     }
if (TXantenna == 3) {
      digitalWrite(A3Pin,1) ;
      lcd.setCursor(8, 1);
      lcd.write(4);

    }
if (TXantenna == 4) {
      digitalWrite(A4Pin,1) ;
      lcd.setCursor(10, 1);
      lcd.write(5);
```

```
    }
if (TXantenna == 5) {
      digitalWrite(A5Pin,1) ;
      lcd.setCursor(12, 1);
      lcd.write(6);

    }
if (TXantenna == 6) {
      digitalWrite(A6Pin,1) ;
      lcd.setCursor(14, 1);
      lcd.write(7);

    }
/*  Check if TX = RX ant
lcd.setCursor(15, 1);
lcd.print(" ");
lcd.setCursor(15, 1);
if (TXantenna = RXantenna) lcd.print("=");  // same RX as TX antenna
*/

}


////////////////////////////////////////////////////////////////

void ReadEEPROMvalues () {      //  Read Antenna's stored in EEPROM
```

```
if (BAND == 4) RXantenna =   EEPROM.read(4);
if (BAND == 4) TXantenna =   EEPROM.read(5);

if (BAND == 6) RXantenna =   EEPROM.read(6);
if (BAND == 6) TXantenna =   EEPROM.read(7);

if (BAND == 10) RXantenna =  EEPROM.read(10);
if (BAND == 10) TXantenna =  EEPROM.read(11);

if (BAND == 12) RXantenna =  EEPROM.read(12);
if (BAND == 12) TXantenna =  EEPROM.read(13);

if (BAND == 15) RXantenna =  EEPROM.read(15);
if (BAND == 15) TXantenna =  EEPROM.read(16);

if (BAND == 17) RXantenna =  EEPROM.read(17);
if (BAND == 17) TXantenna =  EEPROM.read(18);

if (BAND == 20) RXantenna =  EEPROM.read(20);
if (BAND == 20) TXantenna =  EEPROM.read(21);

if (BAND == 30) RXantenna =  EEPROM.read(30);
if (BAND == 30) TXantenna =  EEPROM.read(31);

if (BAND == 40) RXantenna =  EEPROM.read(40);
```

```
if (BAND == 40) TXantenna =  EEPROM.read(41);

if (BAND == 80) RXantenna =  EEPROM.read(80);
if (BAND == 80) TXantenna =  EEPROM.read(81);

if (BAND == 160) RXantenna = EEPROM.read(160);
if (BAND == 160) TXantenna = EEPROM.read(161);
}

//////////////////////////////////////////////////////////////////

void WriteEEPROMvalues () {      //  Write Antenna's in EEPROM

if (BAND == 4) EEPROM.write(4, RXantenna);
if (BAND == 4) EEPROM.write(5, TXantenna);

if (BAND == 6) EEPROM.write(6, RXantenna);
if (BAND == 6) EEPROM.write(7, TXantenna);

if (BAND == 10) EEPROM.write(10, RXantenna);
if (BAND == 10) EEPROM.write(11, TXantenna);

if (BAND == 12) EEPROM.write(12, RXantenna);
if (BAND == 12) EEPROM.write(13, TXantenna);

if (BAND == 15) EEPROM.write(15, RXantenna);
```

```
if (BAND == 15) EEPROM.write(16, TXantenna);

if (BAND == 17) EEPROM.write(17, RXantenna);
if (BAND == 17) EEPROM.write(18, TXantenna);

if (BAND == 20) EEPROM.write(20, RXantenna);
if (BAND == 20) EEPROM.write(21, TXantenna);

if (BAND == 30) EEPROM.write(30, RXantenna);
if (BAND == 30) EEPROM.write(31, TXantenna);

if (BAND == 40) EEPROM.write(40, RXantenna);
if (BAND == 40) EEPROM.write(41, TXantenna);

if (BAND == 80) EEPROM.write(80, RXantenna);
if (BAND == 80) EEPROM.write(81, TXantenna);

if (BAND == 160) EEPROM.write(160, RXantenna);
if (BAND == 160) EEPROM.write(161, TXantenna);
}


//////////////////////////////////////////////////////////////////

void Reset() {                  // RESET memory in EEPROM
 lcd.setCursor(0, 1);
```

```
lcd.print(" HOLD TO RESET");
// erase all EEPROM contents  TODO !!!
delay(2000);
 if ((analogRead(LeftPin)<512) and (analogRead(RightPin)<512))  { // Yes, we have a reset !
 lcd.setCursor(0, 1);
 lcd.print(" RESETTING ...    ");

delay(1000);
     EEPROM.write(4, 0);
     EEPROM.write(5, 0);
     EEPROM.write(6, 0);
     EEPROM.write(7, 0);
     EEPROM.write(10, 0);
     EEPROM.write(11, 0);
     EEPROM.write(12, 0);
     EEPROM.write(13, 0);
     EEPROM.write(15, 0);
     EEPROM.write(16, 0);
     EEPROM.write(17, 0);
     EEPROM.write(18, 0);
     EEPROM.write(20, 0);
     EEPROM.write(21, 0);
     EEPROM.write(30, 0);
     EEPROM.write(31, 0);
     EEPROM.write(40, 0);
     EEPROM.write(41, 0);
```

```
      EEPROM.write(80, 0);
      EEPROM.write(81, 0);
      EEPROM.write(160, 0);
      EEPROM.write(161, 0);
 lcd.setCursor(0, 1);
 lcd.print(" RESTARTING ...    ");
  delay(1000);
    }
}

////////////////////////////////////////////////////////////////////

void ReadButtons () {     // Detect antenna select buttons depressed


if ((analogRead(LeftPin)<512) and (analogRead(TXsimPin)>512) and (buttonpressed == 0)) {    // Left
  tone(TonePin,2000);
  delay (20);
  noTone (TonePin);
  ButtonTime = millis ();
  buttonpressed = 1;
  if (BAND != 0) memorize = 1;
  clearindicator ();
  RXantenna = RXantenna -- ;
  if ((RXantenna == 0) or (RXantenna > 6 ))RXantenna = (6);
  RXindicator ();
```

```
  }

if ((analogRead(RightPin)<512) and (analogRead(TXsimPin)>512) and (buttonpressed == 0)   ) {    // R
  tone(TonePin,2000);
  delay (20);
  noTone (TonePin);
  ButtonTime = millis ();
  buttonpressed = 1;
  if (BAND != 0) memorize = 1;
  clearindicator ();
  RXantenna = RXantenna ++ ;
  if (RXantenna >= 7) RXantenna = (1);
  RXindicator ();
  }


if ((analogRead(LeftPin)<512) and (analogRead(TXsimPin)<512) and (buttonpressed == 0)) {    // Left
  tone(TonePin,2000);
  delay (20);
  noTone (TonePin);
  ButtonTime = millis ();
  buttonpressed = 1;
  if (BAND != 0) memorize = 1;
  clearindicator ();
  TXantenna = TXantenna -- ;
  if ((TXantenna == 0) or (TXantenna > 6 ))TXantenna = (6);
```

```
  TXindicator ();
  }

if ((analogRead(RightPin)<512) and (analogRead(TXsimPin)<512) and (buttonpressed == 0)   ) {    // R
  tone(TonePin,2000);
  delay (20);
  noTone (TonePin);
  ButtonTime = millis ();
  buttonpressed = 1;
  if (BAND != 0) memorize = 1;
  clearindicator ();
  TXantenna = TXantenna ++ ;
  if (TXantenna >= 7) TXantenna = (1);
  TXindicator ();
  }


 //  Timers check
  if (millis () - ButtonTime > 300) { // Button press detect
   buttonpressed = (0);
    }

 // Memorize antenna
  if ((millis () - ButtonTime > memotime) and (memorize == 1)){ // Now memorize antenna after xxx ms
   memorize = 0;
   lcd.setCursor(15, 1);  // print MEMORY indicator
```

```
   lcd.write(1);

   if (TXantenna == 0) TXantenna = RXantenna;   // if TX antenna undefined, set TX antenna same as R
   WriteEEPROMvalues ();
    tone(TonePin,2000);
    delay (150);
    noTone (TonePin);
    delay (50);
    tone(TonePin,2000);
    delay (150);
    noTone (TonePin);

   lcd.setCursor(15, 1);  // erase M indicator
   lcd.print(" ");
   lcd.setCursor(0, 1);
   lcd.print("   ");
   lcd.setCursor(0, 1);
   if (BAND != 0) lcd.print(BAND,DEC);
   if (BAND == 0) lcd.print("Man");

    }


}
```