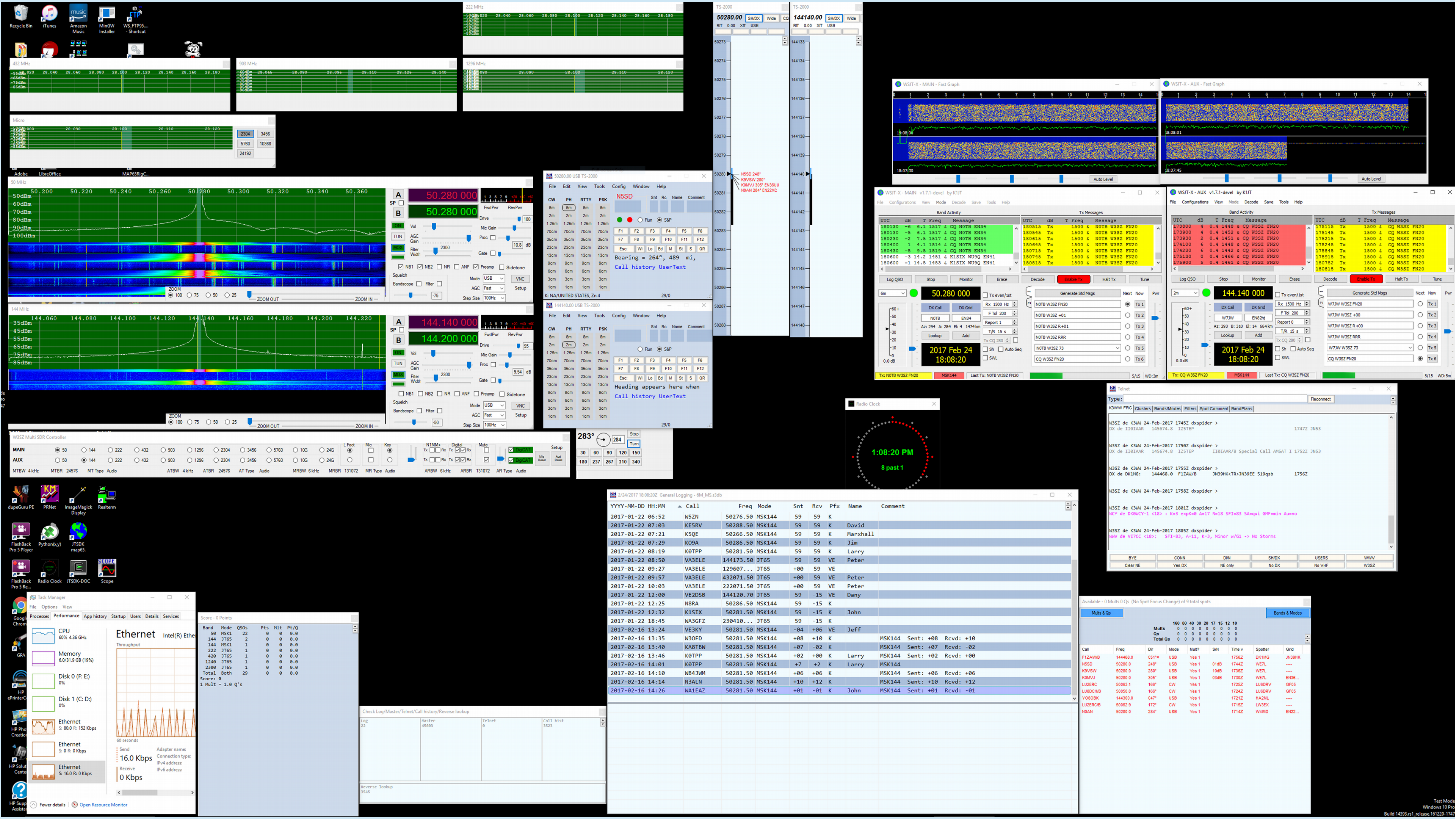
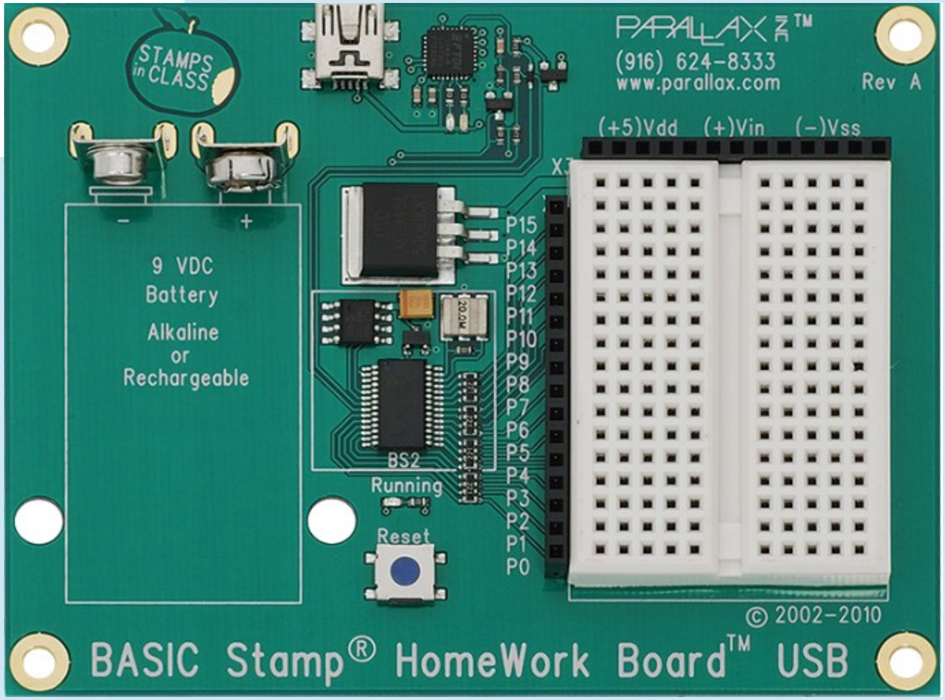
Station Automation --W3SZ



```
YYYY-MM-DD HH:MM    Call        Freq Mode     Snt  Rcv  Pfx  Name       Comment
2017-01-22 06:52    W5ZN    50276.50 MSK144   59   59   K
2017-01-22 07:03    KE5RV   50288.50 MSK144   59   59   K    David
2017-01-22 07:21    K5QE    50266.50 MSK144   59   59   K    Marxhall
2017-01-22 07:29    KO9A    50286.50 MSK144   59   59   K    Jim
2017-01-22 08:19    K0TPP   50281.50 MSK144   59   59   K    Larry
2017-01-22 08:50    VA3ELE 144173.50 JT65     59   59   VE   Peter
2017-01-22 09:27    VA3ELE 129607... JT65     +00  59   VE
2017-01-22 09:57    VA3ELE 432071.50 JT65     +00  59   VE   Peter
2017-01-22 10:03    VA3ELE 222071.50 JT65     +00  59   VE   Peter
2017-01-22 12:00    VE2DSB 144120.70 JT65     59   -15  VE   Dany
2017-01-22 12:25    N8RA    50286.50 MSK144   59   -15  K
2017-01-22 12:32    K1SIX   50281.50 MSK144   59   -15  K    John
2017-01-22 18:45    WA3GFZ 230410... JT65     59   -15  K
2017-02-16 13:24    VE3KY   50281.50 MSK144   -04  +06  VE   Jeff
2017-02-16 13:35    W3OFD   50281.50 MSK144   +08  +10  K               MSK144  Sent: +08  Rcvd: +10
2017-02-16 13:40    KA8TBW  50281.50 MSK144   +07  -02  K               MSK144  Sent: +07  Rcvd: -02
2017-02-16 13:46    K0TPP   50281.50 MSK144   +02  +00  K    Larry      MSK144  Sent: +02  Rcvd: +00
2017-02-16 14:01    K0TPP   50281.50 MSK144   +7   +2   K    Larry      MSK144
2017-02-16 14:10    WB4JWM  50281.50 MSK144   +06  +06  K               MSK144  Sent: +06  Rcvd: +06
2017-02-16 14:14    N3ALN   50281.50 MSK144   +10  +12  K               MSK144  Sent: +10  Rcvd: +12
2017-02-16 14:26    WA1EAZ  50281.50 MSK144   +01  -01  K    John       MSK144  Sent: +01  Rcvd: -01
```

```
W3SZ de K3WW 24-Feb-2017 1745Z dxspider >
DX de II0IAAR   145674.8  IZ5TEP                                 1747Z JN53

W3SZ de K3WW 24-Feb-2017 1750Z dxspider >
DX de II0IAAR   145674.8  IZ5TEP       II0IAAR/8 Special Call AMSAT I 1752Z JN53

W3SZ de K3WW 24-Feb-2017 1755Z dxspider >
DX de DK1MG:    144468.0  F1ZAW/B      JN39HK<TR>JN39EE 519qsb        1756Z

W3SZ de K3WW 24-Feb-2017 1758Z dxspider >

W3SZ de K3WW 24-Feb-2017 1801Z dxspider >
WCY de DK0WCY-1 <18> : K=3 expK=0 A=17 R=18 SFI=83 SA=qui GMF=min Au=no

W3SZ de K3WW 24-Feb-2017 1805Z dxspider >
WWV de VE7CC <18>:   SFI=83, A=11, K=3, Minor w/G1 -> No Storms
```

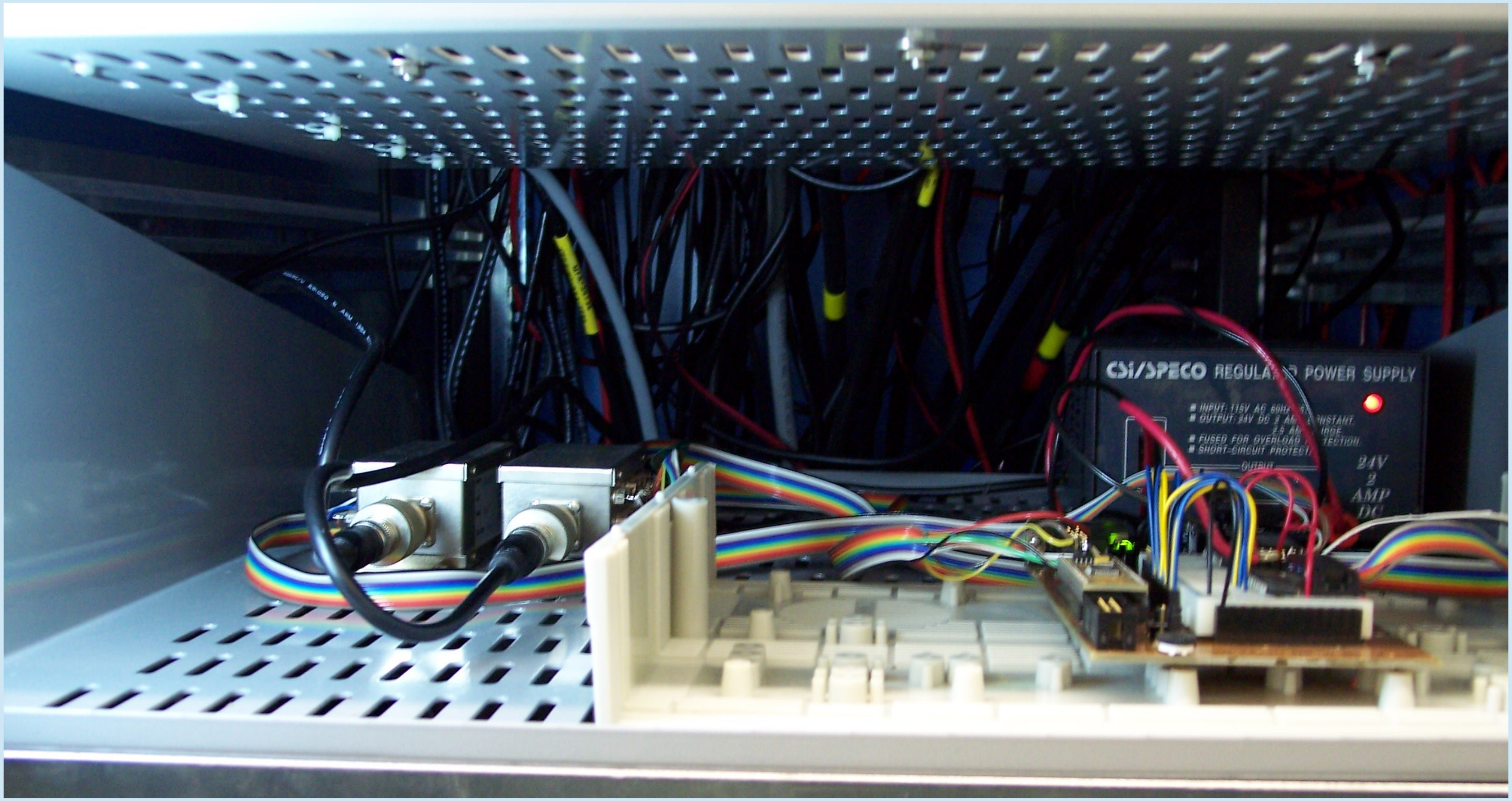
Programmable IF Attenuator



Programmable IF Attenuator

- Get Binary band info from N3FTI device
- Set receive and transmit in-line attenators to provide proper RF signal levels to/from IF radio
- Alan Industries 50 DA63 gives 0-63 dB atten. in 1 dB steps
  - Requires 26 VDC control voltage and power
  - Binary control needs 6 control pins each for Tx, Rx
- Parallax Basic Stamp controller
  - Uses PBASIC language
  - 16 I/O pins (BS 2p24)
- Use ULN2803 Octal Darlington Array IC's to control 26V signal to the attenuator using 5V output from Basic Stamp I/O pins
- Basic Stamp code is at:
  - http://w3sz.com/BasicStampDeviceControlCodeHandout.pdf

Programmable IF Attenuator

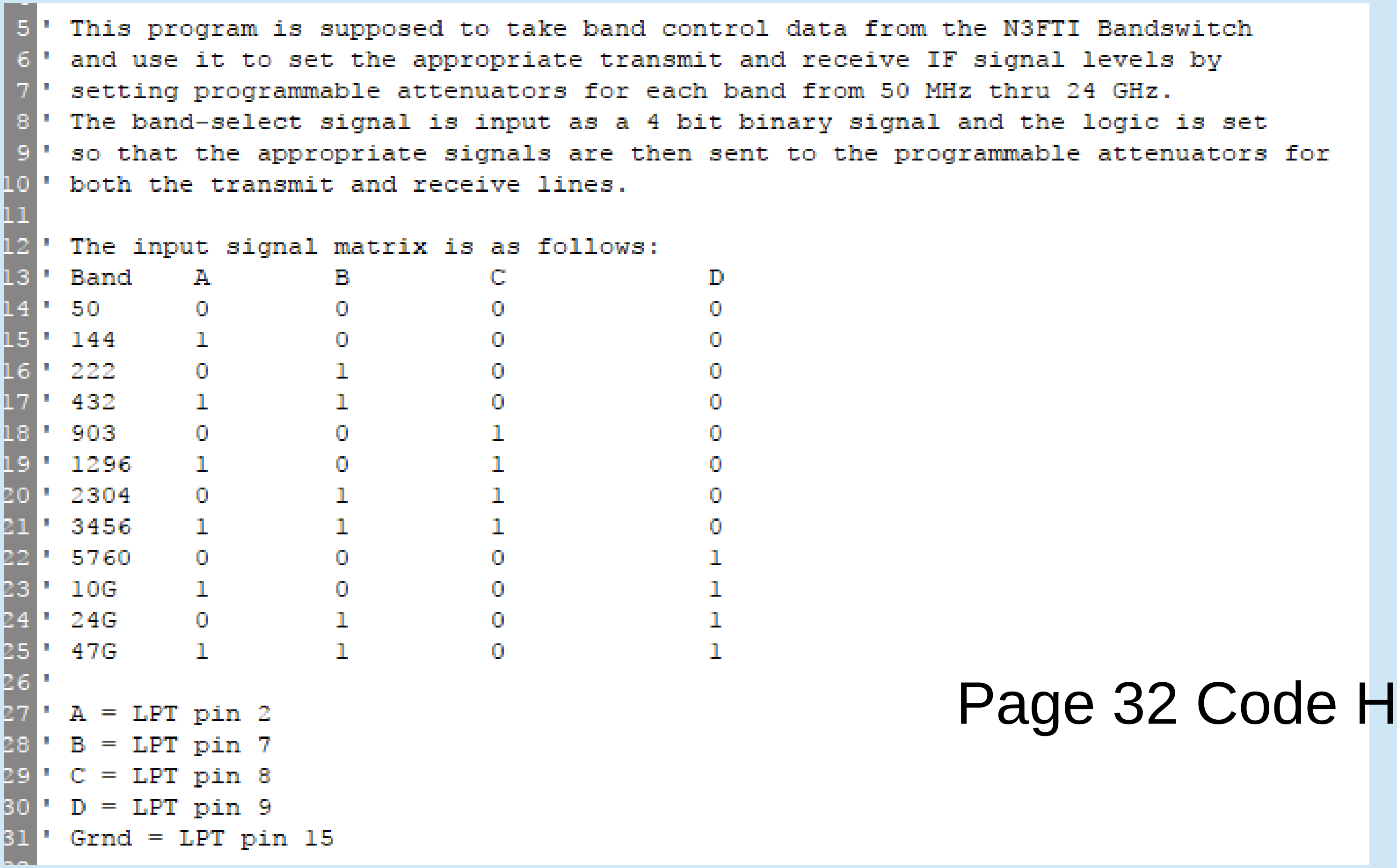


Programmable IF Attenuator

Coding

- No Libraries to declare
- Define and initialize variables
- Define input pins
- Get input from N3FTI device
- Parse input from N3FTI device to determine band
- Define attenuation levels for Tx and Rx based on band
- Determine Binary output pin settings based on attenuation levels
- Define output pins
- Set output pin levels

Programmable IF Attenuator

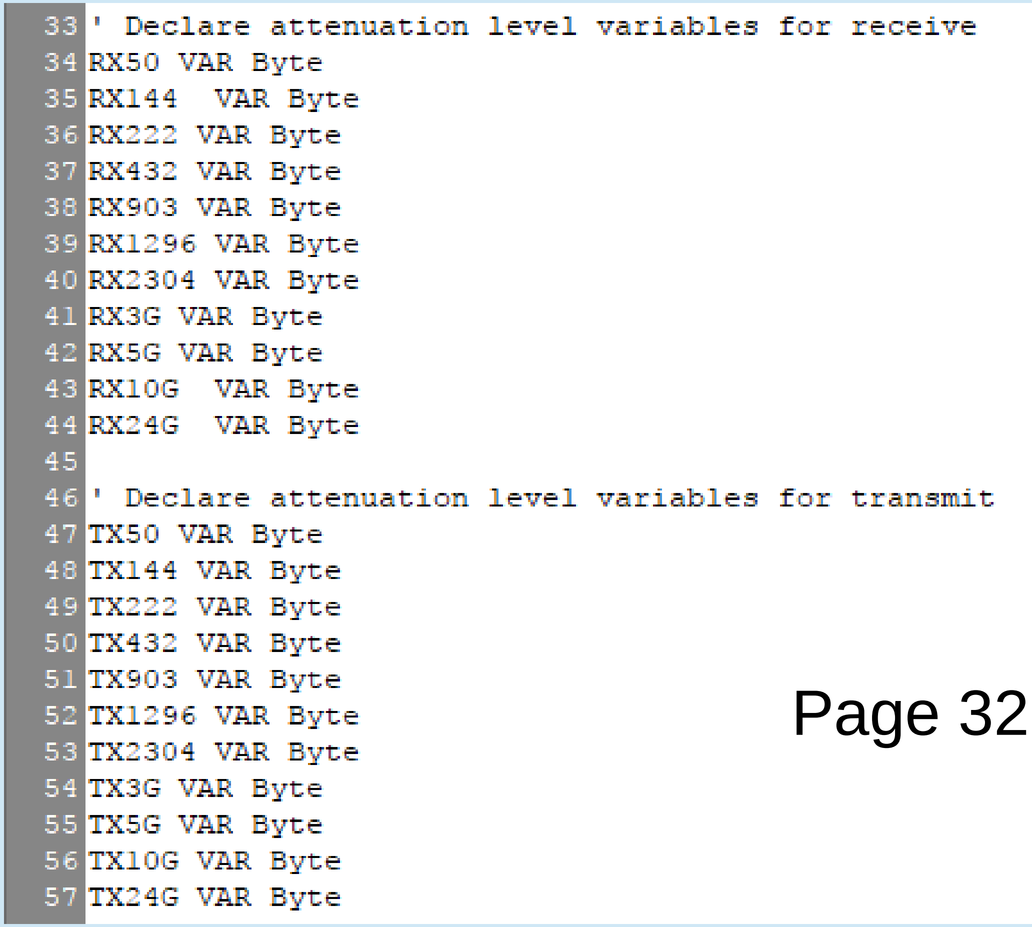


```
' This program is supposed to take band control data from the N3FTI Bandswitch
' and use it to set the appropriate transmit and receive IF signal levels by
' setting programmable attenuators for each band from 50 MHz thru 24 GHz.
' The band-select signal is input as a 4 bit binary signal and the logic is set
' so that the appropriate signals are then sent to the programmable attenuators for
' both the transmit and receive lines.

' The input signal matrix is as follows:
' Band    A        B         C              D
' 50      0        0         0              0
' 144     1        0         0              0
' 222     0        1         0              0
' 432     1        1         0              0
' 903     0        0         1              0
' 1296    1        0         1              0
' 2304    0        1         1              0
' 3456    1        1         1              0
' 5760    0        0         0              1
' 10G     1        0         0              1
' 24G     0        1         0              1
' 47G     1        1         0              1
'
' A = LPT pin 2
' B = LPT pin 7
' C = LPT pin 8
' D = LPT pin 9
' Grnd = LPT pin 15
```

Page 32 Code Handout

Programmable IF Attenuator

Declare variables

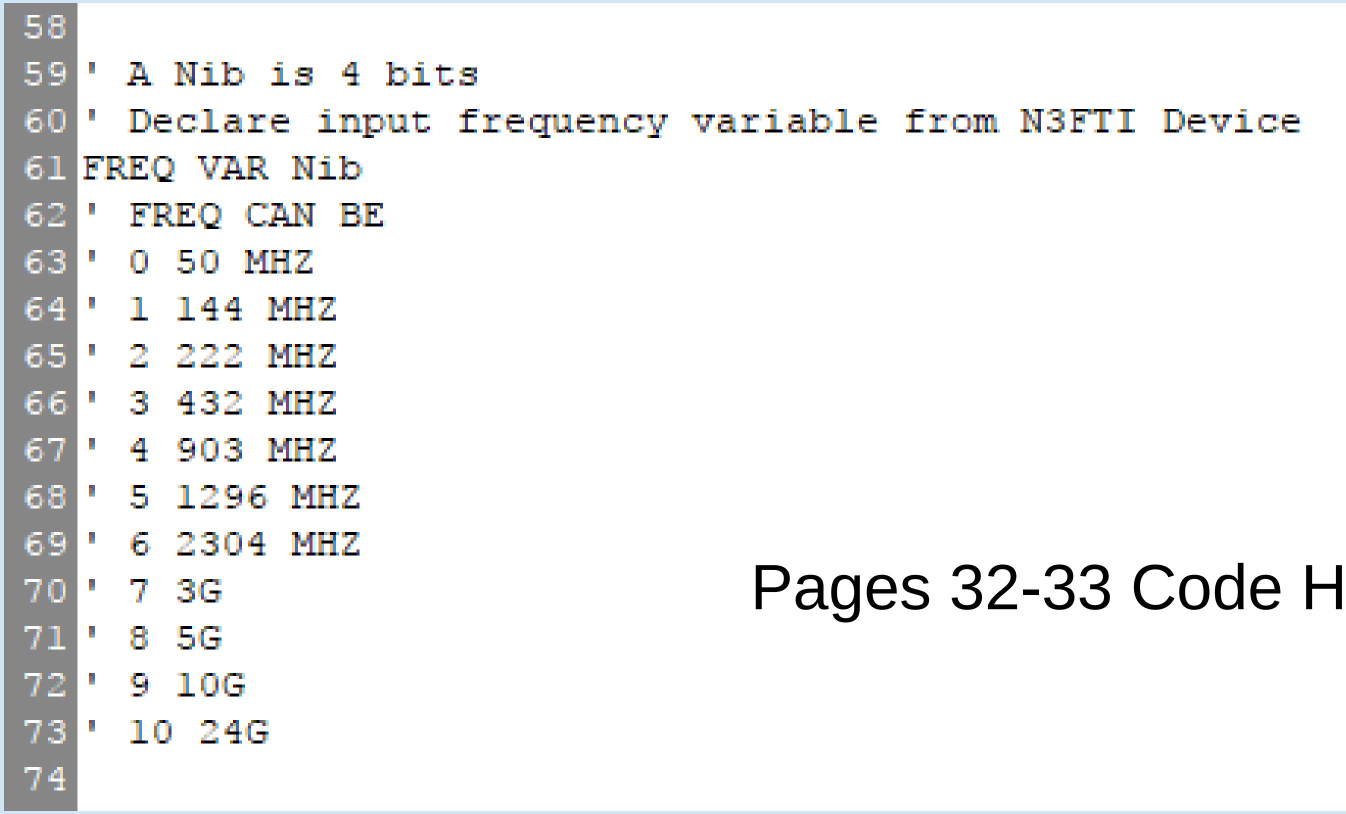


```
' Declare attenuation level variables for receive
RX50 VAR Byte
RX144  VAR Byte
RX222 VAR Byte
RX432 VAR Byte
RX903 VAR Byte
RX1296 VAR Byte
RX2304 VAR Byte
RX3G VAR Byte
RX5G VAR Byte
RX10G  VAR Byte
RX24G  VAR Byte

' Declare attenuation level variables for transmit
TX50 VAR Byte
TX144 VAR Byte
TX222 VAR Byte
TX432 VAR Byte
TX903 VAR Byte
TX1296 VAR Byte
TX2304 VAR Byte
TX3G VAR Byte
TX5G VAR Byte
TX10G VAR Byte
TX24G VAR Byte
```

Page 32 Code Handout

Programmable IF Attenuator

Declare variable FREQ

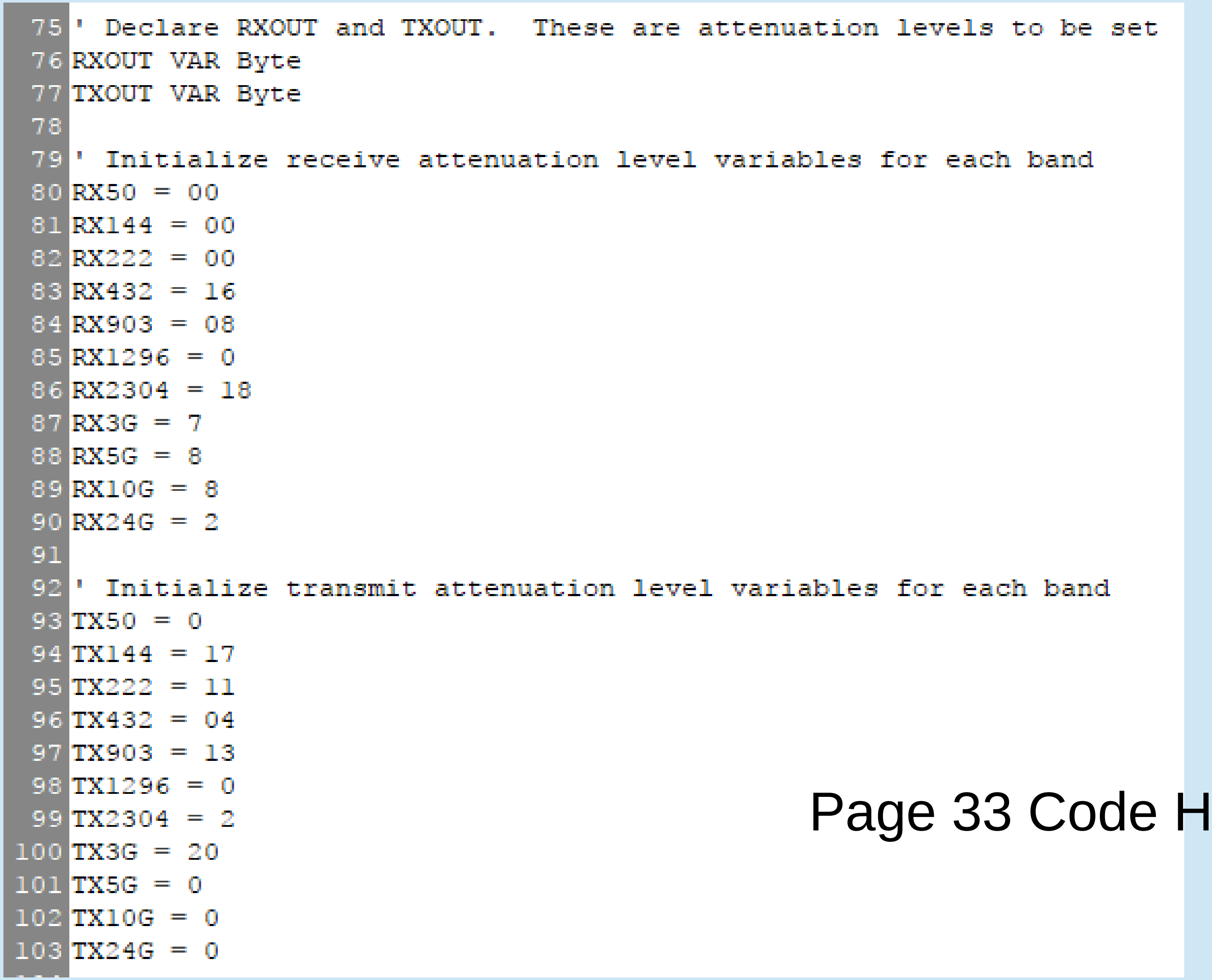


```
' A Nib is 4 bits
' Declare input frequency variable from N3FTI Device
FREQ VAR Nib
' FREQ CAN BE
' 0 50 MHZ
' 1 144 MHZ
' 2 222 MHZ
' 3 432 MHZ
' 4 903 MHZ
' 5 1296 MHZ
' 6 2304 MHZ
' 7 3G
' 8 5G
' 9 10G
' 10 24G
```

Pages 32-33 Code Handout

Programmable IF Attenuator

Declare and initialize more variables

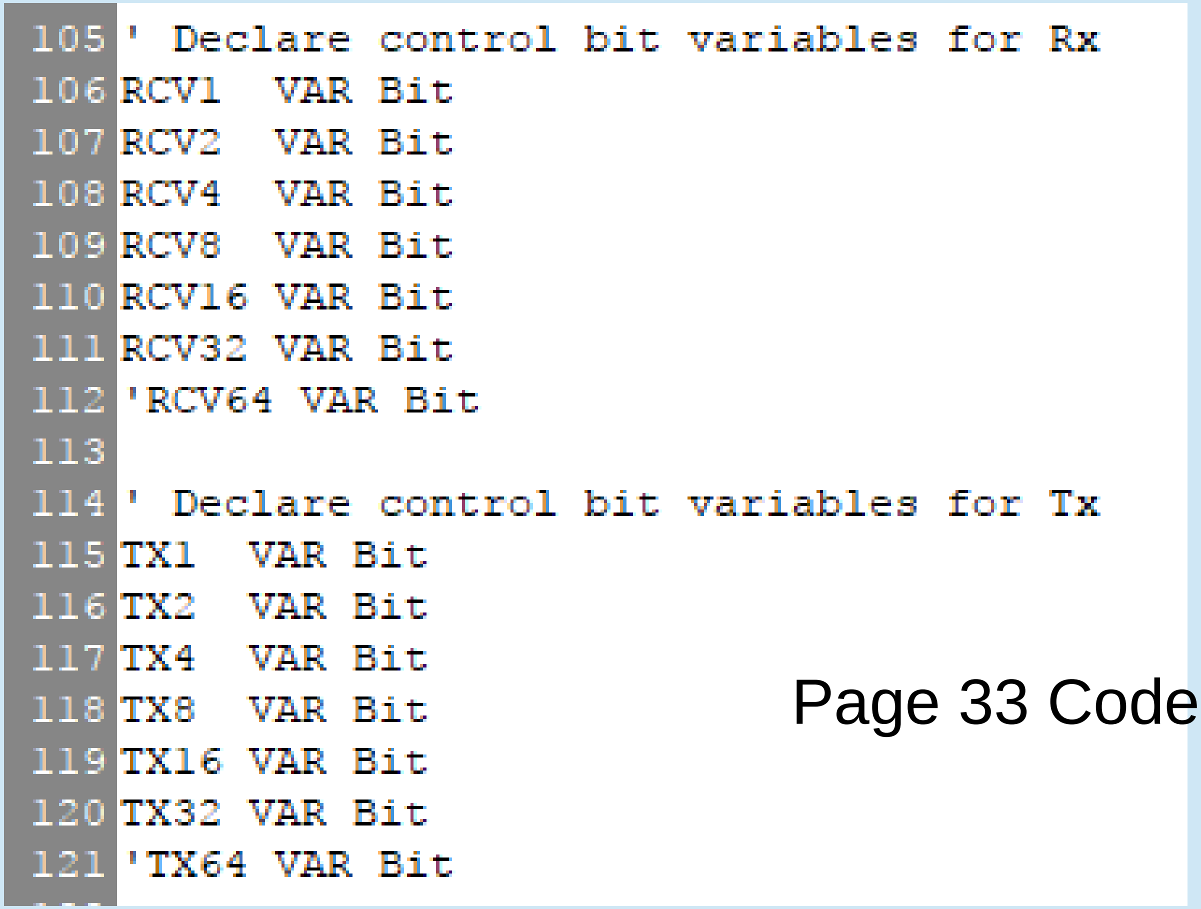


```
' Declare RXOUT and TXOUT.  These are attenuation levels to be set
RXOUT VAR Byte
TXOUT VAR Byte

' Initialize receive attenuation level variables for each band
RX50 = 00
RX144 = 00
RX222 = 00
RX432 = 16
RX903 = 08
RX1296 = 0
RX2304 = 18
RX3G = 7
RX5G = 8
RX10G = 8
RX24G = 2

' Initialize transmit attenuation level variables for each band
TX50 = 0
TX144 = 17
TX222 = 11
TX432 = 04
TX903 = 13
TX1296 = 0
TX2304 = 2
TX3G = 20
TX5G = 0
TX10G = 0
TX24G = 0
```

Page 33 Code Handout

Programmable IF Attenuator

Declare still more variables

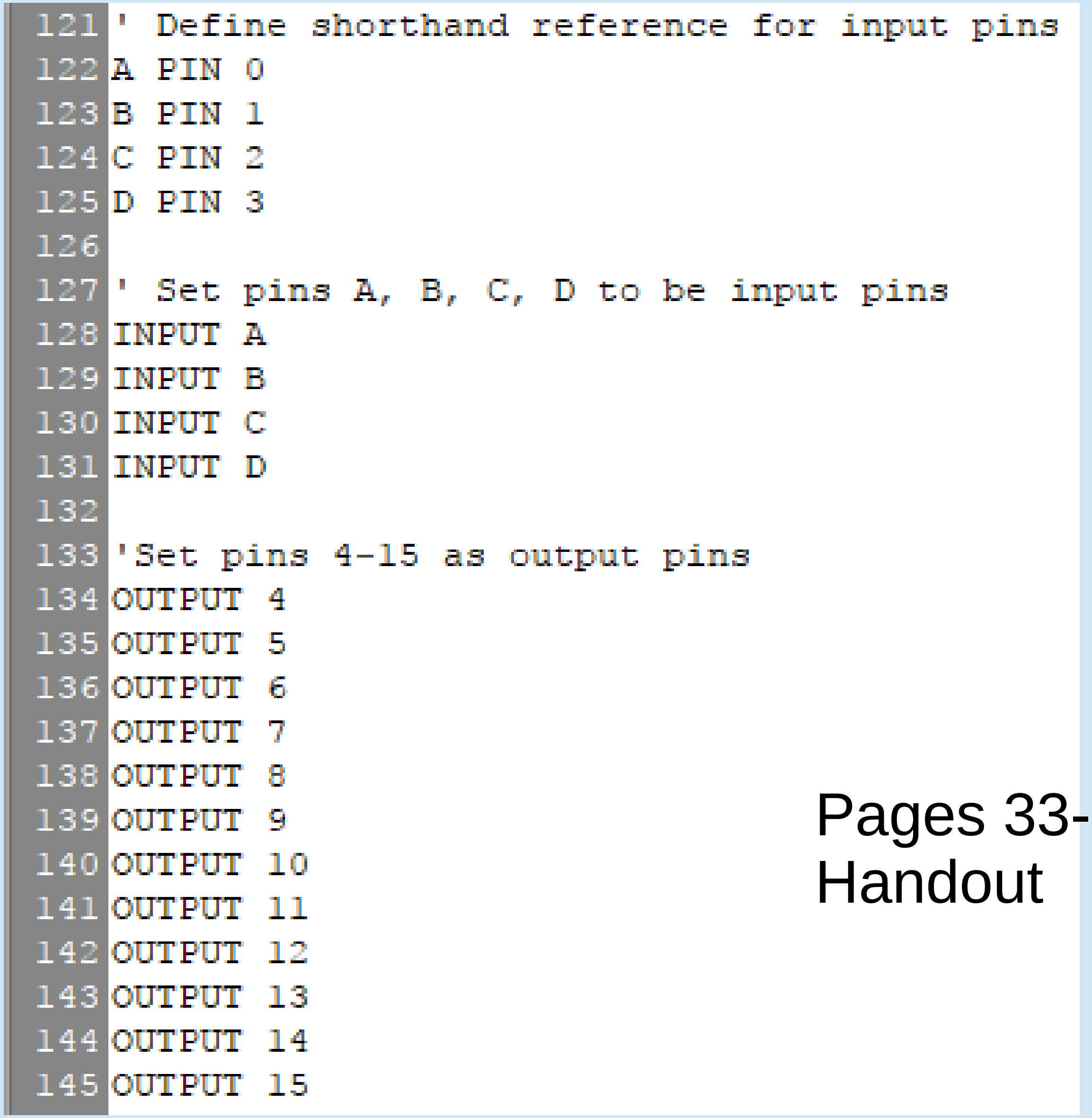


```
' Declare control bit variables for Rx
RCV1  VAR Bit
RCV2  VAR Bit
RCV4  VAR Bit
RCV8  VAR Bit
RCV16 VAR Bit
RCV32 VAR Bit
'RCV64 VAR Bit

' Declare control bit variables for Tx
TX1  VAR Bit
TX2  VAR Bit
TX4  VAR Bit
TX8  VAR Bit
TX16 VAR Bit
TX32 VAR Bit
'TX64 VAR Bit
```

Page 33 Code Handout

Programmable IF Attenuator

Define input and output pins

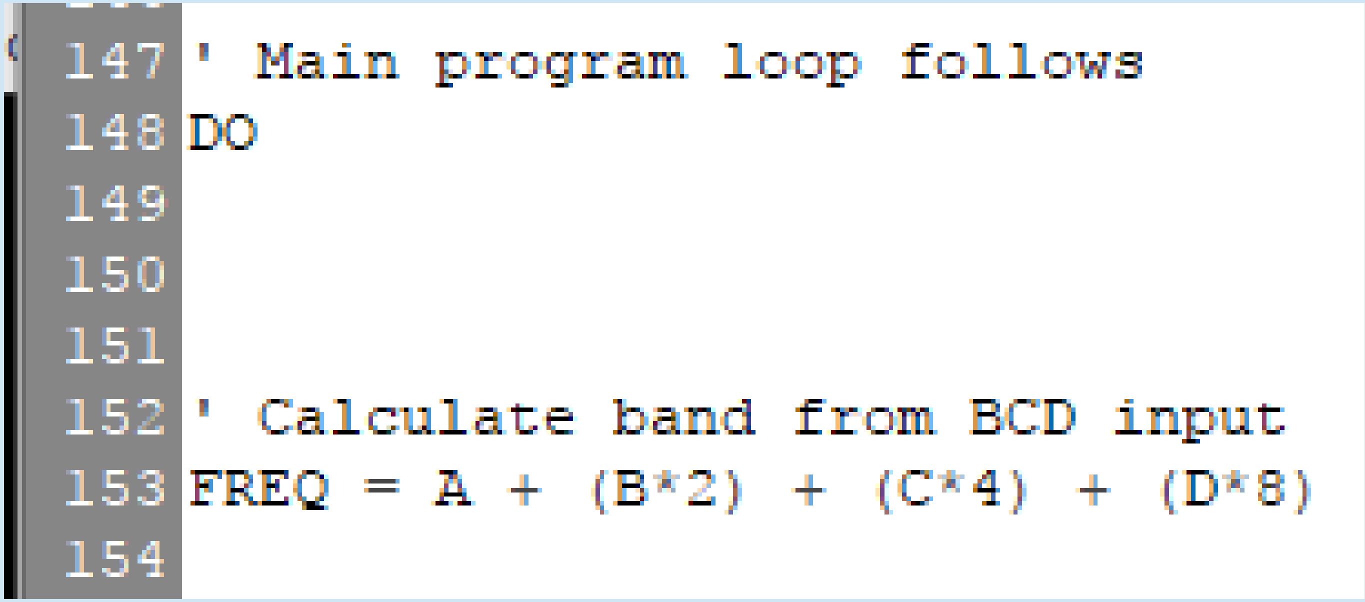


```
' Define shorthand reference for input pins
A PIN 0
B PIN 1
C PIN 2
D PIN 3

' Set pins A, B, C, D to be input pins
INPUT A
INPUT B
INPUT C
INPUT D

'Set pins 4-15 as output pins
OUTPUT 4
OUTPUT 5
OUTPUT 6
OUTPUT 7
OUTPUT 8
OUTPUT 9
OUTPUT 10
OUTPUT 11
OUTPUT 12
OUTPUT 13
OUTPUT 14
OUTPUT 15
```

Pages 33-34 Code Handout

Programmable IF Attenuator

Start Loop, Read Inputs, Calculate Band

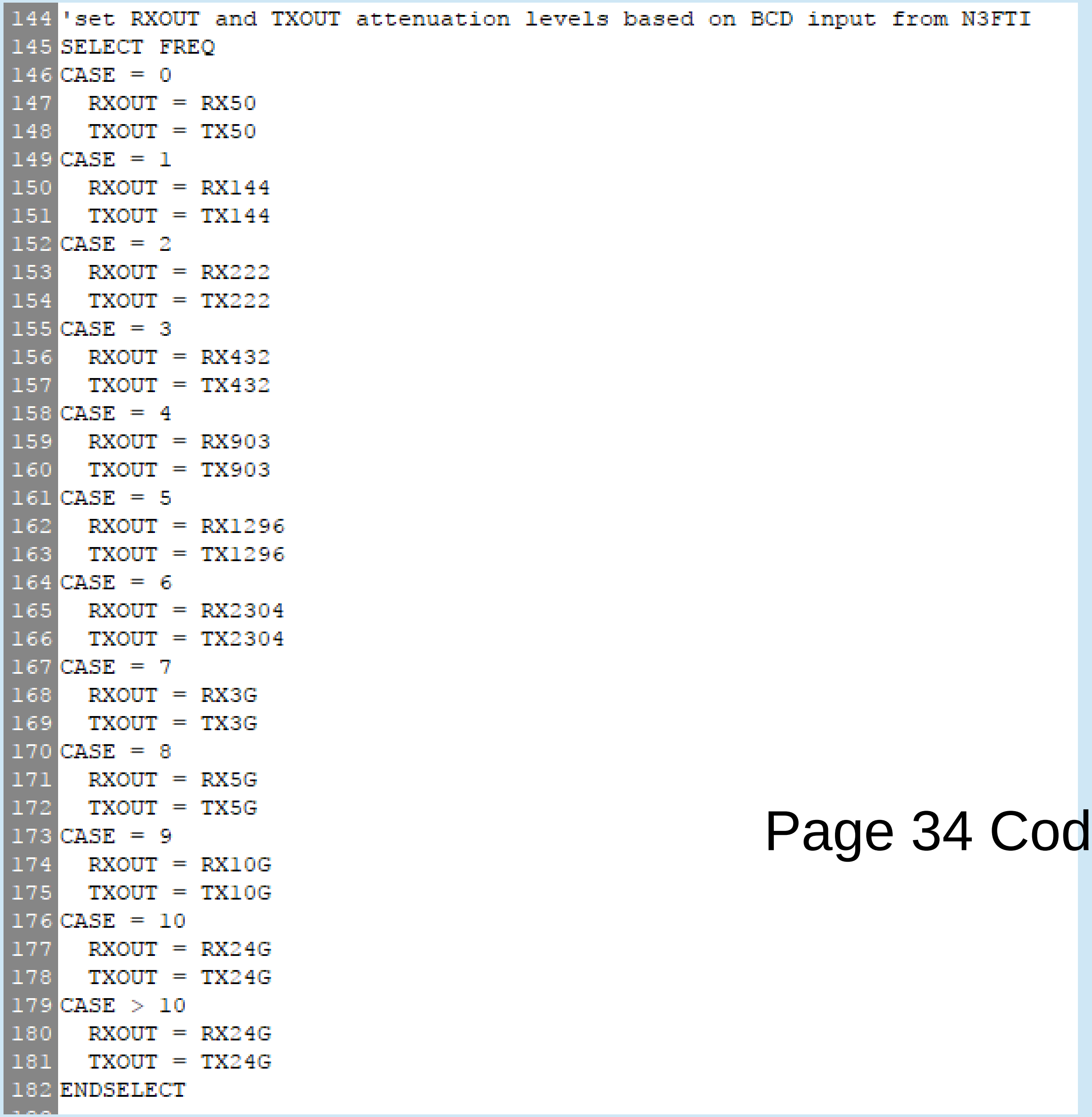


```
' Main program loop follows
DO



' Calculate band from BCD input
FREQ = A + (B*2) + (C*4) + (D*8)

```

Page 34 Code Handout

Programmable IF Attenuator

Determine attenuation levels based on band

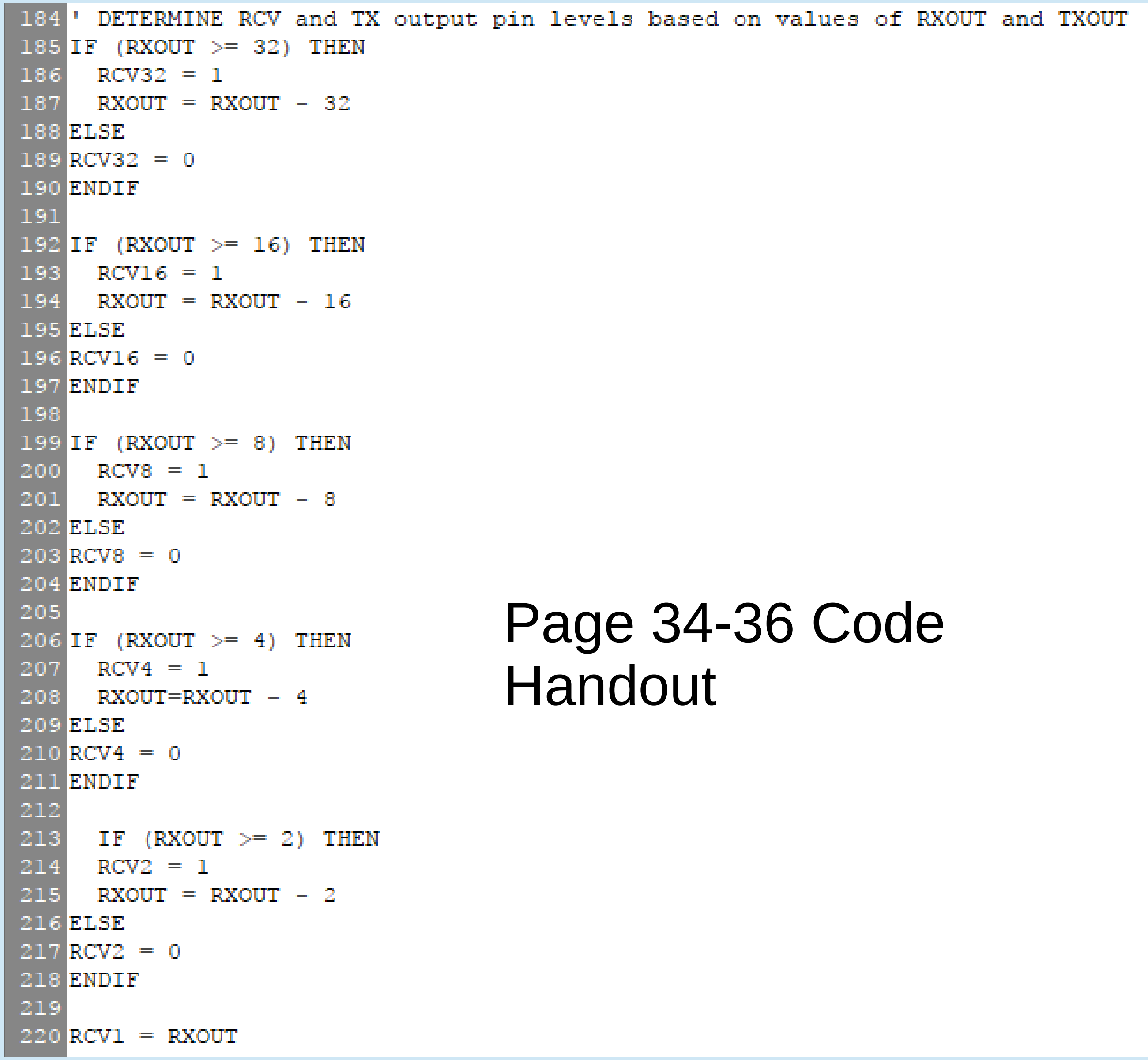


```
'set RXOUT and TXOUT attenuation levels based on BCD input from N3FTI
SELECT FREQ
CASE = 0
  RXOUT = RX50
  TXOUT = TX50
CASE = 1
  RXOUT = RX144
  TXOUT = TX144
CASE = 2
  RXOUT = RX222
  TXOUT = TX222
CASE = 3
  RXOUT = RX432
  TXOUT = TX432
CASE = 4
  RXOUT = RX903
  TXOUT = TX903
CASE = 5
  RXOUT = RX1296
  TXOUT = TX1296
CASE = 6
  RXOUT = RX2304
  TXOUT = TX2304
CASE = 7
  RXOUT = RX3G
  TXOUT = TX3G
CASE = 8
  RXOUT = RX5G
  TXOUT = TX5G
CASE = 9
  RXOUT = RX10G
  TXOUT = TX10G
CASE = 10
  RXOUT = RX24G
  TXOUT = TX24G
CASE > 10
  RXOUT = RX24G
  TXOUT = TX24G
ENDSELECT
```

Page 34 Code Handout

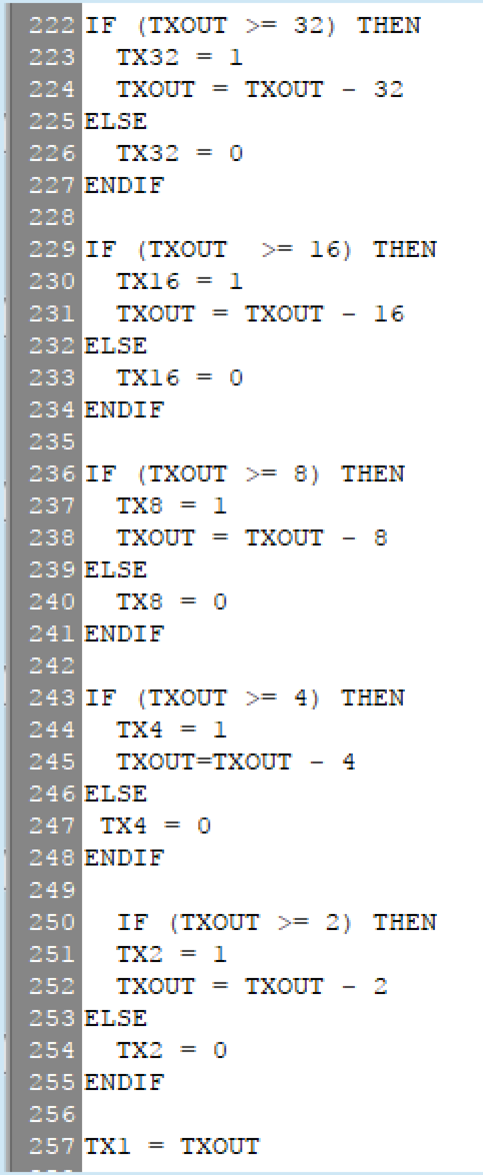
Programmable IF Attenuator

Determine Binary output pin levels based on attenuation levels



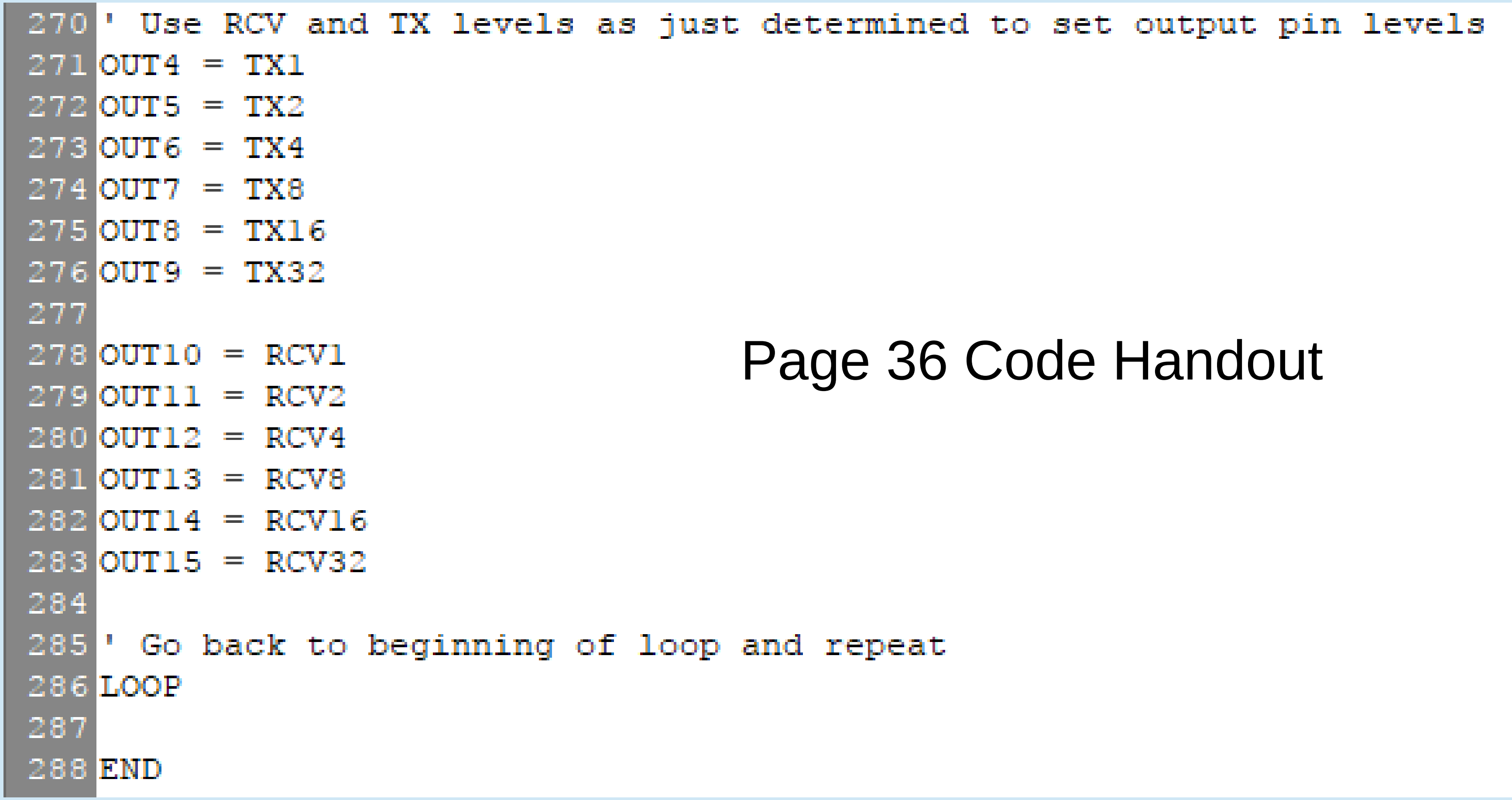
```
' DETERMINE RCV and TX output pin levels based on values of RXOUT and TXOUT
IF (RXOUT >= 32) THEN
  RCV32 = 1
  RXOUT = RXOUT - 32
ELSE
RCV32 = 0
ENDIF

IF (RXOUT >= 16) THEN
  RCV16 = 1
  RXOUT = RXOUT - 16
ELSE
RCV16 = 0
ENDIF

IF (RXOUT >= 8) THEN
  RCV8 = 1
  RXOUT = RXOUT - 8
ELSE
RCV8 = 0
ENDIF

IF (RXOUT >= 4) THEN
  RCV4 = 1
  RXOUT=RXOUT - 4
ELSE
RCV4 = 0
ENDIF

  IF (RXOUT >= 2) THEN
  RCV2 = 1
  RXOUT = RXOUT - 2
ELSE
RCV2 = 0
ENDIF

RCV1 = RXOUT
```

Page 34-36 Code Handout



```
IF (TXOUT >= 32) THEN
  TX32 = 1
  TXOUT = TXOUT - 32
ELSE
  TX32 = 0
ENDIF

IF (TXOUT  >= 16) THEN
  TX16 = 1
  TXOUT = TXOUT - 16
ELSE
  TX16 = 0
ENDIF

IF (TXOUT >= 8) THEN
  TX8 = 1
  TXOUT = TXOUT - 8
ELSE
  TX8 = 0
ENDIF

IF (TXOUT >= 4) THEN
  TX4 = 1
  TXOUT=TXOUT - 4
ELSE
 TX4 = 0
ENDIF

  IF (TXOUT >= 2) THEN
  TX2 = 1
  TXOUT = TXOUT - 2
ELSE
  TX2 = 0
ENDIF

TX1 = TXOUT
```

Programmable IF Attenuator
Set output pin levels



```
' Use RCV and TX levels as just determined to set output pin levels
OUT4 = TX1
OUT5 = TX2
OUT6 = TX4
OUT7 = TX8
OUT8 = TX16
OUT9 = TX32

OUT10 = RCV1
OUT11 = RCV2
OUT12 = RCV4
OUT13 = RCV8
OUT14 = RCV16
OUT15 = RCV32

' Go back to beginning of loop and repeat
LOOP

END
```

Page 36 Code Handout

Why Even Mention Basic Stamp?

- **To illustrate some major points of this symposium**:
  - Following the “prescription” for writing code given in this symposium will allow you to easily write code in any language
    - Use Google to find a piece of previously written code relevant to your objective
    - Read the “found” code and modify it to suit your objective
    - Learn from the “found” code
    - When you run into a roadblock, ask Google
- With access to Google, the language used borders on irrelevant; it is the logic that is important

  THE LOGIC IS ALWAYS THE SAME, REGARDLESS OF WHAT PROGRAMMING LANGUAGE IS USED!!

Lets quickly convert the Basic Stamp PBasic to Arduino C

- 1. Change markers designating “comment” lines
  - Comments in PBasic are indicated by single quote
  - Comments in Arduino C are indicated by double-forward slash
  - Just cut and paste to change all single quotes to “//”
- 2. Change the form of variable and constant declarations
  - “RX50 VAR Byte” is declaration syntax in PBasic
  - “byte RX50” is declaration syntax in Arduino C
  - Use variable type “int” instead of “byte” in Arduino C
  - To change each declaration statement; Just cut and paste to change all “XXXX VAR Byte” to “ int XXXX”
    - When you declare variables, just include the initialization in the declaration statement:

      int RX50 = 8;

Page 45 Code Handout

Convert PBasic to Arduino-C

- 3. Statements in PBasic don't end with a semi-colon but those in Arduino C do, so add a ";" to the end of each statement
- 4. Because of limited memory, some variables in PBasic were type "Nibs" - half a byte. Just make these variables int-type variables in C; we don't need to worry about memory
- 5. Same for "bit" variables in PBasic...just make them int-type variables in Arduino C

Pages 45-47 Code
Handout

Convert PBasic to Arduino-C Handling the GPIO Pins

- 6. Cut the input and output pin definitions and setup portions from the PBasic program and paste them into the definitions and "setup" portions of the Arduino C program
  - For the definition of input pins, "A PIN 0" would become

    const int PinA = A0; (Arduino analog input pin labels start with "A")
  - Add in the setup section, for each input pin:

    pinMode(PinA, INPUT);

    pinMode(PinB, INPUT); etc.

Pages 47-48 Code Handout

- And we need to add in the definitions section a variable to "read" each pin, for example for PinA:

  int A = 0;

Convert PBasic to Arduino-C
Handling the GPIO Pins

- For output pins, “OUTPUT 4” would become

```
PinMode(4, OUTPUT)
```

- But lets improve code and assign a constant label to each output pin for easier pin identification, e.g.:

```
const int TxOUT1 = 4;
```

- And we need to set the mode of each OUTPUT pin and initialize it, so “OUTPUT 4” is replaced by:

```
const int TxOUT1 =4;
PinMode(TxOUT1, OUTPUT);
digitalWrite(TxOUT1, LOW);
```

- Do the same for each Rx and Tx output pin

Page 48 Code Handout

Convert PBasic to Arduino-C
Main Program Loop
Reading and Parsing Input Pins

- The “automatic pin read” in PBasic becomes:

  A = digitalRead(PinA);

- THE LOGIC REMAINS THE SAME:

  //Calculate band from Binary input

  FREQ = A + (B*2) + (C*4) + (D*8);

  IS UNCHANGED EXCEPT FOR ADDING “;”

Page 49 Code Handout

Convert PBasic to Arduino-C Determining Output Levels

- //set RXOUT and TXOUT attenuation levels based on Binary input from N3FTI

```
SELECT FREQ
CASE = 0
    RXOUT = RX50
    TXOUT = TX50
CASE = 1
    RXOUT = RX144
    TXOUT = TX144
```

- //set RXOUT and TXOUT attenuation levels based on Binary input from N3FTI

```
switch (FREQ) {
case 0 : {
   RXOUT = RX50;
   TXOUT = TX50;
    break; }
case 1: {
   RXOUT = RX144;
   TXOUT = TX144;
    break; }
```

Pages 49-50 Code Handout

Convert PBasic to Arduino-C
Main Program Loop
Final SELECT case statement

- PBasic had:

```
CASE > 10
 RXOUT = RX24G
 TXOUT = TX24G
```

Page 50 Code Handout

- C switch statements only support simple integer matches:

```
case 11: {
 RXOUT = RX24G;
 TXOUT = TX24G;
 break;}
```

Convert PBasic to Arduino-C
End of SELECT Statement

- PBasic had:

```
ENDSELECT
```

- In Auduino C this is simply:

```
}
```

Page 50 Code Handout

Convert PBasic to Arduino-C
Set Pin Outputs using Output Levels We Just Determined

```
// DETERMINE RCV
and TX output pin
levels based on values
of RXOUT and TXOUT

IF (RXOUT >= 32) THEN

  RCV32 = 1

  RXOUT = RXOUT - 32

ELSE

RCV32 = 0

ENDIF
```

```
// DETERMINE RCV
and TX output pin
levels based on values
of RXOUT and TXOUT

if (RXOUT >= 32) {

  RCV32 = 1;

  RXOUT = RXOUT – 32;}

else {

RCV32 = 0; }
```

Page 50-51 Code Handout

Convert PBasic to Arduino-C
Set Pin Outputs

```
// Use RCV and TX
levels as just
determined to set
output pin levels

OUT4 = TX1
```

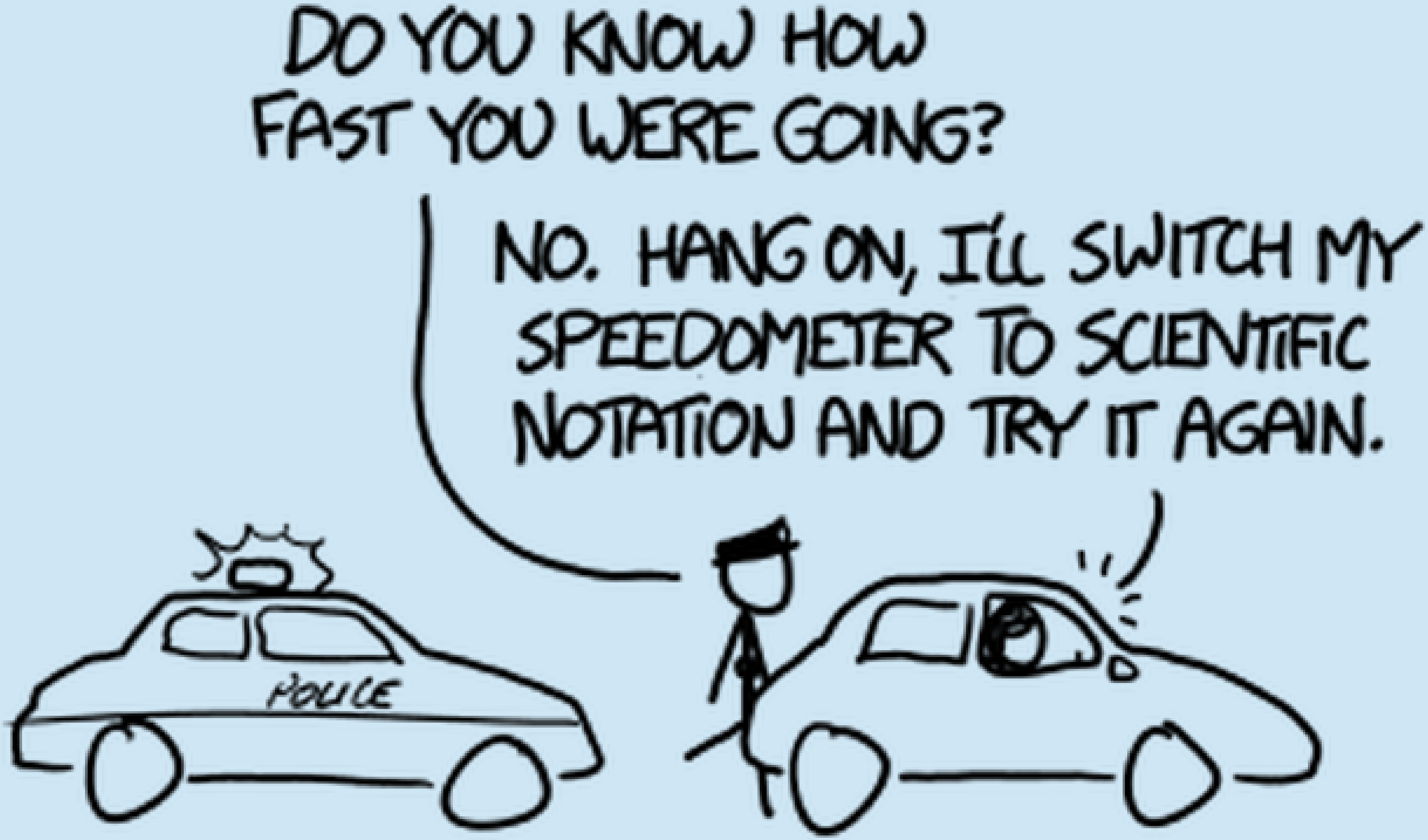
```
// Use RCV and TX
levels as just determined
to set output pin levels

digitalWrite(TxOUT1,TX1);
```

Page 52 Code Handout

Convert PBasic to Arduino-C

- A few more lines of code added for Serial Port output (for debugging)
- Code tested with hardware and is working
- Code is at:
  - http://w3sz.com/IFLevelSet_ContestSettingsNewIno.ino

- We could have converted instead from .bsp to python for use on BeagleBone Black or Raspberry Pi, just as easily
- Again, the language is almost irrelevant...it is the LOGIC that is important


DO YOU KNOW HOW FAST YOU WERE GOING?
NO. HANG ON, I'LL SWITCH MY SPEEDOMETER TO SCIENTIFIC NOTATION AND TRY IT AGAIN.
POLICE