Arduino N1MM Transverter Bandswitch

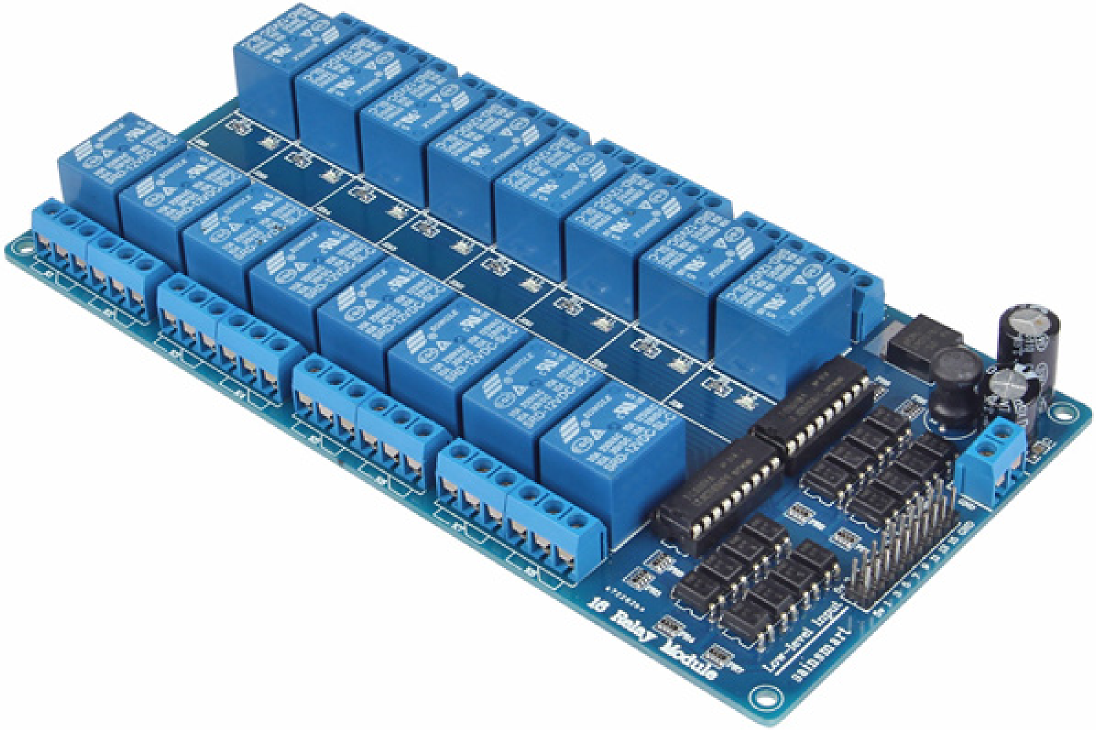
Arduino N1MM Transverter Bandswitch

- Switch transverter band when band changed in N1MM
    - Separate relay for each band, using Sainsmart Relay Board
- Cover 50 MHz thru 76 GHz
- Use USB serial port for communications between N1MM and the Arduino
    - Use OTRSP (Open Two Radio Switching Protocol)
        - Developed by Paul Young, K1XM

Arduino N1MM Transverter Bandswitch

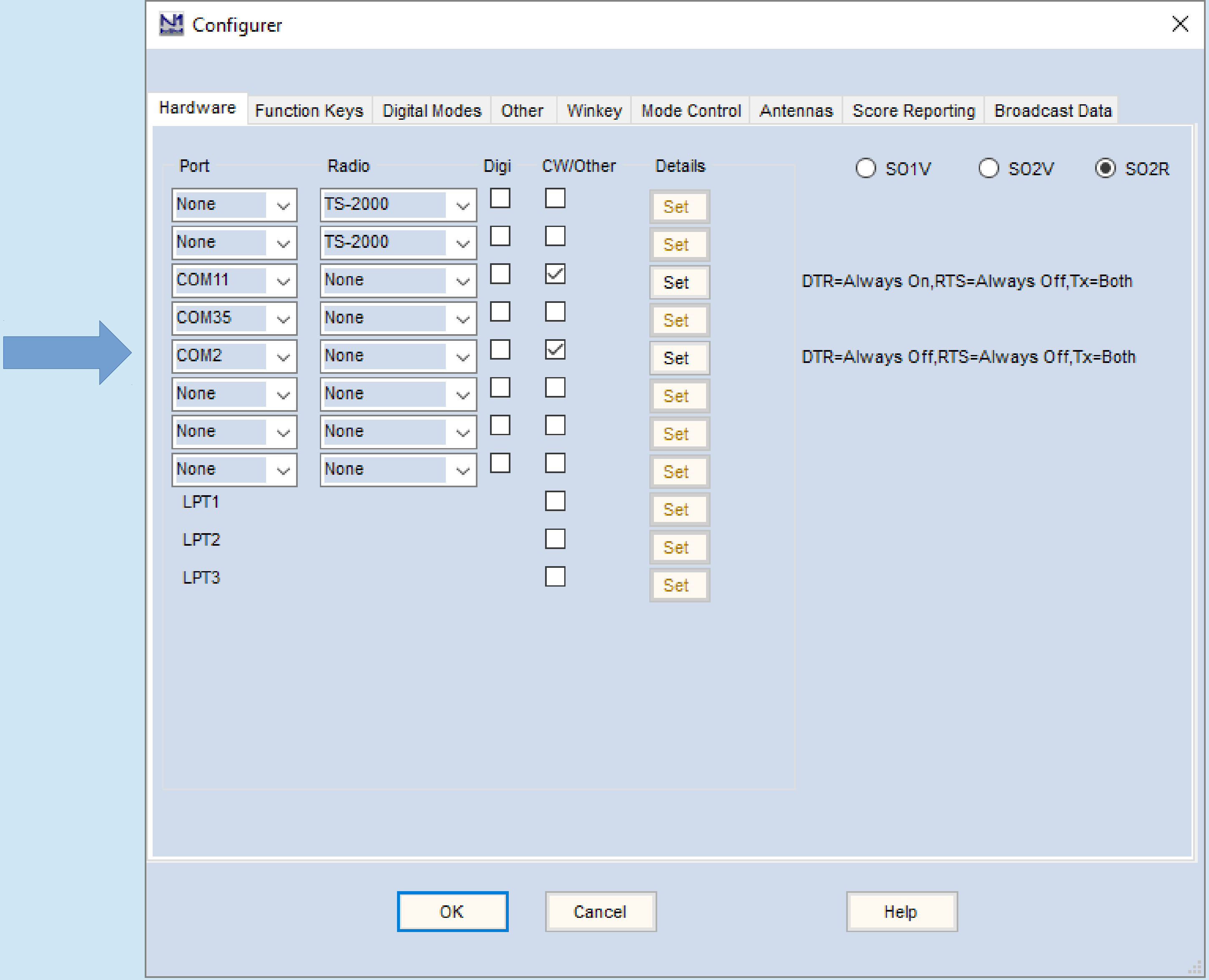
- 50 MHz-76 GHz → 13 bands → 13 GPIO pins
- Uses 19% (6366 of 32256 available bytes for Uno) of program storage space <Flash>)
- Uses 24% (511 of 2048 available bytes for Uno) of SRAM (static random access memory), where variables are placed
- UNO has sufficient GPIO pins and memory

SainSmart 16 Channel 12V Relay Module



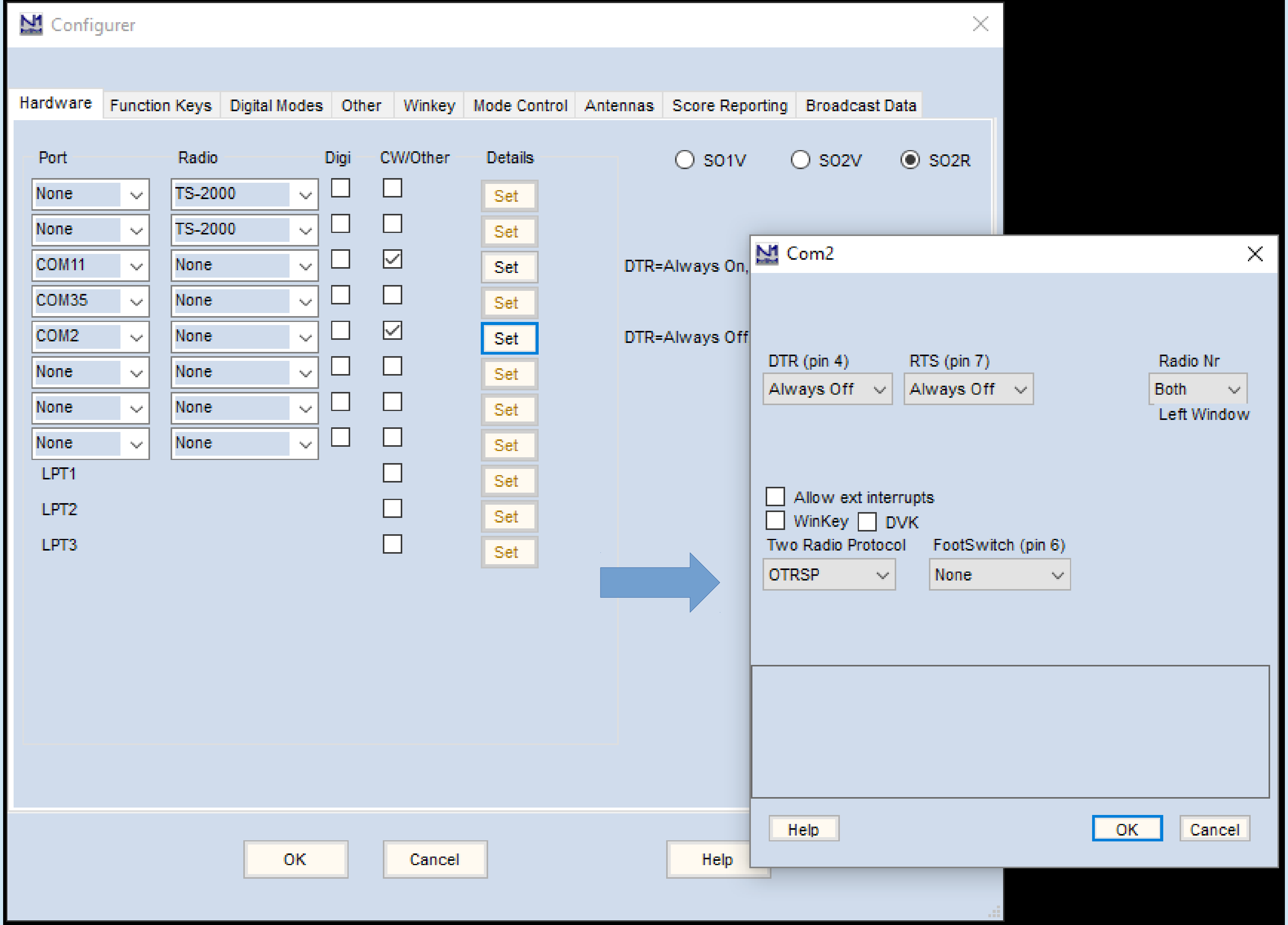
OTRSP

- For this project, need only one-way communication from N1MM to Arduino
- Need only to send N1MM-Radio-Number (n) and Band (bb)
- In OTRSP-speak, this is sent as “AUXnbb”
  - n is either “1” or “2” and bb is 00 – 12
  - bb is defined via N1MM Configure page
  - e.g for Radio 1 and 222 MHz, “AUX102” will be sent by N1MM



○ SO1V ○ SO2V ● SO2R

Port	Radio	Digi	CW/Other	Details	
None	TS-2000	□	□	Set	
None	TS-2000	□	□	Set	
COM11	None	□	☑	Set	DTR=Always On,RTS=Always Off,Tx=Both
COM35	None	□	□	Set	
COM2	None	□	☑	Set	DTR=Always Off,RTS=Always Off,Tx=Both
None	None	□	□	Set	
None	None	□	□	Set	
None	None	□	□	Set	
LPT1			□	Set	
LPT2			□	Set	
LPT3			□	Set	



○ SO1V ○ SO2V ● SO2R

Port	Radio	Digi	CW/Other	Details
None	TS-2000	□	□	Set
None	TS-2000	□	□	Set
COM11	None	□	☑	Set
COM35	None	□	□	Set
COM2	None	□	☑	Set
None	None	□	□	Set
None	None	□	□	Set
None	None	□	□	Set
LPT1			□	Set
LPT2			□	Set
LPT3			□	Set

DTR=Always On,

DTR=Always Off

Com2

DTR (pin 4): Always Off

RTS (pin 7): Always Off

Radio Nr: Both

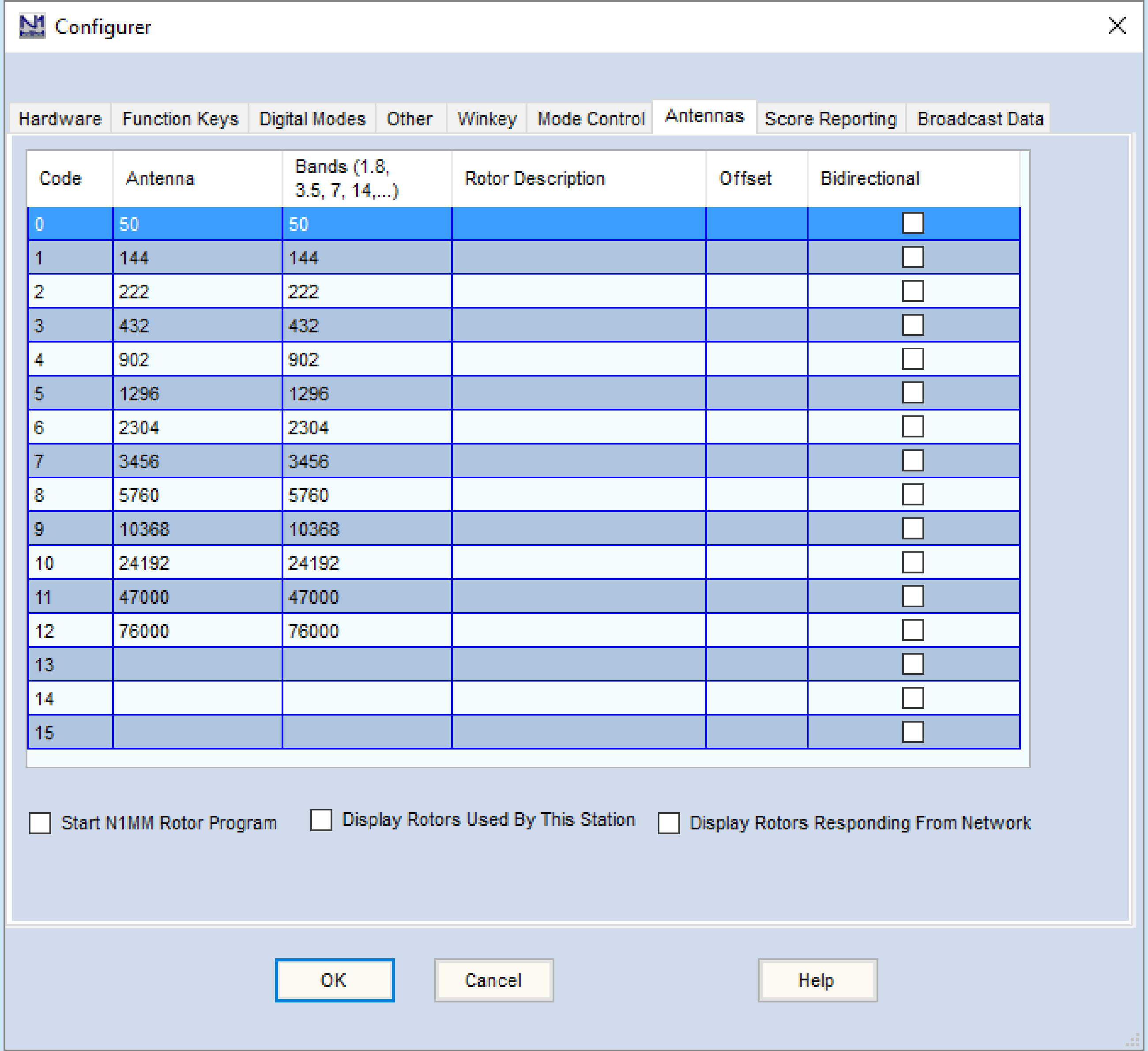
Left Window

□ Allow ext interrupts

□ WinKey □ DVK

Two Radio Protocol: OTRSP

FootSwitch (pin 6): None



Configurer

Code	Antenna	Bands (1.8, 3.5, 7, 14,...)	Rotor Description	Offset	Bidirectional
0	50	50			□
1	144	144			□
2	222	222			□
3	432	432			□
4	902	902			□
5	1296	1296			□
6	2304	2304			□
7	3456	3456			□
8	5760	5760			□
9	10368	10368			□
10	24192	24192			□
11	47000	47000			□
12	76000	76000			□
13					□
14					□
15					□

□ Start N1MM Rotor Program □ Display Rotors Used By This Station □ Display Rotors Responding From Network

Demo of
Arduino N1MM Transverter
Bandswitch

Programming Steps - General

1) Include libraries containing classes with external functions (Optional)
2) Define variables and constants (Optional)
3) Setup ()

   Define and initialize GPIO pins / Analog I/O pins

   Define, start, serial port(s), Ethernet port(s)

4) Loop()

   Receive input from ports / GPIO pins / Analog pins

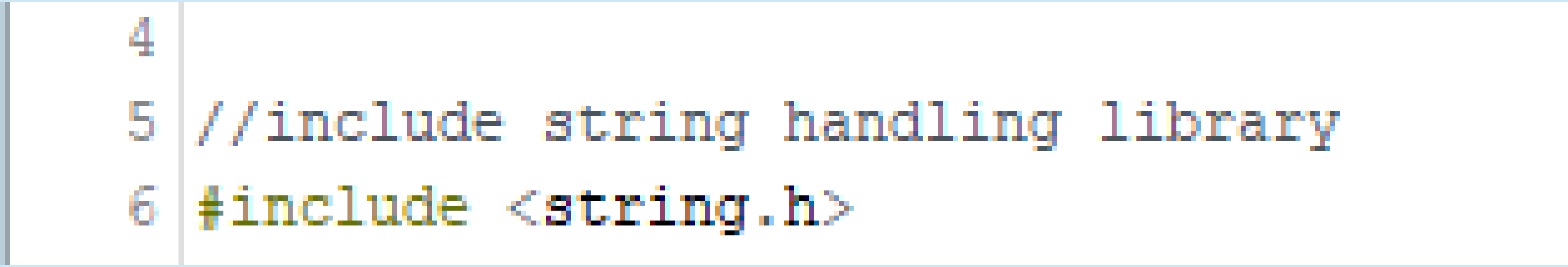
   Parse / process data to extract desired information

   Use information derived from data to perform desired task (e.g. switch GPIO pins) or to send information to client computer

5) From within Loop(), call other functions() as needed (Optional)

Arduino Example

Include Libraries

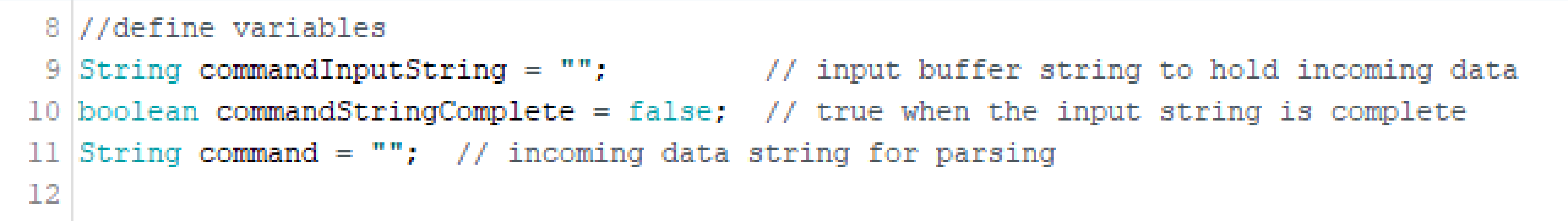


```
//include string handling library
#include <string.h>
```

Preprocessor directive to include string.h library

Arduino Example

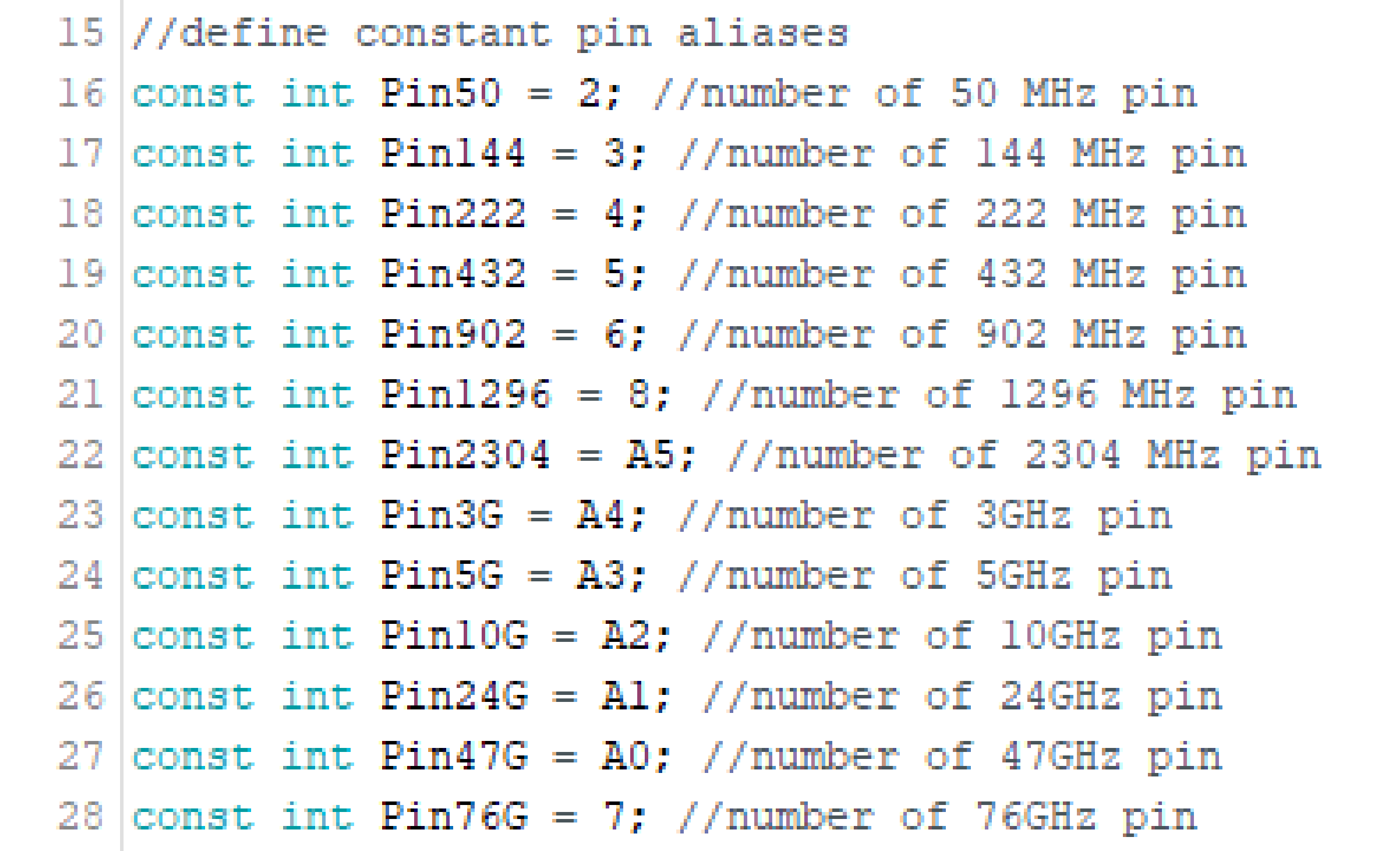
Define **Variables** and Constants



```
//define variables
String commandInputString = "";         // input buffer string to hold incoming data
boolean commandStringComplete = false;  // true when the input string is complete
String command = "";  // incoming data string for parsing

```

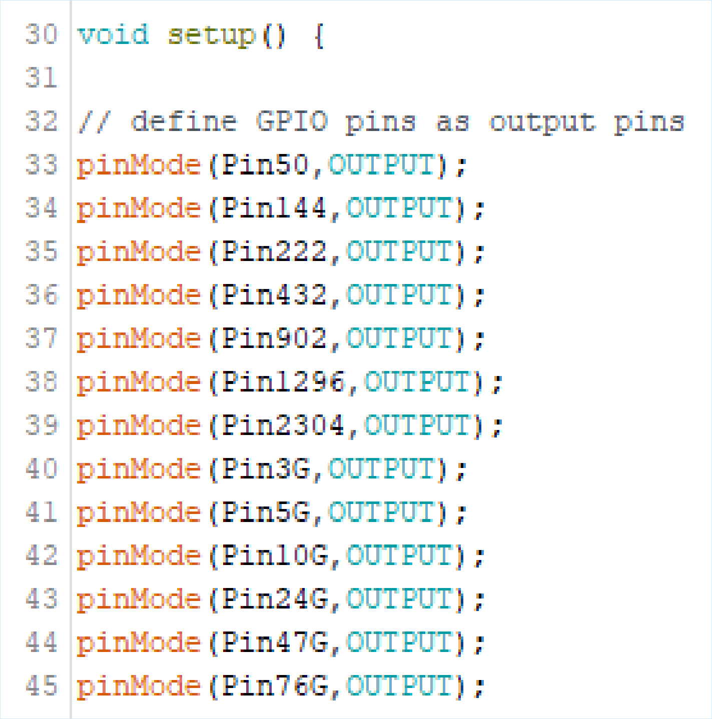
Arduino Example

Define Variables and **Constants**

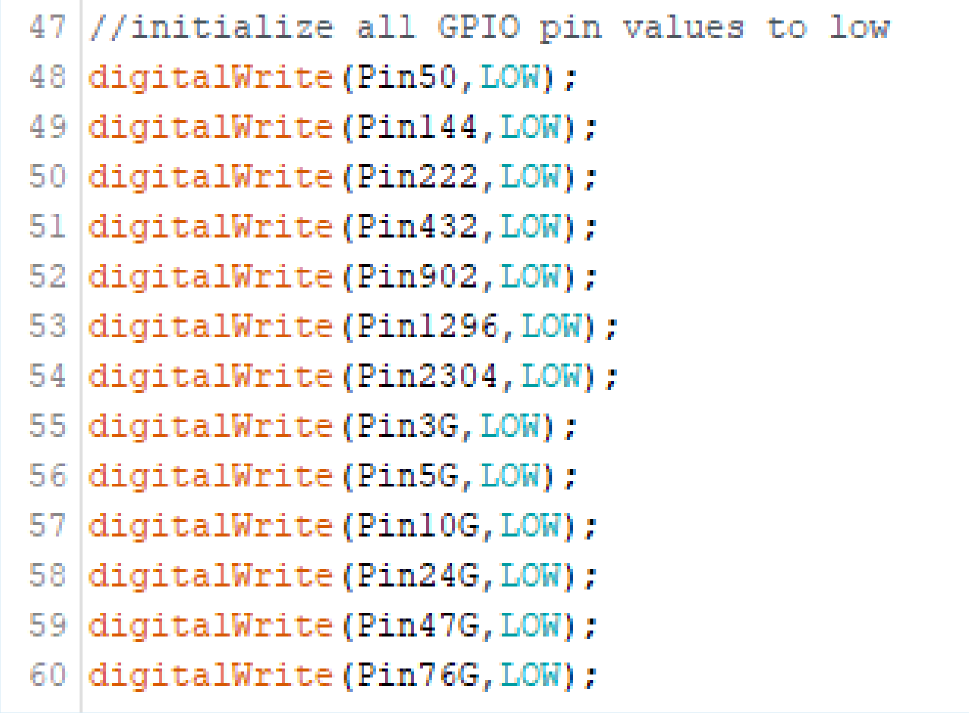


```
//define constant pin aliases
const int Pin50 = 2; //number of 50 MHz pin
const int Pin144 = 3; //number of 144 MHz pin
const int Pin222 = 4; //number of 222 MHz pin
const int Pin432 = 5; //number of 432 MHz pin
const int Pin902 = 6; //number of 902 MHz pin
const int Pin1296 = 8; //number of 1296 MHz pin
const int Pin2304 = A5; //number of 2304 MHz pin
const int Pin3G = A4; //number of 3GHz pin
const int Pin5G = A3; //number of 5GHz pin
const int Pin10G = A2; //number of 10GHz pin
const int Pin24G = A1; //number of 24GHz pin
const int Pin47G = A0; //number of 47GHz pin
const int Pin76G = 7; //number of 76GHz pin
```

Arduino Example
Setup (): Define and Initialize GPIO pins

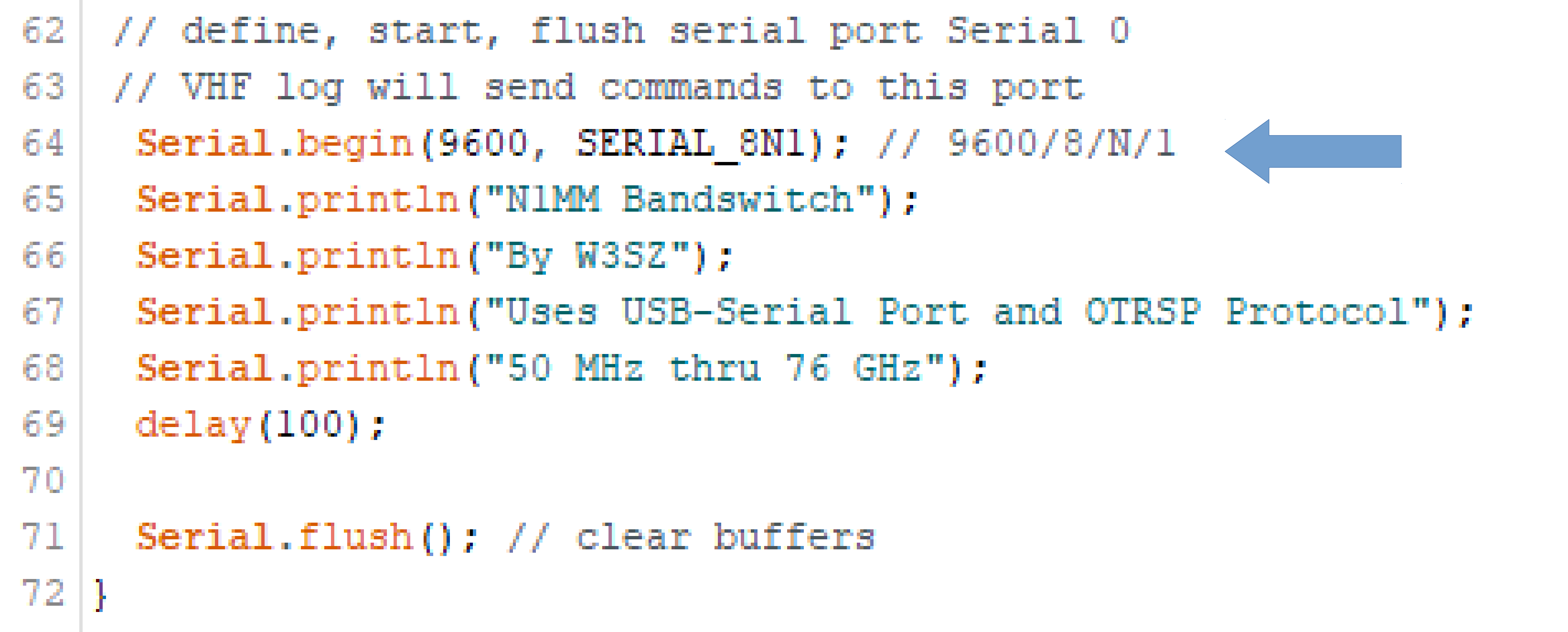


```
void setup() {

// define GPIO pins as output pins
pinMode(Pin50,OUTPUT);
pinMode(Pin144,OUTPUT);
pinMode(Pin222,OUTPUT);
pinMode(Pin432,OUTPUT);
pinMode(Pin902,OUTPUT);
pinMode(Pin1296,OUTPUT);
pinMode(Pin2304,OUTPUT);
pinMode(Pin3G,OUTPUT);
pinMode(Pin5G,OUTPUT);
pinMode(Pin10G,OUTPUT);
pinMode(Pin24G,OUTPUT);
pinMode(Pin47G,OUTPUT);
pinMode(Pin76G,OUTPUT);
```



```
//initialize all GPIO pin values to low
digitalWrite(Pin50,LOW);
digitalWrite(Pin144,LOW);
digitalWrite(Pin222,LOW);
digitalWrite(Pin432,LOW);
digitalWrite(Pin902,LOW);
digitalWrite(Pin1296,LOW);
digitalWrite(Pin2304,LOW);
digitalWrite(Pin3G,LOW);
digitalWrite(Pin5G,LOW);
digitalWrite(Pin10G,LOW);
digitalWrite(Pin24G,LOW);
digitalWrite(Pin47G,LOW);
digitalWrite(Pin76G,LOW);
```

Arduino Example
Setup (): Define, Start, Flush Serial Port



```
 // define, start, flush serial port Serial 0
 // VHF log will send commands to this port
  Serial.begin(9600, SERIAL_8N1); // 9600/8/N/1
  Serial.println("N1MM Bandswitch");
  Serial.println("By W3SZ");
  Serial.println("Uses USB-Serial Port and OTRSP Protocol");
  Serial.println("50 MHz thru 76 GHz");
  delay(100);

  Serial.flush(); // clear buffers
}
```

Serial class is part of the Arduino Language

- No need to add library. Members include:

If(Serial)

available

availableForWrite

begin

end

find

findUntil

flush

parseFloat

parseInt

peek

print

println

read

readBytes

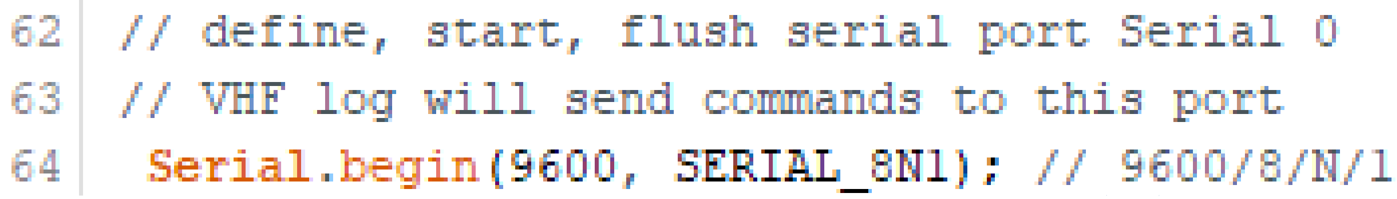
readBytesUntil

setTimeout

write

serialEvent

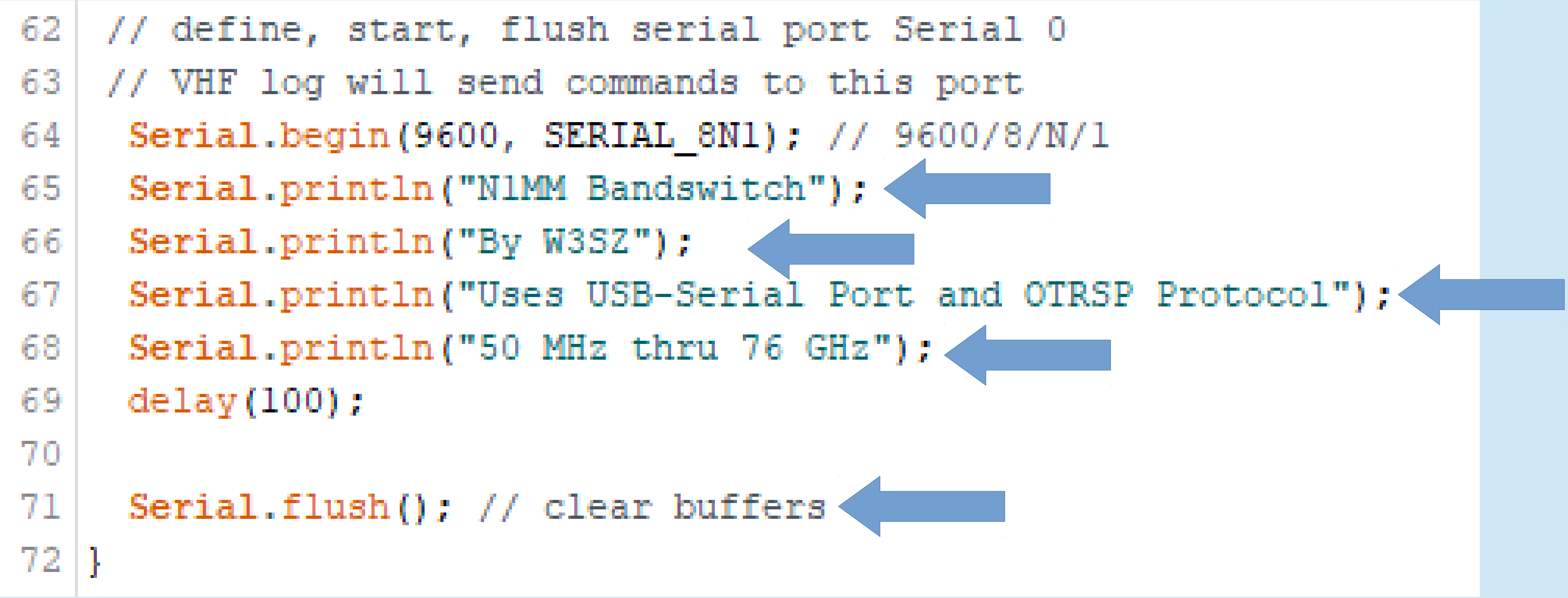
Arduino Example
Setup (): Define, Start, Flush Serial Port



```
// define, start, flush serial port Serial 0
// VHF log will send commands to this port
 Serial.begin(9600, SERIAL_8N1); // 9600/8/N/1
```

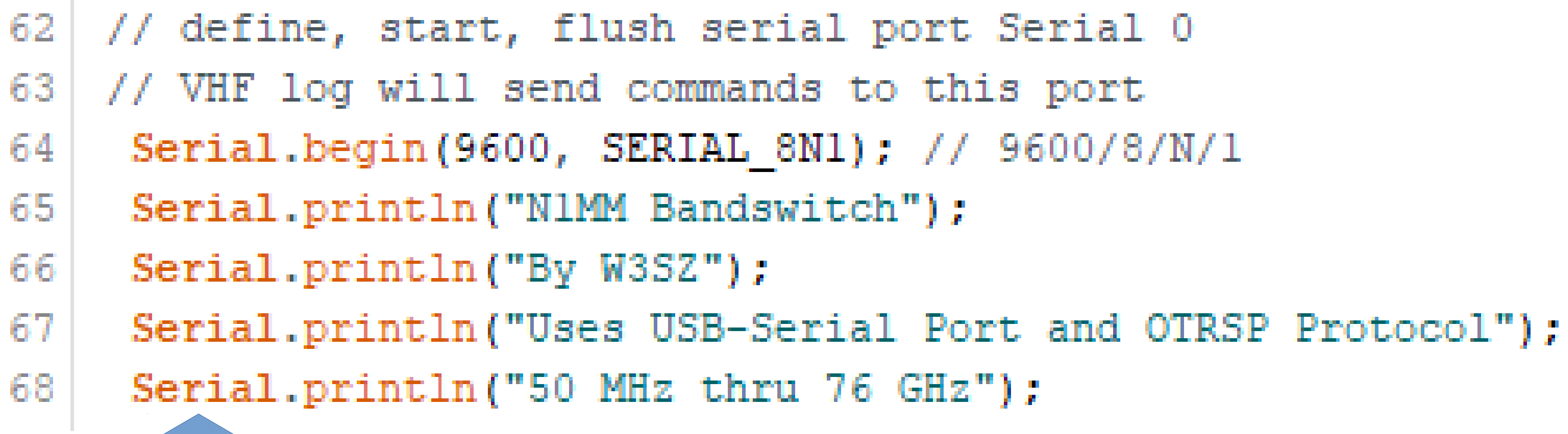
Serial.begin(var1, var2)   var1 sets the data rate in bits per second. An optional second argument var2 configures the data, parity, and stop bits. The default is 8 data bits, no parity, one stop bit. Returns nothing.

Arduino Example
Setup (): Define, Start, Flush Serial Port



```
 // define, start, flush serial port Serial 0
 // VHF log will send commands to this port
  Serial.begin(9600, SERIAL_8N1); // 9600/8/N/1
  Serial.println("N1MM Bandswitch");
  Serial.println("By W3SZ");
  Serial.println("Uses USB-Serial Port and OTRSP Protocol");
  Serial.println("50 MHz thru 76 GHz");
  delay(100);

  Serial.flush(); // clear buffers
}
```

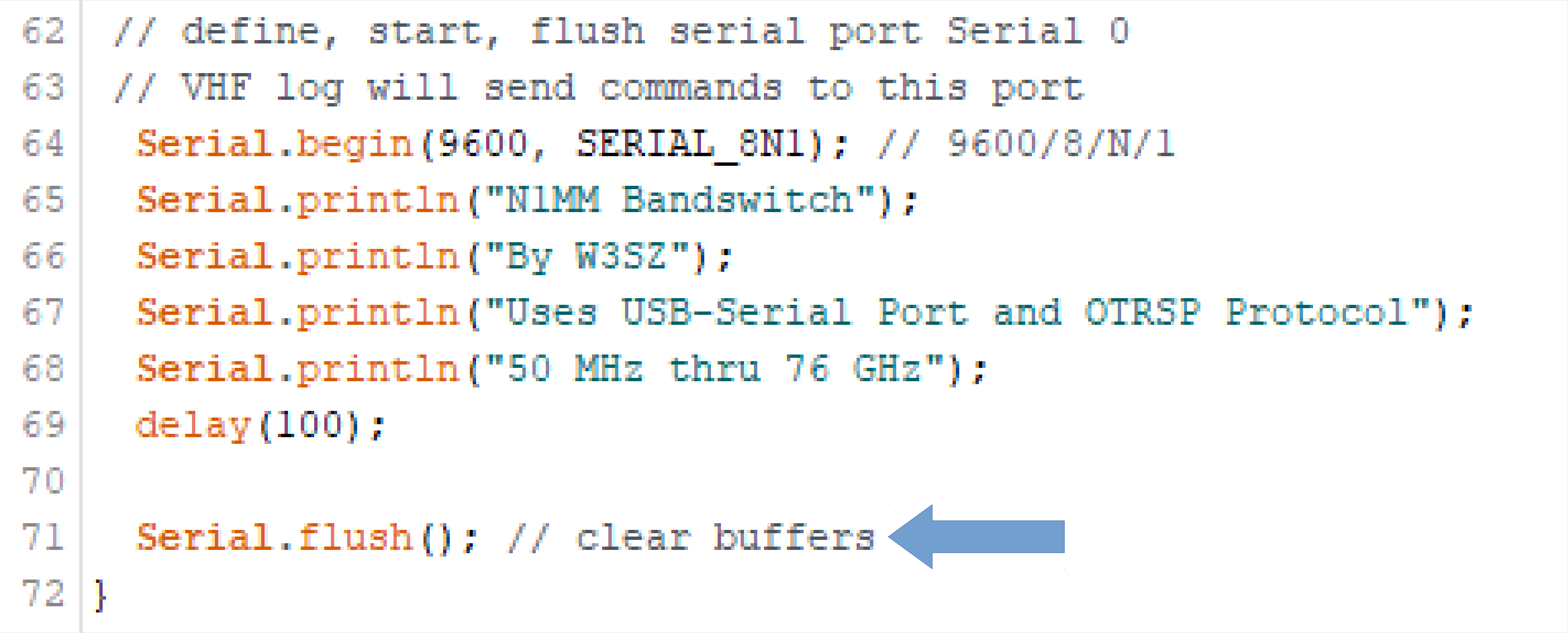
Arduino Example
Setup (): Define, Start, Flush Serial Port



```
// define, start, flush serial port Serial 0
// VHF log will send commands to this port
 Serial.begin(9600, SERIAL_8N1); // 9600/8/N/1
 Serial.println("N1MM Bandswitch");
 Serial.println("By W3SZ");
 Serial.println("Uses USB-Serial Port and OTRSP Protocol");
 Serial.println("50 MHz thru 76 GHz");
```

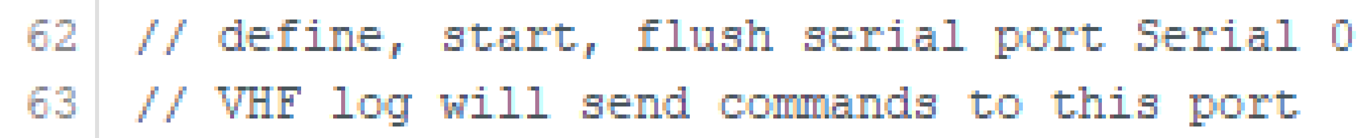
Serial.println(data) Prints data to the serial port as human-readable ASCII text followed by a carriage return character (ASCII 13, or '\r') and a newline character (ASCII 10, or '\n'). Returns the number of bytes written.

Arduino Example
Setup (): Define, Start, Flush Serial Port



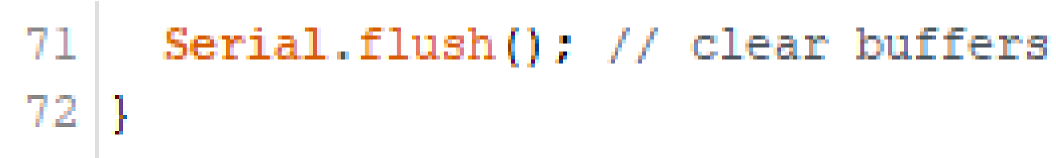
```
 // define, start, flush serial port Serial 0
 // VHF log will send commands to this port
  Serial.begin(9600, SERIAL_8N1); // 9600/8/N/1
  Serial.println("N1MM Bandswitch");
  Serial.println("By W3SZ");
  Serial.println("Uses USB-Serial Port and OTRSP Protocol");
  Serial.println("50 MHz thru 76 GHz");
  delay(100);

  Serial.flush(); // clear buffers
}
```

Arduino Example
Setup (): Define, Start, Flush Serial Port



```
// define, start, flush serial port Serial 0
// VHF log will send commands to this port
```

Serial.flush() Waits for the transmission of outgoing serial data to complete. Returns nothing.



```
  Serial.flush(); // clear buffers
}
```

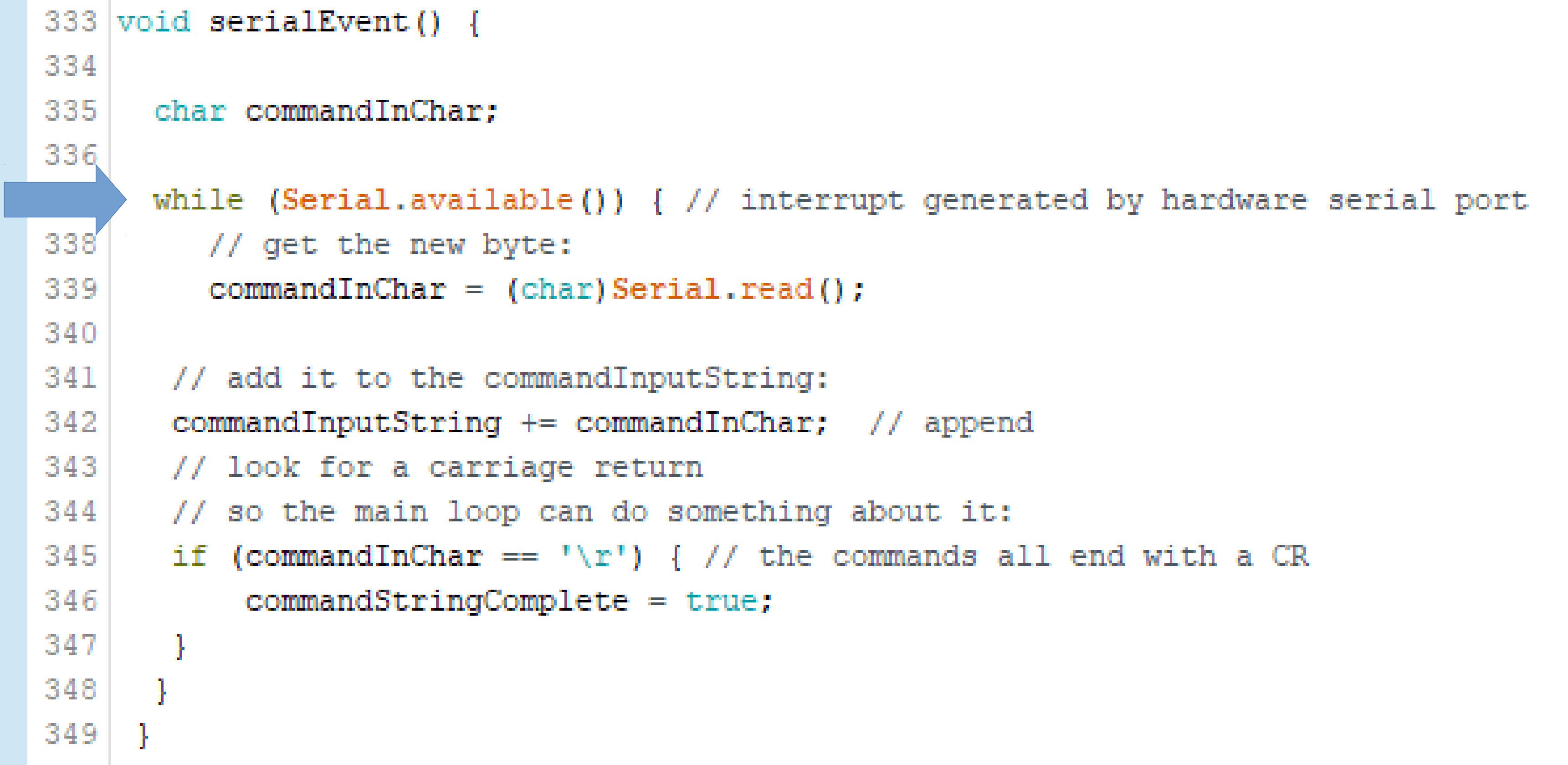
Arduino Example
Loop(): Receive Input from Serial Port
Parse Data to Extract Desired Information
Use Extracted Data to Bandswitch Transverters
Call Other Functions as Needed

- Special built-in function called **serialEvent()** runs at the end of each Loop() if there is new serial data received
- Lets look at this function before examining Loop() itself

Arduino Example

Call Other Functions as Needed

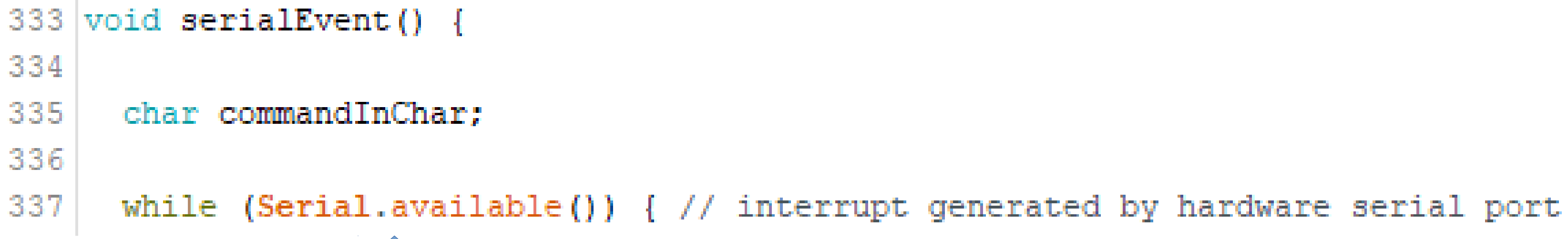
Receive Input from Serial Port



```
void serialEvent() {

  char commandInChar;

  while (Serial.available()) { // interrupt generated by hardware serial port
     // get the new byte:
     commandInChar = (char)Serial.read();

   // add it to the commandInputString:
   commandInputString += commandInChar;  // append
   // look for a carriage return
   // so the main loop can do something about it:
   if (commandInChar == '\r') { // the commands all end with a CR
       commandStringComplete = true;
   }
  }
 }
```

Arduino Example

Call Other Functions as Needed
Receive Input from Serial Port

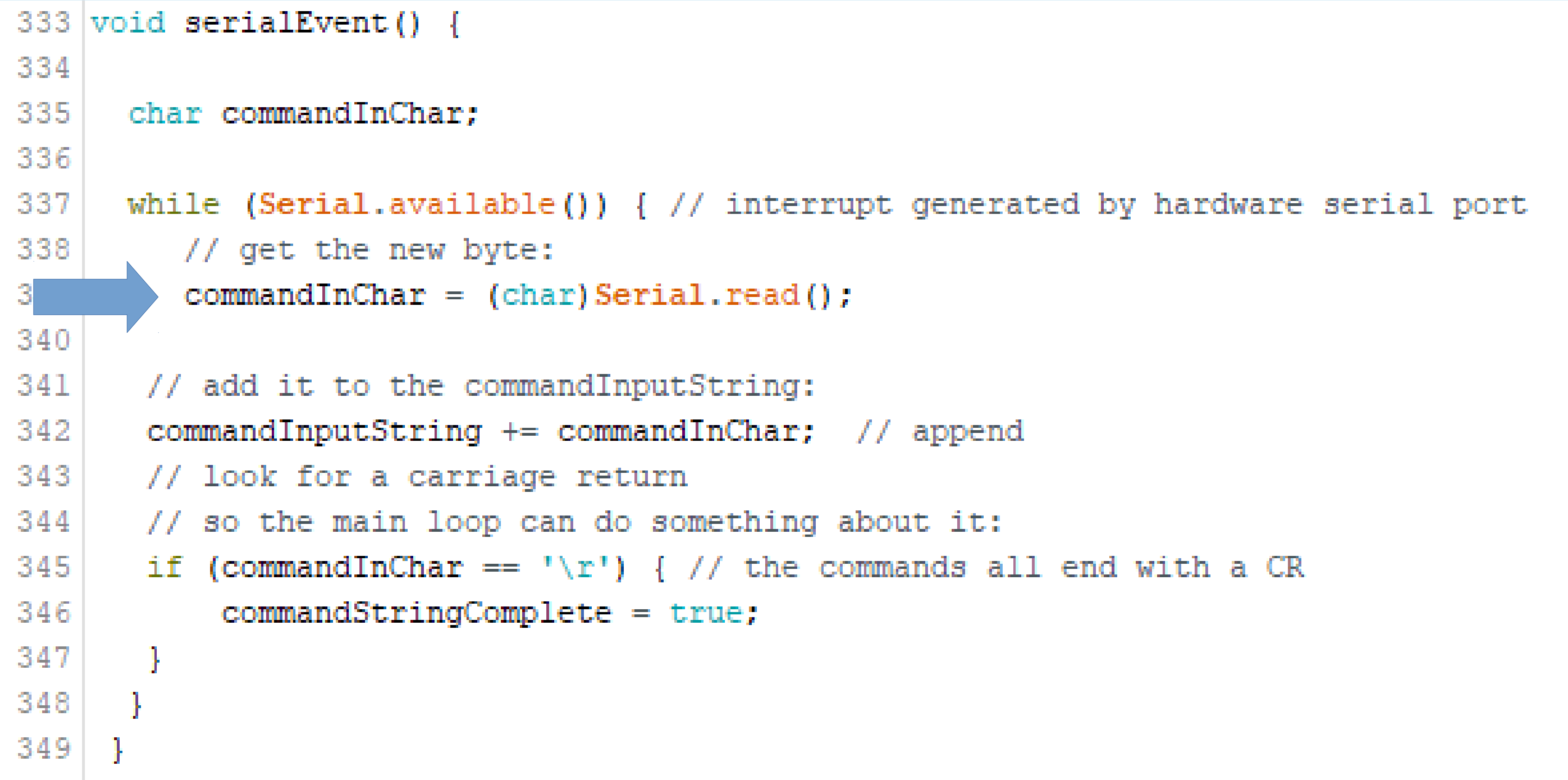


```
void serialEvent() {

  char commandInChar;

  while (Serial.available()) { // interrupt generated by hardware serial port
```

Serial.available()   Gets the number of bytes (characters) available for reading from the serial port. This is data that’s already arrived and stored in the serial receive buffer (which holds 64 bytes). Returns the number of bytes available to read.

Arduino Example

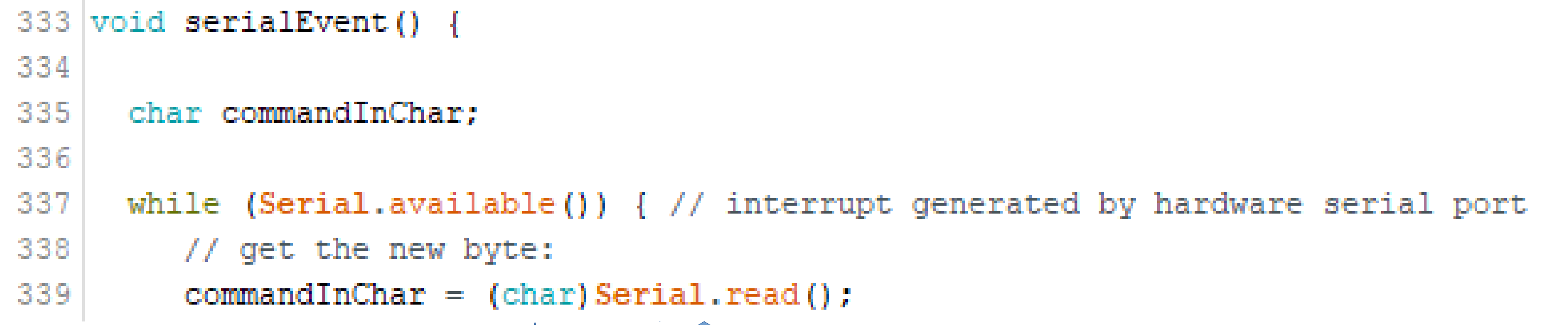
Call Other Functions as Needed
Receive Input from Serial Port



```
void serialEvent() {

  char commandInChar;

  while (Serial.available()) { // interrupt generated by hardware serial port
     // get the new byte:
     commandInChar = (char)Serial.read();

   // add it to the commandInputString:
   commandInputString += commandInChar;  // append
   // look for a carriage return
   // so the main loop can do something about it:
   if (commandInChar == '\r') { // the commands all end with a CR
       commandStringComplete = true;
   }
  }
 }
```

Arduino Example

Call Other Functions as Needed
Receive Input from Serial Port



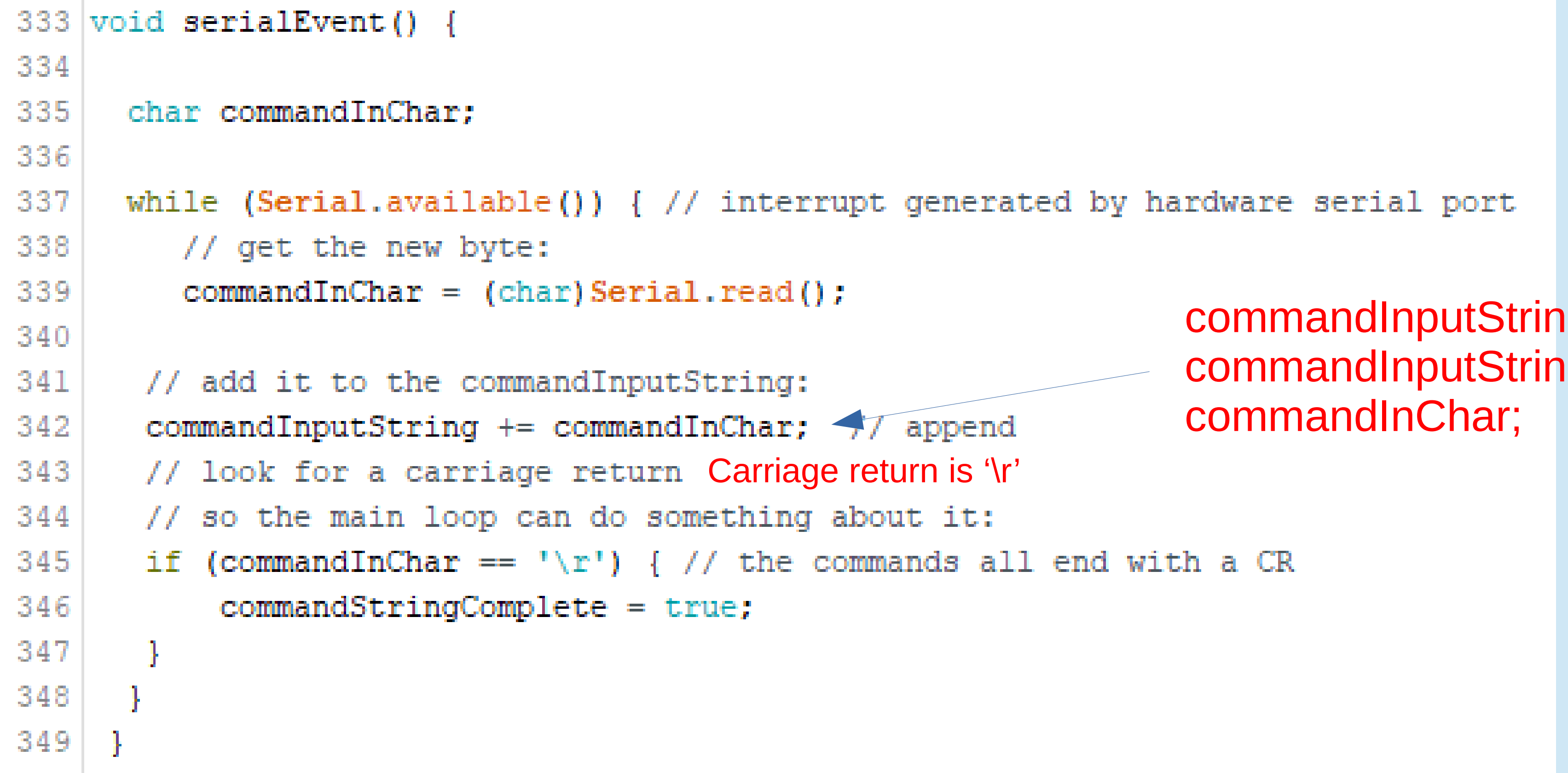
```
void serialEvent() {

  char commandInChar;

  while (Serial.available()) { // interrupt generated by hardware serial port
     // get the new byte:
     commandInChar = (char)Serial.read();
```

Serial.read()   Reads incoming serial data. Returns the first byte of incoming serial data available (or -1 if no data is available)

cast (char)

Arduino Example

Call Other Functions as Needed Receive Input from Serial Port



```
void serialEvent() {

  char commandInChar;

  while (Serial.available()) { // interrupt generated by hardware serial port
     // get the new byte:
     commandInChar = (char)Serial.read();

   // add it to the commandInputString:
   commandInputString += commandInChar;  // append
   // look for a carriage return
   // so the main loop can do something about it:
   if (commandInChar == '\r') { // the commands all end with a CR
       commandStringComplete = true;
   }
  }
 }
```

commandInputString =
commandInputString +
commandInChar;

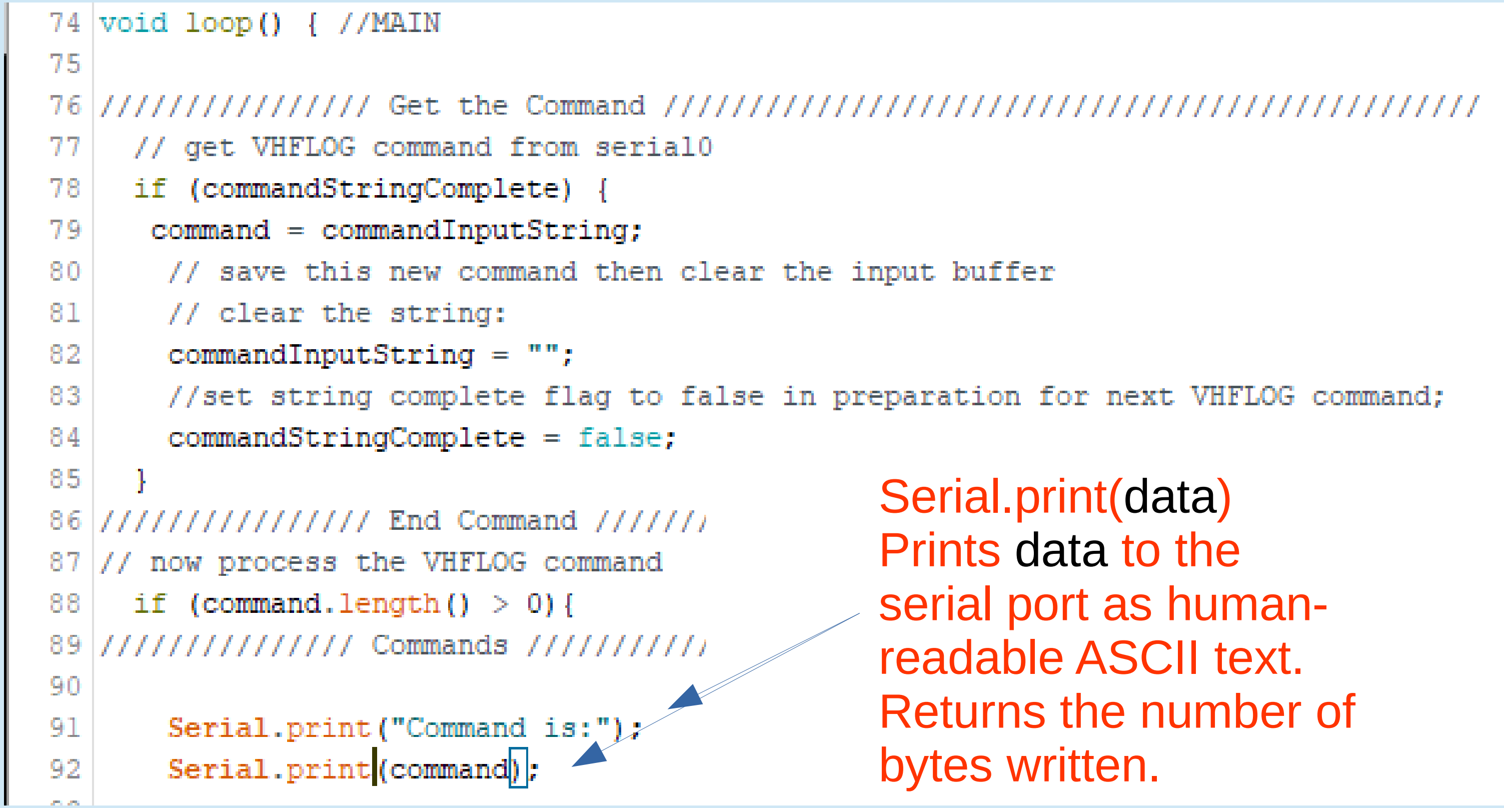
Carriage return is ‘\r’

Arduino Example

Loop(): Receive Input from Serial Port

Parse Data to Extract Desired Information

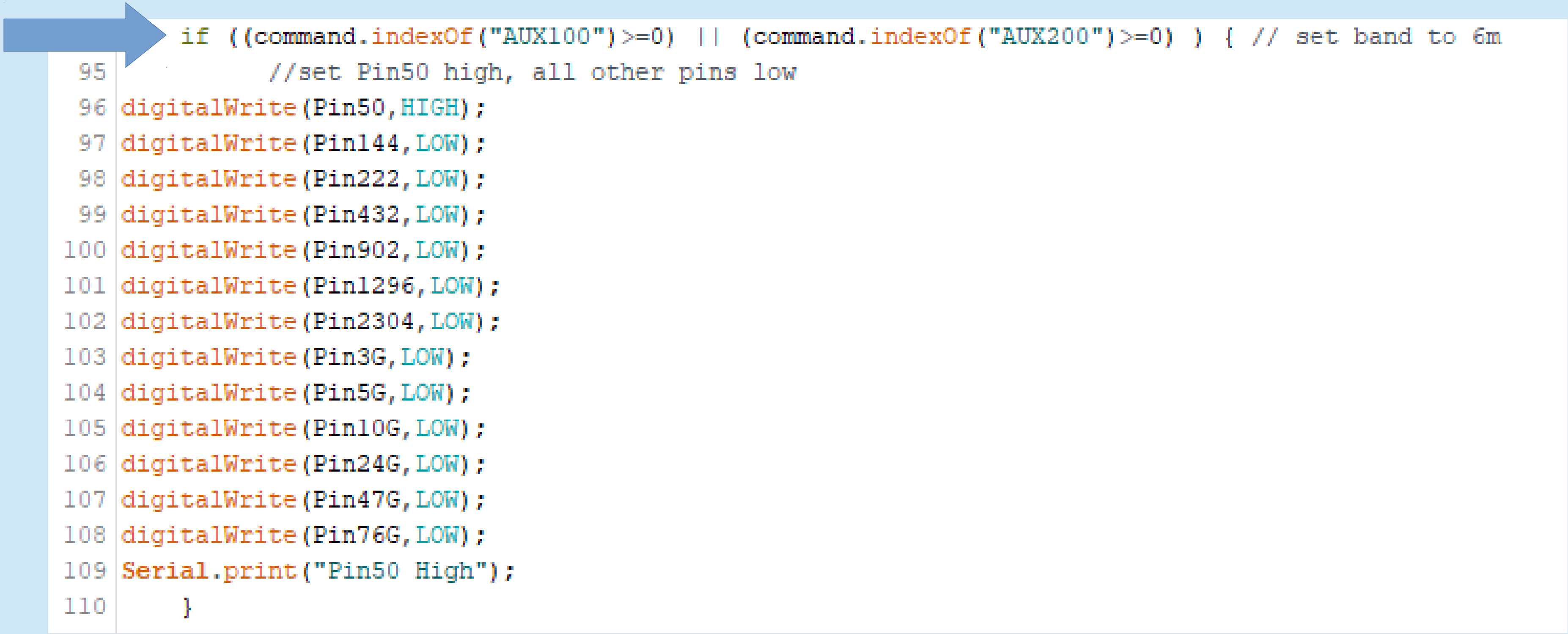
Use Extracted Data to Bandswitch Transverters



```
void loop() { //MAIN

//////////////// Get the Command ////////////////////////////////////////////////
  // get VHFLOG command from serial0
  if (commandStringComplete) {
   command = commandInputString;
    // save this new command then clear the input buffer
    // clear the string:
    commandInputString = "";
    //set string complete flag to false in preparation for next VHFLOG command;
    commandStringComplete = false;
  }
//////////////// End Command ///////
// now process the VHFLOG command
  if (command.length() > 0){
/////////////// Commands ///////////

    Serial.print("Command is:");
    Serial.print(command);
```

Serial.print(data)
Prints data to the serial port as human-readable ASCII text.
Returns the number of bytes written.

Arduino Example
Loop(): Receive Input from Serial Port
Parse Data to Extract Desired Information
Use Extracted Data to Bandswitch Transverters



```
    if ((command.indexOf("AUX100")>=0) || (command.indexOf("AUX200")>=0) ) { // set band to 6m
          //set Pin50 high, all other pins low
digitalWrite(Pin50,HIGH);
digitalWrite(Pin144,LOW);
digitalWrite(Pin222,LOW);
digitalWrite(Pin432,LOW);
digitalWrite(Pin902,LOW);
digitalWrite(Pin1296,LOW);
digitalWrite(Pin2304,LOW);
digitalWrite(Pin3G,LOW);
digitalWrite(Pin5G,LOW);
digitalWrite(Pin10G,LOW);
digitalWrite(Pin24G,LOW);
digitalWrite(Pin47G,LOW);
digitalWrite(Pin76G,LOW);
Serial.print("Pin50 High");
    }
```

Arduino String class

- Members include:

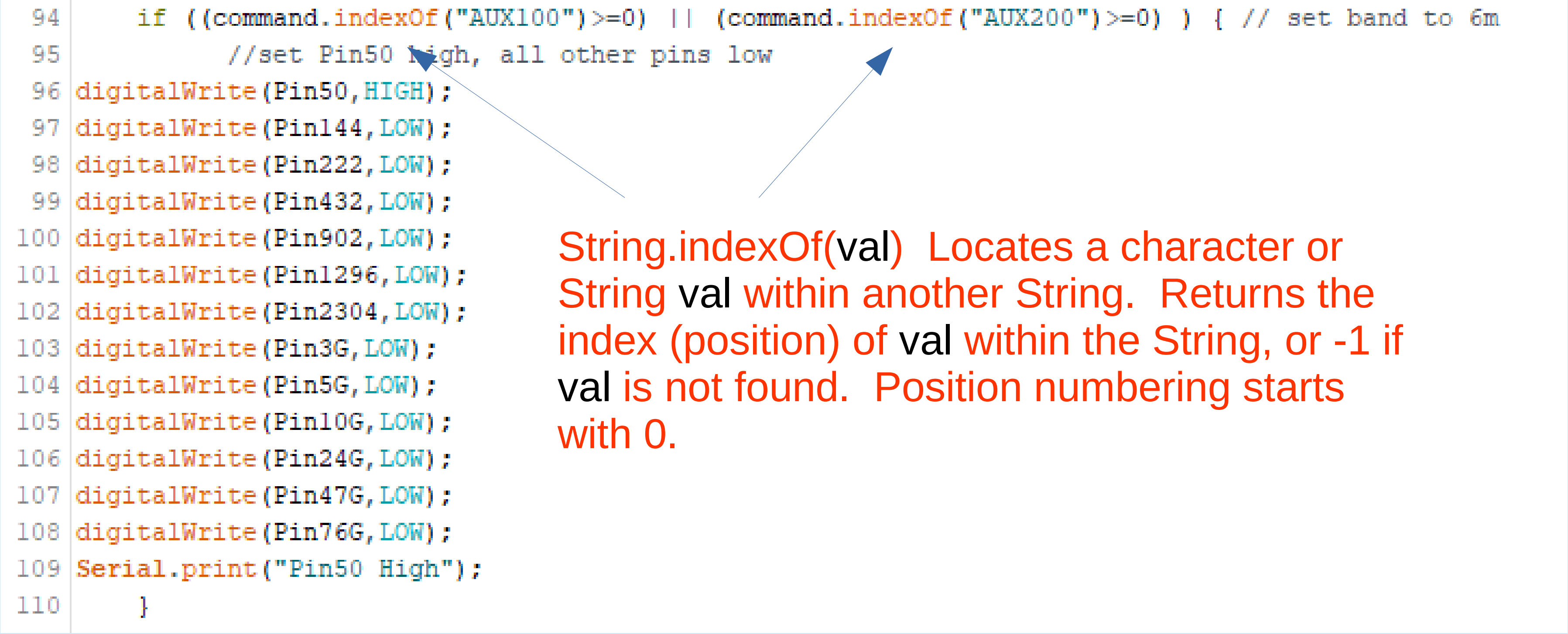
  charAt
  compareTo
  concat
  c_str
  endsWith
  equals
  equalsIgnoreCase
  getBytes
  indexOf
  lastIndexOf
  length
  remove
  replace
  reserve
  setCharAt
  startsWith
  substring
  toCharArray
  toInt
  toFloat
  toLowerCase
  toUpperCase
  trim

Arduino Example

Loop(): Receive Input from Serial Port

Parse Data to Extract Desired Information

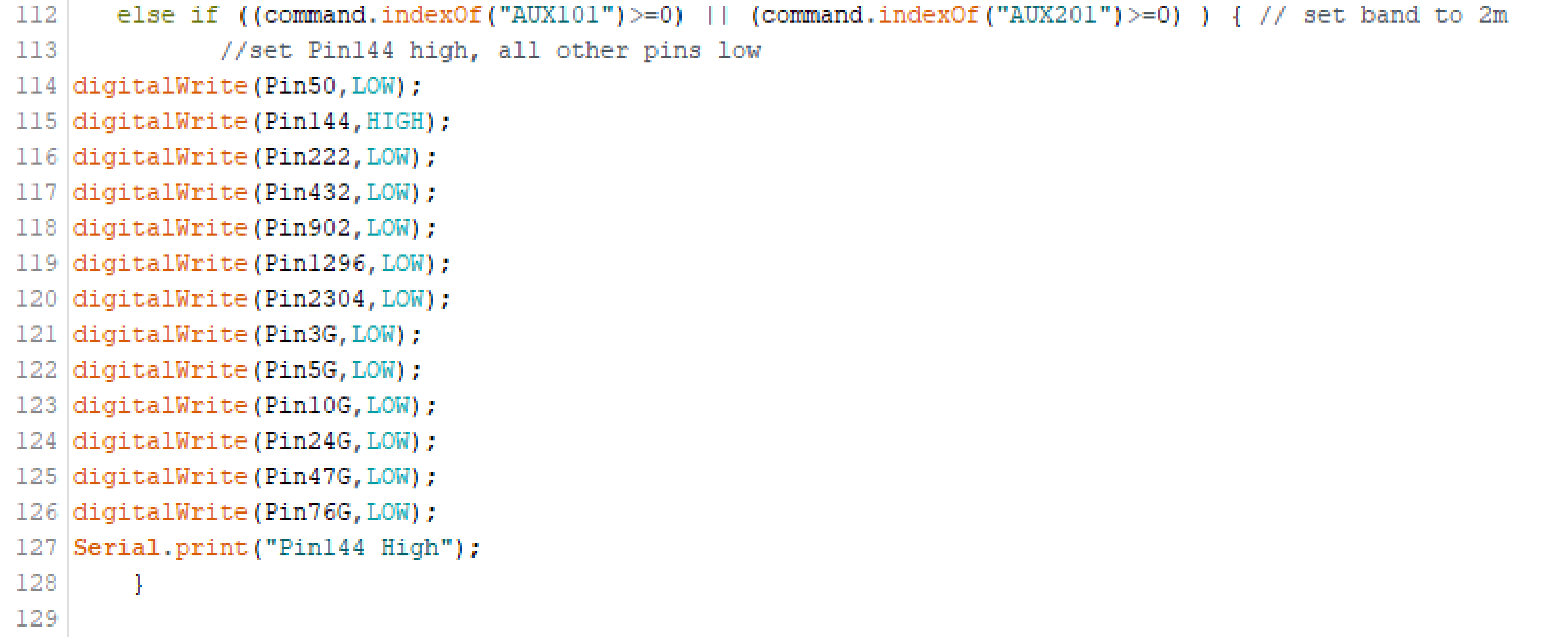
Use Extracted Data to Bandswitch Transverters



```
    if ((command.indexOf("AUX100")>=0) || (command.indexOf("AUX200")>=0) ) { // set band to 6m
          //set Pin50 high, all other pins low
digitalWrite(Pin50,HIGH);
digitalWrite(Pin144,LOW);
digitalWrite(Pin222,LOW);
digitalWrite(Pin432,LOW);
digitalWrite(Pin902,LOW);
digitalWrite(Pin1296,LOW);
digitalWrite(Pin2304,LOW);
digitalWrite(Pin3G,LOW);
digitalWrite(Pin5G,LOW);
digitalWrite(Pin10G,LOW);
digitalWrite(Pin24G,LOW);
digitalWrite(Pin47G,LOW);
digitalWrite(Pin76G,LOW);
Serial.print("Pin50 High");
    }
```

String.indexOf(val) Locates a character or String val within another String. Returns the index (position) of val within the String, or -1 if val is not found. Position numbering starts with 0.

Arduino Example
Loop(): Receive Input from Serial Port
Parse Data to Extract Desired Information
Use Extracted Data to Bandswitch Transverters



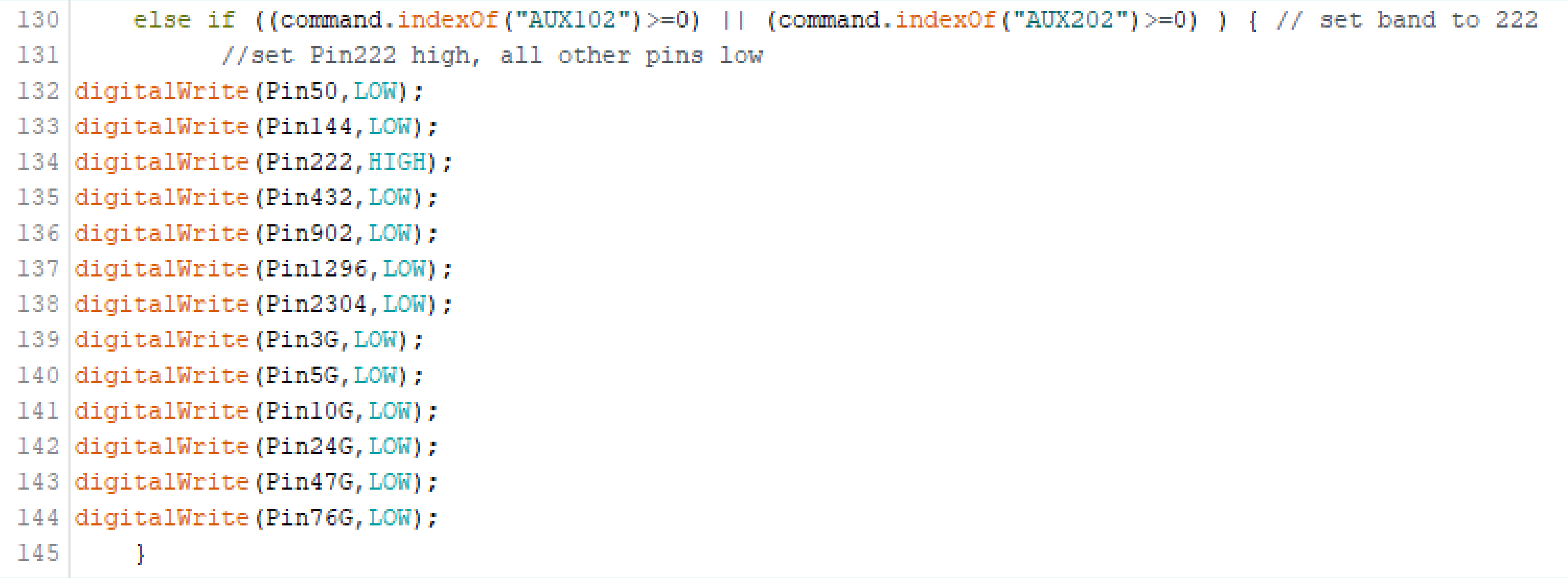
```
   else if ((command.indexOf("AUX101")>=0) || (command.indexOf("AUX201")>=0) ) { // set band to 2m
          //set Pin144 high, all other pins low
digitalWrite(Pin50,LOW);
digitalWrite(Pin144,HIGH);
digitalWrite(Pin222,LOW);
digitalWrite(Pin432,LOW);
digitalWrite(Pin902,LOW);
digitalWrite(Pin1296,LOW);
digitalWrite(Pin2304,LOW);
digitalWrite(Pin3G,LOW);
digitalWrite(Pin5G,LOW);
digitalWrite(Pin10G,LOW);
digitalWrite(Pin24G,LOW);
digitalWrite(Pin47G,LOW);
digitalWrite(Pin76G,LOW);
Serial.print("Pin144 High");
    }

```

Arduino Example

Loop(): Receive Input from Serial Port

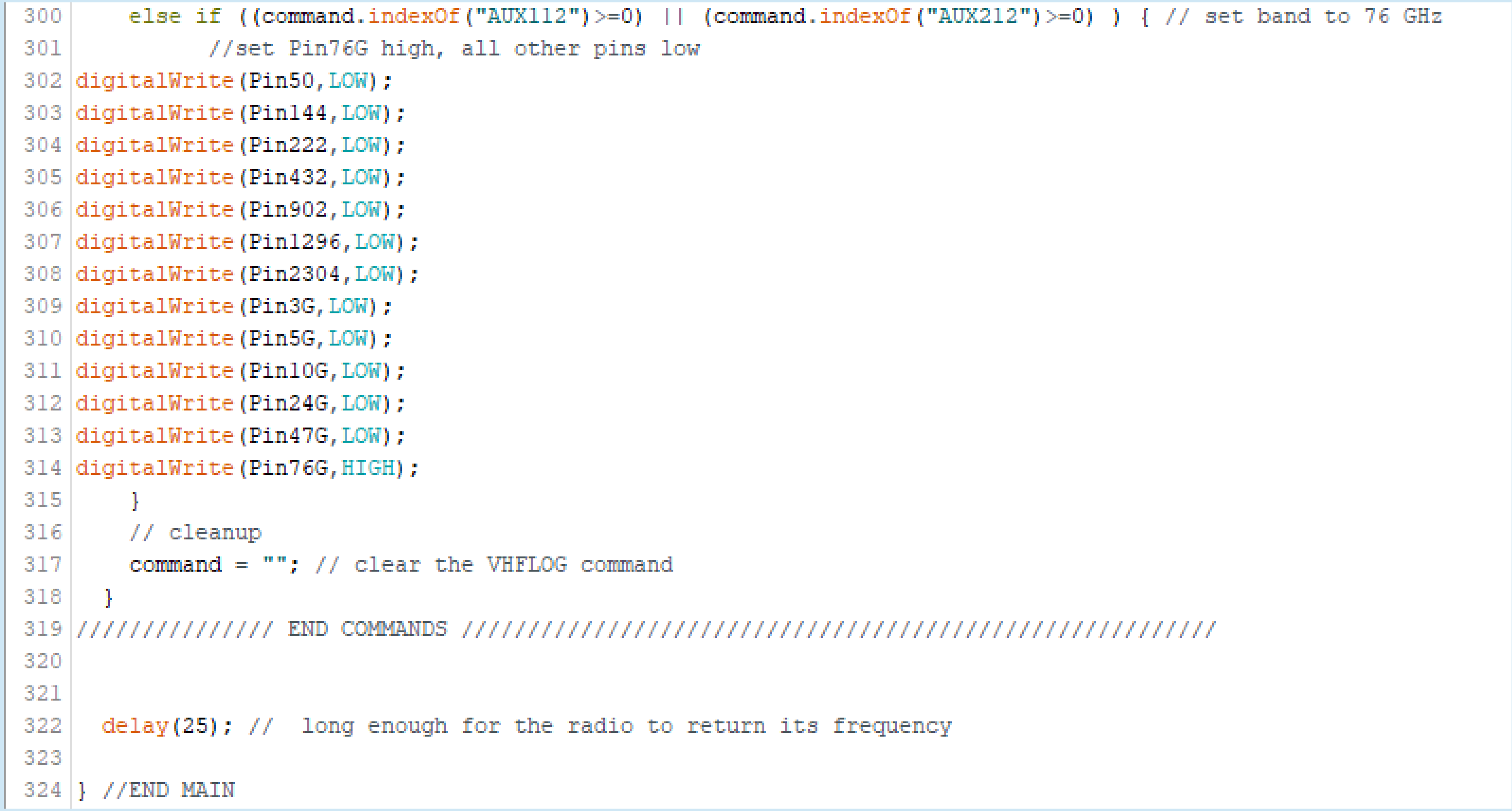
Parse Data to Extract Desired Information

Use Extracted Data to Bandswitch Transverters



```
    else if ((command.indexOf("AUX102")>=0) || (command.indexOf("AUX202")>=0) ) { // set band to 222
          //set Pin222 high, all other pins low
digitalWrite(Pin50,LOW);
digitalWrite(Pin144,LOW);
digitalWrite(Pin222,HIGH);
digitalWrite(Pin432,LOW);
digitalWrite(Pin902,LOW);
digitalWrite(Pin1296,LOW);
digitalWrite(Pin2304,LOW);
digitalWrite(Pin3G,LOW);
digitalWrite(Pin5G,LOW);
digitalWrite(Pin10G,LOW);
digitalWrite(Pin24G,LOW);
digitalWrite(Pin47G,LOW);
digitalWrite(Pin76G,LOW);
    }
```

Arduino Example
Loop(): Receive Input from Serial Port
Parse Data to Extract Desired Information
Use Extracted Data to Bandswitch Transverters

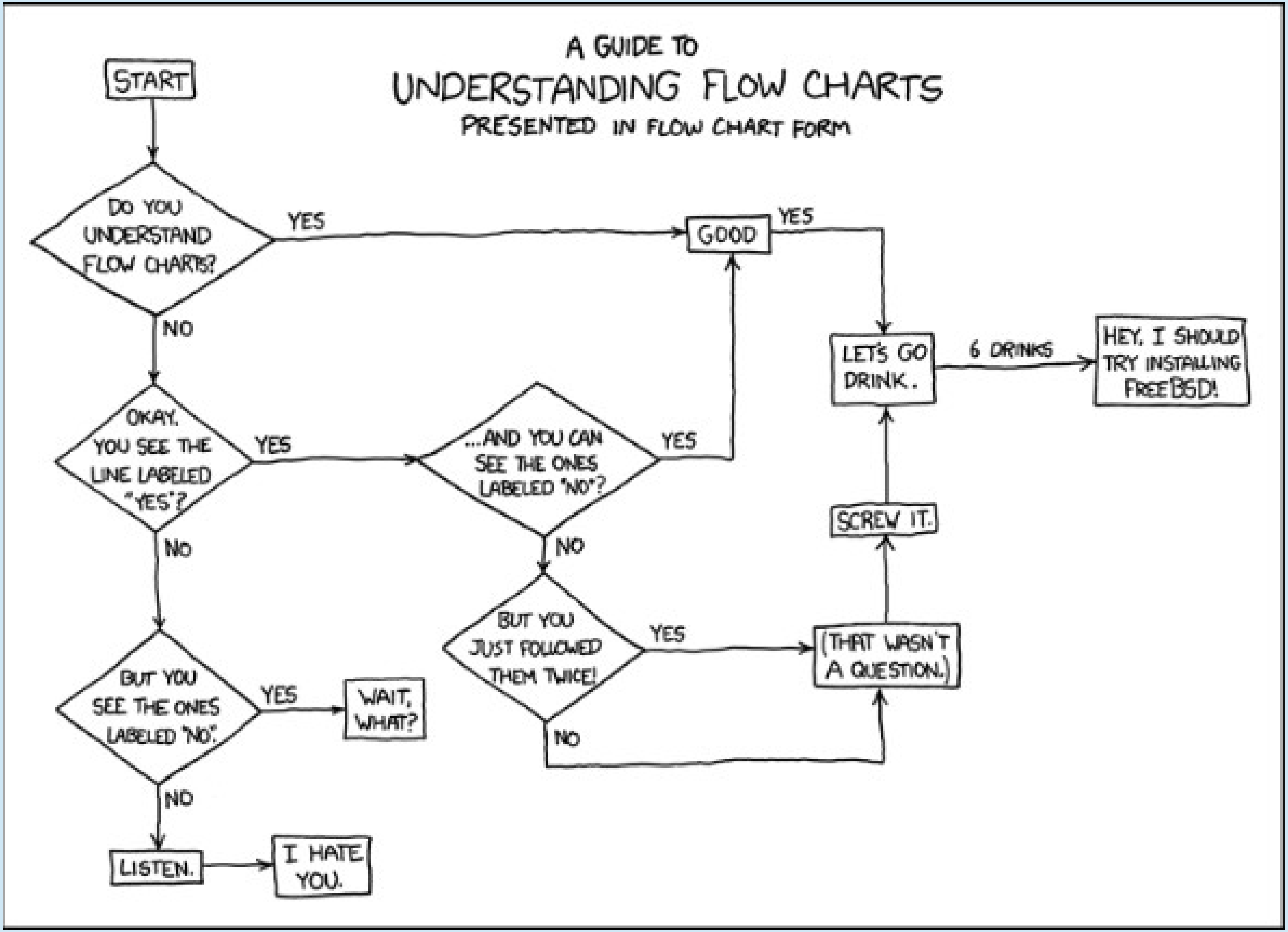


```
    else if ((command.indexOf("AUX112")>=0) || (command.indexOf("AUX212")>=0) ) { // set band to 76 GHz
          //set Pin76G high, all other pins low
digitalWrite(Pin50,LOW);
digitalWrite(Pin144,LOW);
digitalWrite(Pin222,LOW);
digitalWrite(Pin432,LOW);
digitalWrite(Pin902,LOW);
digitalWrite(Pin1296,LOW);
digitalWrite(Pin2304,LOW);
digitalWrite(Pin3G,LOW);
digitalWrite(Pin5G,LOW);
digitalWrite(Pin10G,LOW);
digitalWrite(Pin24G,LOW);
digitalWrite(Pin47G,LOW);
digitalWrite(Pin76G,HIGH);
    }
    // cleanup
    command = ""; // clear the VHFLOG command
  }
/////////////// END COMMANDS /////////////////////////////////////////////////////////


  delay(25); //  long enough for the radio to return its frequency

} //END MAIN
```

Programming Steps

1) Included libraries containing classes with external functions
2) Defined variables and constants
3) Setup ()
   - Defined and initialized GPIO pins
   - Defined, started serial port
4) Loop()
   - Received input from serial port
   - Parsed / processed data to extract desired information
   - Used information derived from data to perform desired task (switch GPIO pins)
5) Called serialEvent() at end of every loop cycle

Questions?


A GUIDE TO
UNDERSTANDING FLOW CHARTS
PRESENTED IN FLOW CHART FORM
START
DO YOU UNDERSTAND FLOW CHARTS?
YES
GOOD
YES
NO
LET'S GO DRINK.
6 DRINKS
HEY, I SHOULD TRY INSTALLING FREEBSD!
OKAY. YOU SEE THE LINE LABELED "YES"?
YES
...AND YOU CAN SEE THE ONES LABELED "NO"?
YES
NO
SCREW IT.
NO
BUT YOU JUST FOLLOWED THEM TWICE!
YES
(THAT WASN'T A QUESTION.)
NO
BUT YOU SEE THE ONES LABELED "NO".
YES
WAIT, WHAT?
NO
LISTEN.
I HATE YOU.