Arduino-to-PicBasicPro Code Translation Examples

Arduino code	PicBasic Pro equivalent
Comments/Remarks: /* A multiple-line comment is bracketed like this */ // Here’ s a single-line comment … code … // end-of-line remark	***Code Structure, Comments/Remarks:*** ‘ A multiple-line comment is ‘ bracketed like this ‘ Here’ s a single-line comment … code … ‘ end-of-line remark

Code Structure:
 	
// Declare global variables
…
 	
// Define and initialize pins and special features
void setup() {
   …
}
 	
// Infinite loop run continuously
void loop() {
   ….
   // Call subroutine/function
   my_function();
   …
}
 	
// Define function
void my_function(void) {
   …
}
	Code Structure:
	
	‘ Declare global variables
	…
	
	‘ Define and initialize pins and special features
	…
	
	‘ Infinite loop run continuously
	Do While (1)
	   ….
	   ‘ Call subroutine/function
	   Gosub my_function
	   …
	Loop
	
	End
	
	‘ Define function
	my_function:
	   …
	Return

Digital Input/Output and Logic:
 	
const int buttonPin = 2;
const int ledPin = 13;
 	
void setup() {
   pinMode(ledPin, OUTPUT);
   pinMode(buttonPin, INPUT);
}
 	
// Turn on the LED if the button is down
if (digitalRead(buttonPin) == HIGH)
   digitalWrite(ledPin, HIGH);
else
   digitalWrite(ledPin, LOW);
	Digital Input/Output and Logic:
	
	buttonPin Var PORTA.3
	ledPin Var PORTB.7
	
	Output ledPin ‘ not required (done by High/Low)
	Input buttonPin ‘ not required (default)
	
	‘ Turn on the LED if the button is down
	If (buttonPin == 1) Then
	    High ledPin
	Else
	   Low ledPin
	Endif

Using a Function or Subroutine:
 	
const int ledPin = 13;
short i;
 	
void setup() {
   pinMode(ledPin, OUTPUT);
}
 	
blink_led();   // blink the LED once
	Using a Function or Subroutine:
	
	ledPin Var PORTB.7
	I Var BYTE
	
	Gosub blink_led ‘ blink the LED once
	
	‘ Blink the LED 5 times
	For I = 1 to 5
	   Gosub blink_led

```
// Blink the LED 5 times
for (i=1; i<=5; i++)
   blink_led();

// Function to blink an LED
void blink_led() {
   digitalWrite(ledPin, HIGH);
   delay(500);
   digitalWrite(ledPin, LOW);
   delay(500);
}
```

```
Next I

‘  Subroutine to blink an LED
blink_led:
   High ledPin
   Pause 500
   Low ledPin
   Pause 500
Return
```

Analog Input and Scaling:

```
int x_raw;
int x_scaled;

x_raw = analogRead(A0);

// Scale raw value from 0-1023 to 0-255 range
x_scaled = map(x_raw, 0, 1023, 0, 255);

// Limit the scaled value to within a range
x_scaled = constrain(x_scaled, 0, 255);
```

***Analog Input and Scaling*:**

```
x_raw Var WORD
x_scaled Var BYTE

Adcin 0, x_raw

‘  Scale raw value from 0-1023 to 0-255 range
x_scaled = x_raw / 4    ‘  scale from 10 bits to 8
```

LCD Output (count-down timer):

```
#include <LiquidCrystal.h>

short i;

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup() {
   lcd.begin (16, 2);
}

// Display count-down from 10 to 0 on the LCD
for (i=10; i>0; i--) {
   /* Clear the lcd and display the count-down
      text and value */
   lcd.clear();
   lcd.print("count down:");
   // move cursor to 2nd row, 2nd col
   lcd.setCursor(1, 1);
   lcd.print(i);
   delay(1000);    // pause for 1 second
}

// Clear LCD
lcd.clear();
```

LCD Output (count-down timer):

```
i Var BYTE

‘  Wait for LCD to power up
Pause 500

‘  Display count-down from 10 to 0 on the LCD
For i = 10 To 0
   ‘  Clear the LCD and display count-down text
   ‘     and value
   Lcdout $FE, 1, "count down: "
   Lcdout $FE, $C0, “  “, Dec i
   Pause 1000;    ‘  pause for 1 second
Next i

‘  Clear the LCD
Lcdout $FE, 1
```

Keypad Serial Interface and Compound Logic:

```
// Define variables
byte key_val;    // button value
byte number;    // value changed by buttons

// Define keypad button codes
const byte key1=0x30;
const byte key2=0x31;
```

Keypad Serial Interface and Compound Logic:

```
‘  Define variables
key_pin  Var  PORTB.0    ‘  input pin
key_val  Var  BYTE    ‘  button value
key_mode  Con  0    ‘  selects 2400 baud
number  Var  BYTE    ‘  value changed by buttons

‘  Define keypad button codes
```

```
const byte key3=0x32;

void setup() {
   /* Initialize serial communication
     (receiving on pin 0) */
   Serial.begin(2400);
}

// Keypad processing loop
while (true) {    // do always (infinite loop)
   // Wait for a keypad button to be pressed
   while (Serial.available() == 0);
   // Read the keypad value from the buffer
   key_val = Serial.read();

   // Perform the appropriate function
   if ((key_val == key1) && (number > 0)) {
        // decrement
        number--;
   }
   else if (key_val == key2) {
        // reset
        number = 0;
   }
   else if ((key_val == key3) && (number < 255)) {
        // increment
        number++;
   }

   // Call a function to process the number
   process_display();
}

// Define function
void process_display (void) {
   …
}
```

```
key1  Con  $30
key2  Con  $31
key3  Con  $32

‘  Keypad processing loop
Do While (1)    ‘  do always (infinite loop)
   ‘  Wait for a keypad button to be pressed
   ‘     and read the value
   Serin key_pin, key_mode, key_val

  ‘  Perform the appropriate function
  If ((key_val = key1) And (number > 0)) Then
      ‘  decrement
      number = number – 1
  ElseIf (key_val = key2) Then
      ‘  reset
      number = 0
  ElseIf ((key_val = key3) And (number < 255)) Then
      ‘  increment
      number = number + 1
  EndIf

   ‘  Call a subroutine to process the number
   Gosub process_display
Loop
End

‘  Define function
process_display:
   …
Return
```

Servo Motor Control:

```
#include <Servo.h>

// Define variables
const int servoPin=9;
const int sensorPin=2;
Servo myServo;
int ang;    // servo angle

// Initialize I/O pins
void setup() {
   pinMode(sensorPin, INPUT);
   myServo.attach(servoPin);
}

/* Continually sweep over the full servo range
      checking a digital sensor every 15 degrees */
void loop() {
   // Start in the 0 degree servo position
   myServo.write(0);
   // Wait for servo to go to (or return to) 0 position
    delay(1000)
```

Servo Motor Control:

```
‘  Servo duty cycle info:
‘  position (degrees) = pulse width (ms) =
‘    duty cycle (%) = Hpwm 0-255 value
‘  0 degrees = 1 ms = 5% = 13
‘  90 degrees = 1.5 ms = 7.5% = 19
‘  180 degrees = 2 ms = 10% = 25
servoFreq  Con  50    ‘  1/(20ms) = 50Hz

‘  Define variables
dutyCycle  Var  WORD
servoPin  Var  PORTB.0  ‘  RB0 (pin 9 set to CCP1)
sensorPin  Var  PORTA.3
duty_cycle  Var  BYTE    ‘  servo angle

‘  Initialize pins
Input sensorPin;
‘  Output servoPin  ‘  not required (done by Hpmw)

‘  Continually sweep over the full servo range
‘     checking a digital sensor every 15 degrees
Do While (1)    ‘  do always (infinite loop)
   ‘  Start in the 0 degree servo position
```

```
    for (ang=15; ang<=180; ang+=15) {
        myServo.write(ang);
        delay(500);  // wait 0.5s for servo to move

        // Read the sensor and react accordingly
        if (digitalRead(sensorPin) == HIGH)
            sensor_react();
    }
}

// Process the sensor detect event
void sensor_react() {
    // Do something here
}
```

```
    Hpwm 1, 13, servoFreq
    ‘  Wait 1s for servo to go to (or return to) 0 position
    Pause 1000

    For dutyCycle = 13 To 25    ‘  step = 1 = 15 deg.
        Hpwm 1, dutyCycle, servoFreq
        ‘  Can use PULSOUT instead to get finer control
        Pause 500  ' wait 0.5s for servo to move

        ‘  Read the sensor and react accordingly
        If (sensorPin) Then Gosub sensorReact
    Next dutyCycle
Loop
End

‘  Process the sensor detect event
sensorReact:
    ‘  Do something here
Return
```

Sending a Song to a Speaker:

```
// Define note pitches (in Hz)
//   (can put in “pitches.h” instead
//    with #include “pitches.h”):
#define NOTE_C  262
#define NOTE_D  294
#define NOTE_E  330
#define NOTE_G  392

// Define variables
const int speakerPin=9;
const int buttonPin=2;
const int n=30;    // number of notes
int i;

// Song notes
int notes[] = { NOTE_E, NOTE_D, NOTE_C,
NOTE_D, NOTE_E, NOTE_E, NOTE_E, 0,
NOTE_D, NOTE_D, NOTE_D, 0,
NOTE_E, NOTE_G, NOTE_G, 0,
NOTE_E, NOTE_D, NOTE_C, NOTE_D,
NOTE_E, NOTE_E, NOTE_E, NOTE_E,
NOTE_D, NOTE_D, NOTE_E, NOTE_D,
NOTE_C, 0 };

// Song note durations (in ms)
int durations[] = { 500, 500, 500,
500, 500, 500, 500, 500,
500, 500, 500, 500,
500, 500, 500, 500,
500, 500, 500, 500,
500, 500, 500, 500,
500, 500, 500, 500,
1500, 500 };

// Initialize I/O pins
void setup() {
    pinMode(speakerPin, OUTPUT);
    pinMode(buttonPin, INPUT);
}
```

Sending a Song to a Speaker:

```
‘  Define note pitches (in Hz)
NOTE_C  Con  262
NOTE_D  Con  294
NOTE_E  Con  330
NOTE_G  Con  392

' Define variables
speakerPin Var  PORTB.0
buttonPin Var  PORTA.3
n  Var  BYTE
i  Var  BYTE

' Song notes
n = 30    ‘  number of notes
notes  Var  WORD[30]
‘ Note – the compact Arraywrite function works
‘    only for BYTE variables
notes[0]=NOTE_E : notes[1]=NOTE_D
notes[2]=NOTE_C : notes[3]=NOTE_D
notes[4]=NOTE_E : notes[5]=NOTE_E
notes[6]=NOTE_E : notes[7]=0
notes[8]=NOTE_D : notes[9]=NOTE_D
notes[10]=NOTE_D : notes[11]=0
notes[12]=NOTE_E : notes[13]=NOTE_G
notes[14]=NOTE_G : notes[15]=0
notes[16]=NOTE_E : notes[17]=NOTE_D
notes[18]=NOTE_C : notes[19]=NOTE_D
notes[20]=NOTE_E : notes[21]=NOTE_E
notes[22]=NOTE_E : notes[23]=NOTE_E
notes[24]=NOTE_D : notes[25]=NOTE_D
notes[26]=NOTE_E : notes[27]=NOTE_D
notes[28]=NOTE_C : notes[29]=0

' Song note durations (in ms)
durations  Var  WORD[30]
durations[0]=500 : durations[1]=500
durations[2]=500 : durations[3]=500
durations[4]=500 : durations[5]=500
durations[6]=500 : durations[7]=500
durations[8]=500 : durations[9]=500
```

```
// Play “Mary Had a Little Lamb” while button down
void loop() {
   if (digitalRead(buttonPin) == HIGH) {
      for (i=0; i<n; i++) {
         tone (speakerPin, notes[i], durations[i]);
         // Add slight pause (50 ms) between notes
         delay (50);
      }
   }
}
```

```
durations[10]=500 : durations[11]=500
durations[12]=500 : durations[13]=500
durations[14]=500 : durations[15]=500
durations[16]=500 : durations[17]=500
durations[18]=500 : durations[19]=500
durations[20]=500 : durations[21]=500
durations[22]=500 : durations[23]=500
durations[24]=500 : durations[25]=500
durations[26]=500 : durations[27]=500
durations[28]=1500 : durations[29]=500

' Initialize I/O pins
‘  Output speakerPin    ‘  not necessary
Input buttonPin

‘  Play “Mary Had a Little Lamb” while button down
Do While (buttonPin)
   For i = 0 To n-1
         Freqout speakerPin, durations[i], notes[i]
         ‘     could use the Sound command instead
         ‘ Add slight pause (50 ms) between notes
         Pause 50;
   Next i
Loop
```

For additional help, compare commands in the Arduino language reference page and the PicBasic Pro manual.