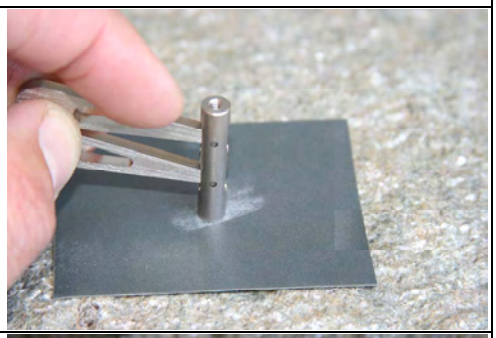
2.3. MECHANICAL TROUBLESHOOTING GUIDE

The Penguin is made from CNC-machined aluminum, molded plastic, screw-machine shoulder bolts, and a variety of electronic components from different factories. There's always the possibility that something will go wrong and parts will bind or move with limited freedom – it's nearly impossible to test every single sub-assembly. For example, tolerances on the shoulder screws and plastic sockets can vary from batch to batch or a shoulder screw bolt could have a slightly larger diameter shaft. It is imperative that your Penguin operate smoothly and without friction to be an effective walker. Contact us if you have any problems. Below are examples of common Penguin assembly difficulties and solutions to them.

Problem	Explanation	Picture
Ball link sockets are tight or binding.	The ball link sockets are molded parts and they may vary in tolerance, though we've chosen a supplier that offers a consistent product. The socket and ball should move freely without any friction when assembled. If you need to loosen up tight-fitting ball links: (a) Use an X-Acto knife to remove any plastic burrs inside the plastic socket. (b) Add a drop of oil into the socket. (c) Reassemble the ball links and check for friction.	
Legs move too much from left to right.	Pressing the left and right legs together should create no more than 1/8" to ¼" of movement (or "slop" as it is called by a machinist). If a particular leg seems to have too much side-to-side play, you can fix this by tightening the shoulder screw down another ¼ turn. If the problem persists, you can gently chamfer the threaded body panel with a drill to sink the shoulder screw further into the body. Be careful if you choose this option because you can remove too much metal and the leg will tighten against the body!	
Ankles are too tight within the feet.	The ankles are the 1¼" long by ¼" diameter round standoffs with two channels machined between. The legs are pinned in these channels. If the round ankle binds inside the foot, slightly shorten the ankle by rubbing the end of it on a piece of sandpaper. Insert the leg assembly back into the foot and check the fit.	
Legs are too tight against the body.	The correct adjustment for the legs depends on the shoulder screws being tightened gently. If the leg is binding against the body, back off the shoulder screws so it can move. Use a dab of Loc-Tite on the threads of the shoulder screw so it stays screwed into position in the body panel.	

3.0 PROGRAMMING THE PENGUIN

3.1. PROGRAMMING CONCEPT

Programming the Penguin to walk is a matter of shifting its center of gravity from side to side with the tilt servo, coordinated with moving the legs forward and backward with the stride servo. The BS2px24's high speed and expanded memory and RAM are useful for storing motion sequences and for performing the computations needed in sensor-based autonomous navigation.

Close the Debug Terminal before downloading a new program, disconnecting the USB cable, or switching off the power to your tethered Penguin! If you fail to close the Debug Terminal before any of these actions, the port may hang up and the BASIC Stamp Editor may freeze when you next try to program your Penguin. If this happens, close the Editor with the Task Manager. Be sure to download the latest version of the BASIC Stamp Editor from www.parallax.com.

3.2. SERVO CALIBRATION

The Penguins servos must be calibrated before the Penguin can walk. This is a crucial step and the Penguin will not walk without it.

1) Run Penguin-ServoCalibration.bpx.
2) Click in the white Transmit area at the top of the Debug Terminal.
3) Press **Enter** to enable the servos.
4) Press **+** and **−** to adjust the tilt servo until the Penguin can stand level.
5) Press the space bar to switch to the stride servo.
6) Press **+** and **−** to adjust the stride servo until the Penguin's feet are in line with each other.
7) Press **S** to save the new settings.
8) Listen for the long beep to ensure that the settings were saved.

Tip: Pressing 1 through 5 changes the coarseness of the adjustments

3.3. PENGUIN'S BASIC STAMP 2PX24 PINOUT

I/O Pin	Description	Direction
P0	Photoresistor – right	Output
P1	Photoresistor – left	Output
P2	Infrared emitter – right	Output
P3	Infrared emitter – left	Output
P4	Infrared receiver	Input
P5	I/O expansion port	Input/Output
P6	Servo stride	Output
P7	Servo tilt	Output
P8	7-segment LED A	Output
P9	7-segment LED B	Output
P10	7-segment LED C	Output
P11	7-segment LED D	Output
P12	Speaker	Output
P13	Hitachi HM55B Compass – CLK (clock)	Output
P14	Hitachi HM55B Compass – CS (chip select)	Output
P15	Hitachi HM55B Compass – DIN/DOUT (data in/out)	Input/Output

3.4. EXAMPLE PROGRAMS

Download the following test programs from the Penguin product page at www.parallax.com and test that each Penguin subsystem is working properly. Complete code listings are also included in this document.

- Penguin-CenterServos.bpx, page 6
- Penguin-SpeakerTest.bpx, page 20
- Penguin-IRTest.bpx, page 20
- Penguin-7SegmentLEDTest.bpx, page 21
- Penguin-PhotoresistorTest.bpx, page 21
- Penguin-CompassTest.bpx, page 22

Once you are familiar with the I/O capabilities of the Penguin, the following programs can be used to demonstrate all of the Penguin's abilities. Download from the Penguin product page at www.parallax.com.

- Penguin-AutonomousNavigation.bpx
- Penguin-MigrateNorth.bpx

3.5. OPTIONAL COMPASS CALIBRATION

To improve the accuracy of the compass, the calibration routines for the Hitachi HM55B Compass Module have been ported to the Penguin. With the calibrated compass routines added to the Penguin's control code, extra RAM and code space is required. Separate versions of the following programs have been created to allow for either more available resources or a more accurate compass.

Original Code	Code Incorporating a Calibrated Compass
Penguin-CompassTest.bpx	Penguin-CalibratedCompassTest.bpx
Penguin-AutonomousNavigation.bpx	Penguin-AutonomousNavigation-CalibratedCompass.bpx
Penguin-MigrateNorth.bpx	Penguin-MigrateNorth-CalibratedCompass.bpx

Before the calibrated compass routines can be used, Penguin-CompassCalibration.bpx must be run to record the compass calibration data. For more information about compass calibration, please read the product documentation and the associated code available from the #29123 product page at www.parallax.com.

3.6. OTHER HELPFUL DOCUMENTATION

The following materials are helpful documents for programming the Penguin; each can be downloaded from its associated product page:

- BASIC Stamp PBASIC Syntax Guide (on-line version of help file installed with the BASIC Stamp Windows Editor)
- HM55B Compass Documentation (#29123)
- HM55B Graphical Viewer for the HM55B (#29123)
- Smart Sensors and Applications (a Stamps in Class Text) (#28029)
- IR Remote for the Boe-Bot Text (a Stamps in Class Text) (#70016)
- Ping))) Ultrasonic Sensor Documentation (#28014)

3.7. TEST CODE LISTINGS

```
' Parallax Penguin Robot
' Penguin-SpeakerTest.bpx

' {$STAMP BS2px}
' {$PBASIC 2.5}

Speaker        PIN     12

i              VAR     Byte
Freq           VAR     Word

'Inverse of 12th root of 2 multiplied by 65536
'Used to divide by one semitone in a 12 note octave
Scale          CON     61858

Freq = 659                                      ' Frequency of an E5
FOR i = 1 TO 12                                 ' 12 semitones = 1 octave
  FREQOUT Speaker, 600, Freq                    ' Play the frequency
  Freq = Freq ** Scale                          ' lower frequency by 1 octave
NEXT
```

```
' Parallax Penguin Robot
' Penguin-IRTest.bpx

' {$STAMP BS2px}
' {$PBASIC 2.5}

REmitter       CON     2
LEmitter       CON     3

IrInput        VAR     IN4

LeftIr         VAR     Bit
RightIr        VAR     Bit

DEBUG CLS                                       ' Prepare debug screen
DEBUG "Infrared sensor status:", CR,
      "Left:", CRSRXY, 15, 1,"Right:"
DO
  GOSUB ReadIr                                  ' Update debug screen with
  DEBUG CRSRXY, 6, 1, DEC LeftIr                ' status of IR sensors
  DEBUG CRSRXY, 22, 1, DEC RightIr
  PAUSE 150
LOOP

ReadIr:
  FREQOUT LEmitter,1,6300                       ' Modulate emitters at 38KHz
  LeftIr = ~IrInput                             ' and check for reflections
  FREQOUT REmitter,1,6300
  RightIr = ~IrInput
RETURN
```

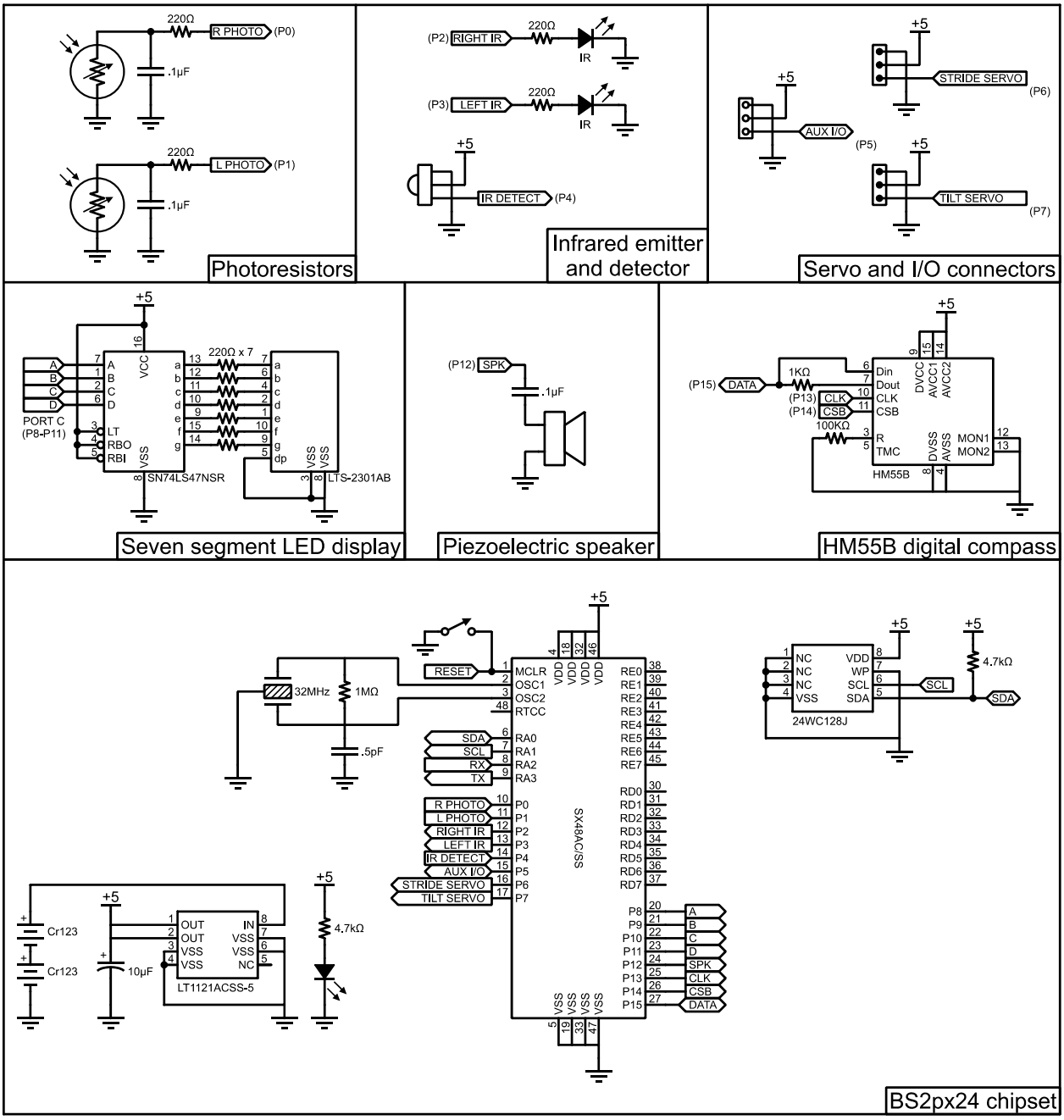
```
' Parallax Penguin Robot
' Penguin-PhotoresistorTest.bpx

' {$STAMP BS2px}
' {$PBASIC 2.5}

REmitter       PIN     2
LEmitter       PIN     3
IrInput        PIN     4

LeftLDR        VAR     Word
RightLDR       VAR     Word

OUTPUT lEmitter                              ' Set IR LEDs as outputs
OUTPUT rEmitter                              ' and the IR receiver as
INPUT  IrInput                               ' an input

HIGH LEmitter                                ' Charge capacitors for
HIGH REmitter                                ' photoresistor R/C circuit

DEBUG CLS                                    ' Prepare debug screen
DEBUG "InfraRed sensor status:", CR,
      "Left:", CRSRXY, 15, 1,"Right:"

DO
  GOSUB READLDR                              ' Update debug screen with
  DEBUG CRSRXY, 6, 1, DEC LeftLDR, "  "      ' status of photoresistors
  DEBUG CRSRXY, 22, 1, DEC RightLDR, "  "
  PAUSE 150
LOOP

ReadLDR:
  RCTIME 0, 1, LeftLDR                       ' Read R/C time for each
  HIGH 0                                     ' photoresitor then recharge
                                             ' their capacitors
  RCTIME 1, 1, RightLDR
  HIGH 1
RETURN
```

```
' Parallax Penguin Robot
' Penguin-7SegmentLEDTest.bpx

' {$STAMP BS2px}
' {$PBASIC 2.5}

LEDDisplay     VAR       OUTC                ' The LED display mirrors OUTC
DIRC = %1111                                 ' Set LED

DO
  FOR LEDDisplay = 0 TO 9                    ' Write directly to LEDDisplay
    PAUSE 750                                ' to change the number displayed
  NEXT
LOOP
```

```
' Parallax Penguin Robot
' Penguin-CompassTest.bpx

' {$STAMP BS2px}
' {$PBASIC 2.5}

Compass        PIN     15                  ' Transceives to/from Din/Dout
Clock          PIN     13                  ' Sends pulses to HM55B's Clock
Enable         PIN     14                  ' Controls HM55B's /Enable

Reset          CON     %0000               ' Reset command for HM55B
Measure        CON     %1000               ' Start measurement command
Report         CON     %1100               ' Get status/axis values command
Ready          CON     %1100               ' 11 -> Done, 00 -> no errors
NegMask        CON     %1111100000000000   ' For 11-bit negative to 16-bits

x              VAR     Word                ' x-axis data
y              VAR     Word                ' y-axis data
Heading        VAR     x                   ' Store Heading measurement
status         VAR     x                   ' Status flags

DEBUG CLS, "Heading:"

DO
  GOSUB ReadCompass
  DEBUG CRSRXY, 9, 0, DEC Heading, "°  "
  PAUSE 150
LOOP

ReadCompass:                               ' Compass module subroutine
  HIGH Enable                              ' Send reset command to HM55B
  LOW Enable
  SHIFTOUT Compass, Clock, MSBFIRST, [Reset\4]

  HIGH Enable                              ' HM55B start measurement command
  LOW Enable
  SHIFTOUT Compass, Clock, MSBFIRST, [Measure\4]

  status = 0                               ' Clear previous status flags

  DO                                       ' Status flag checking loop
    HIGH Enable: LOW Enable                ' Measurement status command
    SHIFTOUT Compass, Clock, MSBFIRST, [Report\4]
    SHIFTIN  Compass, Clock, MSBPOST,  [Status\4]' Get Status
  LOOP UNTIL status = Ready                ' Exit loop when status is ready

  SHIFTIN  Compass,Clock,MSBPOST,[x\11,y\11] ' Get x & y axis values
  HIGH Enable                              ' Disable module

  IF (y.BIT10 = 1) THEN y = y | NegMask    ' Store 11-bits as signed word
  IF (x.BIT10 = 1) THEN x = x | NegMask    ' Repeat for other axis

  Heading = x ATN -y                       ' Convert x and y to brads
  Heading = Heading */ 360                 ' Convert brads to degrees
RETURN
```

4.0 ELECTRONICS

4.1. PENGUIN SCHEMATIC



4.2. CR123 BATTERIES

The Penguin uses CR123 batteries (sometimes also referred to as size "N"). Quality CR123 batteries provide up to one amp-hour of power. CR123 batteries are truly overpriced from readily available sources (drug stores, grocery stores). For example the Energizer and Duracell CR123s are priced around $12.00 per pair.

Parallax has tested all major brands of CR123 batteries (summarized below) and we buy them in bulk. In our tests, we used a fixed current draw of 200 mA at 6V. We measured the time required for the battery voltage to drop below 2.75V. Surprisingly, the tests showed almost no correlation between cost and performance!

The batteries we are selling (Rayovac and Panasonic) have approximately the same capacity as the more expensive ones and they're sold by Parallax at less than half the price of retail outlets! Don't be fooled into paying high prices for CR123 batteries. If you don't buy your CR123 batteries from Parallax then check eBay for the same brand we are selling (usually Rayovac or Panasonic). Pay no more than $2-3 each. The Penguin's current draw during full operation is around about 150 mA. With two Rayovac CR123 batteries it should run for at least six to eight hours.

CR123 Manufacturer	Discharge time
Energizer	270 mins / 4.5 hours
Rayovac	240 mins / 4.0 hours
Duracell	225 mins / 3.8 hours
Panasonic	210 mins / 3.5 hours
Sanyo	150 mins / 2.5 hours
Chinese-label	5 mins / 0.1 hours