# A Micro controller based Furby™ Toy

COMP630
Computer System Design

Prepared by
Juanita D. Heidebrecht
Niagara College of Applied Arts & Technology, Canada

November 27, 2000

Furby™ Toy Project

Table of Contents

## 1.0 Abstract

The project consists of a Furby™ toy whose microprocessors were removed and replaced with PIC16F84 chip on a PIC prototype board.

The toys moves commence a command being received via RS-232 communication or one of the toy's sensors being pressed. The prototype board has a MAX232 driver chip with self-contained charge pump which generates the positive and negative voltages required for the RS-232 interface. The RS-232 specifics are 2400-baud rate; no parity, and one stop bit.

The single-chip PIC micro-controller functions as software UART, receiving a single serial ASCII character that is then interpret as a command for toy's new micro-controller to execute. A shift register is used to take in sensors information form the toy. This shift register is hooked up to the PIC micro-controller where it deciphers the inputs.

The user may view the toys' movements by press pre-defined characters or using the menu features on the front-end c++ software. Also, by pressing sensors on the toy, more movements and wav files are executed. For example, press 't' on the computer keyboard will result in a wav file being played and the character "t" sent to the toy via RS-232 standards. When the "t" is received, the Talking function call will be called and executed; and then the toy will reset in preparation for the next command.

## 2.0 Project Specification

Power Parameters:

Single power supply, 5-6 volt, <250mA

Must output +5 volt, 195-200mA to power the motor

To be hooked up directly to the Furby™ and to the external H-Bridge

Data Input/Output:

RS-232 Serial, 2400 baud, No Parity, 1 Stop, No Flow Control

Shift Register (74HC165): For The Furby™ Sensors

Integrated Circuits Employed:

PIC16F84 single-chip micro controller

MAX232 single-supply RS-232 Driver/Receiver

74HC165 Shift Register

Major Software Functions:

Software UART

User Interface:

Computer Front End: Windows C++ Program. Program consists of a Window with a menu and menu accelerators; each menu item sends a command to the toy to complete a task and the program may play a sound.

Furby(tm) Sensor such as Tummy, Mouth & Back

## 3.0 Concepts & Theory

The Furby™ toy goes thorough a continuous loop waiting for a command. There are two ways in which the toy can receive commands: sensor on the toy being activated or a command received by RS-232 serial communication from the computer.

Each one of the toy's sensors are tied high, therefore each sensor is active low input. There is no de-bounce code to combat mechanical sensors but there is a series of inputs taken and then compared. This eliminates false readings from the sensors.

In the RS-232 serial communications, an ASCII characters (commands) are received and sent via a serial cable that is attached to a COM port at the computer and to a MAX232 chip at the other end. The MAX232 chip converts +-5-15 volt signals into logic 1 (+5 volts) and logic 0 (0 volts) which the PIC then can take in. Characters are received/sent at 2400-baud rate; there is more of a delay and less chance of error compared to 9600-baud rate. The software functions as a UART to assemble received bits into characters. The incoming characters representing a command go through a check to recognize the command and then executed.

The data is received in ASCII, for example the letter "s" is sent as character 0x73.
Possible Parameter types for commands are:

| | | |
|---|---|---|
| 0x52 | R | Reset |
| 0x53 | S | The Surprised look (position) |
| 0x62 | b | Blink eyes motion |
| 0x63 | c | Close Mouth position |
| 0x71 | q | Quite! Closed eyes position |
| 0x72 | r | Reset |
| 0x73 | s | Sleeping motions |
| 0x74 | t | Talking motions |
| 0x77 | w | Wiggle those ears and eyes motion |

Data is also sent to the computer in ASCII. For example, when a sensor is touched such as the tummy, a command is sent to the computer to execute a command to play a wav file.
Possible Parameter types for commands are:

| | | |
|---|---|---|
| 0x54 | T | Tummy Sensor: the command "T" is sent to the "Front End" C++ software to play a wave file |
| 0x46 | f | Feed Sensor: the command "f" |
| 0x42 | B | Back Sensor: the command "B" |

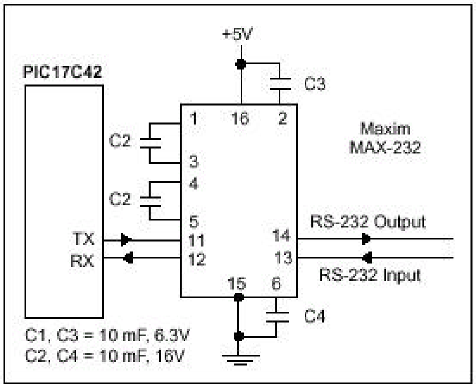
## 4.0 Hardware Design

The prototype was constructed on a PICPROTO board for ease of development. The major components of the system are a Furby™ toy, PIC16F84 single-chip micro controller, a 74HC165 Shift Register, and a MAX232 RS-232 driver/receiver.

### 4.1 MAX232 RS-232 driver/receiver

The MAX232 has two receivers and two drivers; this chip contains a built-in charge-pump that produces both positive and negative 10-volt supplies needed for the drivers. Only one line of each is used for the RS-232 communication in this project.

FIGURE 2: RS-232 INTERFACE TO MAX232



### 4.2 Shift Register

74HC165 is a CMOS based shift register: parallel in and serial out. Eight individual data input lines (parallel) are taken in serially (through one input line to the PIC), where it can be clocked as needed. Another line is needed to the PIC to clock data in.

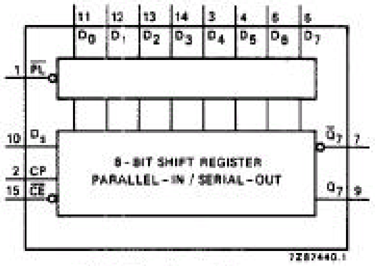
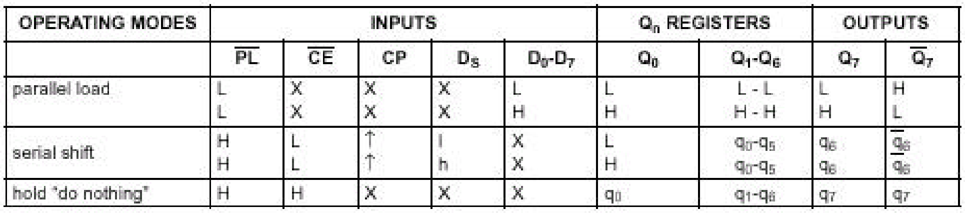


Fig.4 Functional diagram.

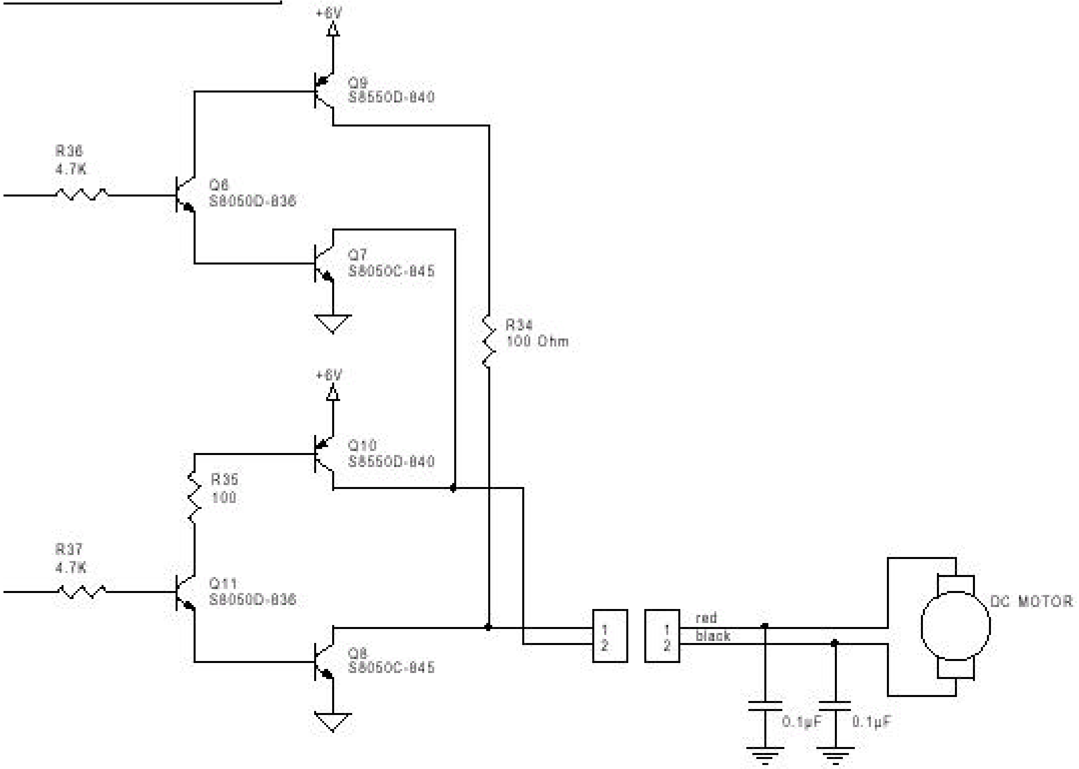
FUNCTION TABLE



| OPERATING MODES | INPUTS | | | | | $Q_n$ REGISTERS | | OUTPUTS | |
|---|---|---|---|---|---|---|---|---|---|
| | $\overline{PL}$ | $\overline{CE}$ | CP | $D_S$ | $D_0$-$D_7$ | $Q_0$ | $Q_1$-$Q_6$ | $Q_7$ | $\overline{Q}_7$ |
| parallel load | L<br>L | X<br>X | X<br>X | X<br>X | L<br>H | L<br>H | L - L<br>H - H | L<br>H | H<br>L |
| serial shift | H<br>H | L<br>L | ↑<br>↑ | l<br>h | X<br>X | L<br>H | $q_0$-$q_5$<br>$q_0$-$q_5$ | $q_6$<br>$q_6$ | $\overline{q}_6$<br>$\overline{q}_6$ |
| hold "do nothing" | H | H | X | X | X | $q_0$ | $q_1$-$q_6$ | $q_7$ | $\overline{q}_7$ |

## 4.3 H-Bridge

An external H-Bridge is not needed unless the H-Bridge on the toy gets ruin somehow. In that event, here is a schematic of a H-bridge to assemble together.



Make Q6 & Q11 2N3904 (NPN), the other NPN transistors (Q8 & Q7) to Darlington transistors TIP120 and the other PNP transistors (Q10 & Q9) to Darlington transistors TIP125. The Darlington transistors can provide current up to 3 A.

## 5.0 Firmware Design

### 5.1 Summary of Software Operation

The main loop of the program continually checks for inputs from the shift register and checks for a start bit present on the serial line.  The shift register takes in inputs from the toy's sensors such as the tummy sensor, back sensor and the feed sensor.  Eight-bit input from the shift register is taken in three times, the start bit from the serial line is checked between each of the three shift register checks.  After the third time the shift register is checked, the three sets of shift register inputs are compared for three consecutive sensor detections.  For example, if the feed input bit is low in each set of inputs then the feed sensor was for sure touched and eliminates errors such as de-bounce.

If a start bit is detected at the serial receive input, the code jumps out of the loop into the software UART routine to assemble received bits into an ASCII character.  After the character is received, the software jumps into a command check routine that is really assemble style if and else statements.

When in the command checking routine, if a command is recognized to be a command; it then jumps out of the routine to the specific command routine.  For example, when the character "s" is received, it is recognized for the command for sleeping.  The software jumps to the sleeping routine where it executes a series of movement commands.

### 5.2 Major Program Features:

#### 5.2.1   Software UART

The software UART is used to assemble incoming bits into a character.  To detect a start bit, bit something of Port B is checked with a bit test instruction as follows:

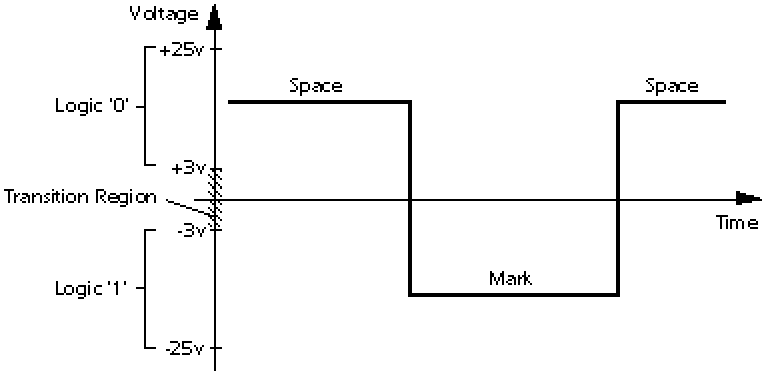
```
btfsc   PORTB,1         ;if line is low, start bit is present
goto    ContinueOn      ;received a high: No start bit yet, re- check
btfss   PORTB,1         ;recieved a low, checks again
btfsc   PORTB,1         ;a low was for sure received and now falls
                         ;through to the delay call for the start bit
goto    ContinueOn
call    StartBitDelay   ;Have to waite 1.5 times the cycle for
                         ;the start
```

Just like the shift register, data is taken in as it is clocked but the clocking is done by a baud rate. The software is designed to take in and send data at a baud rate of 2400 (1/2400th of a second).  As the bits are taken in, it is assembled into an eight-bit ASCII character.

```
btfsc   PORTB, 1                          ;waiting for the start of the next bit
bsf     STATUS,C                          ;set the next bit (1)
btfss   PORTB, 1
bcf     STATUS,C                         ;clear the next bit (0)
```

```
        rrf     ReceivedCommandByte,f          ;shift all the bits to the right
        incf    ReceiveBites,f                 ;increment the bit counter

        call    ReceiveDelay                   ;need 104u second delay between bits
```



When logic 1 is detected by the PIC, it originally came into the MAX232 chip anywhere from –3 to –25 volts and logic 0 was anywhere from +3 to +25 volts. The MAX232 converts these signals into understandable logic levels.

### 5.2.2 Motor Movement

There is a Forward and Reverse routine. Each one makes sure that only one bit is set on at any one time. There are two lines from the PIC that controls the motor and both lines should not be set on at the same time. There is also a stop motor routine that makes sure that both bits are set off.

```
KeepLookingForGEAR_Forward
                call    FORWARD
                call    ShortDelay
                btfss   PORTB,6                        ;Check the Gear
                goto    KeepLookingForGEAR_Forward
```

The Forward Motion and Reverse Motion routine controls how long the motor stays on. A variable named something is set to hold the number of times the gears are to rotate around. Actually the gears go around ten times before it is counted as a gear rotation bundle.

### 5.2.3 Data Acquisition

Besides the UART, the other way that data is acquired is thorough the shift register. Only three lines are needed: one line to take in the data, one to enable and disable the sift register and the other line to clock the shift register. First, enable the shift register and, clock the shift register by creating a trigger effect: set the line one then off right away. Now, check the input line/third line for a high or a low. Take it the data and shift the bits to the right within the input variable (Temp).

```
        bsf     PORTA,0         ;Enable the Shift Register
NextInputBit
```

```
bsf     PORTA,1         ;Create the trigger (clocking)
bcf     PORTA,1

btfsc   PORTA,3         ;waiting for the start of the next bit
bsf     STATUS,C        ;set the next bit (1)
btfss   PORTA,3
bcf     STATUS,C        ;clear the next bit (0)

rrf     Temp,f          ;shift all the bits to the right
```

## 6.0 Performance Tests

1) RS-232 Receive Tests

An incoming ASCII character is received and interpreted as a command

1.1) A command routine is called
1.2) The command routine designates the number of gears to rotate
1.3) Then calls either the Forward or Backwards routines
1.4) Visual movement of the Furby™ appears

2) RS-232 Send and Shift register test

A sensor is activated and an ASCII character is sent as a command

1.1) Press one of the toy's sensors and an ASCII character is sent to the computer and the wiggle eyes command is executed
1.2) Visual evidence of the ears moving appears
1.3) Hear audio evidence of a wave file from the computer

## 7.0 Conclusion

The PIC micro-controlled Furby™ toy as described thus far is pictured in the following pages and was “Kicking Awesome” project to do.

This project has no practical purpose, other than to prove that it is possible to remove the microprocessors already there and have them replaced with a PIC16F84 chip where the toy can be “reprogrammed” to do the same things but when the user wants them to happen.

## 7.1 Improvements

No improvements are planned. This project sits as is for now. However, all source code and resources will be available for anyone to make improvements and changes. There is room for many changes. Since the original source code only took up 428 words out of the 1000 words possible on the PIC16F84 chip; many changes and added features are possible. AND, are encouraged.

Some changes/add features that could be accomplished:

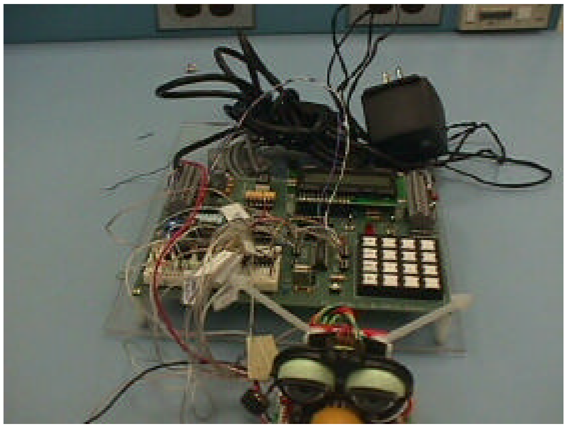
1. After a preset amount of time, if none of the sensors or a command has been received then an event could happen. An event, such as the toy making a movement and a wav file played.
2. Interfacing with the LPC speech processor the original Furby™ allowing the toy itself to “speak”
3. Hooking up the speakers on the toy to the audio output lines on the computer
4. The front end c++ software could be enhanced with active bitmaps

## 8.0 Appendices

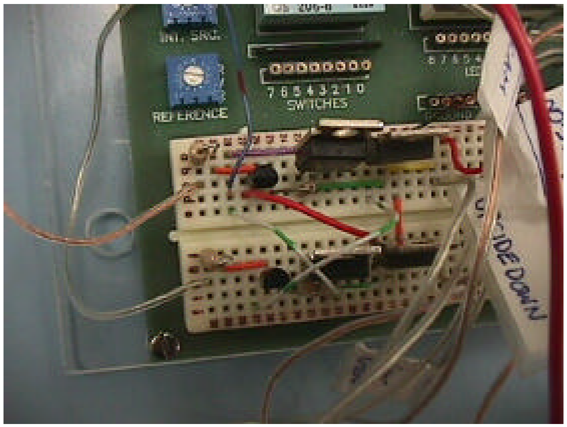
The appendices are as follows:

8.1 Photographs the prototype Furby™ toy and board
8.2 Schematic Diagram
8.3 Flow Chart
8.4 ASM Code Listing
8.5 Front-end C++ Program

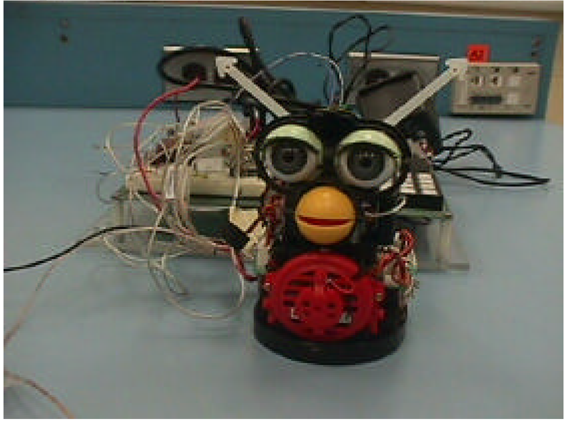
## 8.1 Photographs the prototype Furby™ toy and board



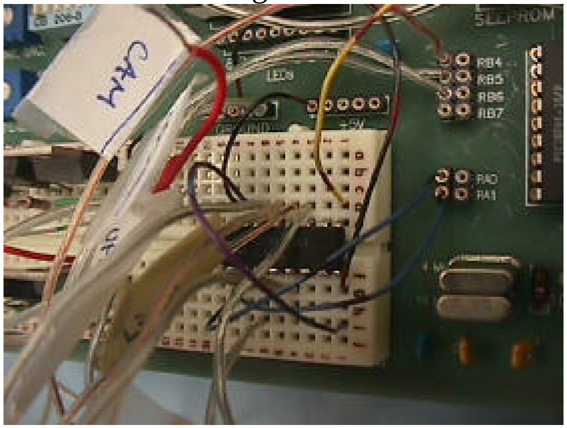
Top View



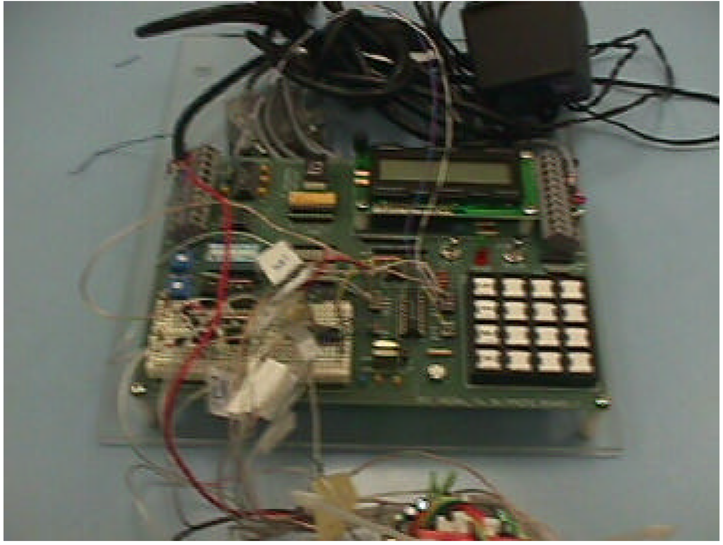
View of the H-Bridge



Front View



View of the Shift Register



Closer look

## 8.2 Schematic Diagram

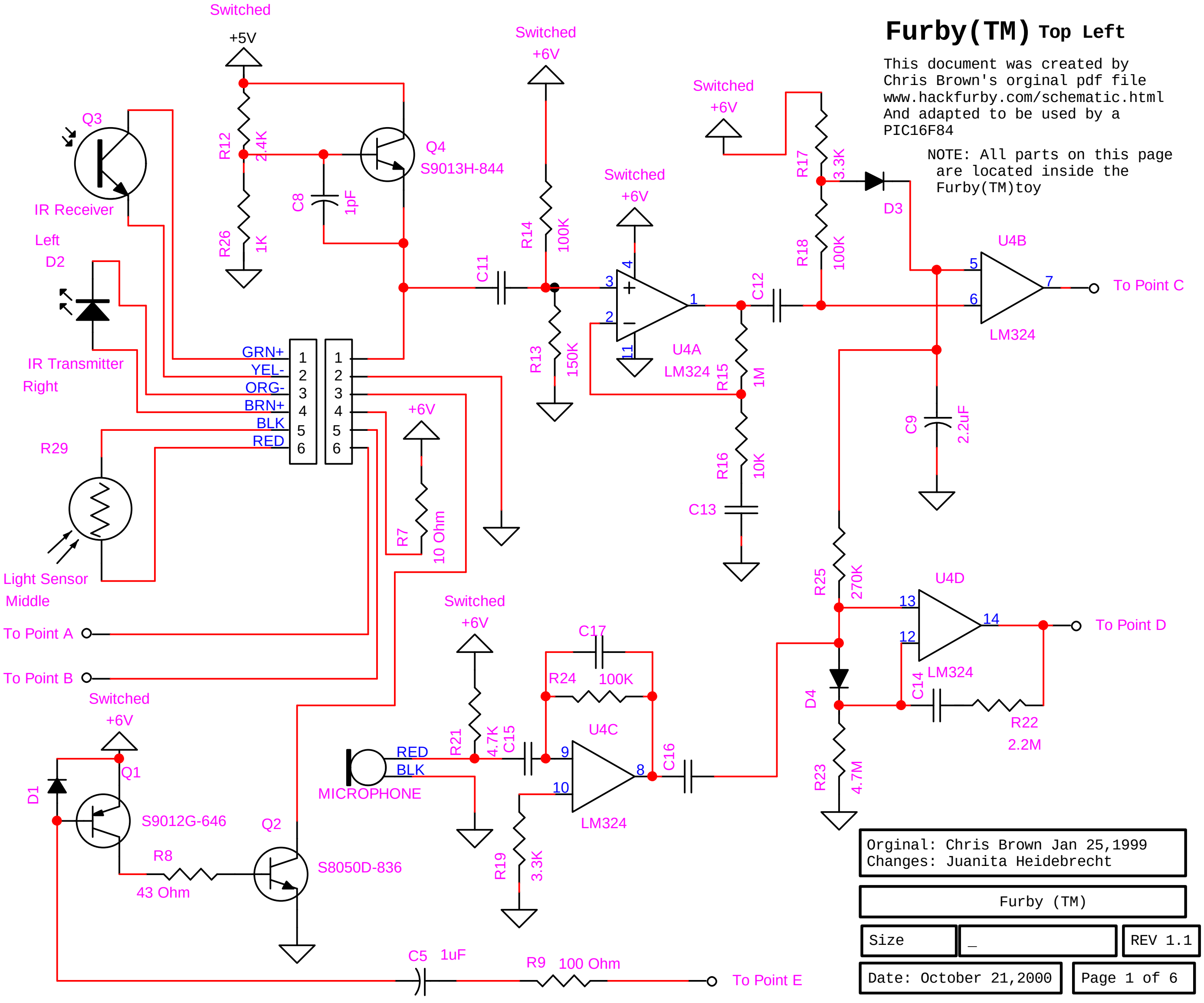
The following 6 pages are a schematic of the Furby™ and the replacement parts.

Top Left<br>
Bottom Left<br>
Top Right<br>
Bottom Left<br>
Original Microprocessor<br>
Replacement Microprocessor

## **Furby(TM)** Top Left

This document was created by
Chris Brown's orginal pdf file
www.hackfurby.com/schematic.html
And adapted to be used by a
PIC16F84

NOTE: All parts on this page
are located inside the
Furby(TM)toy

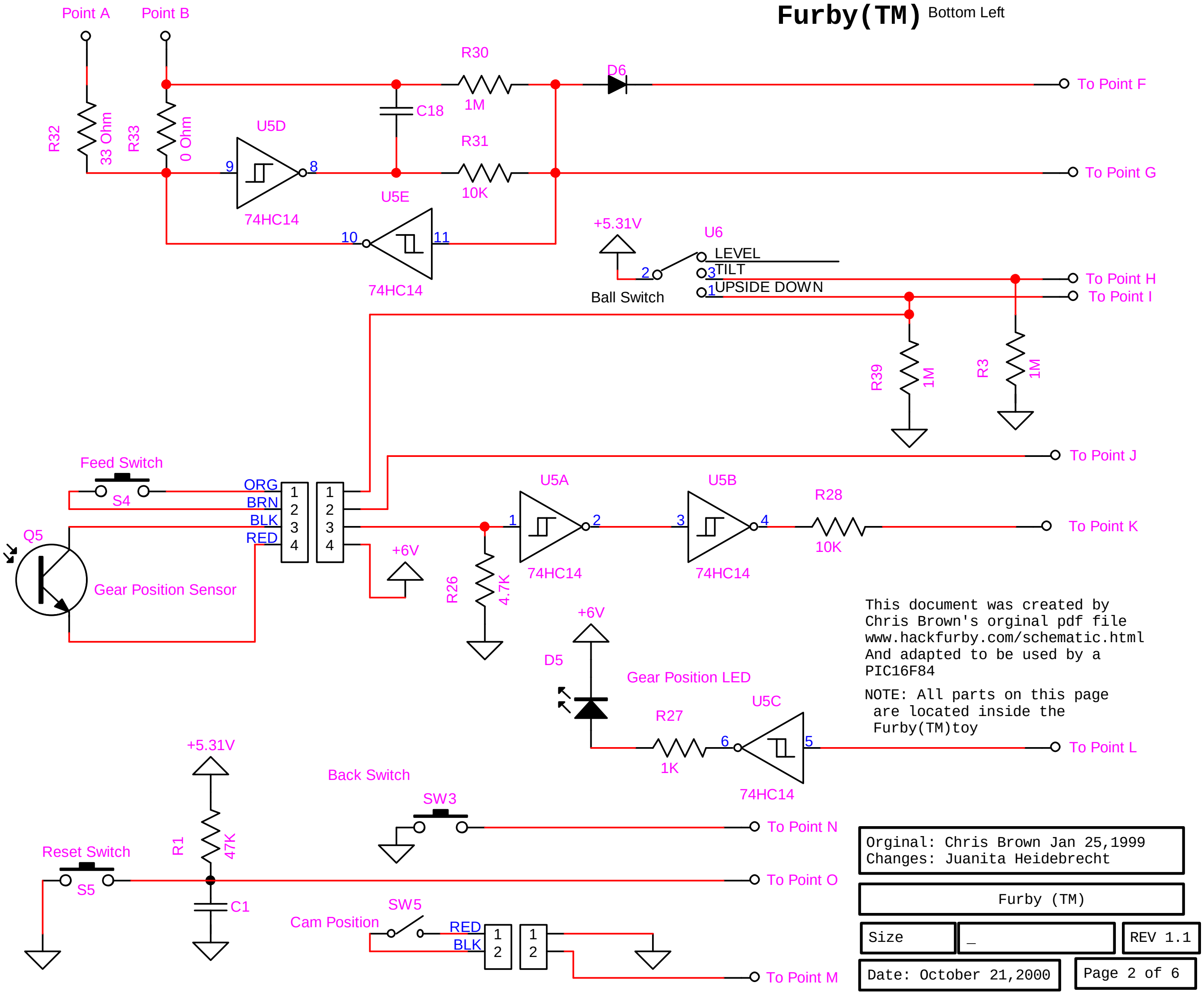


Orginal: Chris Brown Jan 25,1999
Changes: Juanita Heidebrecht

Furby (TM)

| | | |
|---|---|---|
| Size | _ | REV 1.1 |
| Date: October 21,2000 | Page 1 of 6 | |

# Furby(TM) Bottom Left



This document was created by
Chris Brown's orginal pdf file
www.hackfurby.com/schematic.html
And adapted to be used by a
PIC16F84

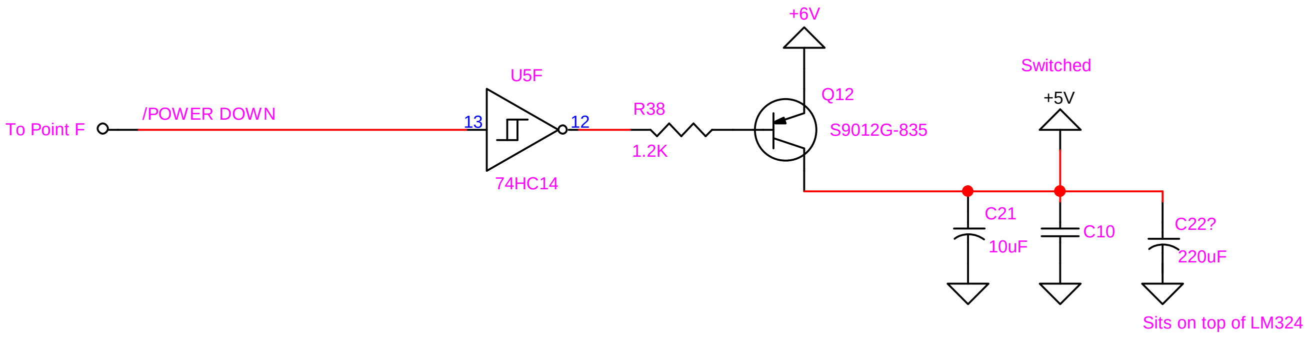
NOTE: All parts on this page
are located inside the
Furby(TM)toy

Orginal: Chris Brown Jan 25,1999
Changes: Juanita Heidebrecht

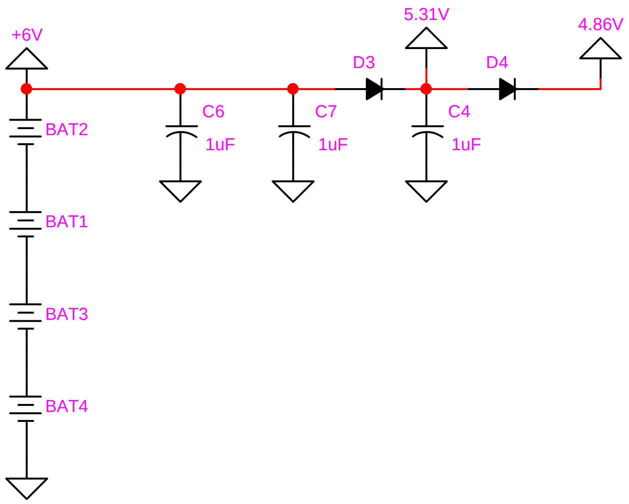
Furby (TM)

| | | |
|---|---|---|
| Size | _ | REV 1.1 |
| Date: October 21,2000 | | Page 2 of 6 |

## Furby(TM) Top Right



## Power Supplies



NOTE: All parts on this page
are located inside the
Furby(TM)toy

This document was created by
Chris Brown's orginal pdf file
www.hackfurby.com/schematic.html
And adapted to be used by a
PIC16F84

Orginal: Chris Brown Jan 25,1999
Changes: Juanita Heidebrecht

Furby (TM)

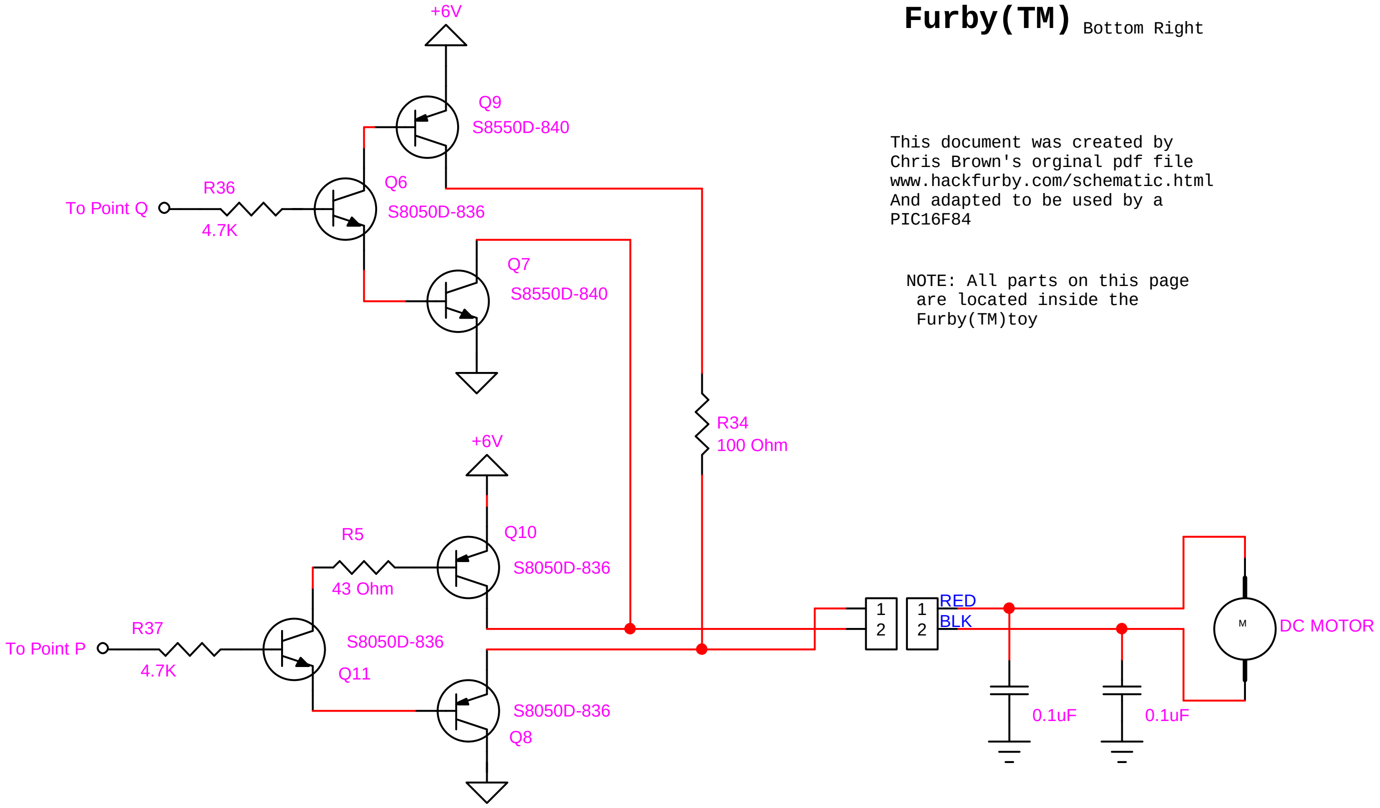
Size | _ | REV 1.1

Date: October 21,2000 | Page 3 of 6

**Furby(TM)** Bottom Right

This document was created by
Chris Brown's orginal pdf file
www.hackfurby.com/schematic.html
And adapted to be used by a
PIC16F84

NOTE: All parts on this page
are located inside the
Furby(TM)toy



Orginal: Chris Brown Jan 25,1999
Changes: Juanita Heidebrecht

Furby (TM)

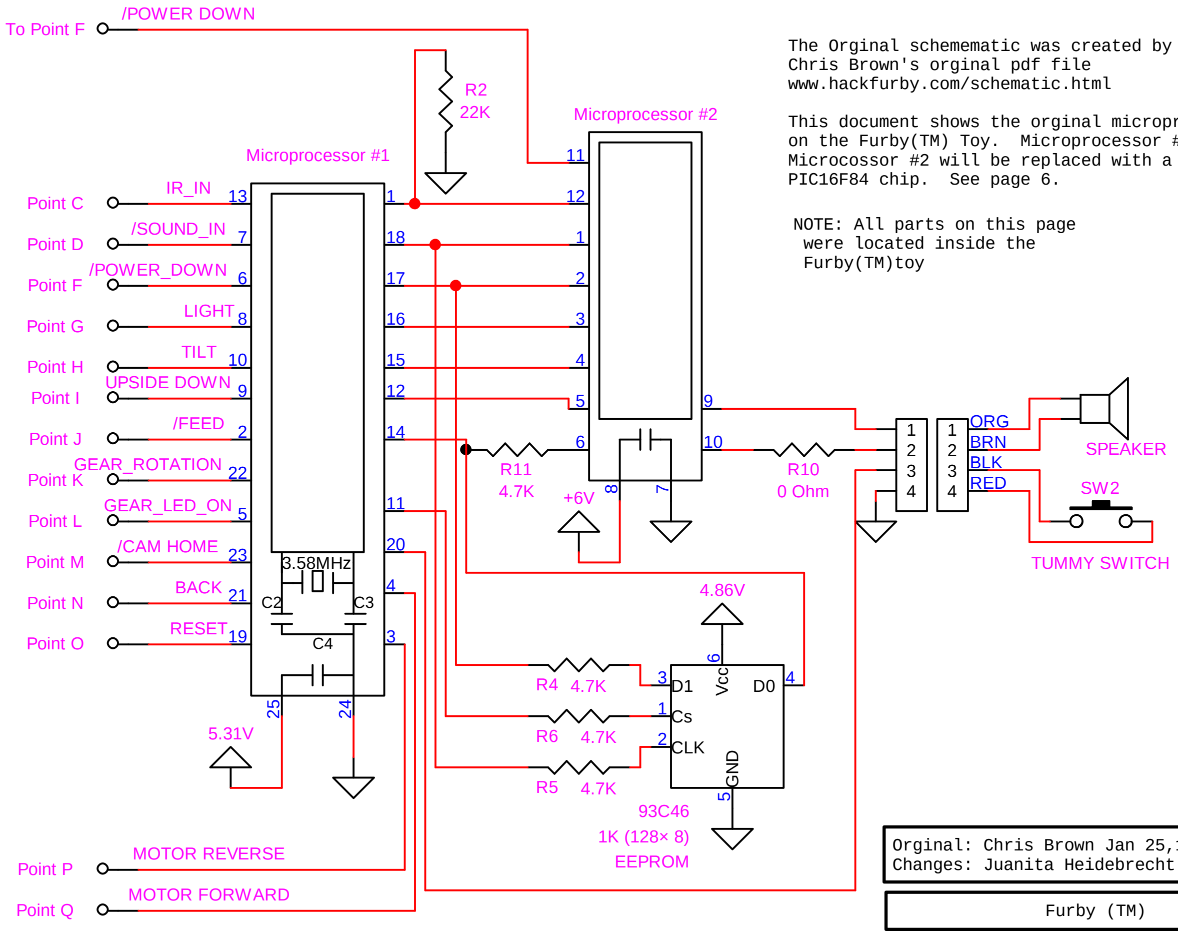
| Size | _ | REV 1.1 |
|---|---|---|
| Date: October 21,2000 | | Page 4 of 6 |

## **Furby(TM)** Orginal Microprocessors

The Orginal schemematic was created by
Chris Brown's orginal pdf file
www.hackfurby.com/schematic.html

This document shows the orginal microprocess on the Furby(TM) Toy. Microprocessor #1 and Microcossor #2 will be replaced with a PIC16F84 chip. See page 6.

NOTE: All parts on this page were located inside the Furby(TM)toy



Orginal: Chris Brown Jan 25,1999
Changes: Juanita Heidebrecht

Furby (TM)

| Size | _ | REV 1.1 |
|---|---|---|
| Date: October 21,2000 | | Page 5 of 6 |

# **Furby(TM)** Replacement Parts

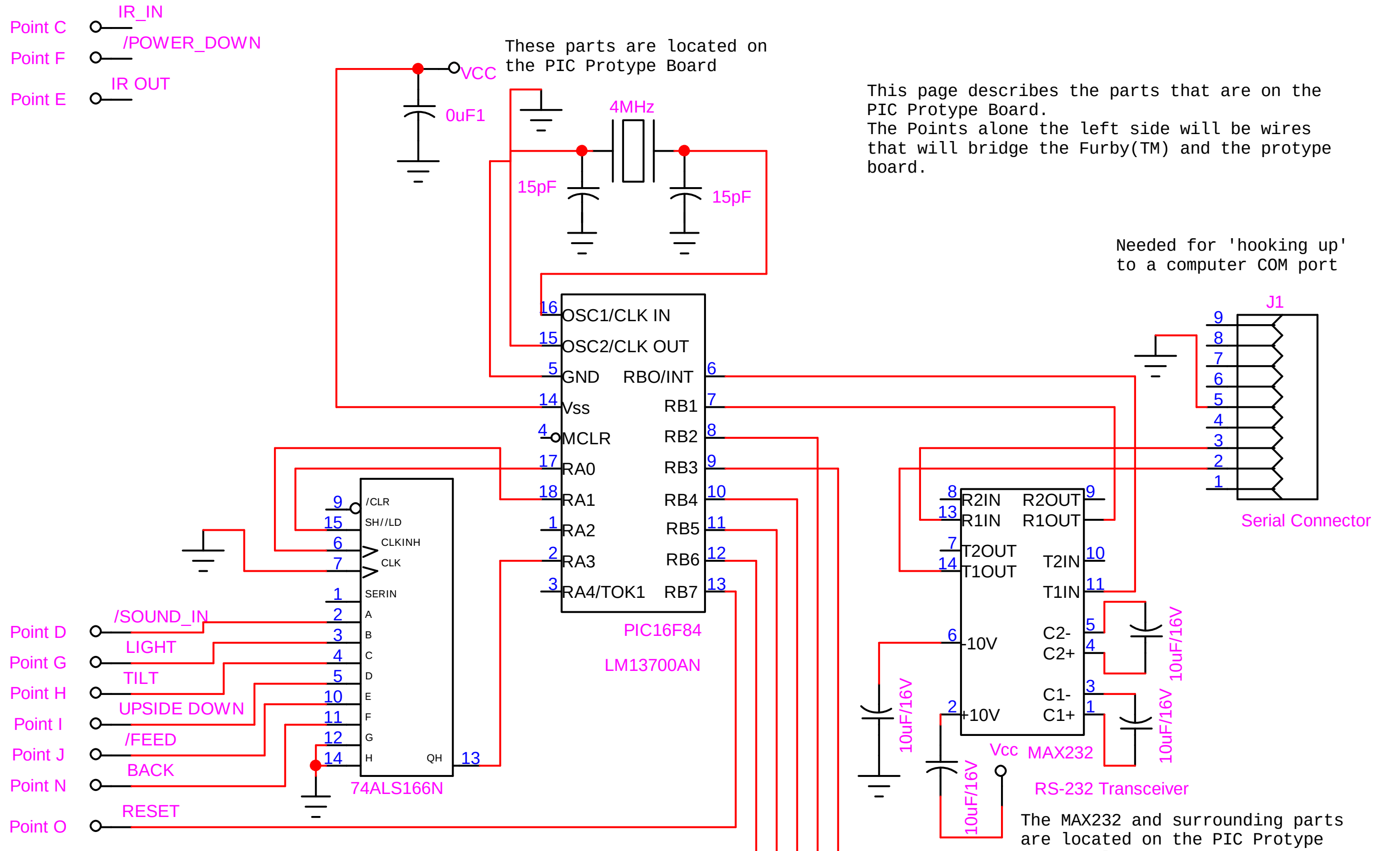
Points that are not used

These parts are located on the PIC Protype Board

This page describes the parts that are on the PIC Protype Board.
The Points alone the left side will be wires that will bridge the Furby(TM) and the protype board.

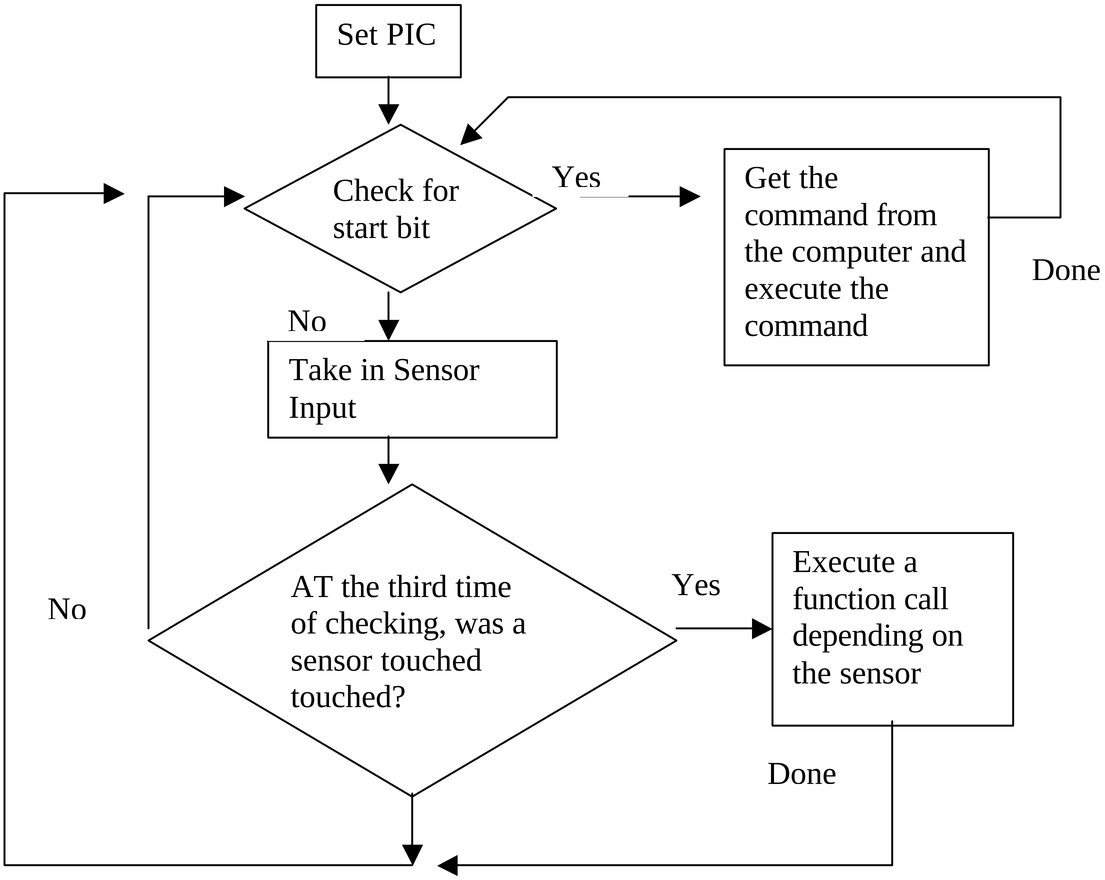
Needed for 'hooking up' to a computer COM port

The MAX232 and surrounding parts are located on the PIC Protype Board unless mentioned otherwise



| | |
|---|---|
| Juanita Heidebrecht | |
| Furby (TM) | |
| Size | _ | REV 1.1 |
| Date: October 21,2000 | Page 6 of 6 |

8.3 Flow Chart


Set PIC
Check for start bit
Yes
Get the command from the computer and execute the command
Done
No
Take in Sensor Input
AT the third time of checking, was a sensor touched touched?
Yes
Execute a function call depending on the sensor
No
Done

## 8.4 ASM Code Listing

```
;*****************************************************************
; Furby(TM)
; Version 008: Finally Version, extra movements have been added
; File Name:   F008.ASM
; Author:      Juanita Heidebrecht, 9308771
; Date:        November 19, 2000
; Class:       COMP630, Computer Engineering Technology
; School:      Niagara College of Applied Arts & Technology
;
;*****************************************************************
;   Target: PIC16F84 MCU     Assembler: MPASM 2.15
;
;  Hardware:
;  Port B
;   RB0   MAX232 TxD1  (Output)
;   RB1   MAX232 RxD1  (Input)
;   RB2   Motor Forward   (Output)
;   RB3   Motor Reverse    (Output)
;   RB4   CAM                  (Input)
;   RB5   Gear_LED_ON   (Output)
;   RB6   Gear_Rotation     (Input)
;   RB7   N/A
;
;  Port A
;   RA0   74HC165 Pin 1 PL / Shift Register enable (Output)
;   RA1   74HC165 Pin 2 CP1 / Clock #1 (Output)
;   RA2   N/A
;   RA3   74HC165 Pin 9 / Serial Output From Last State (Input)
;   RA4   N/A
;   RA5   N/A
;   RA6   N/A
;   RA7   N/A
;   * 74HC165 Pin 15 is now hooked up to ground directly instead of
;     using the PIC.
;   * Extra Hardware layout
;     74HC165 Pin 11  Sound
;     74HC165 Pin 12  Light
;     74HC165 Pin 13  Tilt
;     74HC165 Pin 14  Upsidedown
;     74HC165 Pin 3   Tummy
;     74HC165 Pin 4   Back
;     74HC165 Pin 5   Reset
;     74HC165 Pin 6   n/a - never got accurate info from this one
;   NOTES:
```

```
;   This program uses 2400-baud rate without flow control.  This
;   program looks its best when used with the front end that was
;   made for it.
;*******************************************************************
;   Define type of processor to use and include file of standard EQUs
;
        LIST P=16F84
        include "P16F84.INC"

;*****************************************************************
; Define Registers Used
;*****************************************************************

;Constants
MaxPointer     equu    10     ;3,  maximum number Input Flag Reg.
Bundle         equ    11     ;20, maximum bunch of Gear Sensor
Eight          equ    12     ;8,  maximum number of bits in a byte


;Delay Variables
DelayTemp       equ    13
DelayT2         equ    14
DelayTempS      equ    15
DelayTempSS     equ    16

;Database
FurbyINPUT1    equ    17     ;Input Flag Register
FurbyINPUT2    equ    18
FurbyINPUT3    equ    19
FurbyINPUT4    equ    20

;Gear Variables
EightBites      equ    21     ;Counter, just for eight bytes
Current_State  equ    22     ;Hold the Current postion of Furby(TM)
Gear_Counter  equ    23
Cam_Counter   equ    24
Inc_Counter    equ    25

;Temperary Variables
Temp             equ    26     ;Temperary General Register
Counter          equ    27     ;Temperary General Register/Counter
GearCycles     equ    28     ;Temperary holder for the number gear
                              ;cycles
WantedPosition  equ    29     ;Temperary holder for wanted position

;NOTES:
```

```
;384 cycles needed for 2400-buad rate :. 127  //417us
;95 cycles  needed for 9600-buad rate :. 31  //104us
;16/18 cycles needed for 57200-buad rate :. 5
BuadRate                equ   30
SendCommandByte         equ   31
SentBites               equ   32
ReceivedCommandByte equ   33
ReceiveBites            equ   34

;***************************************************************
; Beginning of the main part of the program
;***************************************************************
main
        ;PORTB::Input:1,4,6/Output:0,2,3,5
        ;Port 7 not used
        movlw   b'11010010'
        tris    PORTB

        ;PORTA::Input:3/Output:1,0
        ;Ports 7-4(not used)
        movlw   b'11111100'
        tris    PORTA

        call    SETPIC      ;Clear all Output ports

RESETPROGRAM
        call    RESET
        call    LongDelay
ContinueToCheckInputs
        clrf    EightBites
        btfsc   PORTB,1     ;if line is low, start bit is present
        goto    ContinueOn  ;received a high: No start bit yet, re- check
        btfss   PORTB,1     ;recieved a low, checks again
        btfsc   PORTB,1     ;a low was for sure received and now falls
                            ;through to the delay call for the start bit
        goto    ContinueOn
        call    StartBitDelay ;Have to waite 1.5 times the cycle for
                              ;the start
        call    RECEIVECOMMAND
        call    CHECKCOMMANDS
        nop

ContinueOn
        call    GETSRINPUT      ;Get Input from any of the
        ;Furby(TM) Sensors
        movf    FSR,w
        sublw    FurbyINPUT4
        btfss     STATUS,Z
        goto      ContinueToCheckInputs
        call       CHECKINPUTS    ;check inputs to determine if there was anything
        call       RESETVARIABLES
        goto      ContinueToCheckInputs

;***************************************************************
;FUNCTION CALLS/METHODS
;  :. Below are all the function call made by the root of the
;     program.  Each function has its own duty, which may call
;     apon other function calls to complete the task.  The most
;     complicated function call may call an endless number of
;     other function calls
;
;***************************************************************

;***************************************************************
; BACK
;     The back sensor was touched.  The command 'B' is then sent
;     to the computer and the command wiggle is then called for
;     execution
;***************************************************************
BACK
        movlw   0x42 ;B
        movwf   SendCommandByte
        call       SENDCOMMAND
        call       WIGGLE
        return
;****************************************************************
; Blink
;     A 'b' was received from the computer serially.  This function
;     call's purpose is to mimic a person blinking their eyes
;****************************************************************
BLINK
        movlw    0x10
        movwf   GearCycles    ;
        call        Move_Forward
        call        Delay
        movlw    0x05
        movwf   GearCycles    ;
        call        Move_Backwards
        call        LongerDelay
        call        RESET
        return
;****************************************************************
```

```
; Check Commands
;     As serial information is called in via RS-232, each character is
;     then checked against a predefined command.  Once recognized,
;     the command is then executed (called)
;************************************************************
CHECKCOMMANDS
        movf     ReceivedCommandByte,w
        sublw    0x74 ;t
        btfss     STATUS,Z
        goto      CheckSleep
        call       Talking
        return
CheckSleep
        movf    ReceivedCommandByte,w
        sublw   0x73 ;s
        btfss     STATUS,Z
        goto     CheckScared
        call      Sleeping
        return
CheckScared
        movf    ReceivedCommandByte,w
        sublw   0x53 ;S
        btfss     STATUS,Z
        goto     CheckWingle
        call      Scared
        return
CheckWingle
        movf    ReceivedCommandByte,w
        sublw   0x77 ;w
        btfss     STATUS,Z
        goto     CheckCLOSE_MOUTH
        call      WIGGLE
        return
CheckCLOSE_MOUTH
        movf    ReceivedCommandByte,w
        sublw   0x63   ;c
        btfss     STATUS,Z
        goto     CheckQUITE
        call    CLOSE_MOUTH
        return
CheckQUITE
        movf    ReceivedCommandByte,w
        sublw   0x71   ;q
        btfss    STATUS,Z
        goto    Checkblink
        call     QUITE
        return
Checkblink
        movf    ReceivedCommandByte,w
        sublw   0x62   ;b
        btfss    STATUS,Z
        goto    CheckReset
        call     QUITE
        return
CheckReset
        movf    ReceivedCommandByte,w
        sublw   0x72  ;r
        btfss     STATUS,Z
        goto     CheckRESET
        call      RESET
        return
CheckRESET
        movf    ReceivedCommandByte,w
        sublw   0x52  ;R
        btfss     STATUS,Z
        goto     NoCommands
        call      RESET
        return
NoCommands
        return
;*****************************************************************
; Check Furby(TM) Inputs
;     This function call checks each sensor.  I did not use
;     interupts and therefore had to be creative in how I would
;     interprete if there was a sensor being used while still being
;     able receive incoming RS-232 commands
;*****************************************************************
CHECKINPUTS
;CheckReset
        btfsc    FurbyINPUT1,0  ;looking for a 0
        goto     CheckBack
        btfsc    FurbyINPUT2,0
        goto     CheckBack
        btfsc    FurbyINPUT3,0
        goto     CheckBack
        call      RESET
        return
CheckBack
        btfsc    FurbyINPUT1,1  ;looking for a 0
        goto     CheckTummy
        btfsc    FurbyINPUT2,1
        goto     CheckTummy
        btfsc    FurbyINPUT3,1
        goto     CheckTummy
        call     BACK
        return
CheckTummy
        btfsc    FurbyINPUT1,2  ;looking for a 0
        goto     CheckFeed
        btfsc    FurbyINPUT2,2
        goto     CheckFeed
        btfsc    FurbyINPUT3,2
        goto     CheckFeed
        call     TUMMY
        return
CheckFeed
        btfss    FurbyINPUT1,3  ;looking for a 1
        return
        btfss    FurbyINPUT2,3
        return
        btfss    FurbyINPUT3,3
        return
        call     FEED
        return
;******************************************************************
; Close Mouth
;     A 'c' was received serially via RS-232.  This function call
;     mimics someone closing their mouth
;******************************************************************
CLOSE_MOUTH
        movlw   0x07
        movwf   GearCycles
        call         Move_Backwards
        call         LongerDelay
        call         RESET
        return
;******************************************************************
; Delay Routines
;     Below is a listing of a varity of delays, each having their
;     own unique function
;******************************************************************
LongerDelay    ;A delay that the user can see
        movlw   .8
        movwf   DelayTempSS
delayler
        call    LongDelay
        decfsz  DelayTempSS,f
        goto    delayler
        return
;...............................................................
LongDelay                   ;Approx 125 mS delay
        movlw   .255
        movwf   DelayT2
ldelaya
        call    Delay
        decfsz  DelayT2,f     ;Decrement this register and
        goto    ldelaya       ; keep going until it hits zero
        return
;.............................................................

Delay                       ;Short delay
        movlw   .255          ;Load Temp register with constant
        movwf   DelayTemp     ;for .3 ms
delaya
        decfsz  DelayTemp,f   ;Decrement until zero
        goto    delaya
        return
;.............................................................
ShortDelay
        movlw   .100
        movwf   DelayTempS
delayS
        decfsz  DelayTempS,f
        goto    delayS
        return
;.............................................................
ShortestDelay
        movlw   .25
        movwf   DelayTempSS
delaySS
        decfsz  DelayTempSS,f
        goto    delaySS
        return
;.............................................................
SendDelay
;9600 need .25 and a nop
        movlw   .119
        movwf   BuadRate
SendLoop
        decfsz  BuadRate,f
        goto    SendLoop
        nop
        nop
        return
```

```
;............................................................
ReceiveDelay
        movlw   .119
        movwf   BuadRate
ReceiveLoop
        decfsz  BuadRate,f
        goto    ReceiveLoop
        return
;............................................................
StartBitDelay
       movlw   .170
       movwf   BuadRate
StartBitLoop
       decfsz  BuadRate,f
       goto    StartBitLoop
       return
;**************************************************************
; FEEDME
;    One of the sensors was touched and now an 'F' is sent to the
;    computer and a little wiggle is executed
;**************************************************************
FEED
        movlw   0x46 ;f
        movwf   SendCommandByte
        call    SENDCOMMAND
        call    WIGGLE
        call    LongerDelay
        return
;**************************************************************
; Forward
;     Forward motion command function call, this was set up
;     orginally so bit 3 and bit 2 are not set on at the same
;     time automatically.
;**************************************************************
FORWARD
        bcf     PORTB,3
        bsf     PORTB,2
        return
;*************************************************************
; GET INPUTS FROM SHIFT REGISTER
;      This function call takes in inputs from the shift register
;      serially thorough a shift register
;*************************************************************
GETSRINPUT
        bsf     PORTA,0         ;Enable the Shift Register
NextInputBit
```

```
        bsf     PORTA,1         ;Create the triger
        bcf     PORTA,1

        btfsc   PORTA,3         ;waiting for the start of the next bit
        bsf     STATUS,C        ;set the next bit (1)
        btfss   PORTA,3
        bcf     STATUS,C        ;clear the next bit (0)

        rrf     Temp,f          ;shift all the bits to the right
        incf    EightBites,f    ;increment the bit counter

        movf    EightBites,w    ;checking for the eight's bit
        sublw   .8              ;to make that byte
        btfss   STATUS,Z
        goto    NextInputBit    ;8 bits have not been received yet - agian
        bcf     PORTA,0         ;8 bits have been received now

        movf    Temp,w          ;Move the contents into the safe place
        movwf   INDF
        incf    FSR,f           ;Increment the pointer
        return
;****************************************************************
; Halloween Take, The
;       I thought that eyes & mouth when open while the ears were
;       straight up made a good scared or surprised position..  This
;       function call is not relevant but was cute.
;****************************************************************
Scared
        ;I want to send a singal to the computer to play a scray noise
        movlw   0x06
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongerDelay
        call    LongerDelay
        call    RESET
        return
;****************************************************************
; Move Forward
;       this function call counts the number of times the gears goes
;       around so I can fake movement.  This is not very accurate but
;       is close enough that when making other packaged function call
;       the toy looks as if it goes to the same place each time.  It
;       is not true.  It depends on many factors and timing has a lot
;       to do with it.
;****************************************************************
Move_Forward
```

```
        movlw   .0
        movwf   Gear_Counter      ;need this one
        movwf   Inc_Counter       ;need this one
KeepGoingForward
        movlw   .0
        movwf   Gear_Counter            ;Clear the Gear Counter
KeepLookingForGEAR_Forward
        call    FORWARD
        call    ShortDelay
        btfss   PORTB,6                 ;Check the Gear
        goto    KeepLookingForGEAR_Forward
        call    STOPMOTOR
        incf    Gear_Counter,f
        movf    Gear_Counter,w
        sublw   Bundle                  ;Move motors 20 pulses
        btfss   STATUS,Z
        goto    KeepLookingForGEAR_Forward
        incf    Inc_Counter,f
        movf    Inc_Counter,w
        subwf   GearCycles,w            ;The End yet?
        btfss   STATUS,Z
        goto    KeepGoingForward        ;Still have to move motors
        return
;*****************************************************************
; Move Backwards
;       this function call counts the number of times the gears goes
;       around so I can fake movement.  This is not very accurate but
;       is close enough that when making other packaged function call
;       the toy looks as if it goes to the same place each time.  It
;       is not true.  It depends on many factors and timing has a lot
;       to do with it.
;*****************************************************************
Move_Backwards
        movlw   .0
        movwf   Gear_Counter      ;need this one
        movwf   Inc_Counter       ;need this one

KeepGoingBackwards
        movlw   .0
        movwf   Gear_Counter            ;Clear the Gear Counter
KeepLookingForGEAR_Backwards
        call    REVERSE
        call    ShortDelay
        btfss   PORTB,6                 ;Check the Gear
        goto    KeepLookingForGEAR_Backwards
        call    STOPMOTOR
        incf    Gear_Counter,f
        movf    Gear_Counter,w
        sublw   Bundle                  ;Move motors 20 pulses
        btfss   STATUS,Z
        goto    KeepLookingForGEAR_Backwards
        incf    Inc_Counter,f
        movf    Inc_Counter,w
        subwf   GearCycles,w            ;The End yet?
        btfss   STATUS,Z
        goto    KeepGoingBackwards       ;Still have to move motors
        return
;***************************************************************
; QUITE!!!
;     A 'q' was received from the computer via RS-232.  This is,
;     if nothings else cute little function call.  Not a compete
;     routine package.
;***************************************************************
QUITE
        movlw   0x10
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongerDelay
        call    RESET
        return
;****************************************************************
; HOME
;     This function call is a primmer.  It is the most important
;     function call.  It is the 'home' position where the toy
;     repositions itself after each function call.  This function
;     calls gives me the ability to fake movements : make furby™
;     appear to being mimicking something.
;****************************************************************
RETURNHOME
KeepLookingForCAM               ;Position Furby(TM) home
        call    REVERSE
        call    STOPMOTOR
        btfsc   PORTB,4         ;Check for CAM
        goto    KeepLookingForCAM
        call    STOPMOTOR
        movlw   .0
        movwf   Current_State   ;Hold current position
        return
;****************************************************************
; Receive a Command from the computer
;     this function call was taking from my lab 3 (RS-232
;     communication).  It allows me to taking in information
;     from the computer and interpret them correctly
;********************************************************************
RECEIVECOMMAND
        ;Just need to receive on byte(a command/option)
        ;ReceivedCommandByte
        clrf    ReceiveBites
NextRXBit
        btfsc   PORTB, 1     ;waiting for the start of the next bit
        bsf     STATUS,C     ;set the next bit (1)
        btfss   PORTB, 1
        bcf     STATUS,C     ;clear the next bit (0)

        rrf     ReceivedCommandByte,f  ;shift all the bits to the right
        incf    ReceiveBites,f  ;increment the bit counter

        call    ReceiveDelay    ;need 104u second delay between bits

        movf    ReceiveBites,w  ;checking for the eight's bit
        subwf   Eight,w          ;to make that byte
        btfss   STATUS,Z
        goto    NextRXBit  ;8 bits have not been received yet - again
        return             ;8 bits have been received - can return now
;********************************************************************
; Reset The Furby(TM)
;********************************************************************
RESET
        call    RESETVARIABLES
        call    RETURNHOME
        call    LongDelay
        return
;********************************************************************
; Reseting Variables
;     Addresses are reset to the beginning position and variables
;     are cleared
;********************************************************************
RESETVARIABLES
        movlw   .0
        movwf   FurbyINPUT1     ;Clear a Input Flag
        movwf   FurbyINPUT2
        movwf   FurbyINPUT3
        movwf   EightBites
        movwf   Temp
        movwf   Counter
        movlw   FurbyINPUT1    ;Making the pointer
        movwf   FSR
        return
```

```
;**************************************************************************
; Reverse
;     This function call is for backward motion.  This was set up
;     originally so bit 3 and bit 2 are not set on at the same
;     time automatically.
;**************************************************************************
REVERSE
        bcf     PORTB,2
        bsf     PORTB,3
        return
;**************************************************************************
; Send a Command to the computer
;     this function call was taking from my lab 3 (RS-232
;     communication).  It allows me to send information
;     to the computer and interpret them correctly
;**************************************************************************
SENDCOMMAND
        ; Just need to send one byte (a command/option)
        clrf    SentBites
        bcf     PORTB,0
        call    SendDelay         ;Start bit
NextTXBit
        btfsc   SendCommandByte,0
        bsf     PORTB,0           ;set the next bit (1)
        btfss   SendCommandByte,0
        bcf     PORTB,0           ;clear the next bit (0)
        rrf     SendCommandByte,f ;shift all the bits to the right
        incf    SentBites,f       ;increment the bit counter
        call    SendDelay
        movf    SentBites,w
        subwf   Eight,w
        btfss   STATUS,Z     ;Check if 8 bits have been sent
        goto    NextTXBit    ;8 bits have not been sent,
                              ;must continue
        bsf    PORTB,0       ;End
        return
;**************************************************************************
; Setup the Pic
; Purpose: Setup the states on the Outputs and initialize any
;              constances
;**************************************************************************
SETPIC
        bsf     PORTB,0         ;RS-232 TxD1
        bcf     PORTB,2         ;Forward Control
        bcf     PORTB,3         ;Reverse Control
        bsf     PORTB,5         ;Turn the GEAR_LED_ON
```

```
                                ;and Leave it on
        bcf     PORTA,0         ;Shift Register Enable line
                                 ;Active Low
        bcf     PORTA,1         ;CP1 Clock Control
        movlw   .3
        movwf   MaxPointer    ;3,  maximum number Input Flag Reg.
        movlw   .10
        movwf   Bundle       ;20, maximum bunch of Gear Sensor
        movlw   .8
        movwf   Eight        ;8,  maximum number of bits in a byte
        return
;*****************************************************************
; Sleeping away
;     A 's' was received from the computer.  The toy will now
;     mimic someone sleeping but standing up :)
;*****************************************************************
; Sleeping Away
Sleeping
        ; I would like to send a command to the computer to play a wave file
        movlw   0x13
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongerDelay
        movlw   0x4
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongerDelay
        movlw   0x6
        movwf   GearCycles    ;
        call    Move_Backwards
        call    LongerDelay
        movlw   0x5
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongerDelay
        movlw   0x5
        movwf   GearCycles    ;
        call    Move_Backwards
        call    LongerDelay
        movlw   0x5
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongerDelay
        movlw   0x5
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongerDelay
        call    LongerDelay
        call    RESET
        return
;*****************************************************************
; Stop the motor
;     Making sure that both bits is set low, as to stop any
;     movement
;*****************************************************************
STOPMOTOR
        bcf     PORTB,2
        bcf     PORTB,3
        return
;*****************************************************************
; Talk
;   A 't' was received from the computer.  This function call makes
;   the toy mimic someone talking
;*****************************************************************
; Talking Away
Talking
        ; I would like to send a command to the computer to play a wave file
         ;movlw   0x26
        movlw   0x05
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongDelay
        movlw   0x07
        movwf   GearCycles    ;
        call    Move_Backwards
        call    LongDelay
        movlw   0x04
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongDelay
        movlw   0x07
        movwf   GearCycles    ;
        call    Move_Backwards
        call    LongDelay
        movlw   0x06
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongDelay
        movlw   0x05
        movwf   GearCycles    ;
        call    Move_Backwards
        call    LongDelay
        movlw   0x05
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongDelay
        movlw   0x05
        movwf   GearCycles    ;
        call    Move_Backwards
        call    LongDelay
        movlw   0x05
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongDelay
        movlw   0x05
        movwf   GearCycles    ;
        call    Move_Backwards
        call    LongDelay
        movlw   0x05
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongDelay
        movlw   0x05
        movwf   GearCycles    ;
        call    Move_Backwards
        call    LongDelay
        movlw   0x05
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongerDelay
        call    LongerDelay
        call    RESET
        return
;****************************************************************
; Tummy was touched
;      The tummy sensor was touched
;****************************************************************
TUMMY
        movlw   0x54 ;T
        movwf   SendCommandByte
        call    SENDCOMMAND
        call    WIGGLE
        return
;****************************************************************
; Wiggle ears
;     A cute and useless function call
;****************************************************************
WIGGLE
```

```
        movlw   0x02
        movwf   GearCycles    ;
        call    Move_Backwards
        call    LongDelay
        movlw   0x03
        movwf   GearCycles    ;
        call    Move_Forward
        call    LongerDelay
        call    RESET
        return
;**************************************************************************
; The End of the Program
;**************************************************************************
       END
;**************************************************************************
```

## 8.5 Front-end C++ Program

Menu Commands:

File

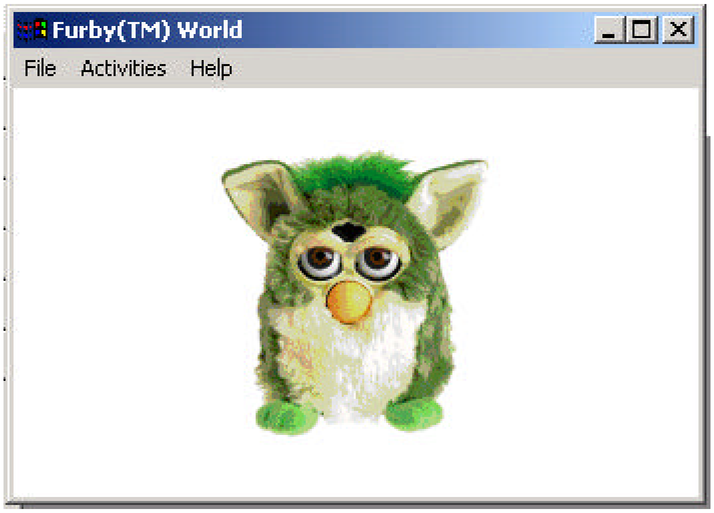
| | |
|---|---|
| COM 1 | Select COM Port 1 to communicate with |
| COM 2 | Select COM Port 2 to communicate with |
| Exit | Exit the program |

Activities

| | |
|---|---|
| Blink | The toy appears to blinks his eyes |
| Close Mouth | The toys appears to close his mouth |
| Quite! | The toys appears to go into a shut down position |
| Reset | The toy resets itself |
| Sing | The toy appears to be singing a song |
| Sleeping | The toy appears to be sleeping |
| Surprised | The toy appears to be surprised or scared |
| Talk | The toy appears to be talking |
| Wiggle | The toy appears to wiggle his ears |

Help

| | |
|---|---|
| About.. | Informs user about the program |


Furby(TM) World
File
Activities
Help