USB QUIZ BOX SOFTWARE

INTERFACE DOCUMENT

Brian’s Boxes

Revision 1.1

7/9/2013

© 2012 McKevett Enterprises

# Introduction

This document has been created to allow users of the Grass Lake Assembly of God USB Quiz Box to create custom software to interface with the quiz box. This custom software can be used to do creative functions such as automatic score keeping, buzz in practice or any other amounts of things that may be done with the information given from the box.

The software should use the HID interface crated by the quiz box firmware when connected via USB. Using this interface should allow for easier software connection and commands with the custom software.

# The basis of this software interface used originally created by Simon Inns for open source use. The original software and tutorial is located at Simon’s website, [www.waitingforfriday.com](http://www.waitingforfriday.com) , under the subheading “Open Source Framework for USB Generic HID devices based on the PIC18F and Windows”. I suggest that you review this tutorial before attempting custom software for the quizbox.

# Features

The quiz box has the following features that may be accessed through the USB commands. These USB features are as follows:

* Sends status of the box in response to the request
* Communicates the first person to buzz in
* Communicates the lag time of the players that buzz in after the first person
* Quiz box can be reset
* Can start 5 second, 30 second, 1 minute, 2 minute, and 3 minute timers
* Starts reaction timing game
* Communicates the reaction time of all players for the reaction timing game

### Reaction Time Game

The quiz box is capable of hosting a reaction time game. This game measures the reaction time of all players and displays the light of the player that had the quickest reaction time. The individual reaction times can be displayed through a status message after the game is completed.

The reaction timer game starts with a command to start the game. It is recommended that a random delay time of 0 and 0.5 seconds be programmed into the software between the time the game button is pushed and the USB message(0x85) is sent to the quiz box. This message initiates a delay of 2 seconds and a status message is returned to the computer. After this delay the center yellow light will turn on (status 2) and the timer will start. The players will then have 1 second to buzz in. All players will be timed from the start of the yellow light. Once the timer has expired the status will change to 8. The player with the fastest reaction time will have their LED light up. To guarantee all players have buzzed in the time bytes should only be looked at once the status goes to 8. Any players that have buzzed in before the yellow LED is lit are penalized by not given a score.

### 5 Second Timer

The quiz box has a feature that runs a 5 second timer that is used to mark the timing period to buzz in after a question is asked by the quiz master. If someone buzzes in before the timer expires then the timer is cleared. If no one buzzes in then the yellow LED will in the center of the quiz box will be lit and the double buzz will go off. At this time the players will be locked out.

A status result will be returned for this timer to acknowledge that the timer has started and that there has not already been a player to buzz in.

### Buzz In

The quiz box will allow a player to buzz in at any time that the box is its idle state.

### 30 Second, 1 Minute, 2 Minute, and 3 Minute Timers

The quiz box has the capability of running timers for responses, time outs, contests, and contest rebuttals. When these extended timers are running the player’s paddles are disabled until the timer has expired.

# USB Commands

The quiz box firmware is configured to react to USB commands. The USB commands are structured as 65 Byte messages that are formatted as the following:

|  |  |  |  |
| --- | --- | --- | --- |
| Byte | 0 | 1 | 2-64 |
| Description | Zeros | Command Header | data |

The reaction and response to each command varies greatly depending on the command. Below are a list of commands and their command header designations.

|  |  |  |
| --- | --- | --- |
| Command Header (hex) | Command Description | Response |
| 0x80 | Clear Command | None |
| 0x85 | Start Reaction Timing Game | Status |
| 0x86 | Status Request | Status |
| 0x87 | Start 5 Sec Timer | Status |
| 0x88 | Start 30 Sec Timer | None |
| 0x89 | Start 1 Min Timer | None |
| 0x8A | Start 2 Min Timer | None |
| 0x8B | Start 3 Min Timer | None |
| 0x8C | Start Infinite Timer | None |
| 0x8D | End Infinite Timer/Buzz | None |

## USB Commands

The computer software can send several USB commands to control the quiz box to do different functions. These same functions can be done with the pushbutton switch inputs on the control box if the quiz box is not connected to a computer.

### Clear Command 0x80

The clear command will clear all counters, LEDs, and any other quiz box variables. This command can be used to exit any timer state or to get the box ready for its next task.

### Start Reaction Timer Game Command 0x85

This command can only be started when the quiz box is in idle mode.

### Status Command 0x86

This command returns a status message from the quiz box.

### Start 5 Second Timer Command 0x87

The start 5 second timer command gives the players 5 seconds to buzz in before they are locked out. This command returns a status message.

### Start 30 Second Timer Command 0x88

This command starts the extended 30 second timer and locks out the paddles until the timer has expired.

### Start 1 Minute Timer Command 0x89

This command starts the extended 1 minute timer and locks out the paddles until the timer has expired.

### Start 2 Minute Timer Command 0x8A

This command starts the extended 2 minute timer and locks out the paddles until the timer has expired.

### Start 3 Minute Timer Command 0x8B

This command starts the extended 3 minute timer and locks out the paddles until the timer has expired.

### Start Infinite Timer Command 0x8C

This command starts a timer that will continuously run until the timer end command is sent. This timer locks out the paddles just like the other timers.

### End Infinite Timer/Buzz Command 0x8B

This command ends the continuous timer in a similar method of the other timers. If this command is sent without continuous timer running it will sound a buzz as the end of a timer.

## USB Response

With certain USB commands there is a response message sent back to the USB master. There is only one response message, but multiple commands will cause the response to be sent. The response message contains the following data:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Byte | 0 | 1 | 2 | 3 | 4-5 | 6-7 | 8-9 | 10-11 | 12-13 | 14-15 | 16-17 | 18-19 | 20-64 |
| Description | 0 | 0 | STATUS | WINNER | TR1 | TR2 | TR3 | TR4 | TG1 | TG2 | TG3 | TG4 | N/A |

### Status Byte

The status byte is used to determine the current status of the quiz box.

0x00 Idle Mode - awaiting command or player to buzz in

0x01 Game pre-start – The reaction time game has been started but the yellow light has not been turned on yet.

0x02 Game Running – The reaction time game is running after the yellow light has started, but the game timer has not yet run out.

0x04 Person Buzzed In – Someone has buzzed in and a timer is running to determine if other players are going to buzz in.

0x08 Done – Either the reaction time game has finished its timer OR the player buzz in timer has finished.

0x10 5 Sec Timer Running – This indicates that the 5 second timer is running and no one has buzzed in yet.

0x20 Extended Timer Running – This indicates that one of the timers of 30 seconds or longer is running. During this state no one can buzz in.

0x40 Startup Sequencing – This is the status code when the quiz box is going through its LED and buzzer check at startup.

Status transition

|  |  |
| --- | --- |
| **Status** | **Goes to** |
| 0x00 | 0x01, 0x04, 0x10, or 0x20 |
| 0x01 | 0x02 |
| 0x02 | 0x08 |
| 0x04 | 0x08 |
| 0x08 | 0x00 Reset command required |
| 0x10 | 0x04 or 0x08 |
| 0x20 | Previous State |
| 0x40 | 0x00 |

### Winner Byte

The winner byte tells who was the first person to buzz in. That person currently has their light enabled.

1. No valid winner
2. 5 second timer has expired
3. n/a
4. R4
5. R3
6. R2
7. R1
8. G1
9. G2
10. G3
11. G4

### Time Bytes – TR1 to TG4

Each player is given two time bytes that combine to create a 16 bit integer. The lower number byte contains the MSB of the integer. There are two times in which the bytes are valid, during the reaction time game and after a player has buzzed for the first time. The timer for the time bytes only last for 1 second after the initial start. The bytes are in counter form. Each counter increment is worth 1.02 milliseconds.

During the reaction time game the time bytes are counted from the time of the initial yellow reaction LED. This allows the winner and all the players to buzz in to view their reaction time.

During the normal operation of the box the time bytes are used to determine which players have buzzed in second, third, and so on. The winning player will have a time byte of zero and the other players have times delayed from the time that the winning player has buzzed in.

Example:

Byte 8: 0x01

Byte 9: 0x45

Integer (16bit) = 0x0145 = 325

Time = 325 \* 1.02 = 331.5 mSec time delay for player R3

# Example Code

Below is example code used by the computer software.

## USB Command Code

This code gives a command to clear the quiz box, command 0x80.

public void greset()

{

// Reset Box

Debug.WriteLine("Reference Application -> Reset Box");

// Declare our output buffer

Byte[] outputBuffer = new Byte[65];

// Declare our input buffer

Byte[] inputBuffer = new Byte[65];

// Byte 0 must be set to 0

outputBuffer[0] = 0;

// Byte 1 must be set to our command

outputBuffer[1] = 0x80;

// Fill the rest of the buffer with known data

int bufferPointer;

Byte data = 0;

for (bufferPointer = 2; bufferPointer < 65; bufferPointer++)

{

// We send the numbers 0 to 63 to the device

outputBuffer[bufferPointer] = data;

data++;

}

// Perform the write command

bool success;

success = writeRawReportToDevice(outputBuffer);

}

## USB Receive Status Code

This code gives a command 0x86 to the quiz box to read the status. The status is returned and the data is decoded.

public gameresult statusrequest()

{

// Game

Debug.WriteLine("Reference Application -> status request");

// Declare our output buffer

Byte[] outputBuffer = new Byte[65];

// Declare our input buffer

Byte[] inputBuffer = new Byte[65];

// Byte 0 must be set to 0

outputBuffer[0] = 0;

// Byte 1 must be set to our command

outputBuffer[1] = 0x86;

// Fill the rest of the buffer with known data

int bufferPointer;

Byte data = 0;

for (bufferPointer = 2; bufferPointer < 65; bufferPointer++)

{

// We send the numbers 0 to 63 to the device

outputBuffer[bufferPointer] = data;

data++;

}

// Perform the write command

gameresult gresulti;

gresulti = new gameresult();

bool success;

success = writeRawReportToDevice(outputBuffer);

if (success)

{

// Perform the read

success = readSingleReportFromDevice(ref inputBuffer);

if (success)

{

// Test the received data; we expect 65 bytes, byte[0] is unused

// bytes 1-64 should be filled with the numbers 0-63

data = 0;

gresulti.gmstatus = inputBuffer[2];

gresulti.gwinner = inputBuffer[3];

for (bufferPointer = 0; bufferPointer < 8; bufferPointer++)

{

gresulti.player[bufferPointer] = bufferPointer + 1;

gresulti.time[bufferPointer] = ((inputBuffer[(bufferPointer \* 2) + 4]) << 8) + inputBuffer[(bufferPointer \* 2) + 5];

}

success = true;

}

}

//RETURN DATA

if (success) gresulti.gstatus = 1;

else gresulti.gstatus = 0;

// We can't tell if the device receieved the data ok, we are

// only indicating that the write was error free.

return gresulti;

}