FIRMWARE / SOFTWARE REFERENCE

MICHAEL WIECKOWSKI

MAY 2013

info@shutterburemote.com

Description of the files

compiler directory

This directory contains all of the files distributed by BlueGiga for compiling a HEX file for upload to the BLE module using a CC debugger from Texas Instruments. We are redistributing them because this is the version (1.1) used in the production Shutterbug Remote. From the command line, use "bgbuild.exe project.xml" to run the compiler.

project.xml , hardware.xml

These two files are required by the BGScript compiler. project.xml describes which files are used in the project and the name of the generated HEX file. hardware.xml sets up the BLE module options including SPI, DC-DC regulator pin, and the hardware interrupt pin.

gatt.xml

This file describes the generic access profile used by the Bluetooth module and the iOS app. Variables called "characteristics" are used to exchange data between the hardware and the app over the air. Each characteristic is given a UUID using an online generator. The characteristics are read from and written to by the app to control the Shutterbug hardware. For example, the shutterFocusControl characteristic is used to open and close the shutter and to activate autofocus. More info about these characteristics is provided later in this document. When the project is compiled, each characteristic is assigned a number (handle) which is documented in a files called attributes.txt (this file is generated by the bgbuild compiler.) The numbers (handles) are used in the BGScripts to refer to the characteristics.

shutterbug_combo.bgs

This is the main BGScript running on a Shutterbug Remote. More detail will be provided in a separate document. Since BGScript does not support functions, we wrote our own function implementation. Every word prepended with the character @ will be recognized as a function and replaced with code from a BGSF file. See functions.py below.

functions.py

This is a Python script used to implement functions. It will take shutterbug_combo.bgs and replace every keyword prepended with @ along with various arguments with the contents of another file. At this time, the only function we use is "@setAlarmCombo" which sets up a timer using either BLE hardware or the RTC. Whenever @setAlarmCombo is called in the main bgscript, it is replaces by the contents of the file "setAlarmCombo.bgsf" (note the different extension to connote a function instead of a script.)

setAlarmCombo.bgsf

This is the function used to set up a timed alarm using either BLE hardware or the Microchip RTC. See functions.py above.

rename directory

This directory contains a separate project used for renaming a Shutterbug using the CC debugger. The name of a Shutterbug is stored in the EEPROM of the RTC. This project programs the BLE hardware to simply write a string into the EEPROM.

GATT Characteristics

The GATT characteristics are the key to writing your own Shutterbug Remote apps for iOS or Android. Once connected to a Shutterbug Remote, the app must read and write these characteristics to take photos, activate focus, and perform the intervalometer functions. Each characteristic is described by name below with its UUID in parenthesis.

cameraRemoteShutter (98BF9A5A0DD011E2A376CC4A6188709B)

This is the primary "service" advertised by the Shutterbug Remote. Your app should look for this service when attempting to connect to the hardware. All of the following characteristics are provided as part of this service.

shutterThousandths (198502EEA8E34D16B79C89928367BEDF)

32 bit integer. Used to set the thousandths portion of the shutter time when in bulb mode. This value is multiplied by two in hardware, so be sure to divide by two before setting this value. i.e. For a shutter time of 1/2 second, set this value to 250.

shutterFocusControl (5807de3b40c4481587b8aa76dfc5893a)

32 bit integer used to control the shutter and focus signals. Defaults to zero when nothing is going on. Set to 1 to take a single photo (focus will be activated followed by shutter.) Set to 4 to activate focus only, no picture will be taken. Set to 2 to run a complex program, such as intervalometer or bulb ramping. Set to 5 to stop all programs, close the shutter, and return to idle. Set to 6 to open the shutter and hold it open (manual BULB mode).

debugTest (f694ed53e048407aa4a288104ede6ce7)

32 bit integer used to read back the current timer value (either using the hardware timer or the real time clock chip.) The app should write a 1 to this characteristic, and the hardware will reply with the current timer value as [days] [hours] [minutes] [seconds] where each byte is encoded as a BCD value.

waitTime (8D7107F8A7B34DCBA2C62978D846280F)

32 bit integer. This value hold 4 bytes packed as follows [days] [hours] [minutes] [seconds]. This characteristic sets the time delay before the shutter fires, and must be enabled in the "setup" characteristic below.

shutterOpenTime (C841B7C746CC405C88966D4522981ADC)

32 bit integer with NOTIFY abilities. This value hold 2 bytes packed as follows [0] [0] [minutes] [seconds]. This characteristic sets the time that the shutter remains open during BULB exposures, and must be enabled in the "setup" characteristic below. When a program is running however, you app can register to receive notifications from this characteristic. The value returned will tell you the current exposure times packed as follows [bulb seconds] [bulb minutes] [ramp seconds] [ramp minutes].

repeatInterval (EF8620CCBE6D426581D24E70E759A7D9)

32 bit integer. This value hold 3 bytes packed as follows [0] [hours] [minutes] [seconds]. This characteristic sets the time in between successive exposures when running as an intervalometer, and must be enabled in the "setup" characteristic below.

repeatCount (468E174B331B462FA0C9D0A61880BE41)

32 bit integer with NOTIFY abilities. This value holds the number of photos to be taken when running in intervalometer mode. Once a program is running however, your app can register to receive notifications from this characteris-

tic. It will send a 32 bit value packed with 4 bytes as follows: [repeat LSB] [repeat MSB] [completed LSB] [completed MSB] where LSB and MSB are the least and most significant bytes respectively. This is how the app can measure the progress of long intervalometer programs (time lapse sequences).

setup (BA8EB16FDA12446F9DE11164BF7411A0)

32 bit integer. This value holds 4 bytes packed as follows [Autofocus and Lockup] [Interval] [Bulb] [Wait]. These bytes serve multiple purposes and will be described one at a time.

Autofocus and Lockup: 0 (default) means no extension of the autofocus pulse and no additional shutter pulse for mirror lockup. 1 means just autofocus extension. 2 means just mirror lockup. 3 means autofocus AND mirror lockup.

Interval: Set to 1 to enable the intervalometer (multiple exposures).

Bulb: For a single bulb exposure, set to 1. For multiple bulb exposures using a ramp, set this value to (1 + pivot frame number). For ramping bulb exposures, 3 times are needed: shutterOpenTime, bulbRamp1, and bulbRamp2. The first exposure will start at shutterOpenTime. Then the exposure will change linearly frame by frame to equal bulbRamp1 when the frame equals the pivot frame. Finally, the exposure will change linearly frame by frame to equal bulbRamp2 when the frame equals the repeatCount. Therefore, to do a bulb ramp, the intervalometer must be enabled with a valid repeat count.

Wait: Set to 1 to delay the first exposure by the waitTime.

status (0D304DBB99D2495BABCDF03588159C5B)

32 bit integer with NOTIFY abilities. When this value changes, your app can register to receive a notification to respond immediately. This value will return the "state" of the hardware state machine. The numbers correspond to:

#0 is "end of exposure" or "end of autofocus"
#1 is "end of EEPROM waiting to write"
#2 is "end of mirror lockup high"
#3 is "end of mirror lockup low"
#4 is the end of the subsecond part of the alarm
#5 is the end of the second resolution of alarms
#6 is "alarm completed" and subsecond resolution

bugName (0C01874EB09B4CAFBBDD5E4A4205A9FE)

16 byte UTF-8 string value. This is the name of the Shutterbug Remote displayed when scanning or connecting to the device. It is stored in the EEPROM of the real time clock chip, so once you write this characteristic, it will persist across power cycles.

shutterRamp1Time (33098AF54B424ADE8B4A44F9E5C9C770)

32 bit integer packed as 4 bytes: [direction] [delta minute] [delta second] [delta thousandths / 2]. This value is used during bulb ramping to achieve a linear change in exposure time. Your app must calculate how much to add or subtract to each exposure to smoothly move between the 3 points on the ramp. A direction of 1 means negative ramping, a direction of 0 means positive. The deltas will be added or subtracted from the current exposure to get the next exposure on each frame. NOTE: The largest "thousandths" delta than you can have is 1/2 second. This is a legacy issue resulting from the calculation of shutter times on an actual camera (1 second, 1/2 second, 1/4 second, 1/8 second, etc.) Care should be taken in the app design to ensure that error is minimized when setting these ramp deltas.

shutterRamp2Time (E67ED966286F4CF68EA8C92C93407791)

32 bit integer packed as 4 bytes: [direction] [delta minute] [delta second] [delta thousandths / 2]. This value is used during bulb ramping to achieve a linear change in exposure time. Your app must calculate how much to add or subtract to each exposure to smoothly move between the 3 points on the ramp. A direction of 1 means negative ramping, a direction of 0 means positive. The deltas will be added or subtracted from the current exposure to get the next exposure on each frame. NOTE: The largest "thousandths" delta than you can have is 1/2 second. This is a legacy issue resulting from the calculation of shutter times on an actual camera (1 second, 1/2 second, 1/4 second, 1/8 second, etc.) Care should be taken in the app design to ensure that error is minimized when setting these ramp deltas.