

4 March 2009

Documented changes to CSR's standard stereo headset application for use on the Blue Mojo reference design.

The Blue Mojo Bluetooth reference design uses two QF1D512 filter ICs (the layout can accept QF1Da512's) attached in series to the PCM port of a CSR BC05-MM processor. The audio data flows from the PCM port as an I2S audio stream, is filtered by the two 1D's and returns to the PCM port, after which it is processed by the codec in the CSR processor. On power-up, the BC05-MM initializes the 1D's and then proceeds with its normal power up processing sequence.

The result is that, as far as the CSR processor is concerned, the presence of the filter IC's has no impact on the normal operation of the Bluetooth profiles. They function to improve the quality of the audio in the headset without affecting its performance with respect to Bluetooth functionality. In order to accomplish this, the standard Stereo Headset application (Stereo-Headset-SDK-2008.R1, based on v4 of BlueLab) was modified. The modifications are limited to four application files and one library file. In addition, seven files associated with the QF1D512's were added. The modifications and additions are fully documented in this document.

The added files are:

QF_Bluetooth_driver.c

Contains the low-level routines for writing configuration information to the IC's.

QF_Bluetooth_driver.h

Defines symbolic names for the PIO pins used.

QF1D512_Bluetooth.c

Contains the high-level routines for writing configuration information to the IC's.

QF_file0.c

Contains the filter configuration information – both registers and coefficients – for filter 0.

QF_file0.h

Contains the symbolic names associated with filter 0.

QF_file1.c

Contains the filter configuration information – both registers and coefficients – for filter 1.

QF_file1.h

Contains the symbolic names associated with filter 1.

The modified CSR application files are:

main.c

Accesses the functions in QF1D512_Bluetooth.c to initialize the two filter IC's, turns on the green LED to indicate that the filters are engaged, and turns on the headset power amp.

headset_amp.c

Modified to turn on headset amp and to ensure that it remains on.

headset_buttons.c

Modified to change the Play/Stop button into an A/B switch button (alternately turning the filters on and

off). Also masked off the PIO's used in interacting with the filters, ensuring that they are not detected as button presses, so as not to affect the normal operation of the headset.

headset_event_handler.c

Modified to preclude operation of standard functions associated with buttons/functions not used in the Blue Mojo (FWD, BACK, AUX1, AUX2, etc.). Changed PLAY function to implement part of A/B switch functionality -- cycle the green LED as filters are toggled on and off.

The modified CSR library file is:

csr_a2dp_decoder_common.c

This is the code that connects, routes, and configures the audio streams. It has been modified to configure the PCM port as an I2S port and to route the stereo audio stream (our filters are only active during stereo A2DP audio) out of the BC05-MM, through our filters, and back into the BC05-MM.

The detailed listing of the added/changed code follows:

(Note: except where indicated, line numbers refer to locations in the resulting modified files. Only those lines of code called out as being from the original file reference the original line numbers).

Main.c

No code was changed in this file, and the following lines of code were added:

Lines 30 & 31

```
#include "QF_Bluetooth_driver.h"
#include "headset_pio.h"
```

Lines 64 – 71:

```
void waitx(void);
void load_config(int config);

void QF1D512_WriteConfigByte0( unsigned int Address, unsigned char Value );
void QF1D512_WriteConfigByte1( unsigned int Address, unsigned char Value );

unsigned int A_B_Switch_State;
```

Lines 293 – 391:

```
/*
*****
/* *** The PIO pins used to initialize and configure (and reconfigure) the
QF1D512 digital filter IC's are set in the state needed for that
purpose. Once the filters are set up, the PIO pins are returned to
a state that will not interfere with the normal operation of the
BlueCore processor. For example, since this code was originally
targeted at the DEV-PC-1645B, which uses more buttons than the
Blue Mojo, the BC5-MM would misinterpret a high state on PIO14 as a
keypress. *** */

/*
PioSetPio(QF_RESET, 1); /* This is the new v4 method for setting a PIO */
PioSetDir((1<<QF_RESET), 1); /* I use the older, but still valid, v3.6.2 method */
PioSet((1<<QF_RESET), (1<<QF_RESET));
PioSetStrongBias((1<<QF_RESET), (1<<QF_RESET));
```

```

/*      PioSetPio(QF_SPI_MOSI, 1);*/
      PioSetDir((1<<QF_SPI_MOSI), 1);
      PioSet((1<<QF_SPI_MOSI), (1<<QF_SPI_MOSI));
      PioSetStrongBias((1<<QF_SPI_MOSI), (1<<QF_SPI_MOSI));

/*      PioSetPio(QF_SPI_CLK, 0); */
      PioSetDir((1<<QF_SPI_CLK), 1);
      PioSet((1<<QF_SPI_CLK), (0<<QF_SPI_CLK));
      PioSetStrongBias((1<<QF_SPI_CLK), (1<<QF_SPI_CLK));

/*      PioSetPio(QF_RESET, 0); */
      PioSet((1<<QF_RESET), (0<<QF_RESET));
      PioSetStrongBias((1<<QF_RESET), (1<<QF_RESET));
      waitx();

/*      PioSetPio(QF_SPI_CS1, 1); */
      PioSetDir((1<<QF_SPI_CS1), 1);
      PioSet((1<<QF_SPI_CS1), (1<<QF_SPI_CS1));
      PioSetStrongBias((1<<QF_SPI_CS1), (1<<QF_SPI_CS1));

/*      PioSetPio(QF_SPI_CS2, 1); */
      PioSetDir((1<<QF_SPI_CS2), 1);
      PioSet((1<<QF_SPI_CS2), (1<<QF_SPI_CS2));
      PioSetStrongBias((1<<QF_SPI_CS2), (1<<QF_SPI_CS2));

/*      PioSetPio(QF_RESET, 1); */
      PioSet((1<<QF_RESET), (1<<QF_RESET));
      PioSetStrongBias((1<<QF_RESET), (0<<QF_RESET));

      waitx(); /* wait for the QF1D512's to finish power-up initialization. */
      waitx();

/*      Load a filter configuration into U1*/
      load_config(0);
/*      Load a filter configuration into U2*/
      load_config(1);

/*      Initialize the variable that indicates whether or not the filters
      are active or bypassed. 0 indicates that they are active. */
      A_B_Switch_State = 0;

/*      PioSetPio(QF_LED, 1); */
      PioSetDir((1<<6), 1);
      PioSet((1<<6), (1<<6));
      PioSetStrongBias((1<<6), (1<<6));

/*      while (1){} */
/*      PioSetPio(QF_SPI_CS1, 1);
      PioSetPio(QF_SPI_CS2, 1); */
      PioSet((1<<QF_SPI_CS1), (1<<QF_SPI_CS1));
      PioSet((1<<QF_SPI_CS2), (1<<QF_SPI_CS2));

```

```

/* Set up initial volume levels
CodecSetRawOutputGainA(6,7);
CodecSetRawOutputGainB(6,7);
*/

/* Here, the PIO pins are returned to a state that will not interfere with the
normal operation of the BlueCore processor. */

PioSetDir((1<<QF_SPI_MOSI), 0);      /* Make SPI_MOSI (Pio14) an input */
PioSet((1<<QF_SPI_MOSI), (0<<QF_SPI_MOSI));      /* Set SPI_MOSI high */
PioSetStrongBias((1<<QF_SPI_MOSI), (1<<QF_SPI_MOSI));      /* Strongly pull down the signal */

/*

PioSetDir((1<<10), 0);      /* Make PIO10 an input */
PioSet((1<<10), (0<<10));      /* Set PIO low */
PioSetStrongBias((1<<10), (1<<10));      /* Strongly pull down the signal */

PioSetDir((1<<15), 0);      /* Make PIO15 an input */
PioSet((1<<15), (0<<15));      /* Set PIO low */
PioSetStrongBias((1<<15), (1<<15));      /* Strongly pull down the signal */

/*****

/* Turn on the headset amp */
PioSetDir((1<<3), 0);      /* Make PIO3 an input */
PioSet((1<<3), (1<<3));      /* Pull PIO3 high */
PioSetStrongBias((1<<3), (1<<3));      /* Strongly pull up the signal */

```

headset_amp.c

Function SET_AMP was modified to force the headset amp on:

```

static void SET_AMP ( hsTaskData * pApp, bool on )
{
#ifdef NATIVE_BUILD
/*      uint32 pio = (uint32)1 << pApp->ampPio; */

/*      Turn on the headset amp */
PioSetDir((1<<3), 0);      /* Make PIO3 an input */
PioSet((1<<3), (1<<3));      /* Pull PIO3 high */
PioSetStrongBias((1<<3), (1<<3));      /* Strongly pull up the signal */

/*

PioSetDir32(pio, pio);

PioSet32(pio, 1);

if (on)
    PioSet32(pio, pio); */
/*      else
    PioSet32(pio, 0); */
#else
/*      uint16 pio = (uint16)1 << pApp->ampPio; */

```

```

    /* Turn on the headset amp */
    PioSetDir((1<<3), 0);      /* Make PIO3 an input */
    PioSet((1<<3), (1<<3));    /* Pull PIO3 high */
    PioSetStrongBias((1<<3), (1<<3)); /* Strongly pull up the signal */

/*
    PioSetDir(pio, pio);

    PioSet32(pio, 1);

    if (on)
        PioSet(pio, pio); */
/*     else
        PioSet(pio, 0); */
#endif
}

```

headset_buttons.c

The following modifications were made:

Line 18 was added:

```
#include "headset_volume.h"
```

Lines 65 – 72 were added:

```

void waitx(void);
void QF1D512_WriteConfigByte0( unsigned int Address, unsigned char Value );
void QF1D512_WriteConfigByte1( unsigned int Address, unsigned char Value );

```

```
/* static headsetTaskData app; */
```

```
extern unsigned int A_B_Switch_State;
```

Line 110 was changed into line 112 in order to mask out PIOs now used to interface to filter ICs:

```

/*
    uint32 pio_bits = PioGet32() | CHARGER_VREG_VALUE |
    CHARGER_CONNECT_VALUE;
*/
    uint32 pio_bits = (PioGet32() & 0xFFFF3800) | CHARGER_VREG_VALUE |
    CHARGER_CONNECT_VALUE;

```

Line 186 was added in order to mask out all PIOs except that used for the A/B switch button:

```
uint32 pio_bits2 = (PioGet32() & 0x00002000);
```

Lines 201 – 254 were added to implement the A/B Switch function:

```

/*****
if (pio_bits2 != 0) {

    hsTaskData * theHeadset = (hsTaskData *) getAppTask();

```

```

if (A_B_Switch_State == 0)
{
    VolumeDown ( theHeadset ) ;
    VolumeDown ( theHeadset ) ;
    VolumeDown ( theHeadset ) ;
    VolumeDown ( theHeadset ) ;
    VolumeDown ( theHeadset ) ;

    waitx();
    waitx();
    waitx();

    QF1D512_WriteConfigByte0( 0x03, 0x00 );
    QF1D512_WriteConfigByte0( 0x03, 0x03 );
    QF1D512_WriteConfigByte1( 0x03, 0x00 );
    QF1D512_WriteConfigByte1( 0x03, 0x03 );

    A_B_Switch_State = 1;

    PioSetDir((1<<6), 1);    /* Make PIO6 an output */
    PioSet((1<<6), (0<<6)); /* Turn off the Green LED */
    PioSetStrongBias((1<<6), (1<<6)); /* Strongly pull up the signal */
}
else
{
    QF1D512_WriteConfigByte0( 0x03, 0x00 );
    QF1D512_WriteConfigByte0( 0x03, 0x01 );
    QF1D512_WriteConfigByte1( 0x03, 0x00 );
    QF1D512_WriteConfigByte1( 0x03, 0x01 );

    A_B_Switch_State = 0;

    VolumeUp ( theHeadset ) ;
    VolumeUp ( theHeadset ) ;
    VolumeUp ( theHeadset ) ;
    VolumeUp ( theHeadset ) ;
    VolumeUp ( theHeadset ) ;

    PioSetDir((1<<6), 1);    /* Make PIO6 an output */
    PioSet((1<<6), (1<<6)); /* Turn on the Green LED */
    PioSetStrongBias((1<<6), (1<<6)); /* Strongly pull up the signal */
}
}
/*****

```

Line 294 was changed in order to mask out PIOs now used to interface to filter ICs:

```

/*      uint32 pio_bits = ((uint32)m->vreg_en_high << VREG_PIN) | ((uint32)m-
>charger_connected << CHG_PIN) | PioGet32(); */
uint32 pio_bits = ((uint32)m->vreg_en_high << VREG_PIN) | ((uint32)m->charger_connected <<
CHG_PIN) | (PioGet32() & 0xFFFF3800);

```

headset_event_handler.c

The following lines were altered:

Line 26 was added:

```
#include "QF_Bluetooth_driver.h"
```

Line 30 was added:

```
#include <pio.h>
```

Lines 42 – 49 were added:

```
void waitx(void);  
void QF1D512_WriteConfigByte0( unsigned int Address, unsigned char Value );  
void QF1D512_WriteConfigByte1( unsigned int Address, unsigned char Value );
```

```
/* static headsetTaskData app; */
```

```
extern unsigned int A_B_Switch_State;
```

In function handleUEMessage(Task task, MessageId id, Message message), the following case statement cases were changed to preclude operation of standard functions associated with buttons/functions not used in the Blue Mojo (FWD, BACK, AUX1, AUX2, etc.):

Lines 633 – 647, case EventPlay:

```
case EventPlay:          /* This is now the A/B Switch */  
/*      if (IApp->buttons_locked)  
      {  
          IIndicateEvent = FALSE ;  
          break;  
      }  
*/  
  
EVENTS_DEBUG(("EventPlay now A/B \n"));  
/* Always indicate play event as will try to connect A2DP if not already connected */  
/*      avrcpEventPlay(IApp);          */  
  
break;
```

Lines 648 – 667, case EventPause and case EventStop:

```
/*  
case EventPause:  
    if (IApp->buttons_locked)  
    {  
        IIndicateEvent = FALSE ;  
        break;  
    }  
EVENTS_DEBUG(("EventPause\n"));  
avrcpEventPause(IApp);  
break;
```

```

        case EventStop:
            if (IApp->buttons_locked)
            {
                IIndicateEvent = FALSE ;
                break;
            }
            EVENTS_DEBUG(("EventStop\n"));
            avrcpEventStop(IApp);
            break;
    */

```

Line 678, case EventSkipForward:
 /* avrcpEventSkipForward(IApp); */

Line 695, case EventSkipBackward:
 /* avrcpEventSkipBackward(IApp); */

Line 711, case EventFastForwardPress:
 /* avrcpEventFastForwardPress(IApp); */

Line 727, case EventFastForwardRelease:
 /* avrcpEventFastForwardRelease(IApp); */

Line 743, case EventFastRewindPress:
 /* avrcpEventFastRewindPress(IApp); */

Line 759, case EventFastRewindRelease:
 /* avrcpEventFastRewindRelease(IApp); */

csr_a2dp_decoder_common.c

This is a library file in C:\Stereo-Headset-SDK-2008.R1\src\lib\csr_a2dp_decoder_common_plugin\ . The following modifications were made:

Lines 118 – 122, which were these:

```

    StreamDisconnect(StreamPcmSource(0), StreamPcmSink(0));
    StreamDisconnect(StreamPcmSource(1), StreamPcmSink(1));

```

```

    PanicFalse(PcmClearRouting(0));
    PanicFalse(PcmClearRouting(1));

```

Were changed into lines 122 – 137, which are these:

```

    /* disconnect port 0 from Left DAC */
    StreamDisconnect(StreamKalimbaSource(0),StreamPcmSink(0));

```

```

    /* Disconnect port 1 from Right DAC */
    StreamDisconnect(StreamKalimbaSource(1),StreamPcmSink(1));

```

```
StreamDisconnect(StreamPcmSource(0), StreamPcmSink(2));
StreamDisconnect(StreamPcmSource(1), StreamPcmSink(3));
```

```
(void) PanicFalse(PcmClearRouting(0));
(void) PanicFalse(PcmClearRouting(1));
(void) PanicFalse(PcmClearRouting(2));
(void) PanicFalse(PcmClearRouting(3));
```

In function CsrA2dpDecoderPluginDisconnect(void), lines 223 & 224, which were these:

```
StreamDisconnect(StreamPcmSource(0), StreamPcmSink(0));
StreamDisconnect(StreamPcmSource(1), StreamPcmSink(1));
```

Were changed into lines 242 – 258, which are these:

```
/* disconnect port 0 from Left DAC */
StreamDisconnect(StreamKalimbaSource(0),StreamPcmSink(0));

/* Disconnect port 1 from Right DAC */
StreamDisconnect(StreamKalimbaSource(1),StreamPcmSink(1));
```

```
StreamDisconnect(StreamPcmSource(0), StreamPcmSink(2));
StreamDisconnect(StreamPcmSource(1), StreamPcmSink(3));
```

```
(void) PanicFalse(PcmClearRouting(0));
(void) PanicFalse(PcmClearRouting(1));
(void) PanicFalse(PcmClearRouting(2));
(void) PanicFalse(PcmClearRouting(3));
```

In function MusicConnectAudio (A2dpPluginTaskdata *task, bool stereo), lines 459 – 465, which were these:

```
PanicFalse(PcmRateAndRoute(0, PCM_NO_SYNC, DECODER->rate, (uint32) 8000,
VM_PCM_INTERNAL_A));
PanicFalse(PcmRateAndRoute(1, 0, DECODER->rate, (uint32) 8000, VM_PCM_INTERNAL_B));

/* plug port 0 into Left DAC */
PanicFalse(StreamConnect(StreamKalimbaSource(0),StreamPcmSink(0)));
/* plug port 1 into Right DAC */
PanicFalse(StreamConnect(StreamKalimbaSource(1),StreamPcmSink(1)));
```

Were changed into lines 496 – 511, which are these:

```
(void) PanicFalse(PcmRateAndRoute(0, PCM_NO_SYNC, DECODER->rate, DECODER->rate,
VM_PCM_EXTERNAL_I2S));
(void) PanicFalse(PcmRateAndRoute(1, 0, DECODER->rate, DECODER->rate,
VM_PCM_EXTERNAL_I2S));
(void) PanicFalse(PcmRateAndRoute(2,PCM_NO_SYNC,DECODER->rate, DECODER-
>rate,VM_PCM_INTERNAL_A));
(void) PanicFalse(PcmRateAndRoute(3,      2,DECODER->rate, DECODER-
```

```
>rate,VM_PCM_INTERNAL_B));
```

```
/* plug port 0 into Left DAC
```

```
PanicFalse(StreamConnect(StreamKalimbaSource(0),StreamPcmSink(0)));*/
```

```
/* plug port 1 into Right DAC
```

```
PanicFalse(StreamConnect(StreamKalimbaSource(1),StreamPcmSink(1)));*/
```

```
(void) PanicFalse(StreamConnect(StreamPcmSource(0),StreamPcmSink(2)));
```

```
(void) PanicFalse(StreamConnect(StreamPcmSource(1),StreamPcmSink(3)));
```

```
(void) PanicFalse(StreamConnect(StreamKalimbaSource(0),StreamPcmSink(0)));
```

```
(void) PanicFalse(StreamConnect(StreamKalimbaSource(1),StreamPcmSink(1)));
```