STM32F4xx Porting Overview

Release: 4.0.1
Febuary, 10, 2014

Louisville, KY www.stonestreetone.com

Table of Contents

1. Introduction

This version of Bluetopia has been pre-ported to work on the STM3240G development platform for the ST Microelectronics STM32F4xx processors. This document describes how to modify the configuration files to a support a different platform or to simply change the UART and pin mapping that is used for the HCI transport and console interfaces.

2. Configuring the HCI transport

2.1 Interrupt Driver configuration

All relevant settings for the HCI UART are contained within "HCITRCFG.h" in the "Bluetopia\hcitrans" directory. This file contains the port and signal configuration as well as a few compiler settings.

The Port settings are as follows:

Name	Default	Description
HCITR_UART	5	The UART (or USART) number to be used for HCI communication. Valid values are 1 to 6.
HCITR_TXD_PORT	C	The port and pin assignment to be used for transmitting data. This pin must be mappable to the TXD signal for the selected UART
HCITR_TXD_PIN	12	
HCITR_RXD_PORT	D	The port and pin assignment to be used for receiving data. This pin must be mappable to the RXD signal for the selected UART
HCITR_RXD_PIN	2	
HCITR_CTS_PORT	D	The port and pin assignment to be used for the Clear To Send flow control input. If hardware flow control is used (USE_SOFTWARE_CTS_RTS is not defined) then this pin must be mappable to the CTS signal for the selected UART, otherwise the pin must be used that can be mapped to an available EXTI line.
HCITR_CTS_PIN	1	
HCITR_RTS_PORT	D	The port and pin assignment to be used for the Ready To Send flow control output. If hardware flow control is used (USE_SOFTWARE_CTS_RTS is not defined) then this pin must be mappable to the RTS signal for the selected UART, otherwise any available GPIO may be used.
HCITR_RTS_PIN	0	
HCITR_RESET_PORT	C	The port and pin assignment to be used for the baseband reset (nSHUTD) signal.
HCITR_RESET_PIN	9	

Other settings:

***SUPPORT_TRANSPORT_SUSPEND*:**

This macro can be defined to support suspending the HCI transport via a call to HCITR_COMSuspend(). When the transport is suspended, it will automatically be woken up when data needs to be transmitted or on the EXTI interrupt for the CTS signal. Note that this functionality requires HCILL to be used and HCITR_COMSuspend() must only be called when HCILL indicates that it is safe.

***USE_SOFTWARE_CTS_RTS*:**

This macro can be defined to force the HCI transport to use software controlled CTS/RTS flow control. If this macro is not defined then software managed CTS/RTS will only be used for the UARTs that do not support hardware flow control (UART4 and UART5). It is recommended that Hardware flow control be used when possible. If software flow control is used, care must be taken to assure that interrupts are not disabled for extended periods or the UART receive register may overflow, losing the data.

***HCITR_CTS_IRQ_HANDLER*:**

This macro can be defined to a custom function name for the CTS IRQ handler when Software controlled CTS/RTS is used. By default this is defined to map the interrupt handler for the selected EXTI line but it may be overwritten if the interrupt overlaps with others that are handled externally. If this macro is defined to be anything other than the interrupt for the CTS EXTI line, then the function must be called on the falling edge of the CTS line.

2.2 Additional DMA configurations

The release may also contain a HCITRANS that uses DMAs that may be used instead of the default interrupt version. The DMA version will be in a separate directory named “hcitrans.dma” which can replace the default “hcitrans” directory. Note that this driver does not support software based flow control and MUST be used with a USART that includes hardware flow control.

It should also be noted that do to limitations in the ST DMA module and lack of a usable idle interrupt on the UART, the DMA driver must periodically poll to determine if data has been received. If this is not desirable, you may considering tying the USART receive pin to another input capable of resetting an internal timer. This timer can then be used to generate an idle interrupt after data has been received.

This driver requires the following configurations in addition to those required for the interrupt driver:

Name	Default	Description
HCITR_DMA_RXD_NUMBER	2	The DMA module (1 or 2), Stream and Channel that maps to the receive input for the USART used by HCITANS.
HCITR_DMA_RXD_STREAM	1	
HCITR_DMA_RXD_CHANNEL	5	
HCITR_DMA_TXD_NUMBER	2	The DMA module (1 or 2), Stream and Channel that maps to the input for the USART used by HCITANS.
HCITR_DMA_TXD_STREAM	6	
HCITR_DMA_TXD_CHANNEL	5	

3. Configuring the Platform

The Platform files (the HAL module in particular) is used by the sample applications for initialization, LED, and console functionality. All relevant settings for the HCI UART are contained within "HALCFG.h" in the "STM3240G-EVAL\Platform" directory. This file contains the port and signal configuration for the console UART and LED.

The Port settings are as follows:

Name	Default	Description
CONSOLE_UART	3	The UART (or USART) number to be used for the console interface. Valid values are 1 to 6.
CONSOLE_TXD_PORT	C	The port and pin assignment to be used for transmitting characters to the console. This pin must be mappable to the TXD signal for the selected console UART.
CONSOLE _TXD_PIN	12	
CONSOLE_RXD_PORT	C	The port and pin assignment to be used for receiving characters from the console. This pin must be mappable to the RXD signal for the selected console UART.
CONSOLE_RXD_PIN	11	
HAL_LED_PORT	G	The port and pin assignments to be used for the status LED.
HAL_LED_PIN	6	

4. File Distributions

File	Contents/Description
HCITRCFG.h	HCI Transport configuration header.
HALCFG.h	Platform configuration header.

4.1 HCI Transport configuration Header File

```
/*****< hcitrcfg.h >**********************************************************/
/*      Copyright 2012 - 2013 Stonestreet One.                                */
/*      All Rights Reserved.                                                  */
/*                                                                            */
/*  HCITRCFG - HCI Transport Layer Configuration parameters.                  */
/*                                                                            */
/*  Author:  Marcus Funk                                                      */
/*                                                                            */
/*** MODIFICATION HISTORY ****************************************************/
/*                                                                            */
/*   mm/dd/yy  F. Lastname    Description of Modification                     */
/*   --------  -----------    ------------------------------------------------*/
/*   11/08/12  M. Funk        Initial creation.                               */
/******************************************************************************/
#ifndef __HCITRCFGH__
#define __HCITRCFGH__

#include "BTAPITyp.h"            /* Bluetooth API Type Definitions.           */

#include "stm32f4xx.h"           /* STM32F register definitions.              */
#include "stm32f4xx_gpio.h"      /* STM32F GPIO control functions.            */
#include "stm32f4xx_usart.h"     /* STM32F USART control functions.           */
#include "stm32f4xx_rcc.h"       /* STM32F Clock control functions.           */
#include "stm32f4xx_exti.h"      /* STM32F Ext interrupt definitions.         */
#include "stm32f4xx_syscfg.h"    /* STM32F system config definitions.         */

   /* The following definitions define the UART/USART to be used by the */
   /* HCI transport and the pins that will be used by the UART.  Please */
   /* consult the processor's documentation to determine what pins are  */
   /* available for the desired UART.                                   */
   /* * NOTE * The TXD and RXD pins MUST be map-able to the selected    */
   /*          UART.  Additionally, if hardware flow control is desired,*/
   /*          the RTS and CTS pins must also be map-able to the        */
   /*          selected UART.  If software managed flow is used, RTS may*/
   /*          be any available GPIO but CTS must be a GPIO that can be */
   /*          mapped to an available EXTI line.  The RESET pin may be  */
   /*          any available GPIO.                                      */
#define HCITR_UART        5

#define HCITR_TXD_PORT    C
#define HCITR_TXD_PIN     12

#define HCITR_RXD_PORT    D
#define HCITR_RXD_PIN     2

#define HCITR_RTS_PORT    D
#define HCITR_RTS_PIN     1

#define HCITR_CTS_PORT    D
#define HCITR_CTS_PIN     0

#define HCITR_RESET_PORT  C
#define HCITR_RESET_PIN   9

   /* Define the following to enable suspend functionality within       */
   /* HCITRANS.  This will shut down the UART when HCITR_COMSuspend() is*/
   /* called and resume normal functionality when data is received in   */
   /* transmitted.                                                      */
   /* * NOTE * This functionality requires using a lower power protocol */
   /*          such as HCILL and the UART should only be suspended when */
   /*          indicated it is safe to do so by the protocol driver.    */
#define SUPPORT_TRANSPORT_SUSPEND

   /* Define the following to explicitly enable software controled      */
   /* CTS/RTS.  This is necessary if the pin specified for the CTS and  */
   /* RTS control lines are not map-able to the specified USART.        */
   /* * NOTE * This is defined automatically for the UARTs that do not  */
   /*          support hardware flow control.                           */
```

```
// #define USE_SOFTWARE_CTS_RTS

   /* Define the following if software managed flow control is being    */
   /* used and the NVIC interrupt for the CTS EXTI line is being also   */
   /* used by another EXTI line.  The specified function can then be    */
   /* called by a global interrupt handler when the CTS EXTI interrupt  */
   /* occurs.                                                           */
   /* * NOTE * If defined when software managed flow control is used,   */
   /*          the NVIC interrupt associated with the CTS EXTI line MUST*/
   /*          be handled externally and call this function.  If not    */
   /*          defined, the interrupt will be handled directly by       */
   /*          HCITRANS.                                                */
// #define HCITR_CTS_IRQ_HANDLER HCITR_CTS_IrqHandler



/*********************************************************************/
/* !!!DO NOT MODIFY PAST THIS POINT!!!                               */
/*********************************************************************/

   /* The following section builds the macros that can be used with the */
   /* STM32F standard peripheral libraries based on the above           */
   /* configuration.                                                    */

   /* Standard C style concatenation macros                             */
#define DEF_CONCAT2(_x_, _y_)          __DEF_CONCAT2__(_x_, _y_)
#define __DEF_CONCAT2__(_x_, _y_)      _x_ ## _y_

#define DEF_CONCAT3(_x_, _y_, _z_)     __DEF_CONCAT3__(_x_, _y_, _z_)
#define __DEF_CONCAT3__(_x_, _y_, _z_) _x_ ## _y_ ## _z_

   /* Determine the Peripheral bus that is used by the UART.            */
#if ((HCITR_UART == 1) ||(HCITR_UART == 6))
   #define HCITR_UART_APB             2
#else
   #define HCITR_UART_APB             1
#endif

   /* Determine the type of UART.                                       */
#if ((HCITR_UART == 1) || (HCITR_UART == 2) || (HCITR_UART == 3) || (HCITR_UART == 6))

   #define HCITR_UART_TYPE             USART

#elif ((HCITR_UART == 4) || (HCITR_UART == 5))

   #define HCITR_UART_TYPE             UART

   /* These UARTs do not support hardware flow control so make sure that*/
   /* software managed flow control is used.                            */
   #ifndef USE_SOFTWARE_CTS_RTS
      #define USE_SOFTWARE_CTS_RTS
   #endif

#else
   #error Unknown HCITR_UART
#endif

   /* The following section builds the macro names that can be used with*/
   /* the STM32F standard peripheral libraries.                         */

   /* UART control mapping.                                             */
#define HCITR_UART_BASE                (DEF_CONCAT2(HCITR_UART_TYPE, HCITR_UART))
#define HCITR_UART_IRQ                 (DEF_CONCAT3(HCITR_UART_TYPE, HCITR_UART, _IRQn))
#define HCITR_UART_IRQ_HANDLER         (DEF_CONCAT3(HCITR_UART_TYPE, HCITR_UART,
_IRQHandler))

#define HCITR_UART_RCC_PERIPH_CLK_CMD  (DEF_CONCAT3(RCC_APB, HCITR_UART_APB,
PeriphClockCmd))
#define HCITR_UART_RCC_PERIPH_CLK_BIT  (DEF_CONCAT3(DEF_CONCAT3(RCC_APB, HCITR_UART_APB,
Periph_), HCITR_UART_TYPE, HCITR_UART))
```

```
#define HCITR_UART_GPIO_AF              (DEF_CONCAT3(GPIO_AF_, HCITR_UART_TYPE,
HCITR_UART))

   /* GPIO mapping.                                                     */
#define HCITR_TXD_GPIO_PORT             (DEF_CONCAT2(GPIO, HCITR_TXD_PORT))
#define HCITR_RXD_GPIO_PORT             (DEF_CONCAT2(GPIO, HCITR_RXD_PORT))
#define HCITR_RTS_GPIO_PORT             (DEF_CONCAT2(GPIO, HCITR_RTS_PORT))
#define HCITR_CTS_GPIO_PORT             (DEF_CONCAT2(GPIO, HCITR_CTS_PORT))
#define HCITR_RESET_GPIO_PORT           (DEF_CONCAT2(GPIO, HCITR_RESET_PORT))

#define HCITR_TXD_GPIO_AHB_BIT          (DEF_CONCAT2(RCC_AHB1Periph_GPIO, HCITR_TXD_PORT))
#define HCITR_RXD_GPIO_AHB_BIT          (DEF_CONCAT2(RCC_AHB1Periph_GPIO, HCITR_RXD_PORT))
#define HCITR_RTS_GPIO_AHB_BIT          (DEF_CONCAT2(RCC_AHB1Periph_GPIO, HCITR_RTS_PORT))
#define HCITR_CTS_GPIO_AHB_BIT          (DEF_CONCAT2(RCC_AHB1Periph_GPIO, HCITR_CTS_PORT))
#define HCITR_RESET_GPIO_AHB_BIT        (DEF_CONCAT2(RCC_AHB1Periph_GPIO,
HCITR_RESET_PORT))

   /* Interrupt mapping.                                                */
#if (defined(SUPPORT_TRANSPORT_SUSPEND) || defined(USE_SOFTWARE_CTS_RTS))

   #if (HCITR_CTS_PIN < 5)
      #define HCITR_CTS_EXTI_NUMBER    HCITR_CTS_PIN
   #elif (HCITR_CTS_PIN < 10)
      #define HCITR_CTS_EXTI_NUMBER    9_5
   #elif (HCITR_CTS_PIN < 15)
      #define HCITR_CTS_EXTI_NUMBER    15_10
   #endif

   /* NOTE: "EXTI" is defined in the STM32F std periph headers so can   */
   /* not be used directly.                                             */
   #define HCITR_CTS_IRQ               (DEF_CONCAT3(EXT, DEF_CONCAT2(I,
HCITR_CTS_EXTI_NUMBER), _IRQn))
   #define HCITR_CTS_EXTI_PORT          (DEF_CONCAT2(EXTI_PortSourceGPIO, HCITR_CTS_PORT))
   #define HCITR_CTS_IRQ_LINE           (DEF_CONCAT2(EXTI_Line, HCITR_CTS_PIN))

   #ifndef HCITR_CTS_IRQ_HANDLER
      #define HCITR_CTS_IRQ_HANDLER    (DEF_CONCAT3(EXT, DEF_CONCAT2(I,
HCITR_CTS_EXTI_NUMBER), _IRQHandler))
   #endif

#endif

#endif
```

4.2 HCI Transport DMA configuration Header File

```
/*****< hcitrcfg.h >*********************************************************/
/*      Copyright 2012 - 2013 Stonestreet One.                               */
/*      All Rights Reserved.                                                 */
/*                                                                           */
/*  HCITRCFG - HCI Transport Layer Configuration parameters.                 */
/*                                                                           */
/*  Author:  Marcus Funk                                                     */
/*                                                                           */
/*** MODIFICATION HISTORY ****************************************************/
/*                                                                           */
/*   mm/dd/yy  F. Lastname    Description of Modification                    */
/*   --------  -----------    ------------------------------------------------*/
/*   11/08/12  M. Funk        Initial creation.                              */
/*****************************************************************************/
#ifndef __HCITRCFGH__
#define __HCITRCFGH__

#include "BTAPITyp.h"            /* Bluetooth API Type Definitions.           */

#include "stm32f4xx.h"           /* STM32F register definitions.              */
#include "stm32f4xx_gpio.h"      /* STM32F GPIO control functions.            */
#include "stm32f4xx_usart.h"     /* STM32F USART control functions.           */
#include "stm32f4xx_rcc.h"       /* STM32F Clock control functions.           */
#include "stm32f4xx_dma.h"       /* STM32F DMA control functions.             */
#include "stm32f4xx_exti.h"      /* STM32F Ext interrupt definitions.         */
#include "stm32f4xx_syscfg.h"    /* STM32F system config definitions.         */

   /* The following definitions define the UART/USART to be used by the */
   /* HCI transport and the pins that will be used by the UART.  Please */
   /* consult the processor's documentation to determine what pins are  */
   /* available for the desired UART.                                   */
   /* * NOTE * The TXD, RXD, RTS and CTS pins MUST be map-able to the   */
   /*          selected UART.  The RESET pin may be any available GPIO. */
   /* * NOTE * The DMA settinsg (Number = 1 or 2, Stream and channel)   */
   /*          must map to the RXD and TXD streams for the selected     */
   /*          UART.                                                    */
#define HCITR_UART               6

#define HCITR_TXD_PORT           G
#define HCITR_TXD_PIN            14

#define HCITR_RXD_PORT           C
#define HCITR_RXD_PIN            7

#define HCITR_RTS_PORT           G
#define HCITR_RTS_PIN            8

#define HCITR_CTS_PORT           G
#define HCITR_CTS_PIN            13

#define HCITR_RESET_PORT         C
#define HCITR_RESET_PIN          9

   /* The following definitons define the DMA infomation for receive and*/
   /* transmit on the HCI UART.  This includes the DMA number (either 1 */
   /* or 2) as well as the stream and channel.                          */
   /* * NOTE * The DMA information MUST map to the receive and transmit */
   /*          DMA for the specified UART (see the DMA sections of the  */
   /*          processor's User Manual).                                */
#define HCITR_DMA_RXD_NUMBER     2
#define HCITR_DMA_RXD_STREAM     1
#define HCITR_DMA_RXD_CHANNEL    5

#define HCITR_DMA_TXD_NUMBER     2
#define HCITR_DMA_TXD_STREAM     6
#define HCITR_DMA_TXD_CHANNEL    5
```

```
   /* Define the following to enable suspend functionality within       */
   /* HCITRANS.  This will shut down the UART when HCITR_COMSuspend() is*/
   /* called and resume normal functionality when data is received in   */
   /* transmitted.                                                      */
   /* * NOTE * This functionality requires using a lower power protocol */
   /*          such as HCILL and the UART should only be suspended when */
   /*          indicated it is safe to do so by the protocol driver.    */
#define SUPPORT_TRANSPORT_SUSPEND

   /* Define the following if software managed flow control is being    */
   /* used and the NVIC interrupt for the CTS EXTI line is being also   */
   /* used by another EXTI line.  The specified function can then be    */
   /* called by a global interrupt handler when the CTS EXTI interrupt  */
   /* occurs.                                                           */
   /* * NOTE * If defined when software managed flow control is used,   */
   /*          the NVIC interrupt associated with the CTS EXTI line MUST*/
   /*          be handled externally and call this function.  If not    */
   /*          defined, the interrupt will be handled directly by       */
   /*          HCITRANS.                                                */
// #define HCITR_CTS_IRQ_HANDLER HCITR_CTS_IrqHandler

   /* Define the following to enable debug logging of HCI traffic.  If  */
   /* this macro is defined, all incomming and outgoing traffic will be */
   /* logged via BTPS_OutputMessage().                                  */
// #define HCITR_ENABLE_DEBUG_LOGGING


/***********************************************************************/
/* !!!DO NOT MODIFY PAST THIS POINT!!!                                 */
/***********************************************************************/

   /* The following section builds the macros that can be used with the */
   /* STM32F standard peripheral libraries based on the above           */
   /* configuration.                                                    */

   /* Standard C style concatenation macros                             */
#define DEF_CONCAT2(_x_, _y_)          __DEF_CONCAT2__(_x_, _y_)
#define __DEF_CONCAT2__(_x_, _y_)      _x_ ## _y_

#define DEF_CONCAT3(_x_, _y_, _z_)     __DEF_CONCAT3__(_x_, _y_, _z_)
#define __DEF_CONCAT3__(_x_, _y_, _z_) _x_ ## _y_ ## _z_

   /* Determine the Peripheral bus that is used by the UART.            */
#if ((HCITR_UART == 1) ||(HCITR_UART == 6))
   #define HCITR_UART_APB              2
#else
   #define HCITR_UART_APB              1
#endif

   /* Determine the type of UART.                                       */
#if ((HCITR_UART == 1) || (HCITR_UART == 2) || (HCITR_UART == 3) || (HCITR_UART == 6))

   #define HCITR_UART_TYPE             USART

#else
   #error Unknown HCITR_UART or UART not supported
#endif

   /* The following section builds the macro names that can be used with*/
   /* the STM32F standard peripheral libraries.                         */

   /* UART control mapping.                                             */
#define HCITR_UART_BASE                (DEF_CONCAT2(HCITR_UART_TYPE, HCITR_UART))

#define HCITR_UART_RCC_PERIPH_CLK_CMD  (DEF_CONCAT3(RCC_APB, HCITR_UART_APB,
PeriphClockCmd))
#define HCITR_UART_RCC_PERIPH_CLK_BIT  (DEF_CONCAT3(DEF_CONCAT3(RCC_APB, HCITR_UART_APB,
Periph_), HCITR_UART_TYPE, HCITR_UART))
#define HCITR_UART_GPIO_AF             (DEF_CONCAT3(GPIO_AF_, HCITR_UART_TYPE,
HCITR_UART))
```

```
   /* GPIO mapping.                                                      */
#define HCITR_TXD_GPIO_PORT            (DEF_CONCAT2(GPIO, HCITR_TXD_PORT))
#define HCITR_RXD_GPIO_PORT            (DEF_CONCAT2(GPIO, HCITR_RXD_PORT))
#define HCITR_RTS_GPIO_PORT            (DEF_CONCAT2(GPIO, HCITR_RTS_PORT))
#define HCITR_CTS_GPIO_PORT            (DEF_CONCAT2(GPIO, HCITR_CTS_PORT))
#define HCITR_RESET_GPIO_PORT          (DEF_CONCAT2(GPIO, HCITR_RESET_PORT))

#define HCITR_TXD_GPIO_AHB_BIT         (DEF_CONCAT2(RCC_AHB1Periph_GPIO, HCITR_TXD_PORT))
#define HCITR_RXD_GPIO_AHB_BIT         (DEF_CONCAT2(RCC_AHB1Periph_GPIO, HCITR_RXD_PORT))
#define HCITR_RTS_GPIO_AHB_BIT         (DEF_CONCAT2(RCC_AHB1Periph_GPIO, HCITR_RTS_PORT))
#define HCITR_CTS_GPIO_AHB_BIT         (DEF_CONCAT2(RCC_AHB1Periph_GPIO, HCITR_CTS_PORT))
#define HCITR_RESET_GPIO_AHB_BIT       (DEF_CONCAT2(RCC_AHB1Periph_GPIO,
HCITR_RESET_PORT))

   /* DMA Mapping.                                                       */
#define HCITR_TXD_DMA_AHB_BIT          (DEF_CONCAT2(RCC_AHB1Periph_DMA,
HCITR_DMA_TXD_NUMBER))
#define HCITR_TXD_DMA_CHANNEL          (DEF_CONCAT2(DMA_Channel_, HCITR_DMA_TXD_CHANNEL))
#define HCITR_TXD_DMA_STREAM           (DEF_CONCAT2(DEF_CONCAT3(DMA,
HCITR_DMA_TXD_NUMBER, _Stream), HCITR_DMA_TXD_STREAM))
#define HCITR_RXD_DMA_AHB_BIT          (DEF_CONCAT2(RCC_AHB1Periph_DMA,
HCITR_DMA_RXD_NUMBER))
#define HCITR_RXD_DMA_CHANNEL          (DEF_CONCAT2(DMA_Channel_, HCITR_DMA_RXD_CHANNEL))
#define HCITR_RXD_DMA_STREAM           (DEF_CONCAT2(DEF_CONCAT3(DMA,
HCITR_DMA_RXD_NUMBER, _Stream), HCITR_DMA_RXD_STREAM))

#define HCITR_TXD_DMA_FLAG_TCIF        (DEF_CONCAT2(DMA_FLAG_TCIF, HCITR_DMA_TXD_STREAM))
#define HCITR_TXD_DMA_FLAG_HTIF        (DEF_CONCAT2(DMA_FLAG_HTIF, HCITR_DMA_TXD_STREAM))
#define HCITR_TXD_DMA_FLAG_TEIF        (DEF_CONCAT2(DMA_FLAG_TEIF, HCITR_DMA_TXD_STREAM))
#define HCITR_TXD_DMA_FLAG_DMEIF       (DEF_CONCAT2(DMA_FLAG_DMEIF,
HCITR_DMA_TXD_STREAM))
#define HCITR_TXD_DMA_FLAG_FEIF        (DEF_CONCAT2(DMA_FLAG_FEIF, HCITR_DMA_TXD_STREAM))

#define HCITR_RXD_DMA_FLAG_TCIF        (DEF_CONCAT2(DMA_FLAG_TCIF, HCITR_DMA_RXD_STREAM))
#define HCITR_RXD_DMA_FLAG_HTIF        (DEF_CONCAT2(DMA_FLAG_HTIF, HCITR_DMA_RXD_STREAM))
#define HCITR_RXD_DMA_FLAG_TEIF        (DEF_CONCAT2(DMA_FLAG_TEIF, HCITR_DMA_RXD_STREAM))
#define HCITR_RXD_DMA_FLAG_DMEIF       (DEF_CONCAT2(DMA_FLAG_DMEIF,
HCITR_DMA_RXD_STREAM))
#define HCITR_RXD_DMA_FLAG_FEIF        (DEF_CONCAT2(DMA_FLAG_FEIF, HCITR_DMA_RXD_STREAM))

#define HCITR_TXD_IRQ                  (DEF_CONCAT3(DEF_CONCAT3(DMA,
HCITR_DMA_TXD_NUMBER, _Stream), HCITR_DMA_TXD_STREAM, _IRQn))
#define HCITR_RXD_IRQ                  (DEF_CONCAT3(DEF_CONCAT3(DMA,
HCITR_DMA_RXD_NUMBER, _Stream), HCITR_DMA_RXD_STREAM, _IRQn))
#define HCITR_TXD_IRQHandler           (DEF_CONCAT3(DEF_CONCAT3(DMA,
HCITR_DMA_TXD_NUMBER, _Stream), HCITR_DMA_TXD_STREAM, _IRQHandler))
#define HCITR_RXD_IRQHandler           (DEF_CONCAT3(DEF_CONCAT3(DMA,
HCITR_DMA_RXD_NUMBER, _Stream), HCITR_DMA_RXD_STREAM, _IRQHandler))

   /* Location of the Data register for the UART in use.                */
#define HCITR_UART_DR_REGISTER_ADDRESS (((unsigned int)(DEF_CONCAT3(HCITR_UART_TYPE,
HCITR_UART, _BASE))) + 4)

   /* Interrupt mapping.                                                 */
#ifdef SUPPORT_TRANSPORT_SUSPEND

   #if (HCITR_CTS_PIN   < 5)
      #define HCITR_CTS_EXTI_NUMBER    HCITR_CTS_PIN
   #elif (HCITR_CTS_PIN < 10)
      #define HCITR_CTS_EXTI_NUMBER    9_5
   #elif (HCITR_CTS_PIN < 16)
      #define HCITR_CTS_EXTI_NUMBER    15_10
   #endif

   /* NOTE: "EXTI" is defined in the STM32F std periph headers so can   */
   /* not be used directly.                                              */
   #define HCITR_CTS_IRQ               (DEF_CONCAT3(EXT, DEF_CONCAT2(I,
HCITR_CTS_EXTI_NUMBER), _IRQn))
   #define HCITR_CTS_EXTI_PORT         (DEF_CONCAT2(EXTI_PortSourceGPIO, HCITR_CTS_PORT))
   #define HCITR_CTS_IRQ_LINE          (DEF_CONCAT2(EXTI_Line, HCITR_CTS_PIN))
   #ifndef HCITR_CTS_IRQ_HANDLER
      #define HCITR_CTS_IRQ_HANDLER    (DEF_CONCAT3(EXT, DEF_CONCAT2(I, 
HCITR_CTS_EXTI_NUMBER), _IRQHandler))
   #endif

#endif

#endif
```

4.3 Platform Configuration Header File

```
/*****< halcfg.h >************************************************************/
/*      Copyright 2012 - 2013 Stonestreet One.                               */
/*      All Rights Reserved.                                                 */
/*                                                                           */
/*  HALCFG - Hardware Abstraction Layer Configuration parameters.            */
/*                                                                           */
/*  Author:  Marcus Funk                                                     */
/*                                                                           */
/*** MODIFICATION HISTORY ****************************************************/
/*                                                                           */
/*   mm/dd/yy  F. Lastname    Description of Modification                    */
/*   --------  -----------    ------------------------------------------------*/
/*   11/08/12  M. Funk        Initial creation.                              */
/*****************************************************************************/
#ifndef __HCITRCFGH__
#define __HCITRCFGH__

#include "BTAPITyp.h"            /* Bluetooth API Type Definitions.           */

#include "stm32f4xx.h"           /* STM32F register definitions.              */
#include "stm32f4xx_gpio.h"      /* STM32F GPIO control functions.            */
#include "stm32f4xx_usart.h"     /* STM32F USART control functions.           */
#include "stm32f4xx_rcc.h"       /* STM32F Clock control functions.           */
#include "stm32f4xx_exti.h"      /* STM32F Ext interrupt definitions.         */
#include "stm32f4xx_syscfg.h"    /* STM32F system config definitions.         */

   /* The following definitions define the UART/USART to be used by the */
   /* HCI transport and the pins that will be used by the UART.  Please */
   /* consult the processor's documentation to determine what pins are  */
   /* available for the desired UART.                                   */
   /* * NOTE * The TXD and RXD pins MUST be map-able to the selected    */
   /*          UART.  Additionally, if hardware flow control is desired,*/
   /*          the RTS and CTS pins must also be map-able to the        */
   /*          selected UART.  If software managed flow is used, RTS may*/
   /*          be any available GPIO but CTS must be a GPIO that can be */
   /*          mapped to an available EXTI line.  The RESET pin may be  */
   /*          any available GPIO.                                      */
#define CONSOLE_UART      3

#define CONSOLE_TXD_PORT  C
#define CONSOLE_TXD_PIN   10

#define CONSOLE_RXD_PORT  C
#define CONSOLE_RXD_PIN   11

#define HAL_LED_PORT      G
#define HAL_LED_PIN       6

/*********************************************************************/
/* !!!DO NOT MODIFY PAST THIS POINT!!!                               */
/*********************************************************************/

   /* The following section builds the macros that can be used with the */
   /* STM32F standard peripheral libraries based on the above           */
   /* configuration.                                                    */

/* Standard C style concatenation macros                            */
#define DEF_CONCAT2(_x_, _y_)          __DEF_CONCAT2__(_x_, _y_)
#define __DEF_CONCAT2__(_x_, _y_)      _x_ ## _y_

#define DEF_CONCAT3(_x_, _y_, _z_)     __DEF_CONCAT3__(_x_, _y_, _z_)
#define __DEF_CONCAT3__(_x_, _y_, _z_) _x_ ## _y_ ## _z_


   /* Determine the Peripheral bus that is used by the UART.            */
#if ((CONSOLE_UART == 1) ||(CONSOLE_UART == 6))
   #define CONSOLE_UART_APB              2
#else
```

```
   #define CONSOLE_UART_APB              1
#endif

   /* Determine the type of UART.                                       */
#if ((CONSOLE_UART == 1) || (CONSOLE_UART == 2) || (CONSOLE_UART == 3) || (CONSOLE_UART
== 6))

   #define CONSOLE_UART_TYPE             USART

#elif ((HCITR_UART == 4) || (HCITR_UART == 5))

   #define CONSOLE_UART_TYPE             UART

#else

   #error Unknown CONSOLE_UART

#endif

   /* The following section builds the macro names that can be used with*/
   /* the STM32F standard peripheral libraries.                         */

   /* UART control mapping.                                             */
#define CONSOLE_UART_BASE               (DEF_CONCAT2(CONSOLE_UART_TYPE, CONSOLE_UART))
#define CONSOLE_UART_IRQ                (DEF_CONCAT3(CONSOLE_UART_TYPE, CONSOLE_UART,
_IRQn))
#define CONSOLE_UART_IRQ_HANDLER        (DEF_CONCAT3(CONSOLE_UART_TYPE, CONSOLE_UART,
_IRQHandler))

#define CONSOLE_UART_RCC_PERIPH_CLK_CMD  (DEF_CONCAT3(RCC_APB, CONSOLE_UART_APB,
PeriphClockCmd))
#define CONSOLE_UART_RCC_PERIPH_CLK_BIT  (DEF_CONCAT3(DEF_CONCAT3(RCC_APB,
CONSOLE_UART_APB, Periph_), CONSOLE_UART_TYPE, CONSOLE_UART))
#define CONSOLE_UART_GPIO_AF            (DEF_CONCAT3(GPIO_AF_, CONSOLE_UART_TYPE,
CONSOLE_UART))

   /* GPIO mapping.                                                     */
#define CONSOLE_TXD_GPIO_PORT           (DEF_CONCAT2(GPIO, CONSOLE_TXD_PORT))
#define CONSOLE_RXD_GPIO_PORT           (DEF_CONCAT2(GPIO, CONSOLE_RXD_PORT))
#define HAL_LED_GPIO_PORT               (DEF_CONCAT2(GPIO, HAL_LED_PORT))

#define CONSOLE_TXD_GPIO_AHB_BIT        (DEF_CONCAT2(RCC_AHB1Periph_GPIO,
CONSOLE_TXD_PORT))
#define CONSOLE_RXD_GPIO_AHB_BIT        (DEF_CONCAT2(RCC_AHB1Periph_GPIO,
CONSOLE_RXD_PORT))
#define HAL_LED_GPIO_AHB_BIT            (DEF_CONCAT2(RCC_AHB1Periph_GPIO, HAL_LED_PORT))

#endif
```