RemoTI HID Dongle Developer's Guide

Document Number: SWRA298

TABLE OF CONTENTS

1 REFERENCES 3

2 INTRODUCTION 4

2.1 PURPOSE 4
2.2 SCOPE 4

3 REMOTI HID DONGLE APPLICATION 4

3.1 FEATURES 4
3.2 BUILD CONFIGURATIONS 4
3.3 FILES 5
3.4 ARCHITECTURE 8
3.5 HID INTERFACES 10

4 CUSTOMIZING HID INTERFACES 16

5 KEY EVENT HANDLER 17

6 FLASH PAGE MAP AND MEMORY MAP 17

7 STACK AND HEAP 19

8 IEEE ADDRESS 22

9 NETWORK LAYER CONFIGURATION 25

10 DMA, PERIPHERAL IO AND TIMERS 26

11 GENERAL INFORMATION 27

11.1 DOCUMENT HISTORY 27

12 ADDRESS INFORMATION 27

13 TI WORLDWIDE TECHNICAL SUPPORT 27

Acronyms and Definitions

ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
API	Application Programming Interface
CCM	Counter with CBC-MAC (CCM), a mode of operation for cryptographic block ciphers
CERC	Consumer Electronics Remote Control, name of a profile of Zigbee RF4CE
DMA	Direct Memory Access
DV	Dynamic Value
FIFO	First In First Out
GPIO	General Purpose Input Output
HAL	Hardware Abstraction Layer
HID	Human Interface Devices
IAR	IAR Systems, a software development tool vendor
IDATA	Internal Data memory
IEEE	Institute of Electrical & Electronics Engineers, Inc.
IO	Input Output
ISR	Interrupt Service Routine
LED	Light Emitting Diode
NIB	Network Information Base
NV	Non-Volatile, or Non-volatile memory
NWK	Network
OSAL	Operating System Abstraction Layer
SRAM	Static Random Access Memory
TI	Texas Instruments Incorporated
UART	Universal Asynchronous Receiver-Transmitter
XDATA	eXternal Data memory
Zigbee RF4CE	An 802.15.4 based remote control protocol standard

1 References

[1] RemoTI Developer’s Guide, SWRU198
[2] RemoTI API, SWRA268
[3] HAL Drivers API, SWRA193
[4] OSAL API, SWRA194
[5] CC253X System-on-Chip Solution for 2.4-GHz IEEE 802.15.4/ZigBee/RF4CE User’s Guide, SWRU191
[6] RemoTI Sample Applications User’s Guide, SWRU201
[7] Universal Serial Bus Specification, Revision 2.0, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips
[8] Device Class Definition for Human Interface Devices (HID), Version 1.11, USB Implementer’s Forum
[9] HID Usage Tables, Version 1.12, USB Implementer’s Forum

2 Introduction

2.1 Purpose

This document explains the RemoTI HID dongle application and topics related to customizing the application.

2.2 Scope

This document describes concepts and settings of the Texas Instruments RemoTI release with respect to Zigbee RF4CE target node HID dongle development using the RemoTI release and the CC2531 USB dongle. Readers of this document are expected to be familiar with Zigbee RF4CE, USB, HID as well as CC2531.

As to the general concept of Zigbee RF4CE and RemoTI architecture, please refer to [1].

As to the specifications of USB, HID and HID usage tables, please refer to [7], [8] and [9].

Please refer to [5] for CC2531 details.

3 RemoTI HID Dongle Application

The RemoTI development kit includes a HID dongle application. This chapter describes the features of the application and the organization of the source code and project files.

3.1 Features

The following summarize the features of the RemoTI HID dongle application:

- Compliance with Zigbee RF4CE network layer specification, as a target node
- Compliacne with Zigbee RF4CE CERC profile, as a target node
- Zigbee RF4CE network layer security
- HID-compliant keyboard
- HID-compliant consumer control device
- Two activity LEDs

3.2 Build configurations

RemoTI HID dongle application project is located in the Projects\RemoTI\HidDongle\CC2531USB folder of the RemoTI software installation.

Table 1 explains compile flags (preprocessor defined symbols) used in the project configuration.

Table 1 – Compile flags

Compile Flag	Description
xPOWER_SAVING	This compile flag is added to explicitly indicate that POWER_SAVING compile flag must not be added. When POWER_SAVING compile flag is defined, CC2531

Compile Flag	Description
	power modes 2 and 3 are enabled. Power mode 2 and 3 are not compatible with USB operation and hence POWER_SAVING compile flag must not be defined.
GENERIC=__generic	This compile flag shall always be defined as GENERIC=__generic to be compatible with the RemoTI library files. The compile flag was devised to add IAR specific compiler keyword to certain function parameters.
HAL_UART	This compile flag is and must be set to FALSE to indicate that UART driver is not in use.
HAL_UART_DMA	This compile flag is and must be set to 0 to indicate that DMA based UART is not in use.
HAL_UART_ISR	This compile flag is and must be set to 0 to indicate that ISR based UART is not in use.
HAL_UART_USB	This compile flag is and must be set to 0 to indicate that virtual UART over USB is not in use.
HAL_UART_USB_SUSPEND	This compile flag is and must be set to TRUE to indicate that USB suspend mode is supported.
HAL_HID	This compile flag is and must be set to TRUE to indicate that HID class is in use for the USB.

Besides the compile flags, other settings such as code model were also set to fit the configuration. For instance, the banked code model is used.

3.3 Files

C source files and library files are explained in Table 2 in the order they appear in the IAR workspace window. Note that there are more files than those listed in the table, such as C header files that define constants and function prototypes. Even workspace project itself does not list all header files referenced by the C files.

Note that certain driver modules are included although they are not actually used, just for potential use of the drivers by a custom application.

Table 2 – Project files

File name	Description

File name	Description
Application	
hidapp.c	Application file.
rcn_config.c	Network layer configuration file. The file contains global variables with initial values which are used as configuration parameters by RemoTI network layer. The configuration parameters are explained in chapter 9.
CLIB	
chipcon_cstartup.s51	Assembly routines to override default C libraries for banked code
HAL / USB HID class specific modules	
usb_class_requests.c	USB class request handler specific to USB HID class
usb_firmware_library_config.c	USB library configuration
usb_hid.c	USB HID class specific application support module, such as initialization routine and polling routine, which are used by hal_drivers.c module. It also contains keyboard and mouse event handler functions but they are not in use by RemoTI HID dongle application.
usb_hid_descriptor.s51	USB descriptors specific to RemoTI HID dongle
usb_hid_hooks.c	hook functions for various USB request processing, specific to USB HID class
usb_hid_reports.c	Library of HID report generation functions. Only subset of the reports are used by USB HID dongle application.
HAL / USB generic firmware library for CC2531	
usb_board_config.h	Collection of macros abstracting the hardware details for USB control.
usb_interrupt.c	The USB interrupt initialization routine and the USB interrupt service routine
usb_suspend.c	USB suspend mode related subroutines.

File name	Description
usb_descriptor_parser.c	Parser for USB descriptor structures
usb_firmware.c	Main interface routines for USB generic library
usb_standard_request.c	Handlers for USB standard requests
HAL	
hal_assert.c	HAL assertion library
hal_drivers.c	Entry point for congregation of HAL drivers, such as initialization for all HAL drivers, HAL task, as an OSAL task, entry point (event handler) and polling entry point.
hal_adc.c	ADC device driver
hal_aes.c	AES device driver
hal_board_cfg.h	RemoTI hardware platform specific configuration parameters and macros used by HAL. Application may also use HAL feature flags (HAL_KEY, HAL_LED, etc).
hal_ccm.c	CCM implementation using AES device driver
hal_dma.c	DMA device driver
hal_key.c	Key switch driver
hal_led.c	LED driver (not in use)
hal_sleep.c	Sleep mode (PM1, PM2 and PM3) control implementation. This module is not used but it has to be included in project in order to link with OSAL power saving module.
hal_timer.c	Timer module for hardware timer 1, 3 and 4. This module is not used.
hal_flashRtiCc2530.c	Flash device driver
Libraries	
rcnsuper-CC2530-banked.lib	RemoTI network layer library built for banked code model.

File name	Description
OSAL	
OSAL.c	OSAL implementation for messaging and main event handling loop
OSAL_Clock.c	OSAL clock tick implementation
OSAL_Memory.c	OSAL heap implementation
OSAL_Nv.c	OSAL non-volatile memory manager
OSAL_PwrMgr.c	OSAL power management scheme implementation
OSAL_Timers.c	OSAL timer implementation
RTI	
rti.c	RemoTI application framework implementation
rti_testmode.c	RemoTI test mode API function implementation

3.4 Architecture

This section briefly explains the interactions and relationship among the modules represented by files described in previous section.

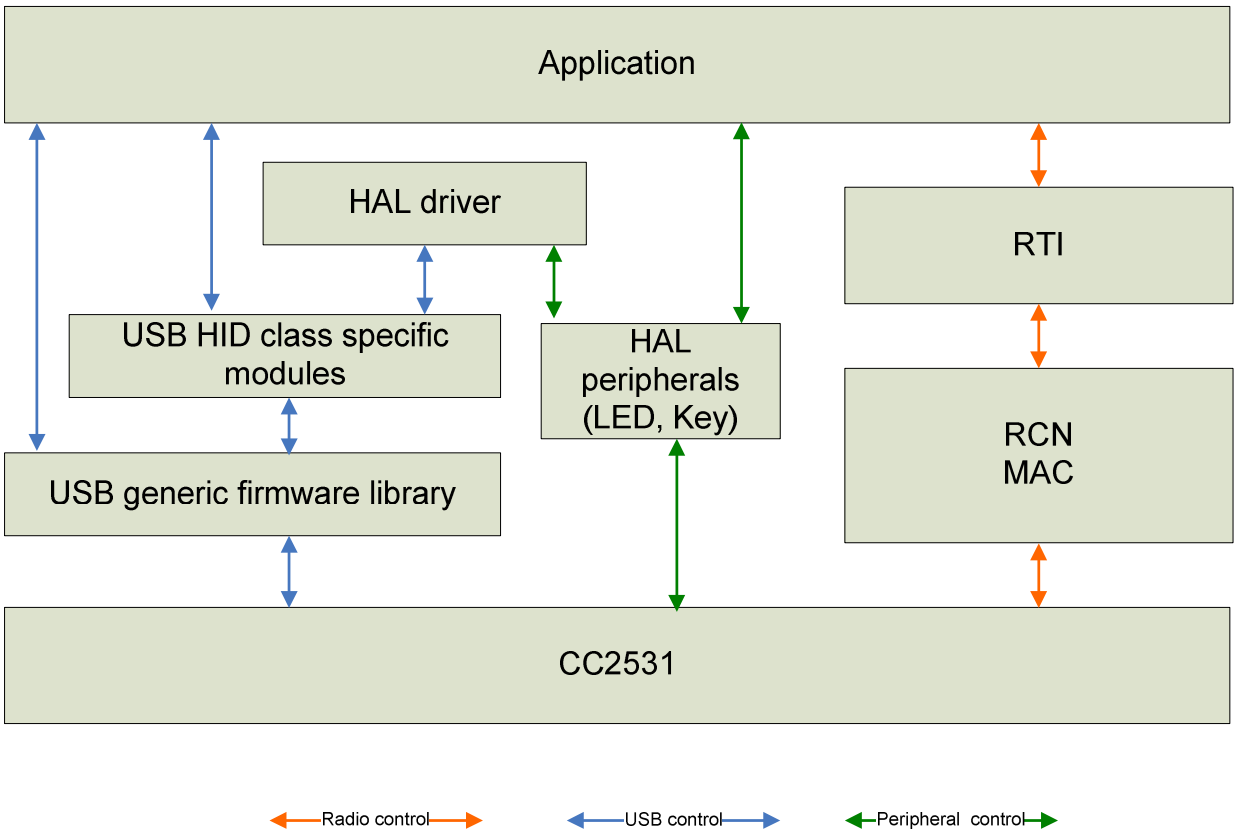

Figure 1 – RemoTI HID dongle component architecture

Figure 1 illustrates inter-module interactions within RemoTI HID dongle.
Each network processor module acronym is explained below:

- RCN – RemoTI network layer
- RTI – RemoTI application framework
- MAC – Medium Access Control layer (it is part of RemoTI network layer library in the file list)

The application controls radio through RemoTI application framework, initializing RemoTI stack as a target node. As the target node receives a CERC command from a paired controller device, the received data is handled by the application and when appropriate, the application generates HID reports using USB HID class specific modules.

When the CC2531 USB dongle receives output reports from USB host, an interrupt is triggered and handled by the USB generic firmware library. The interrupt service routine sets up event flags which could be polled by an application callback function. The application reads USB FIFO buffer from such a callback function when received data is present. The HAL driver module contains hooks to call the polling functions.

3.5 HID Interfaces

HID dongle interfaces can be found from usb_hid_descriptor.s51 file.

RemoTI HID dongle defines three interfaces: a keyboard interface, a consumer control device interface and a vendor specific interface. Besides the control end-point (end-point 0), the USB end-point 1 is used as keyboard input, the end-point 2 is used for consumer control device input and the end-point 3 is used for vendor specific commands both as input and output.

The keyboard report descriptor is defined as follows:

```
Usage Pg (Generic Desktop)
Usage (Keyboard)
Collection: (Application)

Report Size (1)
Report count (8)
Usage Pg (Key Codes)
Usage Min (224)
Usage Max (231)
Log Min (0)
Log Max (1)
Input: (Data, Variable, Absolute)

Report Count (1)
Report Size (8)
Input: (Constant)

Report Count (5)
Report Size (1)
Usage Pg (Pg# for LEDs)
Usage Min (1)
Usage Max (5)
Output: (Data, Variable, Absolute)
Report Count (1)
Report Size (3)
Output: (Constant)

Report Count (6)
Report Size (8)
Log Min (0)
Log Max (101)
Usage Pg (Key Codes)
Usage Min (0)
Usage Max (101)
Input: (Data, Array)

End Collection
```

A keyboard input report is formed as illustrated in Table 3 based on the above descriptor.

Table 3 – Keyboard input report format

Number	8	8	8	8	8	8	8	8

of bits								
Field	modifiers	reserved	Key code	Key code	Key code	Key code	Key code	Key code

Note that the USB host software must not rely on Table 3 but that it rather has to parse HID descriptors and decode input reports accordingly.

A keyboard output report is not used by the HID dongle application.

The consumer control device report descriptor is defined as follows:

```
Usage Pg (Consumer Devices)
Usage (Consumer Control)
Collection (Application)
  Usage (Numeric Key Pad)
  Collection (Logical)
    Usage Pg (Button)
    Usage Min (Button 1)
    Usage Max (Button 10)
    Logical Min (1)
    Logical Max (10)
    Report Size (4)
    Report Count (1)
    Input (Data, Ary, Abs)
  End Collection
  Usage Pg (Consumer Devices)
  Usage (Channel)
  Logical Min (-1)
  Logical Max (1)
  Report Size (2)
  Report Count (1)
  Input (Data, Var, Rel, Null)
  Usage (Volume Up)
  Usage (Volume Down)
  Logical Min (0)
  Report Size (1)
  Report Count (2)
  Input (Data, Var, Abs)
  Usage (Mute)
  Usage (Power)
  Usage (Recall Last)
  Usage (Assign Selection)
  Usage (Play)
  Usage (Pause)
  Usage (Record)
  Usage (Fast Forward)
  Usage (Rewind)
  Usage (Scan Next)
  Usage (Scan Prev)
  Usage (Stop)
  Logical Min (1)
  Logical Max (12)
  Report Size (4)
  Report Count (1)
  Input (Data, Ary, Abs)
```



```
    Usage (Selection)
    Collection (Logical)
      Usage Pg (Button)
      Usage Min (Button 1)
      Usage Max (Button 3)
      Logical Min (1)
      Logical Max (3)
      Report Size (2)
      Input (Data, Ary, Abs)
    End Collection
    Logical Min (2)
    Input (Const, Var, Abs)

  End Collection
```

Based on the above descriptor, a consumer control device input report with its report ID set to 1 must be formatted as Table 4. Note that the USB host software must not rely on Table 4 to decode input reports but that it must instead parse report descriptors to decode input reports.

Table 4 – Consumer control device input report format

Number of bits	4	2	1	1	4	2	2
Field	Numeric key button	Channel	Volume Up	Volume Down	Other buttons	Selection buttons	reserved

Vendor specific reports are used to control the RemoTI HID dongle to perform RemoTI specific operations such as pairing. The report descriptor for the vendor specific reports is defined as follows:

```
Usage Pg (Vendor defined = 0xFF00)
Usage (RemoTI Target)
Collection (Application)
  Report ID (1) – 2 byte commands
  Logical Min (-128)
  Logical Max (127)
  Usage (Vendor Usage 1)
  Usage (Vendor Usage 2)
  Report Size (8)
  Report Count (2)
  Output (Data, Var, Abs)

  Usage (Vendor Usage 1)
  Usage (Vendor Usage 2)
  Input (Data, Var, Abs)

  Report ID (2) – paired entry
  Usage (Vendor Usage 2)
  Report Count (1)
  Input (Data, Var, Abs)
  Usage (Vendor Usage 3)
  Report Size (16)
```



```
        Input (Data, Var, Abs)
        Usage (Vendor Usage 4)
        Report Size (64)
        Input (Data, Var, Abs)

    End Collection
```

Table 5 is the usage table of the TI vendor defined page.

Table 5 – TI vendor defined page 0xFF00

Usage ID	Usage Name	Usage Type
00	Undefined	
01	Command Identifier	DV
02	Index or status	DV
03	Vendor identifier	DV
04	IEEE address	DV
05	RemoTI Target	CA
06 – FFFF	Reserved	

The valid values of a command identifier (Usage ID 2) are listed in Table 6.

Table 6 – Command identifier (vendor defined usage 2) semantics

Command Identifier Value	Description
-128 ~ 0	Reserved
1	Allow pair
2	Unpair
3	Clear entire pairing table
4	Get pairing table size
5	Get pairing entry
6	Get current MAC channel

7 ~ 127	Reserved

A vendor defined report with its report ID set to 1 is formatted as Table 7 either for an input report or an output report. Note that the USB host must not rely on Table 7 to decode an input report or to compose an output report but rather it has to parse report descriptors and decode or build reports accordingly.

The index or status field of the report is used to indicate a pairing entry index when the command identifier is either 2 (Unpair) or 5 (Get pairing entry). Otherwise, it is used to indicate a status for an input report and it is ignored for an output report. The output reports are used by the USB host to request a certain action or information and the USB device reports the status of the action or the information in response to such a request by an input report with its report ID set to either 1 or 2. An unpair command input report is also used to notify an unpair event triggered by a paired controller device, besides as a response to an unpair command output report.

Table 7 –Vendor defined report (ID=1) format

Number of Bits	8	8
Field	Command identifier	Index or status

A vendor defined input report with its report ID set to 2 is formatted as Table 8. Note that the USB host must not rely on Table 8 to decode an input report but rather it has to parse report descriptors to decode reports accordingly. The index field in this report indicates a pairing entry index.

Table 8 –Vendor defined input report (ID=2) format

Number of bits	8	16	64
Field	Index	Vendor identifier	IEEE address

Note that the vendor defined usage reports could have been embedded with consumer control device reports instead of using its own interface and USB end points. However, the latest Linux (Ubuntu 9.04) at the time of the release of the RemoTI HID dongle software does not work with a consumer control device interface mixed with vendor defined usage reports, and hence they are separated on purpose.

Figure 2 illustrates an example sequence of keyboard reporting. Note that the keyboard report is not repeated when CERC user control repeated commands are received by the dongle. It turned out that both Microsoft Windows™ and Linux work better if the keyboard reports are not repeated while a key is depressed. When the key is released, a report with null keys must be generated to notify the release of the key.

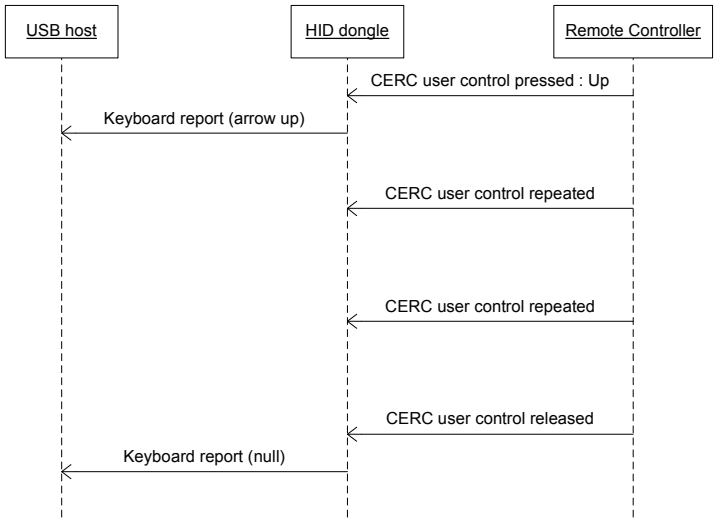

Figure 2 – Keyboard report sequence

Figure 3 illustrates an example sequence of consumer control device reports, triggered by remote controller key press. The reports are generated in the same sequence as keyboard reports.

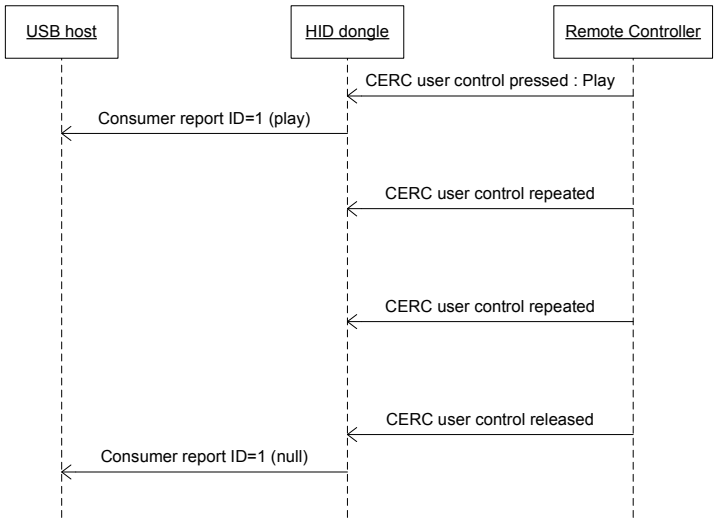

Figure 3 – Consumer control device report sequence

Figure 4 illustrates an example sequence of vendor defined reports. An output report is sent by the USB host and the HID dongle application sends back an input report upon completion of the action triggered by the output report.

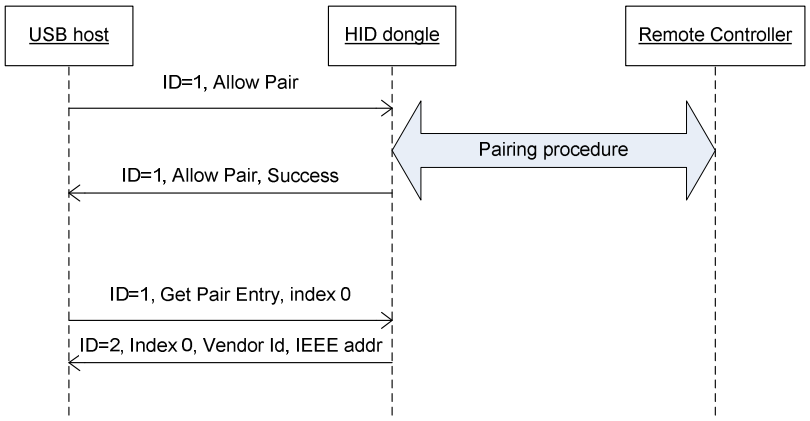

Figure 4 – Vendor defined report sequence

4 Customizing HID interfaces

For various reasons, different USB dongle products may have different HID interfaces. For instance, different products may use different button keys in their consumer control device reports. Certain products may want to have a consumer control device interface alone without a keyboard interface. Certain products may want to have a mouse interface. Certain products may want to have a game controller interface.

Modifying HID interfaces requires generic USB HID knowledge more so than the implementation specific knowledge. The descriptors are all defined in usb_hid_descriptor.s51 file. Interface change has to be reflected in the descriptors.

If the format of an input report changes, report generation functions in usb_hid_reports.c file requires modifications. If the format of an output report changes, hidappOutputReport() function in hidapp.c file requires modification.

If a new output endpoint is assigned for a new interface, hidappOutputReport() function should select the new endpoint and poll received data at the end of polling of endpoint 2, as follows:

```
void usbHidAppPoll(void)
{
  uint8 controlReg;
  uint16 bytesNow;
  uint8 oldEndpoint;
  uint8 *pBuf;

  // Save the old index setting, then select endpoint 0 and fetch the control register
  oldEndpoint = USBFW_GET_SELECTED_ENDPOINT();
  USBFW_SELECT_ENDPOINT(3);

  // Read registers for interrupt reason
  controlReg = USBCSOL;
```



```
  // Receive OUT packets
  if (controlReg & USBCSOL_OUTPKT_RDY)
  {
...
      // application specific handling
      hidappOutputReport();
    }
  }
```

```
  USBFW_SELECT_ENDPOINT(4);

  // Read registers for interrupt reason
  controlReg = USBCS0L;

  // Receive OUT packets
  if (controlReg & USBCS0L_OUTPKT_RDY)
  {
...
      // application specific handling
      hidappEP4OutputReport();
    }
  }
```

```
  // Restore the old index setting
  USBFW_SELECT_ENDPOINT(oldEndpoint);
}
```

Note that new endpoint output report handling function has to be duplicated from hidappOutputReport() function. In the above example, hidappEP4OutputReport() is such a function.

5 Key event handler

A key press event triggers a callback function call. The callback function is hidappKeyCback() in hidapp.c file. For now, the code simply exercises LED flashing. This function can be modified either to remove such LED exercise code or to add a new functionality upon a key event.

6 Flash page map and memory map

Figure 5 illustrates the flash page map of the HID dongle image. One flash page is 2048 bytes as specified in [5].

The code starts at the first page (lowest address page) up. OSAL non-volatile memory pages occupy configurable number of pages from the second last page down. The last flash page includes lock bits (last 16 bytes. See [5] for details) and commissioned IEEE address (8 bytes, prior to lock bits). IEEE address is explained more in chapter 8. The remainder of this last flash page can be used for additional code if the code fills up the rest of the space. Find more information about the last flash page and flash lock bits in [5].

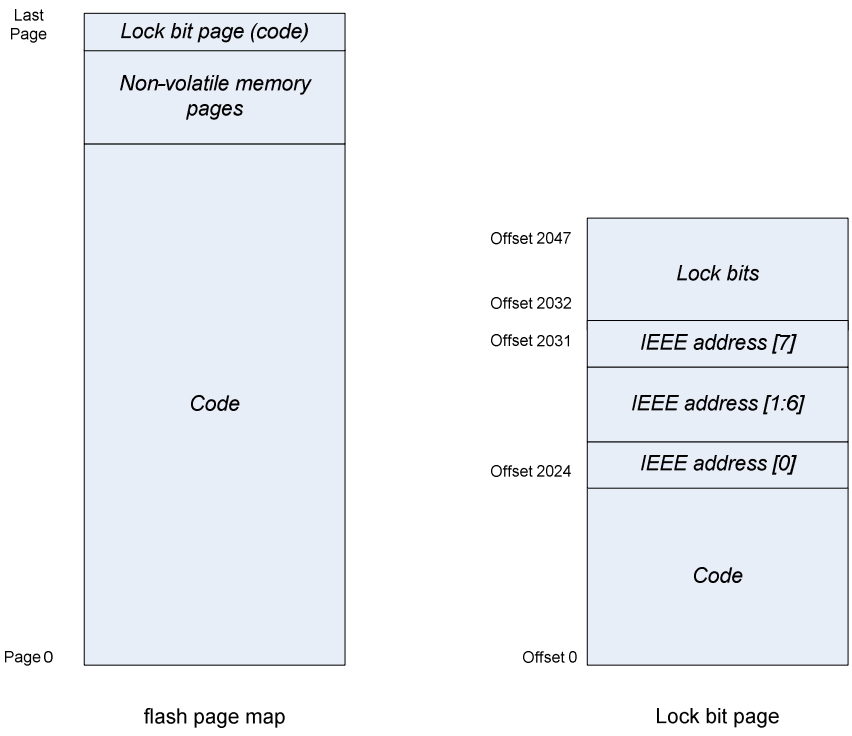

Figure 5 – Flash page map

The number of pages used for the OSAL non-volatile memory system is defined in hal_board_cfg.h file. The configurable constants and their values are listed in Table 9:

Table 9 – Non-volatile memory configuration in hal_board_cfg.h

Constant Name	Description	Values
HAL_NV_PAGE_END	Last OSAL NV page plus one	127
HAL_NV_PAGE_CNT	Number of OSAL NV pages	6

In order to change the number of pages used for the non-volatile memory system, both hal_board_cfg.h file and the linker command file have to be updated. In hal_board_cfg.h file, change the HAL_NV_PAGE_CNT definition. For instance, if you wish to use 4 flash pages and OSAL NV pages, change hal_board_cfg.h file as follows:

```
...
#define HAL_NV_PAGE_CNT          4
...
```


The linker command file used by HID dongle application is Projects\RemoTI\common\cc2530\ti_51ew_cc2530f256.xcl file, under your installation folder.

In the linker command file, find _ZIGNV_ADDRESS_SPACE_START definition and change the starting address to match the number of pages defined. The file by default has the following lines:

```
...
-D_ZIGNV_ADDRESS_SPACE_START=0x7C800
...
```

If you want four pages for non-volatile memory instead, the non-volatile memory page should be located at 12th page of the last bank (16 - 1 - 3), and the address should be 0x78000 + (0x800 * (12 - 1)) = 0x7D800. See further below in this section, for flash pages per bank and address ranges. The linker command file in this case has to be updated as follows:

```
...
-D_ZIGNV_ADDRESS_SPACE_START=0x7D800
...
```

The XDATA memory map and the CODE memory space are described in [5].

The HID dongle application uses a banked code model and the bank area is dynamically mapped to a flash bank (comprised of 16 pages) in use. The code address space is represented in virtual code address. The virtual addresses for code banks are listed in Table 10.

Table 10 – Virtual address of banked code

Code Bank	Bank 0	Bank 1	Bank 2	Bank 3	Bank 4	Bank 5	Bank 6	Bank 7
Address Range	0x00000 – 0x07FFF	0x18000 – 0x1FFFF	0x28000 – 0x2FFFF	0x38000 – 0x3FFFF	0x48000 – 0x4FFFF	0x58000 – 0x5FFFF	0x68000 – 0x6FFFF	0x78000 – 0x7FFFF

Bank 0 is constantly mapped to common area (0x0000 – 0x7FFF) and the other banks are mapped to the bank area (0x8000 – 0xFFFF) dynamically.

The bank set up is determined at the link time and it is configured through the linker configuration file.

7 Stack and Heap

The 8051 micro-controller uses a variety of data memory access methods. Generic data memory (i.e. not one specific for register access) is the internal data memory with 8 bit address space (IDATA) and the external data memory with 16 bit address space (XDATA). CC2531 maps both memory address spaces to the same internal SRAM. See [5] for details. The IAR compiler generates code to use a stack from both IDATA and XDATA. How a compiled code uses IDATA and XDATA for stack is highly dependent on the compiler itself.

With the IAR 8051 compiler version 7.51A, a RemoTI development kit 1.1 HID dongle uses about 239 bytes of the XDATA stack and 46 bytes of the IDATA stack. However, the depth of the used stacks could change with even a slight modification of the code as how compiler generates code to use stack is unpredictable.

Hence, 384 bytes of the XDATA stack and 192 bytes of the IDATA stack were reserved in project settings for a RemoTI development kit 1.1 HID dongle. Stack sizes can be adjusted after profiling the stack usage with the final application code, by browsing the stack memory space through a debugger.

For instance, the XDATA stack is located between addresses 0x100 and 0x27F and IDATA stack is located between addresses 0x40 and 0xFF, as could be found from a generated map file.

The IAR embedded workbench populates the value 0xCD to the entire XDATA stack and IDATA stack space when the debugger resets CC2530.

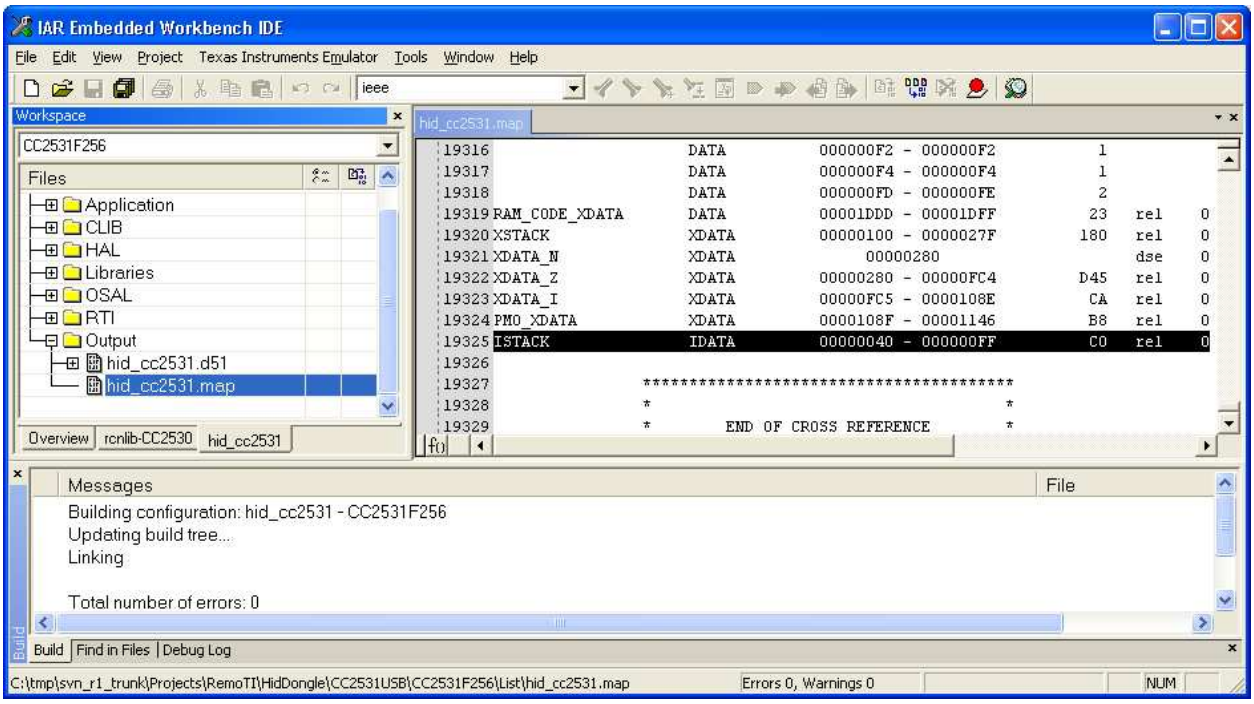

Figure 6 – Finding stack location

After running the application for the use cases picked for the deepest stack usage, the stack memory space can be browsed to determine how much stack was in use. In Figure 7, the XDATA stack was used down to 0x191, which makes the stack depth in this use case to be 0x27F – 0x191 + 1 = 239 bytes.

The IDATA stack usage can be profiled likewise. Just select IData to browse the IData memory.

Once stack usage is profiled, the stack size can be adjusted from project settings (General Options category, Stack/Heap tab).

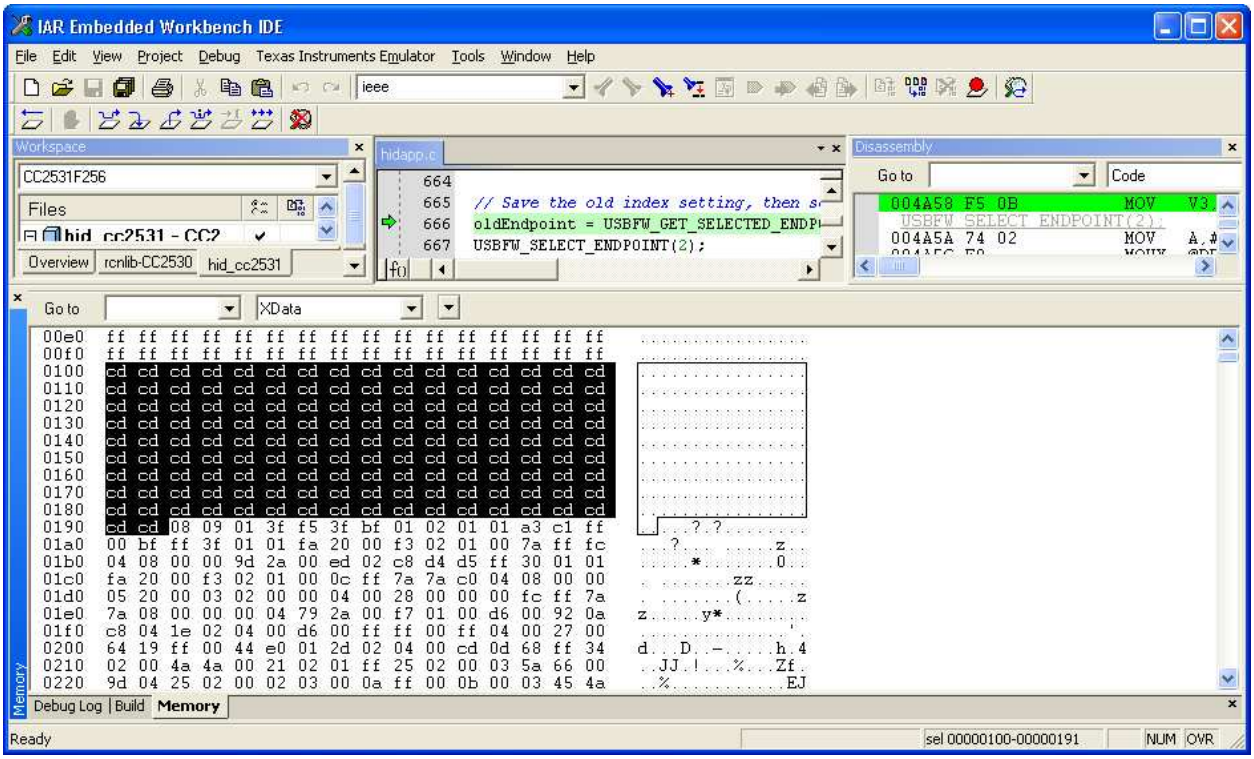

Figure 7 – XDATA Stack Profiling

RemoTI software uses heap through the OSAL memory management module. The default heap size is set to 2048 bytes. Heap usage varies drastically per use case even with the same software image. In other words, heap size has to be determined based on the supported use cases of the products.

In order to profile heap usage, some OSAL code has to be instrumented. Unlike stack memory space, heap memory space is not initialized with a certain pattern of data by a debugger.

In order to initialize heap memory space with a certain pattern of data, define OSALMEM_PROFILER as TRUE in preprocessor definition. Then, OSAL memory module initializes the heap with 0x58 (a code for character 'X').

With the new image, after running the use case with maximum heap usage, break the debugger and check the *_theHeap* memory space.

The address range of *_theHeap* variable can be found from the generated map file.

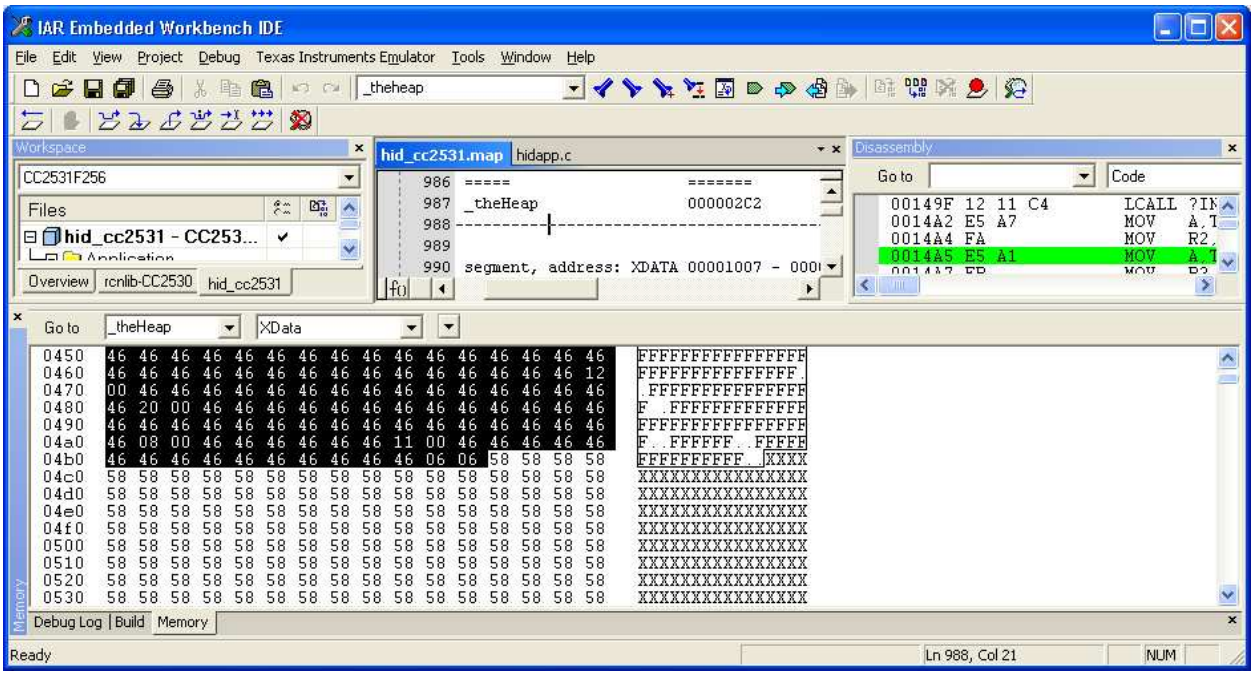

Figure 8 – Heap usage profiling

If the _theHeap variable occupies 0x2C2 to 0xAC1 address space for example, search from 0x2C2 up to 0xAC1 for any foot print of memory usage. In Figure 8, 0x4BB is the highest address of memory space that was used in the heap space. That amounts to 0x4BB - 0x2C2 + 1 = 506 bytes of heap usage.

Once heap size is profiled, the heap size can be adjusted by adding INT_HEAP_LEN definition as compile flag. For instance, adding INT_HEAP_LEN=1024 to preprocessor defined symbols window in IAR makes heap size to be set to 1,024 bytes.

8 IEEE address

CC2530 has its own IEEE address built into the chip (information page IEEE address). RemoTI network layer uses this IEEE address unless the IEEE address is overridden with a custom IEEE address by *RCN_NlmeSetReq*() call for *RCN_NIB_IEEE_ADDRESS* attribute. Once the IEEE address is overridden, network layer uses the custom IEEE address till this custom IEEE address is overwritten with another *RCN_NlmeSetReq*() call. If upper layer writes 0xFFFFFFFFFFFFFFFF as the custom IEEE address, network layer uses this null IEEE address till next power cycle. From next power cycle, network layer will start using the IEEE address built into the chip again.

RemoTI application framework, rti.c module, uses *RCN_NlmeSetReq()* to prioritize an IEEE address programmed to a specific last flash page location. See *rtiProgramIeeeAddr*() function for the source code. This function is called upon every system reset and the function reads the commissioned IEEE address in the special location and if it is valid (non-0xFFFFFFFFFFFFFFFF), this IEEE address is set to the network layer using *RCN_NlmeSetReq*() call. The special location is offset 0x7E8 of the last page stored in little-endian order, which neighbors lock bits which starts from offset 0x7F0. This is the location where SmartRF programmer will program the secondary IEEE address.

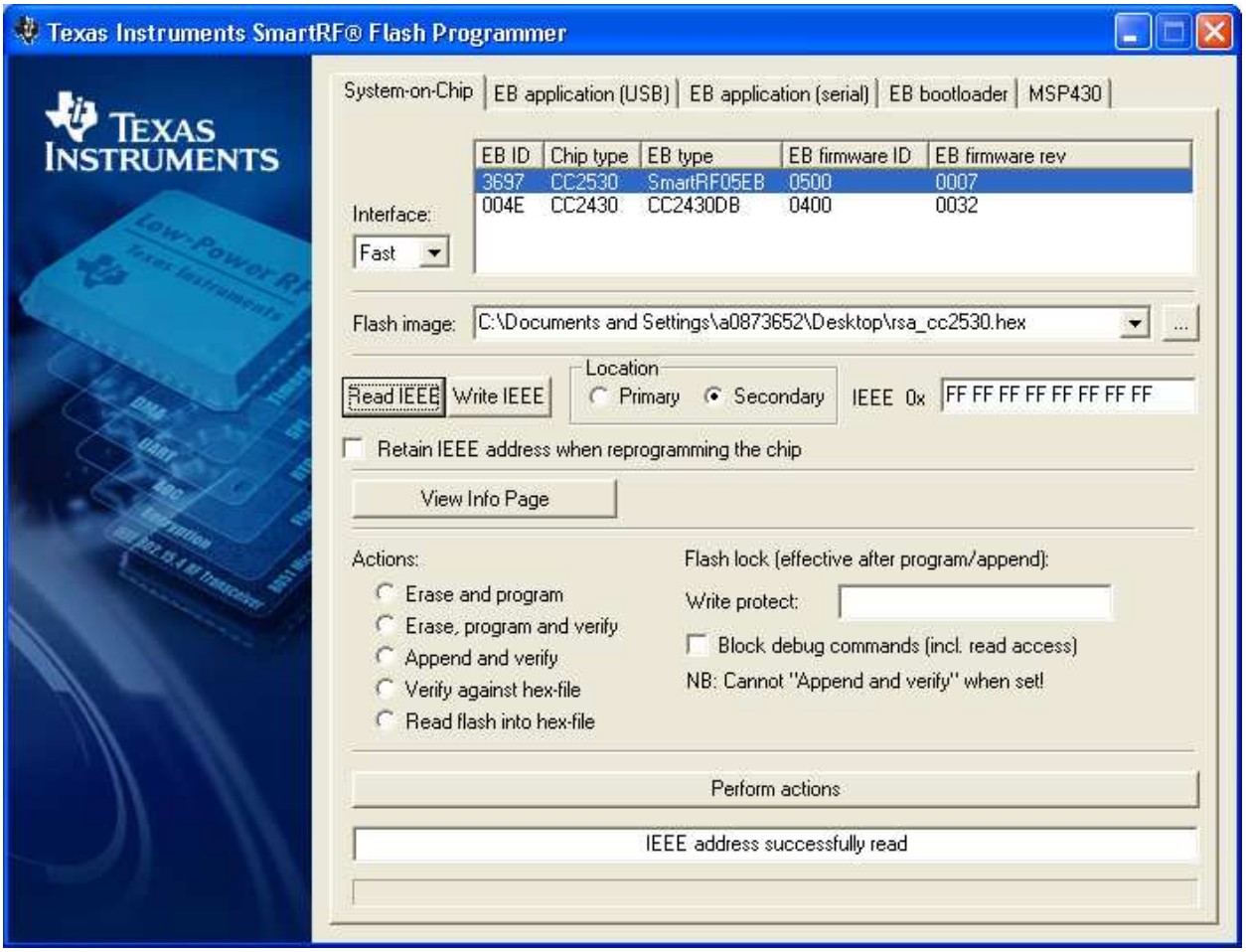

Figure 9 – SmartRF programmer

Hence, with RemoTI application framework, the hierarchy of IEEE address upon CC2530 reset is as follows:

- If the commissioned IEEE address is valid, use the commissioned IEEE address
- Otherwise, use the information page IEEE address

Figure 10 illustrates the flow chart of selecting the network layer IEEE address, during startup of a device.

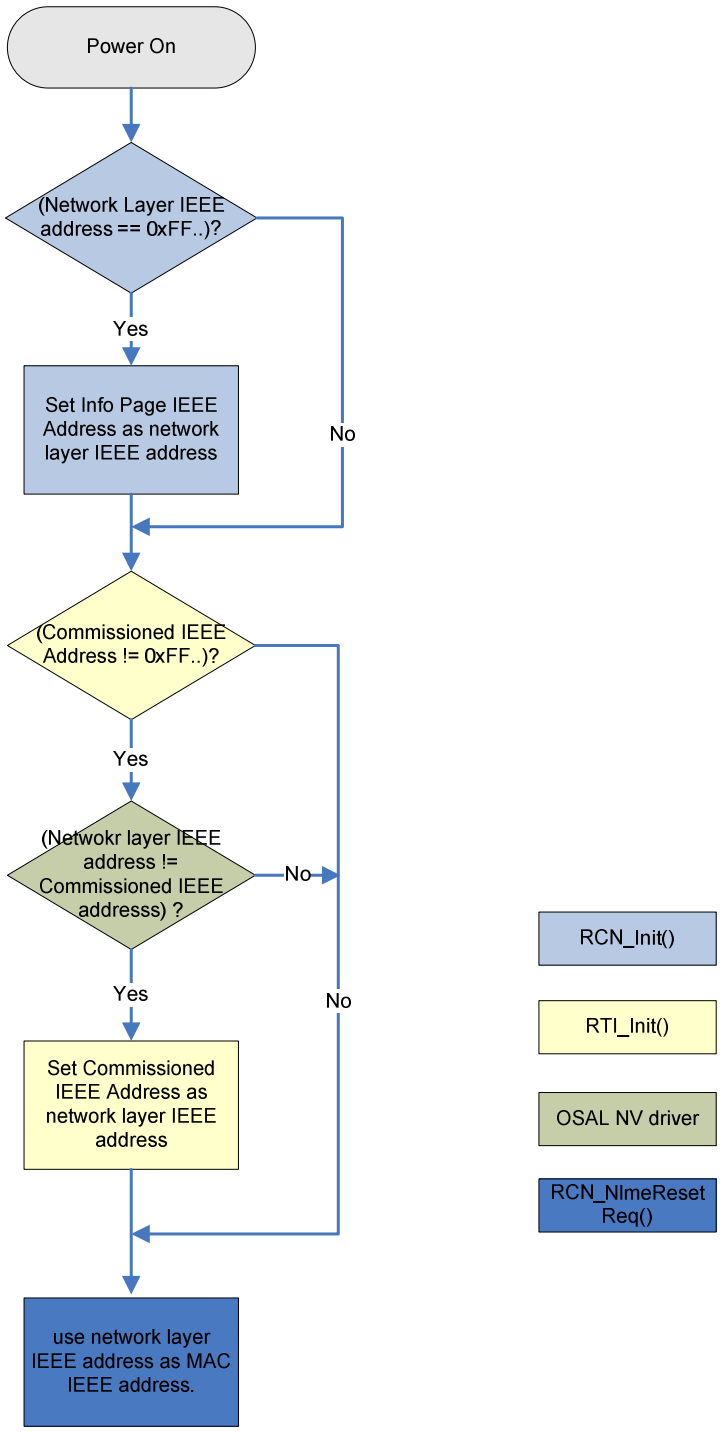

Figure 10 – IEEE address selection flow during startup

9 Network layer configuration

The standard NIB attributes can be configured and updated at run time through *RTI_WriteItem*() function or *RCN_NlmeSetReq*() function in case rti.c module is not used.

In rti.c module, *rtiResetSA*() function implementation shows example of *RCN_NlmeSetReq*() calls to set standard defined NIB attributes.

Network layer attributes that can be used with either *RTI_WriteItem* or *RCN_NlmeSetReq*() are enumerated in rcn_attribs.h file. Note that several non-standard attributes are also provided.

The following Table 11 explains non-standard attributes.

Table 11 – Network layer custom attributes

Attribute identifier	Description
RCN_NIB_NWK_NODE_CAPABILITIES	This attribute corresponds to standard constant nwkcNodeCapabilities. The value of this attribute should not change in product.
RCN_NIB_NWK_VENDOR_IDENTIFIER	This attribute corresponds to standard constant nwkcVendorIdentifier. The value of this attribute should not change in product.
RCN_NIB_NWK_VENDOR_STRING	This attribute corresponds to standard constant nwkcVendorString. The value of this attribute should not change in product.
RCN_NIB_STARTED	It is an attribute to indicate whether network layer has started (‘1’) or not (‘0’). This attribute is useful for application to determine whether it has to perform cold boot procedure or warm boot procedure. RTI module (rti.c) uses this attribute to determine cold boot or warm boot procedure.
RCN_NIB_IEEE_ADDRESS	IEEE address attribute. By default, network layer will program IEEE address using chip IEEE addresss. Application can override chip IEEE address with this attribute. Note that RTI module (rti.c) writes into this attribute upon system reset. Application should consider conflict with RTI module when writing this attribute. See chapter 8.

Attribute identifier	Description
RCN_NIB_AGILITY_ENABLE	Enable/disable frequency agility
RCN_NIB_TRANSMIT_POWER	Set transmission power level as this attribute value in dBm.

Note that other non-standard attributes such as RCN_NIB_PAN_ID and RCN_NIB_SHORT_ADDRESS are not configurable items. Those attribute values can be read in order for a debug purpose.

A certain set of network layer implementation parameters can also be modified at build time by changing rcn_config.c file. The file is configured with default recommended values.

10 DMA, peripheral IO and timers

A RemoTI HID dongle uses the following resources:

- USB controller
- Peripheral IO pins P1_0 and P1_1 for LED control
- Peripheral IO pins P1_2 and P1_3 for key switch input
- DMA channel 0 for non-volatile memory access
- DMA channel 1 and 2 for security encryption and decryption engine
- Timer2 (MAC timer) and sleep timer.

11 General Information

11.1 Document History

Table 12 – Document History

Revision	Date	Description/Changes
swra298	2009-09-18	Initial release

12 Address Information

Texas Instruments Norway AS
Gaustadalléen 21
N-0349 Oslo
NORWAY
Tel: +47 22 95 85 44
Fax: +47 22 95 85 46
Web site: http://www.ti.com/lpw

13 TI Worldwide Technical Support

Internet

TI Semiconductor Product Information Center Home Page: support.ti.com
TI Semiconductor KnowledgeBase Home Page: support.ti.com/sc/knowledgebase
TI LPRF forum E2E community http://www.ti.com/lprf-forum

Product Information Centers

Americas

Phone: +1(972) 644-5580
Fax: +1(972) 927-6377
Internet/Email: support.ti.com/sc/pic/americas.htm

Europe, Middle East and Africa

Phone:

Belgium (English)	+32 (0) 27 45 54 32
Finland (English)	+358 (0) 9 25173948
France	+33 (0) 1 30 70 11 64
Germany	+49 (0) 8161 80 33 11
Israel (English)	180 949 0107
Italy	800 79 11 37
Netherlands (English)	+31 (0) 546 87 95 45
Russia	+7 (0) 95 363 4824
Spain	+34 902 35 40 28
Sweden (English)	+46 (0) 8587 555 22
United Kingdom	+44 (0) 1604 66 33 99

Fax: +49 (0) 8161 80 2045

Internet:	support.ti.com/sc/pic/euro.htm	
Japan		
Fax	International	+81-3-3344-5317
	Domestic	0120-81-0036
Internet/Email	International	support.ti.com/sc/pic/japan.htm
	Domestic	www.tij.co.jp/pic
Asia		
Phone	International	+886-2-23786800
	Domestic	Toll-Free Number
	Australia	1-800-999-084
	China	800-820-8682
	Hong Kon	800-96-5941
	India	+91-80-51381665 (Toll)
	Indonesia	001-803-8861-1006
	Korea	080-551-2804
	Malaysia	1-800-80-3973
	New Zealand	0800-446-934
	Philippines	1-800-765-7404
	Singapore	800-886-1028
	Taiwan	0800-006800
	Thailand	001-800-886-0010
Fax		+886-2-2378-6808
Email		tiasia@ti.com or ti-china@ti.com
Internet		support.ti.com/sc/pic/asia.htm

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		Applications	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

