User manual

DA14580 Software Porting Guide

UM-B-016

Abstract

This document contains information about porting DA14580 software from the version 2.0.4.to version 3.0.2.

Contents

Tables

1 Terms and definitions

2 References

1. DA14580 Datasheet, Dialog Semiconductor
2. DA14580 Development Kit, UM-B-014, Dialog Semiconductor
3. DA14580 Sleep Mode Specifications, UM-B-006, Dialog Semiconductor
4. DA14580 Software Patching over the Air, SPotA, UM-B-007, Dialog Semiconductor
5. DA14580 Software Development Guide, UM-B-003, Dialog Semiconductor

3 Introduction

This document provides information for porting DA14580 software from version 2.0.4 to release 3.0.2. The status of each API or software module, that possible modifications of it could affect the application running on the system, is presented in detail in the sections below.

4 Application structure

4.1 Directory tree

Application task software in dk_apps\src\module\app\ is re-organized in new subdirectories. The new directory tree is outlined in Table 1.

Table 1: Directory Tree

Directories/files				Description
Api				common code header files
	app.h, app_task.h, app_sec.h app_sec_task.h			Common code header files.
	app_api.h			Common application hooks to project code.
	app_task_handlers.h			Application's message handlers
Src				Application code
	app.c, app_task.c, app_sec.c, app_sec_task.c			Application's common code
	app_profiles			Profile specific application code
		accel		Accelerometer profile application code
		diss		Device Information Service Server application code
		…		Other supported profiles application code.
	app_project			Application (project) specific code
		dice		Dice sample application
			system	app_sleep, periph_setup. Project specific.
		prox_reporter_fh		Proximity reporter sample application
			system	app_sleep, periph_setup. Project specific.
		…		Other sample applications
	app_utils			Special functionality shared by different applications.
		app_alt_pair		Pairing/Bonding to alternative devices
		app_console		Debugging info over serial iface

4.2 Application hooks

New common code hook functions defined.

Table 2: Application Tree

Initialization	
app_db_init_complete_func	Handles completion of databases creation. i.e.
Device configuration	
app_set_dev_config_complete_func	Called upon device's configuration completion
app_update_params_complete_func	Called upon connection parameters update completion
app_update_params_rejected_func	Called upon connection parameters update rejection
Advertise	
app_adv_direct_complete	Handles direct advertising completion
app_adv_undirect_complete	Handles undirect advertising completion
Security	
app_sec_encrypt_ind_func	Handles encryption indication
app_paired_func	Project's action when device is paired
app_send_pairing_rsp_func	Sends pairing response message Called upon pairing request message reception.
app_sec_encrypt_complete_func	Project's actions when encryption request is completed successfully
app_sec_encrypt_ind_func	Handles encryption indication
app_validate_encrypt_req_func	Validates encryption request message
app_send_tk_exch_func	Send GAPC_TK_EXCH. Called in gapc_bond_req_ind_handler(tk_type == GAP_TK_KEY_DISPLAY)
app_send_irk_exch_func	Send GAPC_IRK_EXCH. Called in gapc_bond_req_ind_handler/GAPC_IRK_EXCH
app_send_csrk_exch_func	Send GAPC_CSRK_EXCH. Called in gapc_bond_req_ind_handler/GAPC_CSRK_EXCH
app_send_ltk_exch_func	Send GAPC_LTK_EXCH. Called in gapc_bond_req_ind_handler/GAPC_LTK_EXCH

5 Peripheral Drivers

DA14580 SDK comes with a core driver provided for each interface together with several peripheral driver examples.

Core drivers:

- *GPIO*
- *SPI*
- *UART*
- *ADC*
- *Wakeup / Quadrature*
- *Timers / PWM*
- *RF*
- Peripheral examples:
- *Accelerometer*
- *SPI Flash*

- *EEPROM I2C*
- *Battery*

6 Project Configuration

Configuration Header files da14580_config.h and da14580_stack_config.h added in all projects directories. Both Header files are pre-included in projects. Directives defined in these files replaced defined pre-processor symbols in project configuration. Header file da14580_stack_config.h defines directives related to stack configuration and should not be altered by developer. Directives defined in da14580_config.h determines project configuration.

Table 3: Configuration settings

Directive	Defined	Undefined
CFG_APP	Integrated host application	External processor host application
CFG_PRF_<profile>	Profile included	Profile not include
CFG_APP_<application>	Application Identifier. Must be defined in integrated host applications	
CFG_NVDS	NVDS structure used	NVDS structure not used
CFG_APP_SEC	BLE security enabled	BLE security disabled
CFG_LUT_PATCH	Coarse calibration enabled	Coarse calibration disabled
CFG_WDOG	Watchdog timer enabled	Watchdog timer disabled
CFG_EXT_SLEEP CFG_DEEP_SLEEP	Default sleep mode. Only one must be defined	
BLE_CONNECTION_MAX_USER	Max connections number	
DEVELOPMENT__NO_OTP	Development mode, OTP memory not used	Production.
CFG_LP_CLK	Low power clock selection (XTAL32 or RCX20)	
REINIT_DESCRIPT_BUF	Scatterfile configuration	
USE_MEMORY_MAP	Scatterfile configuration	

7 BLE Stack API

GAPM_SET_DEV_CONFIG_CMD

Field uint16_t max_mtu added in gapm_set_dev_config_cmd structure.

GAPC_GET_CON_CHANNEL_MAP

GAPC_GET_CON_CHANNEL_MAP

operation added in GAPC_GET_INFO_CMD for retrieving the connection channel map.

The response will be a GAPC_CON_CHANNEL_MAP_IND event followed by a GAPC_CMP_EVT when the operation is completed.

GATTC_WRITE_CMD_CFM

GATTC_WRITE_CMD_CFM command added to support sending write confirmations from upper layers.

GATTC_EVENT_IND

Field's length type is changed from uint8_t to uint16_t igattc_event_ind structure.

8 KERNEL API

- **app_timer_set() MUST be used instead of ke_timer_set(). The prototype of this function is in app.h.**

9 Sleep API

- Sleep API functions added.
- void **app_force_active_mode()**: ***Disables sleep. Stores the sleep mode used by the application.***
- void **app_restore_sleep_mode()**: ***Restores the selected mode of operation, if active mode is not requested.***
- ***void app_ble_ext_wakeup_on(): Puts the BLE core into permanent sleep waiting for a wakeup from the ARM (external wakeup mode). If the system decides to wake-up the BLE then it must call app_ble_ext_wakeup_off() or the BLE won't be able to wake up in order to serve Bluetooth events.***
- ***void app_ble_ext_wakeup_off(): Restore the operation of the BLE core to the default mode. The BLE core will wake up every 10sec if no BLE events are scheduled. If a BLE event has been scheduled then the BLE core will wake up in time to serve it.***
- ***bool app_ble_ext_wakeup_get(): Returns the current mode of operation of the BLE core (external wakeup or default).***
- app_sleep.h file including sleep architecture hook functions definitions is project specific and moved to dk_apps\src\module\app\app_project\<project>\system. This file or function definitions cannot be deleted.

10 Jump Table

Jump table structure modified to comply with new ROM code's jump table structure.

11 OTP Header

No changes in OTP Header.

12 NVDS

Additional fields in NVDS structure:

- NVDS_TAG_BLE_CA_TIMER_DUR = 2000. Default Channel Assessment Timer duration (20s - Multiple of 10ms)
- NVDS_TAG_BLE_CRA_TIMER_DUR = 6. Default Channel Reassessment Timer duration (Multiple of Channel Assessment Timer duration)
- NVDS_TAG_BLE_CA_MIN_RSSI = 0x90. Default Minimal RSSI Threshold - -48dBm
- NVDS_TAG_BLE_CA_NB_PKT = 100. Default number of packets to receive for statistics
- NVDS_TAG_BLE_CA_NB_BAD_PKT = 10. Default number of bad packets needed to remove a channel

13 Patch area

SDK 2.0.4 Patched function s have been removed. More information about new patched functions is included in the Release Notes.

14 Other

Other modifications that are included in the 3.0.2 versus 2.0.4 release are:

- Attribute specification `__attribute__((section("retention_mem_area0"), zero_init))` must be used for global variables placed in retention memory instead of `__attribute__((section("exchange_mem_case1")))`.

15 Keyboard application modifications

15.1 Structure

The code structure of the Keyboard application is depicted below.

Directories/files		Description
dk_apps/keil_projects/hid/keyboard/		Project files
	da14580_config.h	Project configuration file.
	da14580_stack_config.h	Stack configuration for this project (do not alter!).
	da14580_scatter_config.h	Scatter file configuration (do not alter!).
	fh_keyboard_sdk.uvproj	Keil project file for this application.
	unused.txt	File generated automatically by the linker to allow removal of any unused code.
dk_apps/src/modules/app/src/app_project/keyboard/		Application code
	app_kbd.c	Functions for key scanning, debouncing, deghosting etc are included in this file. The interrupt handlers for the Keyboard Controller and the Systick Timer are also included.
	app_kbd.h	Header file for app_kbd.c
	app_kbd_config.h	This is the configuration file of this application. It may be used together with da14580_config.h to configure the Keyboard app. The user can set various options of the application in here, like LEDs, EEPROM, MITM etc. The definition of the specific MATRIX_SETUP that will be used is done here together with the parameters for debouncing and scanning times. Finally, the various timeouts used by the high-level FSM and the preferred connection parameters (which may be overridden by the host, of course) are set here.
	app_kbd_debug.h	Header file with definitions used when CFG_PRINTF is on and debugging is done via UART.
	app_kbd_fsm.c	This is the high-level FSM of the application.
	app_kbd_fsm.h	Header file for app_kbd_fsm.c
	app_kbd_key_matrix.h	In this file, the pins used for the LEDs and the I2C are defined. Also, the GPIO declarations for the

		GPIO driver are done in here.
	app_kbd_leds.c	The LEDs' functionality is implemented in this file.
	app_kbd_leds.h	Header file for app_kbd_leds.c
	app_kbd_matrix.h	In this file, the key matrix (input array x output array) is defined. Also, the "keymap" which includes the scan code that corresponds to each "intersection" is defined for each available matrix setup.
	app_kbd_proj.c	The implementation of the various hooks of the common code is included here.
	app_kbd_proj.h	Header file for app_kbd_proj.c
	app_kbd_proj_task.c	The HID Report Map is declared in this file.
	app_kbd_proj_task.h	Header file for app_kbd_proj_task.c
	app_kbd_scan_fsm.c	Low level FSM of the application. This FSM handles the key scanning.
	app_kbd_scan_fsm.h	Header file for app_kbd_scan_fsm.c

15.2 Updates since 2.0.4

The most significant updates of the Keyboard application compared to 2.0.4 are:

1. A high level FSM has been implemented. This FSM satisfies all requirements stated in the HID Over GATT Specification. When the application starts, it will stay in IDLE mode waiting for a key press. The BLE will be deactivated during this period. When a key is pressed, the system will wake up and will either start directed advertising to a previously bonded host or, if no bonded host exists, advertising will be started as normally.
2. LED functionality was moved to a separate FSM.
3. HID reporting depends on the current application state. It may or may not be active.
4. "White list" may be used (configurable) when directed advertising fails. If this option is chosen then the system will be able to bond only to 1 host. Other options (like NormallyConnectable) have also been implemented.
5. Modified the code so that if encryption is not activated by the host within a specified time interval (this can happen with iOS devices when MITM is not being used) and the connection is still active then the key reporting is activated.
6. Modified the code to support entry to Deep Sleep mode when in IDLE_ST (at the beginning or after a user-specified inactivity timeout).
7. A low level FSM for key scanning has been implemented. The processing of the scan results, the debouncing and the deghosting are done right after the completion of the scanning of the last active row. Then "idle" time follows until the completion of the scan cycle time. In the next state, update of the full_scan flag is being made and a decision is taken whether to scan again or exit (and enter into sleep) since there's no key activity.
8. Optimized debouncing and deghosting.
9. Keyboard scanning is always active. This means that we will always get an interrupt for a new key press. Whether it will be reported or not (HID reporting) depends on the high level FSM state.

10. Patched the code so that the calculation of the "idle" time (when the Keyboard Controller is active) takes into account the fact than after wake up we are running based on the RC16 clock which is smaller than 16MHz (approx. 14.5MHz).
11. The configuration of the application is done in the file app_kbd_config.h. The configuration of the debouncing periods and of the full and partial scan periods has been simplified.
12. Roll-over has been implemented (when more than 6 keys are being pressed at the same time).
13. 'Fn + 'c' works regardless of whether we are connected or not. Clearing the EEPROM was decided to be possible at all times.
14. Optimized app_multi_bond.c. The EEPROM is accessed only when it is absolutely necessary in order to save power. BLE_ALT_PAIR was changed to CFG_MULTI_BOND. The availability of an EEPROM is no longer depending on the CFG_MULTI_BOND. There can be an EEPROM without Multiple Bonding support. The opposite is not possible.
15. The application uses the common I2C and WKUP Controller Handlers.
16. Fixed a bug that resulted in not sending HID reports even though key events were left pending.
17. Got rid of most #ifdefs for the keyboard application. The code uses normal 'if (HAS_flag)' statements based on '#ifdef flag_ON' definitions. This way the application is checked under all possible configurations when it is compiled.

16 SPOTA Modifications

This chapter outlines the code and structural changes of the SPOTA Receiver software, project fh_spotar.uvproj.

16.1 Application Modifications

A new define directive has been added to the project's configuration header file da14580_config.h. The following define determines if the RAM patch buffers should be in retention RAM or non-retainable RAM:

```
#ifdef CFG_PRF_SPOTAR
// Placed in the RetRAM when SPOTAR_PATCH_AREA is 0 and in SYSRAM when 1
#define SPOTAR_PATCH_AREA   0
#endif
```

The header file is included in the *scatterfile_common.sct* where according to the identifier value

the `spotar_patch_area` is reserved in the relevant RAM section

More information is given in [4].

16.2 Profile Modifications

SIG has assigned a 16bit UUID for the SPOTAR service (0xFEF5). Also, using an UUID generator, six propriotry 128bit UUIDs were allocated for the service characteristics.

The profile attribute database has been modified to reflect the above. Note that the characteristic descriptors that describe a characteristic value have been removed from the database to reduce the amount of memory used for the database. The files modified are located in the following directory:

```
C:\Users\edamaska\Documents\580\src_es5\release\ble_sw\ble_sw\dk_apps\src\
ip\ble\hl\src\profiles\spota\spotar
```

17 RCX Low power clock

17.1 Configuration

A definition directive has been added in projects' da14580_config.h for low power clock source selection:

```
#define CFG_LP_CLK   0x00
```

Possible values:

0x00: XTAL32

0xAA: RCX

0xFF: Select LP clock from OTP Header field.

17.2 Clock calibration / frequency calculation

A calibration mechanism is developed to measure RCX clock's frequency changes over temperature. This mechanism consists of calibrate_rcx20() and read_rcx_freq() functions. If RCX is selected as low power clock calibrate_rcx20() initiates the h/w process to measure the number of XTAL16 (16MHz) ticks elapsed during the countdown of provided number of RCX ticks. RCX evaluation under temperature cycling proved that a calibration process of 20 RCX ticks provides adequate precision in current frequency calculation. calibrate_rcx20() is called in SLP interrupt handler to start the process in h/w while processor services the BLE event. read_rcx_freq() checks that calibration process is completed in h/w, reads the number of XTAL16 clocks ticks and calculates current RCX frequency. read_rcx_freq() is called at the end of BLE connection event, before entering in sleep mode.

17.3 Time units conversions

Three time unit convertion functions are re-implemented for RCX20 clock:

rwip_slot_2_lpcycles_rcx() -> BLE slots (0.625 ms) to low power cycles.

lld_sleep_us_2_lpcycles_rcx_func() -> ms to low power cycles

lld_sleep_lpcycles_2_us_rcx_func() -> low power cycles to ms

The latest 2 are members of jump table structure and used in ROM code.

18 Radio Preferred settings

Radio preferred settings are stored in the file \ble_sw\dk_apps\src\plf\refip\src\arch\system_settings.h

19 Revision history

Revision	Date	Description
1.0	28 Mar 2014	Initial version.

Status definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

Disclaimer

Information in this document is believed to be accurate and reliable. However, Dialog Semiconductor does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information. Dialog Semiconductor furthermore takes no responsibility whatsoever for the content in this document if provided by any information source outside of Dialog Semiconductor.

Dialog Semiconductor reserves the right to change without notice the information published in this document, including without limitation the specification and the design of the related semiconductor products, software and applications.

Applications, software, and semiconductor products described in this document are for illustrative purposes only. Dialog Semiconductor makes no representation or warranty that such applications, software and semiconductor products will be suitable for the specified use without further testing or modification. Unless otherwise agreed in writing, such testing or modification is the sole responsibility of the customer and Dialog Semiconductor excludes all liability in this respect.

Customer notes that nothing in this document may be construed as a license for customer to use the Dialog Semiconductor products, software and applications referred to in this document. Such license must be separately sought by customer with Dialog Semiconductor.

All use of Dialog Semiconductor products, software and applications referred to in this document are subject to Dialog Semiconductor's Standard Terms and Conditions of Sale, unless otherwise stated.

RoHS Compliance

Dialog Semiconductor complies to European Directive 2001/95/EC and from 2 January 2013 onwards to European Directive 2011/65/EU concerning Restriction of Hazardous Substances (RoHS/RoHS2).
Dialog Semiconductor's statement on RoHS can be found on the customer portal https://support.diasemi.com/. RoHS certificates from our suppliers are available on request.

Contacting Dialog Semiconductor

Germany Headquarters
Dialog Semiconductor GmbH
Phone: +49 7021 805-0

United Kingdom
Dialog Semiconductor (UK) Ltd
Phone: +44 1793 757700

The Netherlands
Dialog Semiconductor B.V.
Phone: +31 73 640 8822

Email:
enquiry@diasemi.com

North America
Dialog Semiconductor Inc.
Phone: +1 408 845 8500

Japan
Dialog Semiconductor K. K.
Phone: +81 3 5425 4567

Taiwan
Dialog Semiconductor Taiwan
Phone: +886 281 786 222

Web site:
www.dialog-semiconductor.com

Singapore
Dialog Semiconductor Singapore
Phone: +65 64 849929

China
Dialog Semiconductor China
Phone: +86 21 5178 2561

Korea
Dialog Semiconductor Korea
Phone: +82 2 3469 8291