WICED Studio

WICED™ DCT (Device Configuration Tables)

Associated Part Family: BT CYW2070x

Doc. No.: 002-19004 Rev. *A

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
www.cypress.com

Contents

About This Document

Purpose and Scope

This document provides instructions for how to use the WICED DCT data area to provide persistent storage for system information and application data. Using the sample Applications, and API's, you will be able to save data between power cycling your device.

Note: This document applies to **WICED SDK 3.7.0** or higher.

Audience

This document is for software developers who are using the WICED Development System to create applications for secure embedded wireless networked devices.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Cypress documents, go to www.cypress.com/glossary.

IoT Resources and Technical Support

Cypress provides a wealth of data at www.cypress.com/internet-things-iot to help you to select the right IoT device for your design, and quickly and effectively integrate the device into your design. Cypress provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates. Customers can acquire technical documentation and software from the Cypress Support Community website (community.cypress.com/).

Terminology

Bootloader : The initial prog ram run when power is applied. Initializes hardware and decides which Application to run.

DCT : Device Configuration Tables – Data stored to FLASH, both System DCT information and Application information.

LUT : Look Up Table – in the Wiced Multi-Application Framework, this is a simple directory where the system (App, DCT, Resources) is located in FLASH.

1 Introduction

1.1 DCT Description & Overview

The DCT stores System and Application data persistently so that the device can use the information between power cycles. The layout of the data is extremely important, and may change between SDKs.

Using the OTA facility (described in another Application Note), an Application can be updated, and it is important that the DCT is still usable. Upgrading an Application using the same SDK as the original Application (with no changes to the System DCT area) ensures a smooth update.

Starting with Wiced-SDK-3.7.0, it is possible to upgrade to a new SDK and upgrade the DCT layout so that the part of the DCT layout that matches what is used by the Bootloader of the original SDK is intact, and there will be no problems.

1.2 Bootloader Application

The Bootloader is a small application that loads any needed system resources and launches the Current Application. It will check for button presses for factory reset operation.

The Bootloader is finalized at shipment and is not updated through the product cycle.

There is one portion of the DCT that the Bootloader uses: the sub-structure `platform_dct_header_t`, which is always at the start of the DCT area of the FLASH.

1.3 Two DCT Areas

There are two DCT areas defined in the FLASH, designated as DCT1 and DCT2. When there are changes to the DCT, we use these in a flip-flop arrangement, copying the “current” DCT to the opposite area with the changes requested, then indicating that the “new” area is the “current” area. Then we mark the “old” area as not in use. This ensures that if power is lost in any part of the update procedure, there is a viable DCT area upon power up.

2 DCT Layout Importance

It is extremely important that the DCT Layout change as little as possible. Consider this structure layout and change:

```
typedef struct my_struct_1_s
{
        Uint16   data1;
        Uint16   signature;
} my_struct_1_t;

typedef struct my_struct_2_s
{
        Uint16   data1;
        Uint16   data2;
        Uint16   signature;
} my_struct_2_t;
```

In structure 1, the offset for data1 from the beginning of the structure is 0x00. That is, if you start at the memory address of the start of the structure, the first 2 bytes (Uint16) will be data1. And the offset of the signature is 0x02.

In structure 2, the offset of the signature has moved, and is now 0x04.

If you write data to the FLASH DCT area using structure 1, and then change the code to use structure 2, when you go to verify the data integrity by looking at the signature, the code will fail. The code is now looking 0x04 bytes from the start of the structure for the signature. But the old code wrote the signature with an offset of 0x02 bytes. The new code will find this an invalid data structure. Therefore it is important to consider the structures when updating the SDK and developers should always maintain a backward compatible version of the common platform DCT structures.

2.1 Defining the SDK used by the Bootloader

When building an Update Application, knowledge of the SDK that the Bootloader was built with needs to be passed to the make system so that the System DCT can be written in such a way as to be compatible with the Bootloader. To do this, designate the SDK of the Bootloader on the make command line:

```
<application_name>-<platform_name> UPDATE_FROM_SDK=<bootloader_sdk>
```

Where `<bootloader_sdk>` is one of:

```
3_1_2
3_3_0
3_3_1
3_4_0
3_5_1
3_5_2
3_6_0
3_6_1
3_6_2
```

2.2 Defining the use of optional structures

There were optional structures in the System DCT that are now always included. They are the Bluetooth (BT), Peer to Peer (P2P) and Over The Air 2 (OTA2) sub-structures. This information must also be specified so that the code knows which (if any) of the optional structures were used in the original application build. This is supported with the following additions to the make command line:

```
<application_name>-<platform_name> <optional_struct> <optional_struct> <optional_struct>
```

Where `<optional_struct>` is one of:

```
APP_USED_BT=1
APP_USED_P2P=1
APP_USED_OTA2=1
```

3 Application DCT data

The DCT system allows for Applications to add Data to the end of the DCT area which will be maintained when the DCT is updated (copied to new DCT area when data is changed). Please see the application dct_read_write for an example of usage.

NOTE: If your application uses Homekit, the Homekit support uses part of the Application DCT for its own use. Please see the section on Homekit below.

`/apps/snip/dct_read_write`

3.1 Defining Application DCT Data

- Add 2 files to your application directory
  - `/<application>/< application >_dct.h`
  - `/<application>/<application>_dct.c`
- Add a line to your application makefile

```
APPLICATION_DCT     := <application>_dct.c
```

3.2 Application DCT Structure Definition

Using `/apps/snip/dct_read_write/dct_read_write.h` as an example, add a structure definition to describe the data you wish to add to the DCT:

```
typedef struct
{
            uint32_t number_var;
    char     string_var[ MY_DCT_STRING_SIZE ];
} my_app_dct_t;
```

3.3 Default Application DCT Data

Using `/apps/snip/dct_read_write/dct_read_write.c` as an example, add a structure definition to describe the data you wish to add to the DCT:

```
DEFINE_APP_DCT(my_app_dct_t)
{
    .number_var         = 99999999,
    .string_var         = "The DCT says hi!"
};
```

3.4 Using Application DCT Data

To read Application DCT data, use `wiced_dct_read_lock()` which will allocate the destination buffer. You must release the memory by calling `wiced_dct_read_unlock()`.

3.4.1 wiced_dct_read_lock()

To read Application DCTdata, use `wiced_dct_read_lock()` which *may* allocate destination RAM. When using Internal FLASH for DCT storage, the second argument determines if RAM is allocated for the read (if not, the returned pointer points directly to FLASH). When using external FLASH for storing DCT, the second argument is ignored – RAM is always allocated.

```
{
    uint32_t        *number;
    wiced_dct_read_lock((void**)&number, WICED_TRUE, DCT_APP_SECTION,
                        OFFSETOF(my_app_dct_t, number_var), sizeof( uint32_t ) );
    ...
```

3.4.1.1 DCT stored in Internal FLASH

Setting the second argument to *WICED_TRUE* allocates RAM.

Setting the second argument to *WICED_FALSE* does not allocate RAM, and will point directly to the data in FLASH.

3.4.1.2 DCT stored in External SFLASH

With the DCT in external FLASH, the second argument is ignored. For clarity, always use *WICED_TRUE*.

3.4.2 wiced_dct_read_unlock()

Regardless of the value you choose to pass for the second argument `wiced_dct_read_lock()`, always use the same argument when calling `wiced_dct_read_unlock()`.

```
    ...
    wiced_dct_read_unlock((void*)number, WICED_TRUE);
}
```

3.4.3 wiced_dct_write()

Call `wiced_dct_write()` to write data to the DCT. You do not need to call `wiced_dct_read_lock()` in order to write data to the DCT.

```
{
    uint32_t  new_number_value = 0x32;
    wiced_dct_write( (void*) &new_number_value, DCT_APP_SECTION,
                     OFFSETOF(my_app_dct_t, number_var), sizeof(uint32_t) );
}
```

3.5 Using Application DCT Data with Homekit

Homekit support uses part of the Application DCT Data area to store Homekit information. Please review the Homekit applications in apps/snip/apple_homekit/lightbulb_service. Looking at file apps/snip/apple_homekit/lightbulb/homekit_application_common_dct.h, you will see how the Application DCT is defined:

```
/**
 * Application should define its DCT structure here
 * The homekit_application_dct_structure_t given below is
 * just an example. Application is free to redefine this structure
 * based on its need and use it.
 */
typedef struct
{
    uint8_t      is_factoryrest;
    char         dct_random_device_id[18];
    uint16_t     placeholder_a;
    char         placeholder_b[20];
} homekit_application_dct_structure_t;

/**
 * Pass in the Application DCT structure to this macro
 */
APPEND_APPLICATION_DCT_TO_HOMEKIT_DCT( homekit_application_dct_structure_t )
```

Looking at file apps/snip/apple_homekit/lightbulb/homekit_application_common_dct.c, you will see how to define default values for the Application DCT.

```
DEFINE_APP_DCT(wiced_homekit_app_shared_dct_t)
{
     HOMEKIT_RESERVED_DCT_SECTION,
     .app_dct.is_factoryrest            = 1,
     .app_dct.dct_random_device_id      = "XX:XX:XX:XX:XX:XX",
     .app_dct.placeholder_a             = 1,
     .app_dct.placeholder_b             = { "Some Initial Value" },
};
```

3.5.1 Accessing the Application DCT with Homekit Support

You will need to use different values for accessing the Application DCT data. Use DCT_HOMEKIT_APP_SECTION for the section, and use HOMEKIT_APP_DCT_OFFSET(<field>) to get the appropriate offset into the data.

```
{
    Uint16_t        *placeholder_a_RAM;
    wiced_dct_read_lock((void**)&placeholder_a_RAM, WICED_TRUE, DCT_HOMEKIT_APP_SECTION,
                        HOMEKIT_APP_DCT_OFFSET(placeholder_a), sizeof( uint16_t ) );
    wiced_dct_read_unlock((void*)placeholder_a_RAM, WICED_TRUE);
}

{
    uint16_t  placeholder_a_RAM = 0x32;
    wiced_dct_write( (void*) &placeholder_a, DCT_HOMEKIT_APP_SECTION,
                     HOMEKIT_APP_DCT_OFFSET(placeholder_a), sizeof(uint16_t) );
}
```

3.6 Using the same DCT for multiple applications

When you have multiple applications defined for your product, you must use the same DCT layout for all applications.

The snip programs used for switching between applications during low power mode shows one example of how to use the same DCT definition across multiple applications.

`apps/snip/multi_image_0`
`apps/snip/multi_image_1`
Notes:

DCT fields specific to the app `multi_image_0` are defined in:

`apps/snip/multi_image_0/app_0_dct.h`
DCT fields specific to the app `multi_image_1` are defined in:

`apps/snip/multi_image_1/app_1_dct.h`
DCT fields common to both applications are defined in:

`apps/snip/multi_image_0/common_dct.h`
Look at the two makefiles to see how the common_dct is defined for the build:

```
apps/snip/multi_image_0/multi_image_0.mk
      APPLICATION_DCT    := common_dct.c

apps/snip/multi_image_1/multi_image_1.mk
      APPLICATION_DCT    := .. /multi_image_0/common_dct.c
```

4 DCT Layout History

In <Wiced-SDK>/WICED/platform/include/platform_dct_old_sdk.h, there are defines for each change to the System DCT structures that are used when building an update Application to run with an older Bootloader (built with an older SDK).

4.1 DCT Changes Between SDK Versions

SDK change	Field(s) Added	Field(s) Changed / Moved within struct	Field(s) Removed
3.1.0→ 3.1.1	■ platform_dct_header_t added apps_locations[DCT_MAX_APP_COUNT]		
3.1.1 → 3.1.2	■ platform_dct_header_t added padding ■ platform_dct_bt_config_t New **OPTIONAL** struct		
3.1.2 → 3.3.0	■ platform_dct_ethernet_config_t New structure ■ platform_dct_network_config_t New structure		
3.3.0 → 3.3.1		■ platform_dct_network_config_t ■ changed char hostname[HOSTNAME_SIZE + 1]; ■ to wiced_hostname_t hostname;	
3.3.1 → 3.4.0	■ platform_dct_bt_config_t ■ added bluetooth_device_class ■ changed padding		
3.4.0 → 3.5.1	■ platform_p2p_config_t ■ New OPTIONAL struct		
3.5.1 → 3.5.2	■ platform_dct_header_t ■ added CRC, sequence number, initial_write ■ platform_dct_ota2_config_t ■ New OPTIONAL struct	■ platform_dct_header_t ■ moved magic_number, write_incomplete	■ platform_dct_header_t ■ removed is_current_dct

SDK change	Field(s) Added	Field(s) Changed / Moved within struct	Field(s) Removed
3.5.2 → 3.6.0		■ platform_dct_ota2_header_t ■ changed padding[1] to force_factory_reset ■ structure size unchanged	
3.6.0 → 3.6.1			
3.6.1→ 3.6.2			
3.6.2 → 3.6.3			
3.6.3→ 3.7.0	■ platform_dct_header_t ■ remove CDC, sequence_number, initial_write (added in SDK 3.5.2) ■ platform_dct_version_t ■ New structure ■ Make all structs **non-OPTIONAL**	■ platform_dct_header_t ■ move magic_number and ■ write_incomplete ■ (back to locations in pre-SDK-3.5.2)	■ platform_dct_header_t ■ add is_current_dct ■ (as in pre-SDK-3.5.2)

4.2 DCT sizes and offsets:

```
* SDK-3.0.1:
 * Starting WICED v3.0.1
 * DCT                                 offset   size
 * platform_dct_data_t              :          0x1c5c
 * platform_dct_header_t            :  0x0000  0x0064
 * platform_dct_mfg_info_t          :  0x0064  0x009c
 * platform_dct_security_t          :  0x0100  0x1810
 * platform_dct_wifi_config_t       :  0x1910  0x034c
 *
 * SDK-3.1.0:
 * Starting WICED v3.1.0
 * DCT                                 offset   size
 * platform_dct_data_t              :          0x1c5c
 * platform_dct_header_t            :  0x0000  0x0064
 * platform_dct_mfg_info_t          :  0x0064  0x009c
 * platform_dct_security_t          :  0x0100  0x1810
 * platform_dct_wifi_config_t       :  0x1910  0x034c
 *
 * SDK-3.1.1:
 * Starting WICED v3.1.1
 * DCT                                 offset   size
 * platform_dct_data_t              :          0x1d7c
 * platform_dct_header_t            :  0x0000  0x0184 ** new fields
 * platform_dct_mfg_info_t          :  0x0184  0x009c
 * platform_dct_security_t          :  0x0220  0x1810
 * platform_dct_wifi_config_t       :  0x1a30  0x034c
 *
 * SDK-3.1.2:
 * Starting WICED v3.1.2
 * DCT                                 offset   size
 * platform_dct_data_t              :          0x1e80
 * platform_dct_header_t            :  0x0000  0x0184 ** optional padding added
 * platform_dct_mfg_info_t          :  0x0184  0x009c
 * platform_dct_security_t          :  0x0220  0x1810
 * platform_dct_wifi_config_t       :  0x1a30  0x034c
 * platform_dct_bt_config_t         :  0x1d7c  0x0101  ** new struct
 *
 * SDK-3.3.0:
 * Starting WICED v3.3.0
 * DCT                                 offset   size
```

```
* platform_dct_data_t              :          0x1eac
* platform_dct_header_t            :  0x0000  0x0184
* platform_dct_mfg_info_t          :  0x0184  0x009c
* platform_dct_security_t          :  0x0220  0x1810
* platform_dct_wifi_config_t       :  0x1a30  0x034c
* platform_dct_ethernet_config_t   :  0x1d7c  0x0008 ** new struct
* platform_dct_network_config_t    :  0x1d84  0x0024 ** new struct
* platform_dct_bt_config_t         :  0x1da8  0x0101
*
* SDK-3.3.1:
* Starting WICED v3.3.1
* DCT                                offset   size
* platform_dct_data_t              :          0x1eac
* platform_dct_header_t            :  0x0000  0x0184
* platform_dct_mfg_info_t          :  0x0184  0x009c
* platform_dct_security_t          :  0x0220  0x1810
* platform_dct_wifi_config_t       :  0x1a30  0x034c
* platform_dct_ethernet_config_t   :  0x1d7c  0x0008
* platform_dct_network_config_t    :  0x1d84  0x0024 ** field change, size is the same
* platform_dct_bt_config_t         :  0x1da8  0x0101
*
* SDK-3.4.0:
* Starting WICED v3.4.0
* DCT                                offset   size
* platform_dct_data_t              :          0x1eb0
* platform_dct_header_t            :  0x0000  0x0184
* platform_dct_mfg_info_t          :  0x0184  0x009c
* platform_dct_security_t          :  0x0220  0x1810
* platform_dct_wifi_config_t       :  0x1a30  0x034c
* platform_dct_ethernet_config_t   :  0x1d7c  0x0008
* platform_dct_network_config_t    :  0x1d84  0x0024
* platform_dct_bt_config_t         :  0x1da8  0x0105 ** added field
*
* SDK-3.5.1:
* Starting WICED v3.5.1
* DCT                                offset   size
* platform_dct_data_t              :          0x1f24
* platform_dct_header_t            :  0x0000  0x0184
* platform_dct_mfg_info_t          :  0x0184  0x009c
* platform_dct_security_t          :  0x0220  0x1810
* platform_dct_wifi_config_t       :  0x1a30  0x034c
* platform_dct_ethernet_config_t   :  0x1d7c  0x0008
```

```
* platform_dct_network_config_t    :  0x1d84  0x0024
* platform_dct_bt_config_t         :  0x1da8  0x0105
* platform_dct_p2p_config_t        :  0x1eb0  0x0074 ** new struct
*
* SDK-3.5.2:
* Starting WICED v3.5.2
* DCT                               offset   size
* platform_dct_data_t              :          0x1f30
* platform_dct_header_t            :  0x0000  0x018c ** changed fields
* platform_dct_mfg_info_t          :  0x018c  0x009c
* platform_dct_dct_security_t      :  0x0228  0x1810
* platform_dct_wifi_config_t       :  0x1a38  0x034c
* platform_dct_ethernet_config_t   :  0x1d84  0x0008
* platform_dct_network_config_t    :  0x1d8c  0x0024
* platform_dct_bt_config_t         :  0x1db0  0x0105
* platform_dct_p2p_config_t        :  0x1eb8  0x0074
* platform_dct_ota2_config_t       :  0x1f2c  0x0004 ** new struct
*
* Starting WICED v3.6.0
* DCT                               offset   size
* platform_dct_data_t              :          0x1f30
* platform_dct_header_t            :  0x0000  0x018c
* platform_dct_mfg_info_t          :  0x018c  0x009c
* platform_dct_dct_security_t      :  0x0228  0x1810
* platform_dct_wifi_config_t       :  0x1a38  0x034c
* platform_dct_ethernet_config_t   :  0x1d84  0x0008
* platform_dct_network_config_t    :  0x1d8c  0x0024
* platform_dct_bt_config_t         :  0x1db0  0x0105
* platform_dct_p2p_config_t        :  0x1eb8  0x0074
* platform_dct_ota2_config_t       :  0x1f2c  0x0004
*
* SDK-3.6.1:
* Starting WICED v3.6.1
* DCT                               offset   size
* platform_dct_data_t              :          0x1f30
* platform_dct_header_t            :  0x0000  0x018c
* platform_dct_mfg_info_t          :  0x018c  0x009c
* platform_dct_dct_security_t      :  0x0228  0x1810
* platform_dct_wifi_config_t       :  0x1a38  0x034c
* platform_dct_ethernet_config_t   :  0x1d84  0x0008
* platform_dct_network_config_t    :  0x1d8c  0x0024
* platform_dct_bt_config_t         :  0x1db0  0x0105
```

```
* platform_dct_p2p_config_t        :  0x1eb8  0x0074
* platform_dct_ota2_config_t       :  0x1f2c  0x0004
*
* SDK-3.6.2:
* Starting WICED v3.6.2-RC1
* DCT                               offset   size
* platform_dct_data_t               :          0x1f30
* platform_dct_header_t             :  0x0000  0x018c
* platform_dct_mfg_info_t           :  0x018c  0x009c
* platform_dct_security_t           :  0x0228  0x1810
* platform_dct_wifi_config_t        :  0x1a38  0x034c
* platform_dct_ethernet_config_t    :  0x1d84  0x0008
* platform_dct_network_config_t     :  0x1d8c  0x0024
* platform_dct_bt_config_t          :  0x1db0  0x0105
* platform_dct_p2p_config_t         :  0x1eb8  0x0074
* platform_dct_ota2_config_t        :  0x1f2c  0x0004 ** Field changed, size stayed the same
*
* SDK-3.7.0:
* DCT                               offset   size
* platform_dct_data_t               :          0x1f38
* platform_dct_header_t             :  0x0000  0x0184 ** changed
* platform_dct_mfg_info_t           :  0x0184  0x009c
* platform_dct_dct_security_t       :  0x0220  0x1810
* platform_dct_wifi_config_t        :  0x1a30  0x034c
* platform_dct_ethernet_config_t    :  0x1d7c  0x0008
* platform_dct_network_config_t     :  0x1d84  0x0024
* platform_dct_bt_config_t          :  0x1da8  0x0105
* platform_dct_p2p_config_t         :  0x1eb0  0x0074
* platform_dct_ota2_config_t        :  0x1f24  0x0004
* platform_dct_version_t            :  0x1f28  0x0010 ** new struct
```

Document Revision History

Document Title: WICED™ DCT (Device Configuration Tables)

Document Number:002-19004

Revision	ECN	Issue Date	Description of Change
**		06/07/2016	**WICED-DCT 0.1** : Initial release
		06/20/2016	**WICED-DCT 0.2** : Clean up
		01/30/2017	**WICED-DCT 0.3** : Added info on using Application DCT and also when Homekit is supported Added info on use of one DCT across multiple applications
*A		03/22/2017	Converted to Cypress template format

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709