WICED Studio

WICED™ Resource Filesystem (Read Only Resource Filesystem)

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

Resource Filesystem - A pre-built, read only file system in the device's FLASH.

Resource Filesystem Description

The Resource Filesystem stores data in the FLASH that the Application will use during runtime. It is expected that the data stored in the filesystem is not to be changed over the lifecycle of the device.

Examples are the WiFi Firmware, documents to serve as web pages, etc.

1 Putting Files in the Resource Filesystem

Files are added to the Resource Filesystem at build time. The directory path <Wiced-SDK>/resources is where Resources are stored for inclusion in the Resource Filesystem.

Examples:

The WiFi Firmware is stored for different devices in:

```
<Wiced-SDK>/resources/firmware/<device>/<version>.bin
```

The `snip/resource_read` example application uses two files in the Resource Filesystem:

```
<Wiced-SDK>/resources/apps/resource_read/test_file01.txt
<Wiced-SDK>/resources/apps/resource_read/test_file02.txt
```

- Create a folder with the same name as your application
  - ☐ Add a file to <Wiced-SDK>/resources/apps/<application_name>
- Add the file(s) to this folder
  - ☐ Ex: <Wiced-SDK>/resources/apps/<application_name>/my_file.bin
- Define the files to add in the Application makefile (example from `snip/resource_read/resource_read.mk`)

```
$(NAME)_RESOURCES :=  apps/resource_read/test_file01.txt apps/resource_read/test_file02.txt
```

If you have problems during the build, look at tools/makefiles/wiced_resources.mk for the list of file types allowed in the resource file system. You may need to add a new file type here.

Current file types (as of 1/25/2017):

```
# filters for resources
TEXT_FILTERS   := %.html %.htm %.txt %.eml %.js %.css %.dat %.cer %.pem %.json %.xml %.py %.key
BINARY_FILTERS := %.jpg %.jpeg %.png %.ico %.gif %.bin %.flac %.wav
```

2 Reading from the Resource Filesystem

The build system names the resources based on their file path, and defines a resource handle for use in the application. The start of the path is always "resource_", followed by the directories, then the filename. The directory and resource file name are separated with "_DIR_" and the "." for the file extension is replaced with a "_". The resource size can be accessed off of the resource handle by using the ".size" field. For more information about the automatic resource name generation convention, refer to <Wiced-SDK>/resources/README.txt.

From the example program <Wiced-SDK>/ apps/resource_read/resource_read.c:

```
#define RES_01 "test_file01.txt"
#define RES_02 "test_file02.txt"
{
    uint32_t size_out;
    const void * buffer;
    resource_result_t result;

    // RES_01
    printf( "Loading resource %s...\n\n", RES_01 );
    // for automatic resource name generation convention, refer to resources/README.txt
    result = resource_get_readonly_buffer (&resources_apps_DIR_resource_read_DIR_test_file01_txt,
                                           0, 0x7fffffff, &size_out, &buffer );
    if ( result != RESOURCE_SUCCESS )
    {
        printf( "Failed to read %s\n", RES_01 );
        return;
    }
    printf( "resource %s size = %d\n\n", RES_01, (int) size_out );
    /* because buffer does not end with NULL terminator, print buffer to stdout with size */
    fwrite( buffer, 1, size_out, stdout );
    printf("\n\n");
    resource_free_readonly_buffer(&resources_apps_DIR_resource_read_DIR_test_file01_txt, buffer);

    // RES_02
    printf( "Loading resource %s...\n\n", RES_02 );
    // for automatic resource name generation convention, refer to resources/README.txt
    result = resource_get_readonly_buffer (&resources_apps_DIR_resource_read_DIR_test_file02_txt,
                                           0, 0x7fffffff, &size_out, &buffer );
    if ( result != RESOURCE_SUCCESS )
    {
        printf( "Failed to read %s\n", RES_02 );
        return;
    }
    printf( "resource %s size = %d\n\n", RES_02, (int) size_out );
    /* because buffer does not end with NULL terminator, print buffer to stdout with size */
    fwrite( buffer, 1, size_out, stdout );
    printf("\n\n");
    resource_free_readonly_buffer(&resources_apps_DIR_resource_read_DIR_test_file02_txt, buffer);
}
```

Document Revision History

Document Title: WICED™ Resource Filesystem (Read Only Resource Filesystem)

Document Number:002-19004

Revision	ECN	Issue Date	Description of Change
**		01/25/2017	WICED-FS 0.1: Initial release
*A		03/21/2017	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