



ONCHIP Technologies
Embedding Intelligence

www.onchiptech.com
(408)-416-3761

ONCHIP USB-Host **Embedded USB Host Stack**

Free Trial Version for NXP LPC2468 Microcontroller

Disclaimer

The content provided in this document is believed to be accurate, but not guaranteed to be entirely free from errors. ONCHIP Technologies accepts no liability whatsoever and assumes that all users understand risks involved.

All rights reserved. All other brands, logos and products are trademarks or registered trademarks of their respective companies.

Copyright notice

None of the materials provided in this document may be used, reproduced or transmitted, in any form or by any means without written permission from the ONCHIP Technologies.

© 2007-2008 ONCHIP Technologies LLC. Seattle, WA, 98056, U.S.A

© 2006-2008 ONCHIP Technologies India Pvt. Ltd. Hyderabad - 500048, INDIA

Licensing

This software is provided for FREE evaluation, if you plan on using this software in a commercial product you need to contact ONCHIP Technologies to obtain license to use in your product. The license will allow you to access ALL the source code, complete documentation and support.

For information on pricing, contact

ONCHIP Technologies, LLC
2100, Lake Washington Blvd N
I 106
Renton, WA, 98056
U.S.A
Phone (408) 416-3761
Fax (408) 608-0325
Email sales@onchiptech.com
Web www.onchiptech.com

ONCHIP Technologies Pvt. Ltd.
#102 C-61 Mahathi Enclave
Madura Nagar
Hyderabad, AP, 500038
INDIA
Phone +91 99 89995551
Phone +91 40 64521046
Email sales@onchiptech.com
Web www.onchiptech.com

Table of Contents

| | |
|---|----|
| Disclaimer | 2 |
| Copyright notice | 2 |
| Licensing | 2 |
| Introduction | 6 |
| 1.1 Advantages..... | 6 |
| 1.2 Intended Audience | 6 |
| Universal Bus (USB) Overview..... | 6 |
| 2.1 USB Host..... | 6 |
| 2.2 USB Device..... | 7 |
| 2.3 USB Data Transfer Types | 7 |
| 2.4 USB Dual-Role (OTG) | 7 |
| USB Host Stack Architecture | 8 |
| 3.1 USB Class Drivers | 8 |
| 3.2 USB Core Driver | 9 |
| 3.3 USB Host Controller Driver (HCD)..... | 9 |
| 3.4 OS Abstraction | 9 |
| USB Host Stack Class Drivers..... | 10 |
| 4.1 Mass Storage Class (MSC) | 10 |
| 4.2 Human Interface Device Class (HIDC) | 10 |
| 4.3 Communication Device Class (CDC)..... | 10 |
| 4.4 Printer Class..... | 10 |

| | | |
|--------|---|----|
| 4.5 | Audio Class..... | 10 |
| 4.6 | Video Class..... | 11 |
| | Getting Started with USB Host Mass Storage Example..... | 12 |
| 5.1 | Embedded Artists LPC2468 OEM board..... | 12 |
| 5.2 | Hardware/Software requirements..... | 12 |
| 5.3 | Jumper Settings on LPC2468 OEM Board..... | 13 |
| 5.4 | Hyper Terminal Settings..... | 13 |
| 5.5 | Download and Run USB Host Project..... | 14 |
| | Software Configuration..... | 15 |
| 6.1 | UART Configuration..... | 15 |
| 6.2 | USB Host Stack Configuration..... | 15 |
| 6.3 | Demo Application Configuration..... | 16 |
| | USB Host Stack Programming..... | 16 |
| 7.1 | USB Host API..... | 16 |
| 7.1.1 | Initialize USB Host Stack: USBHost_Init()..... | 16 |
| 7.1.2 | Determine Device Connection: FS_IsDevConn()..... | 17 |
| 7.1.3 | Open File: FILE_Open()..... | 17 |
| 7.1.4 | Close File: FILE_Close()..... | 17 |
| 7.1.5 | Read File: FILE_Read()..... | 17 |
| 7.1.6 | Write File: FILE_Write()..... | 18 |
| 7.1.7 | Delete File: FILE_Del()..... | 18 |
| 7.1.8 | Create Directory: FILE_CreatDir()..... | 19 |
| 7.1.9 | Delete directory: FILE_DelDir()..... | 19 |
| 7.1.10 | Start Navigation: FILE_NavStart()..... | 19 |
| 7.1.11 | Navigate Child: FILE_NavChild()..... | 19 |

| | | |
|--------|--|----|
| 7.1.12 | Navigate Next: FILE_NavNext()..... | 20 |
| 7.1.13 | Get File Information: FILE_GetInfo() | 20 |
| 7.2 | Error Codes..... | 21 |
| | About ONCHIP Technologies | 22 |

Introduction

This document describes the general overview of ONCHIP USB-Host software stack and provides instructions to evaluate the software with a mass storage example to access the file system on USB flash drives. The section xxx describes the steps to try the example on Embedded Artists (EA) OEM Board that uses NXP LPC2468 micro controller.

The ONCHIP USB-Host stack is fully featured, small memory foot print and high quality USB Host stack software. It is built from the ground up for the embedded applications with portability, reliability and robustness as primary goal.

1.1 Advantages

- Cleanest and programmer-friendly documented USB software
- RTOS independent
- Highly configurable for the target application
- Supports OHCI, EHCI and other host controller standards
- Designed to work with a wide range of hardware for example, NXP LPC2468, Atmel AT91RM9200, Cogent CSB637, Intel PXA270, Philips ISP116x/1362 etc.

1.2 Intended Audience

ONCHIP USB-Host software stack evaluation version is intended at firmware developers who want to evaluate the USB Host features on the Embedded Artists (EA) OEM LPC2468 Board. This example in section xxx demonstrates the use of USB flash drives with the LPC2468 USB Host port.

Universal Bus (USB) Overview

This chapter provides a short overview of the universal serial bus. For complete information about USB, visit http://www.usb.org/developers/docs/usb_20_05122006.zip.

Universal Serial Bus (USB) is a connectivity specification developed by Intel and other technology industry leaders. There are two industry standards for USB, the USB1.1 standard supports bus speeds of 1.5 Mb/s for low-speed USB devices and 12 Mb/s for full-speed USB devices. The USB2.0 standard supports speed of 480Mb/s for high-speed devices and is fully backward compatible with USB1.1.

2.1 USB Host

A USB Host is a hardware/software that interacts with the USB Devices through Host controller hardware. The host is responsible for detecting insertion and removal of devices, providing power, assigning a unique address to the attached devices, and managing control and data flow between the host and devices. The data on the USB bus is transferred via endpoints that act as communication channels between the host and the USB device.

2.2 USB Device

USB device or peripheral is a function such as a Pen Drive, Printer, and Scanner, Modem or other peripheral. USB Device contains a transceiver and a controller together handles the USB protocol at the bus level. The hardware will implement series of memory buffers called endpoints that enable the software to read and write packets over the USB.

2.3 USB Data Transfer Types

USB specification defines four types of transfers.

- Control Transfers
- Bulk Transfer
- Interrupt Transfers
- Isochronous transfers

Every USB device must support control transfers. Control transfers are generally used for command and status type of operations. The Control transfers are used in device enumeration process and some USB devices use control transfers for class-specific requests.

Bulk transfers are used to transfer large amounts of data, but with less priority over other transfers. Bulk transfers have guaranteed data delivery but no guarantee on latency. Only full and high speed devices support bulk transfers

Interrupt transfers are periodic. The interrupt transfers have guaranteed latency.

Isochronous transfers are used when the data needs to be delivered periodically with bounded timing. Isochronous transfers don't have retry, toggle, CRC check, or guarantee of delivery. Isochronous transfers can be supported by full & high speed devices only.

2.4 USB Dual-Role (OTG)

USB On-The-Go (OTG) defines a dual-role device, which can act as either a host or peripheral, and can connect to a PC or other portable devices through the same connector. Portable devices such as handhelds, cell phones and digital cameras that today connect to the PC as a USB peripheral will connect to other USB devices directly using USB OTG port. This means users can perform such functions as sending photos from a digital camera and cell phone to a printer, sharing music files from an MP3 player to another portable player.

USB Host Stack Architecture

ONCHIP-USB-Host software architecture design conforms to USB 2.0 specification. It's modular design adapts to different RTOSs (Real-Time Operating Systems), and USB host controllers. Figure 3-1 shows a block diagram of the different layers of ONCHIP-USB-Host.

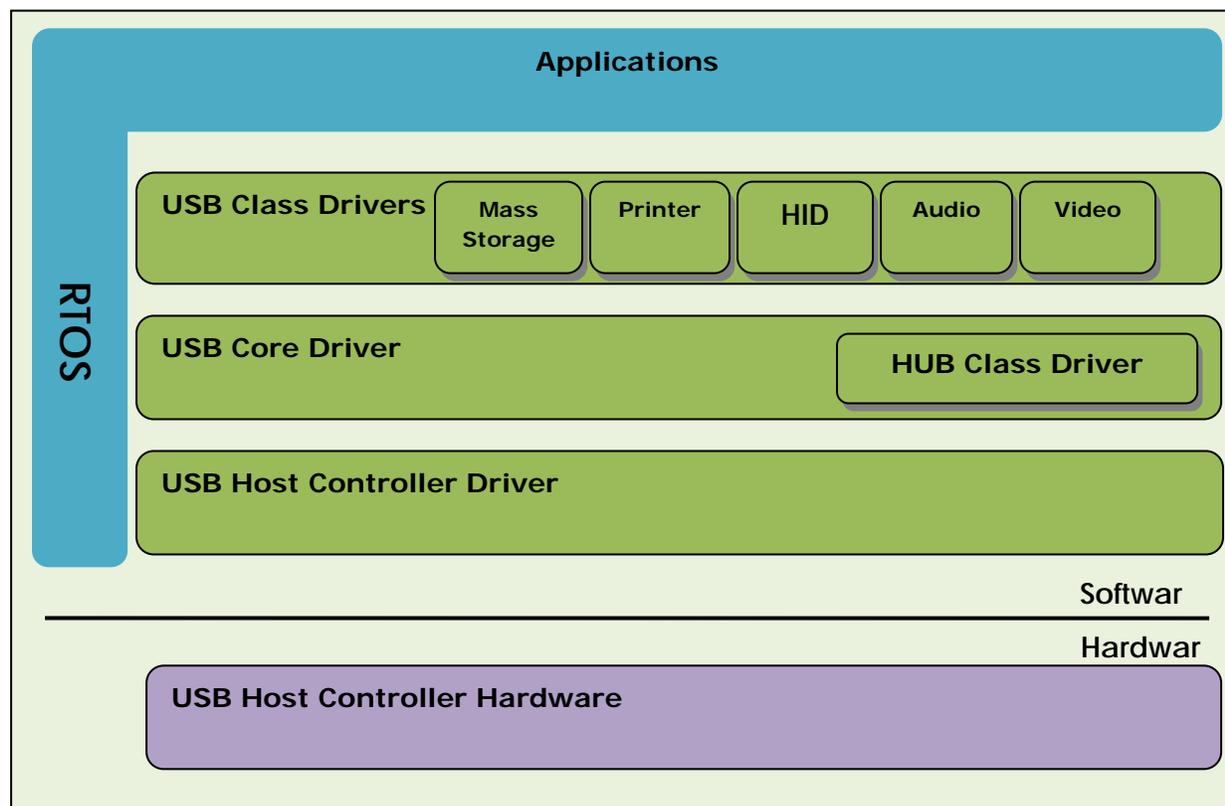


Figure 3-1. ONCHIP-USB-Host Stack Architecture

3.1 USB Class Drivers

The USB Specification defines several classes that characterize the USB device functionality. ONCHIP-USB-Host provides USB Class Drivers for different USB device classes listed in the following table.

| USB Class | Target Devices |
|--------------|---|
| Mass Storage | Flash drives, Zip Drives, MP3 Players, Digital Cameras, External Hard Drives, |
| HID | Keyboard, Mouse and Sensors |
| CDC | USB Modems and USB to Serial bridges |
| Printer | USB Printers |
| Video | Webcam and Camcorders |

| | |
|-----------------------|---|
| Vendor specific class | USB devices with the user defined functionality |
|-----------------------|---|

More information on each class is provided in the following sections of this document.

3.2 USB Core Driver

The **USB Core Driver** manages the USB devices and provides framework for USB Class Drivers. It contains the **Hub Driver** for monitoring the hub ports for device attach or detach events. When a new USB device is attached, the **Hub Driver** enumerates and invokes the appropriate USB Class Driver the device and when the device is detached it performs necessary steps.

3.3 USB Host Controller Driver (HCD)

The **Host Controller Driver (HCD)** communicates with the Host Controller hardware to transfer data across the USB bus. ONCHIP provides **Host Controller Drivers** for full-speed OHCI interface and high-speed EHCI standard interface. Microcontroller such as Atmel AT91RM9200, Intel PXA270 and Phillips ISP116x/1362 have full-speed OHCI interface.

3.4 OS Abstraction

RTOS services such as semaphore and mutex are used by the stack are abstracted in an RTOS-independent manner to enable the stack to be used with any RTOS by implementing these services.

USB Host Stack Class Drivers

ONCHIP-USB-Host offers a variety of class drivers for different USB classes. These USB classes include the standard ones that are defined by usb.org. Vendor-Specific classes can also be developed using the stack framework. Some of the class drivers offered by the ONCHIP-USB-Host stack are detailed in the following sections.

4.1 Mass Storage Class (MSC)

The Mass Storage Class supports mass storage peripherals such as USB flash drives, MP3 players and USB hard disks, floppy drives, zip drives etc,. For the overview on MSC class see the link http://www.usb.org/developers/devclass_docs/usb_msc_overview_1.2.pdf

Mass Storage class specification for Bulk-Only transport available on-line at http://www.usb.org/developers/devclass_docs/usbmassbulk_10.pdf

4.2 Human Interface Device Class (HIDC)

The HID Class Driver supports devices that are used by humans to control the operation of computer systems. Typical examples of HID class devices include Keyboards and pointing devices, Front-panel controls that might be found on devices such as telephones, VCR remote games etc,.

The device class definition for HID is available on-line at http://www.usb.org/developers/devclass_docs/HID1_11.pdf.

4.3 Communication Device Class (CDC)

The CDC Class Driver supports devices such as modem, Ethernet, Fax, ATM, Dongles, Wi-Fi etc. The class definitions for communication devices are available on-line at http://www.usb.org/developers/devclass_docs/usbcdc11.pdf

4.4 Printer Class

The Printer Class Driver supports different USB printer devices. These USB printers are one of the rapidly developing products in the market and they are replacing the old parallel port printers. The device class definition for printer class device is available on-line at http://www.usb.org/developers/devclass_docs/usbprint11.pdf

4.5 Audio Class

The Audio Class Driver supports the different USB audio devices such as LINE IN, MIC IN, and USB audio output devices such as head sets, ear phones, speakers etc. The device class definition for audio devices is available on-line at http://www.usb.org/developers/devclass_docs/audio10.pdf.

All audio data format types are defined in the Audio Data format specification which is available on-line at http://www.usb.org/developers/devclass_docs/frmts10.pdf

All terminal types are defined in the terminal types specification which is available on-line at http://www.usb.org/developers/devclass_docs/termt10.pdf

The class definition for MIDI devices is available on-line at http://www.usb.org/developers/devclass_docs/midi10.pdf

4.6 Video Class

The Video Class Driver supports the different types of video devices such as video cameras, webcams, digital video recorders, digital television tuners etc. The device class definition for video devices is available on-line at http://www.usb.org/developers/devclass_docs/USB_Video_Class_1_1.zip

2. Keil uVision3 IDE.
3. Serial cable to see the log messages on the Hyper Terminal.
4. **ONCHIP-USB-Host-Trial.zip** file. This zip file contains the Keil project for ONCHIP's USB Host stack evaluation. The files included in the zip files are described below.
 - Keil Project located at *Project/UsbHost.Uv2*
 - User Guide located at *Doc/ ONCHIP USB-Host-Free Trial Version User's Guide-v1.0.doc*
 - USB Host stack binary library located at *App/UsbHost.lib*
 - API header files *Include/file_api.h* and *Include/usbh_api.h*
 - Mass storage example application located at *App/app.c*
 - Configuration options header file located at *App/app_cfg.h*

5.3 Jumper Settings on LPC2468 OEM Board

The following jumper setting must be present for the example.

| ISP Jumpers | | UART Jumpers | | USB Host Jumpers | |
|-------------|---------------|--------------|-----------|------------------|---------------|
| P2.10 | NOT CONNECTED | P3.22 | CONNECTED | P0.14 | 2-3 CONNECTED |
| RESET | NOT CONNECTED | P3.18 | CONNECTED | P1.30 | 2-3 CONNECTED |
| | | P3.17 | CONNECTED | USB-B+ | 2-3 CONNECTED |
| | | P3.20 | CONNECTED | USB-B- | 2-3 CONNECTED |
| | | P3.19 | CONNECTED | | |
| | | P3.21 | CONNECTED | | |
| | | P3.16 | CONNECTED | | |
| | | P3.30 | CONNECTED | | |

5.4 Hyper Terminal Settings

Connect a serial cable to UART #1 on the LPC2468 board. Configure the Hyper Terminal settings as 115200-8-N-1 and "No flow control" as show in Fig 5-2

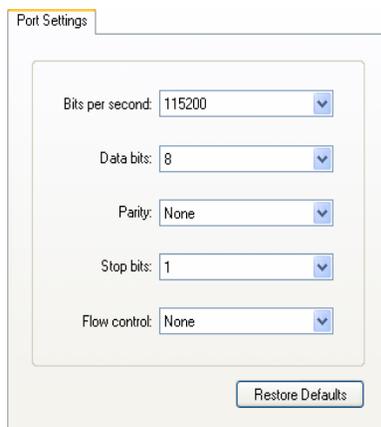


Figure 5-2. Hyper Terminal Settings

5.5 Download and Run USB Host Project

- Connect the power supply by using USB or optional power supply.
- Connect the Keil U-Link USB J-Tag
- Connect the serial cable at UART#1 to see the log messages.
- Open the hyper terminal and make settings as shown in hyper terminal settings.
- Open the OT/USB-HOST directory. In that open Project directory.
- Open the project RtxUsbHost.Uv2 by double clicking on it.
- The directory structure appears in IDE as shown in the figure 5-3.

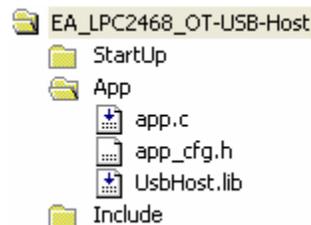


Figure 5-3. Directory structure

- Goto project tab and click 'Rebuild all target files' to compile the project or click the "Rebuild all target files" button on the build tool bar as shown in Fig 5-4.

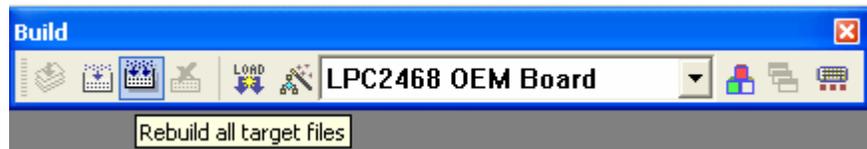


Figure 5-4. Rebuild All Files

- Goto Flash tab and click 'Download' or click the "Download to flash memory" button on the build tool bar as shown in Fig 5-5



Figure 5-5. Download to Flash

- Connect a USB flash drive at USB host port.
- The following output shall appear on the hyper terminal window as shown in Fig 5-6

```
USB HOST Starting...
OHCI Applying Hardware Reset...
OHCI Applying Software Reset...
OHCI Enabling Interrupts...
Manufacturer      : ONCHIP TECHNOLOGIES
Product           : OHCI ROOT HUB
USB HOST Started.
Connect a Mass Storage device
Port 2 : FULL Speed Device Connected.
Manufacturer      : JetFlash
Product           : Mass Storage Device
MaxLUN : 0
Initializing Mass Storage device...
Disk Capacity : 1959 MB
No. of Sectors : 4014077      Sector Size : 512
Copying from A:/MsRead.txt to A:/MsWrite.txt
Copy completed
```

Figure 5-6. Output

Software Configuration

6.1 UART Configuration

The following configuration flag is used for UART pin selection which is specific to the appropriate board.

For Keil MCB2300 board,

```
#define USBH_CFG_UART_SEL    USBH_CFG_UART_KEIL_MCB2300
```

For Embedded Artists LPC2468 OEM board,

```
#define USBH_CFG_UART_SEL    USBH_CFG_UART_EA_OEM_LPC2468
```

6.2 USB Host Stack Configuration

The following constant defines the task priority of the USB host asynchronous task.

```
#define USBH_CFG_TASK_PRIO    2
```

6.3 Demo Application Configuration

The following constant defines the demo application task stack size

```
#define APP_CFG_TASK_STK_SIZE 512
```

The following constant defines the demo application task priority

```
#define APP_CFG_TASK_PRIO 11
```

The following constants provide the user to build and run various file operations. Only one option should be enabled at a time.

```
#define APP_CFG_TEST_COPY DEF_DISABLED
#define APP_CFG_TEST_DEL_FILE DEF_DISABLED
#define APP_CFG_TEST_CREAT_DIR DEF_DISABLED
#define APP_CFG_TEST_DEL_DIR DEF_DISABLED
#define APP_CFG_TEST_NAV DEF_ENABLED
```

USB Host Stack Programming

The USB Host stack provides rich API for writing applications that access the USB devices connected to the USB Host port. In this example we describe the API calls for initializing the USB Host stack and to access the files on the USB drive.

7.1 USB Host API

7.1.1 Initialize USB Host Stack: USBHost_Init()

This function initializes the internals of the stack and the USB Host controller hardware.

```
CPU_INT32S USBHost_Init (CPU_INT32U priority);
```

| Argument | Description |
|----------|-------------------------------|
| priority | USB Host Stack task priority. |

Return Value: OK - if host initialization success otherwise error.

7.1.2 Determine Device Connection: FS_IsDevConn()

This function is used to determine if the USB Drive is connected. This function will return 1 if it is connected else 0.

```
CPU_INT08U FS_IsDevConn (void);
```

Argument: None

Return Value: 1 if the device is connected
0 if the device is not connected

7.1.3 Open File: FILE_Open()

This function opens a file. The user must provide absolute path of the file.

```
CPU_INT32U FILE_Open (CPU_INT08U *file_path, CPU_INT32U mode, CPU_INT32S *err);
```

| Argument | Description |
|------------------|---|
| file_path | Absolute file path of the file to be opened. For ex: "A:/Dir/File.txt" |
| Mode | RDONLY Open a file in read only mode. The file must be present. RDWR Open a file in read/write mode. If the file doesn't exist, create it. |
| Err | If the function is success err = OK else error code. |

Return Value: If the function is success returns file descriptor or 0

7.1.4 Close File: FILE_Close()

This function closes an opened file.

```
void FILE_Close (CPU_INT32U fd);
```

| Argument | Description |
|-----------|---|
| fd | The file descriptor of the opened file. |

Return Value: None

7.1.5 Read File: FILE_Read()

This function reads data from file.

```
CPU_INT32U FILE_Read (CPU_INT32U fd,
```

```
CPU_INT08U *buffer,
CPU_INT32U nbr_byts,
CPU_INT32S *err)
```

| Argument | Description |
|-----------------|--|
| fd | The file descriptor of the opened file from which the data has to be read. |
| buffer | The buffer supplied by the user into which the data has to be read. |
| nbr_byts | The number of bytes requested by the user |
| err | If reading from file is success, returns OK otherwise error code |

Return Value: Number of bytes actually read.

7.1.6 Write File: FILE_Write()

This function writes data to file.

```
CPU_INT32U FILE_Write (CPU_INT32U fd,
CPU_INT08U *buffer,
CPU_INT32U nbr_byts,
CPU_INT32S *err)
```

| Argument | Description |
|-----------------|---|
| fd | The file descriptor of the opened file to which the data has to be written. |
| buffer | The buffer supplied by the user which holds the data to be written to file. |
| nbr_byts | The number of bytes contained in the buffer |
| err | If writing to file is success, returns OK otherwise error code |

Return Value: Number of bytes actually written.

7.1.7 Delete File: FILE_Del()

This function is used to delete file. The user must provide absolute path of the file.

```
CPU_INT32S FILE_Del (CPU_INT08U *file_path)
```

| Argument | Description |
|------------------|---------------------------------------|
| file_path | This is the absolute path of the file |

Return Value: OK - If the file deleted successfully. Otherwise error.

7.1.8 Create Directory: FILE_CreatDir()

This function creates a directory.

```
CPU_INT32S FILE_CreatDir (CPU_INT08U *file_path)
```

| Argument | Description |
|------------------|-------------------------------|
| file_path | The absolute path of the file |

Return Value: OK, if the directory created successfully otherwise error.

7.1.9 Delete directory: FILE_DelDir()

This function deletes a directory.

```
CPU_INT32S FILE_DelDir (CPU_INT08U *file_path)
```

| Argument | Description |
|------------------|-------------------------------|
| file_path | The absolute path of the file |

Return Value: OK, if the directory created successfully otherwise error.

7.1.10 Start Navigation: FILE_NavStart()

This function is used to navigate through the file system. It starts navigation from a given file or directory.

```
CPU_INT32S FILE_NavStart (CPU_INT08U *file_path, FILE_NAV *file_nav)
```

| Argument | Description |
|------------------|---|
| file_path | The absolute path of the file / directory |
| file_nav | Pointer to the FILE_NAV structure. |

Return Value: OK, if the file/directory exists otherwise error.

7.1.11 Navigate Child: FILE_NavChild()

This function is used to navigate from the given directory to its first child. The parent FILE_NAV structure provided to this function must belong to a directory and not to a file.

```
CPU_INT32S FILE_NavChild (FILE_NAV *parent_nav, FILE_NAV *child_nav)
```

| Argument | Description |
|-------------------|--|
| parent_nav | The navigation structure pointer of the parent directory |
| child_nav | This is the navigation structure pointer of the child |

Return Value: OK, if the child file/directory is found otherwise error.

7.1.12 Navigate Next: FILE_NavNext()

This function is used to get the next entry of file or directory the **file_nav** structure is pointed to.

```
CPU_INT32S FILE_NavNext (FILE_NAV *file_nav, FILE_NAV *nxt_nav)
```

| Argument | Description |
|-----------------|--|
| file_nav | The navigation structure pointer of the file/directory from which we have to navigate next |
| nxt_nav | The navigation structure pointer of the next file/directory |

Return Value: OK, if the next file/directory found otherwise error.

7.1.13 Get File Information: FILE_GetInfo()

This function is used to get the information about a file or directory pointed by the FILE_NAV structure. The file information is stored in a FILE_INFO structure.

The FILE_INFO structure is defined as

```
typedef struct file_info {
    CPU_INT08U DIR_Lfn[256]; /* Long file name */
    CPU_INT08U DIR_Sfn[13]; /* Short file name */
    CPU_INT08U DIR_Attr; /* File Attributes */
    CPU_INT08U DIR_NTRes; /* Reserved for Win NT */
    CPU_INT08U DIR_CrtTimeTenth; /* Milli second time stamp */
    CPU_INT16U DIR_CrtTime; /* Creation time */
    CPU_INT16U DIR_CrtDate; /* Creation date */
    CPU_INT16U DIR_LstAccDate; /* Last access date */
    CPU_INT16U DIR_FstClusHI; /* High byte of the starting cluster */
    CPU_INT16U DIR_WrtTime; /* Time of last write */
    CPU_INT16U DIR_WrtDate; /* Date of last write */
    CPU_INT16U DIR_FstClusLO; /* Low byte of the starting cluster */
    CPU_INT32U DIR_FileSize; /* File size */
} FILE_INFO;
```

```
void FILE_GetInfo (FILE_NAV *file_nav, FILE_INFO *file_info)
```

| Argument | Description |
|------------------|--|
| file_nav | This is the navigation structure pointer of the file/directory for which the information is required |
| file_info | This is the FILE_INFO structure pointer in which the required information will be stored |

Return Value: None

7.2 Error Codes

| Number | Error Code | Description |
|--------|--------------------------|---|
| -2001 | ERR_WRONG_BOOTSIG | Boot signature is not 0xAA55 |
| -2002 | ERR_NO_FREE_LFO | Maximum open file instances limit reached. |
| -2003 | ERR_NO_FREE_DFO | Maximum open files limit reached. |
| -2004 | ERR_INVALID_PATH | File path given is not absolute path |
| -2005 | ERR_NO_SPACE_IN_ROOT_DIR | Not enough free space in the root directory |
| -2006 | ERR_LONG_PATH | The path given exceeds 260(max limit) characters |
| -2007 | ERR_NAME_ALREADY_EXISTS | File name with the name specified already exists |
| -2008 | ERR_DIR_NOT_EMPTY | Directory specified is not an empty directory |
| -2009 | ERR_FILE_CANT_HAVE_CHILD | A file cannot have files or directories as its Childs |
| -2010 | ERR_EMPTY_DIR | The directory specified is empty |
| -2011 | ERR_NO_MORE_ENT | Last entry reached. No more entries in the cluster |
| -2012 | ERR_FILE_NOT_FOUND | File with the name specified doesn't exist |
| -2013 | ERR_DISK_FULL | Not enough free space in the disk |
| -2014 | ERR_READ_FAIL | Read operation failed |
| -2015 | ERR_WRITE_FAIL | Write operation failed |
| -2016 | ERR_RDONLY_MODE | Could not write. The file is opened in_RDONLY mode |

About ONCHIP Technologies

OnChip Technologies is a leading supplier of ready to use embedded software components. OnChip is highly committed to deliver high quality software with clean source code, documentation and support. OnChip's products and software design services help companies bring up their products quickly and cost effectively.

| | |
|---|---|
|  <p>OnChip Technologies www.onchiptech.com</p> | <p>Primary Contact</p> <p><i>Surendra Ippagunta</i> <i>Managing Director</i> <i>Tel +1 (408) 416 - 3761(U.S.A)</i> <i>Tel +91(998) 999 - 5551(India)</i> <i>Email surendrai@onchiptech.com</i></p> |
| <p>US Office</p> <p><i>OnChip Technologies LLC</i> <i>Seattle, USA</i> <i>Tel: +1 (408) 416 - 3716</i> <i>Fax: +1 (408) 608 - 0325</i></p> | <p>India Office</p> <p><i>OnChip Technologies India Pvt Ltd</i> <i>Flat# 102, C-61, Madura Nagar</i> <i>Hyderabad - 500038, INDIA</i> <i>Tel: +91 40 64521046</i></p> |