UDC Library Reference Guide

V1.1

Publication Release Date: Jun. 2012

Support Chips:

W55FA Series

Support Platforms:

Non-OS

Table of Contents

1. UDC Library 4

1.1. Programming Guide 4

System Overview 4
UDC Library Property Definition 5
Programming Flow 7

1.2. USB Device (UDC) API 8

udcOpen 8
udcClose 8
udcInit 8
udcDeinit 9
udcIsAttached 9
udcIsAttachedToHost 10

1.3. Mass Storage Class (MSCD) API 10

mscdInit 10
mscdDeinit 11
mscdFlashInit 11
mscdFlashInitNAND 12
mscdFlashInitExtend 13
mscdSdEnable 14
mscdNandEnable 15
mscdSdUserWriteProtectPin 15
mscdSdUserCardDetectPin 16
mscdMassEvent 17

1.4. USB Video Class (UVCD) API 18

uvcdInit 18
uvcdSendImage 19
uvcdIsReady 19

1.5. Example code 20

2. Revision History 21

1. UDC Library

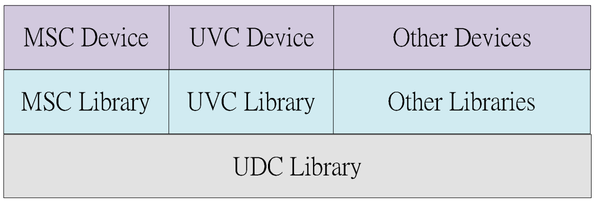
This library is designed to make user application to use FA95/VA95 UDC more easily.
The UDC library has the following features:

- Support all Basic USB operations.
- Pass USB-IF Chapter 9.

SDK Non-OS provide two usb class libraries for the USB class reference sample. User can refer to the libraries to develop him own class libraries. Mass Storage Class device: mscd library.

  - Pass the USB-IF Mass Storage Class Test
  - Provide flash options to build MSC device as a Composite device with RAM disk, NAND Disk, and SD Card Reader.
- USB Video Class device : uvcd library.
  - Pass the USB-IF Video Class Test
  - Provide a video cam sample to send two test patterns to PC.

User can use UDC library to implement all USB basic operations (Send descriptors, Reset command and etc.), and a USB class library (like MSCD) to provide USB class functions.



1.1. Programming Guide

System Overview

The USB device controller interfaces the AHB bus and the UTMI bus. The USB controller contains both the AHB master interface and AHB slave interface. CPU programs the USB controller registers through the AHB slave interface. For IN or OUT transfer, the USB device controller needs to write data to memory or read data from memory through the AHB master interface. The USB device controller is complaint with USB 2.0 specification and it contains four configurable endpoints in addition to control endpoint. These endpoints could be configured to BULK, INTERRUPT or ISO. The USB device controller has a built-in DMA to relieve the load of CPU.

Features

- USB Specification version 2.0 compliant.
- Interfaces between USB 2.0 bus and the AHB bus.
- Supports 16-bit UTMI Interface to USB2.0 Transceiver.
- Support direct register addressing for all registers from the AHB bus.
- Software control for device remote-wakeup.
- AHB bus facilitates connection to common micro controllers and DMA controllers.
- Supports 4 configurable endpoints in addition to Control Endpoint
- Each of these endpoints can be Isochronous, Bulk or Interrupt and they can be either of IN or OUT direction.
- Three different modes of operation of an in-endpoint (Auto validation mode, manual validation mode, Fly mode.)
- DP RAM is used as end point buffer.
- DMA operation is carried out by AHB master
- Supports Endpoint Maximum Packet Size up to 1024 bytes.

UDC Library Property Definition

The UDC library provides property structure to set UDC property more easily.

USBD_INFO_T (The fields for internal used are not in the table)

Name	Description
Descriptor pointer	
pu32DevDescriptor	Device Descriptor pointer
pu32QulDescriptor	Device Qualifier Descriptor pointer
pu32HSConfDescriptor	Standard Configuration Descriptor pointer for High speed
pu32FSConfDescriptor	Standard Configuration Descriptor pointer for Full speed
pu32HOSConfDescriptor	Other Speed Configuration Descriptor pointer for High speed
pu32FOSConfDescriptor	Other Speed Configuration Descriptor pointer for Full speed
pu32StringDescriptor[5]	String Descriptor pointer
Descriptor length	
u32DevDescriptorLen	Device Descriptor Length
u32QulDescriptorLen	Device Qualifier Descriptor pointer Length
u32HSConfDescriptorLen	Standard Configuration Descriptor Length for High speed
u32FSConfDescriptorLen	Standard Configuration Descriptor Length for Full speed
u32HOSConfDescriptorLen	Other Speed Configuration Descriptor Length for High speed
u32FOSConfDescriptorLen	Other Speed Configuration Descriptor Length for Full speed
u32StringDescriptorLen[5]	String Descriptor Length
USBD Init	

pfnHighSpeedInit	High speed USB Device Initialization function
pfnFullSpeedInit	Full speed USB Device Initialization function
Endpoint Number	
i32EPA_Num	Endpoint Number for EPA (-1 : Not used)
i32EPB_Num	Endpoint Number for EPB (-1 : Not used)
i32EPC_Num	Endpoint Number for EPC (-1 : Not used)
i32EPD_Num	Endpoint Number for EPD (-1 : Not used)
Endpoint Call Back	
pfnEPACallBack	Callback function pointer for Endpoint A Interrupt
pfnEPBCallBack	Callback function pointer for Endpoint B Interrupt
pfnEPCCallBack	Callback function pointer for Endpoint C Interrupt
pfnEPDCallBack	Callback function pointer for Endpoint D Interrupt
Class Call Back	
pfnClassDataINCallBack	Callback function pointer for Class Data IN
pfnClassDataOUTCallBack	Callback function pointer for Class Data OUT
pfnDMACompletion	Callback function pointer for DMA Complete
pfnReset	Callback function pointer for USB Reset Interrupt
pfnSOF	Callback function pointer for USB SOF Interrupt
pfnPlug	Callback function pointer for USB Plug Interrupt
pfnUnplug	Callback function pointer for USB Un-Plug Interrupt
VBus status	
u32VbusStatus	VBus Status

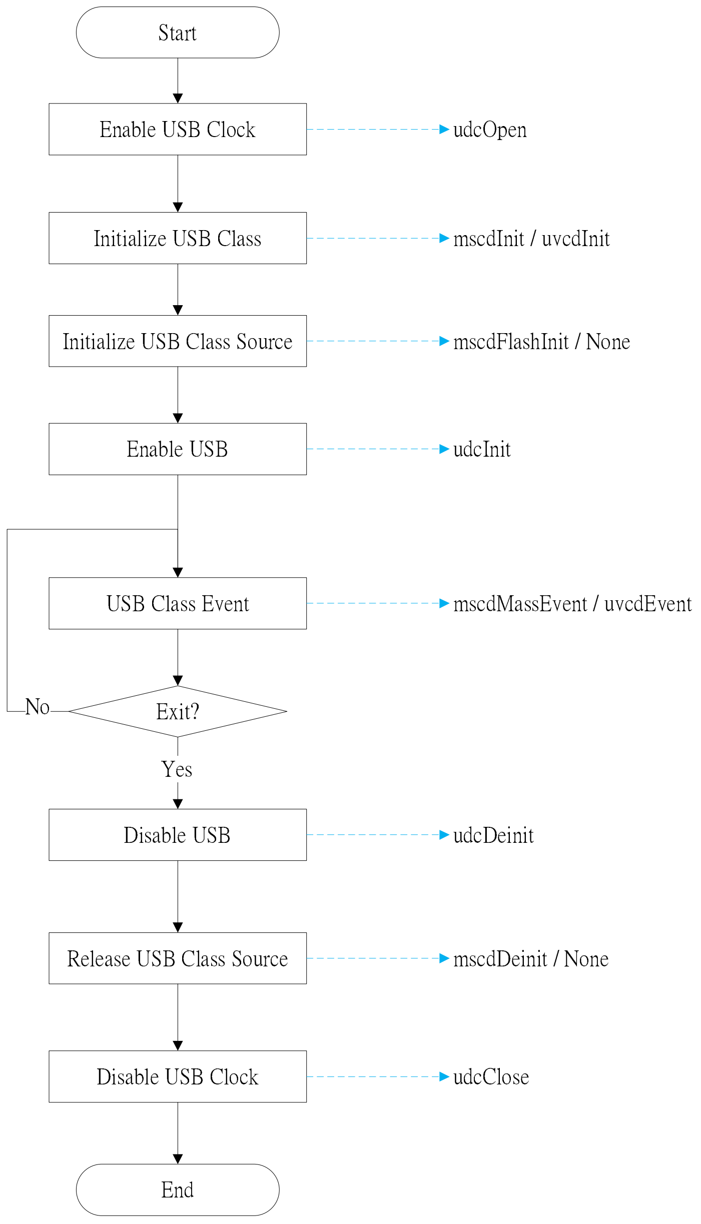
The USB Device initial function initializes the basic setting of USB device controller including endpoints buffer allocate, endpoint number, endpoint type, speed mode, and interrupt, etc. User can modify the function to change USB speed and endpoint properties.

- *pfnHighSpeedInit*
  - mscdHighSpeedInit
  - uvcdHighSpeedInit
- *pfnFullSpeedInit*
  - mscdFullSpeedInit
  - uvcdFullSpeedInit

PC classifies USB device according to the descriptors. With Non-OS SDK structure, the descriptors are initialized in the class Init functions. The functions set proper descriptors and the callback functions.

- *mscdInit*
- *uvcdInit*

Programming Flow



1.2. USB Device (UDC) API

udcOpen

Synopsis

VOID udcOpen(VOID)

Description

This function enables the engine clock.

Parameter

None

Return Value

None

Example

```
udcOpen ();
```

udcClose

Synopsis

VOID udcClose (VOID)

Description

This function disables the engine clock.

Parameter

None

Return Value

None

Example

```
udcClose ();
```

udcInit

Synopsis

VOID udcInit(VOID)

Description

This function initializes the software resource, enables its interrupt, and set VBus detect function.

Parameter

None

Return Value

None

Example

udcInit ();

udcDeinit

Synopsis

VOID udcDeinit (VOID)

Description

Disable VBus detect function

Parameter

None

Return Value

None

Example

udcDeinit ();

udcIsAttached

Synopsis

BOOL udcIsAttached(VOID)

Description

This function can get USB attach status.

Parameter

None

Return Value

TRUE	- USB is attached.
FALSE	- USB isn’t attached.

Example

```
/* Check USB attach status */
if(udcIsAttached ())
        sysprintf(“USB is attached\n”);
else
        sysprintf(“USB isn’t attached\n”);
```

udcIsAttachedToHost

Synopsis

BOOL udcIsAttachedToHost (VOID)

Description

This function can get USB current attach device status.

Parameter

None

Return Value

TRUE	- USB is attached to Host now.
FALSE	- USB doesn’t get any command from Host now.

Example

```
/* Check USB HOST attach status */
if(udcIsAttachedToHost ())
        sysprintf(“USB is attached to Host now\n”);
else
        sysprintf(“USB doesn’t get any command from Host \n”);
```

Note

It takes time for Host to sent command to device. So usr may set a timeout tme to check the status, i.e., user needs to polling the status during the timeout time.

1.3. Mass Storage Class (MSCD) API

mscdInit

Synopsis

VOID mscdInit(VOID)

Description

This function initializes software source (descriptors, callback functions, buffer configuration)

Parameter

None

Return Value

None

Example

```
mscdInit ();
```

mscdDeinit

Synopsis

VOID mscdDeinit (VOID)

Description

This function release software source (allocated by mscdInit)

Parameter

None

Return Value

None

Example

```
mscdDeinit ();
```

mscdFlashInit

Synopsis

UINT8 mscdFlashInit (NDISK_T *pDisk, INT SDsector);

Description

Initial the Flash capacity for usb device controller use.(One chip selector NAND flash and one port SD)

Parameter

pDisk	The internal data for NAND disk information.
SDsector	Total sector for SD disk.

Return Value

0	- Fail

1	- Success

Example

```
NDISK_T MassNDisk;
INT SDsector;
SDsector = sicSdOpen0();
mscdFlashInit(&MassNDisk, SDsector);
```

Note

1. User can assign the export NAND flash (CS0/CS1/CS2) by mscdNandEnable.(Default is CS0 if user doesn’t use mscdNandEnable)
2. User can assign the export SD card (Port0/Port1/Port2) by **mscdSdEnable**. (Default is Port0 if user doesn’t use mscdSdEnable)
3. The API can only single port SD and single CS NAND by mscdSdEnable or mscdNandEnable.
4. If user wants to export only SD, please link w55fa95_MSC_ SD.a
5. If user wants to export only NAND, please link w55fa95_MSC_ NAND.a
6. If user wants to export both SD and NAND, please link w55fa95_MSC_NAND_SD.a

mscdFlashInitNAND

Synopsis

```
UINT8
mscdFlashInitNAND (
        NDISK_T *pDisk,
        NDISK_T *pDisk1,
        NDISK_T *pDisk2,
        INT SDsector
);
```

Description

Initial the Flash capacity for usb device controller use (thress chip selector NAND flash and one port SD) .

Parameter

pDisk	The internal data for NAND disk information for CS0.
pDisk1	The internal data for NAND disk information for CS1.
pDisk2	The internal data for NAND disk information for CS2.
SDsector	Total sector for SD disk.

Return Value

0	- Fail
1	- Success

Example

```
NDISK_T MassNDisk, MassNDisk1, MassNDisk2;
INT SDsector;
SDsector = sicSdOpen0();
mscdFlashInitNAND (&MassNDisk, &MassNDisk1, &MassNDisk2, SDsector);
```

Note

1. User can assign the export NAND flash (CS0/CS1/CS2) by mscdNandEnable.(Default is CS0 if user doesn’t use mscdNandEnable)
2. User can assign the export SD card (Port0/Port1/Port2) by mscdSdEnable. (Default is Port0 if user doesn’t use mscdSdEnable)
3. The API can only single port SD by mscdSdEnable.
4. If user wants to export only SD, please link w55fa95_MSC_ SD.a
5. If user wants to export only NAND, please link w55fa95_MSC_ NAND.a
6. If user wants to export both SD and NAND, please link w55fa95_MSC_NAND_SD.a

mscdFlashInitExtend

Synopsis

```
UINT8
mscdFlashInitExtend (
        NDISK_T *pDisk,
        NDISK_T *pDisk1,
        NDISK_T *pDisk2,
        INT SDsector0,
        INT SDsector1,
        INT SDsector2,
        INT RamSize
);
```

Description

Initial the Flash capacity for usb device controller use (thress chip selector NAND flash and three ports SD).

Parameter

pDisk	The internal data for NAND disk information for CS0.
pDisk1	The internal data for NAND disk information for CS1.
pDisk2	The internal data for NAND disk information for CS2.
SDsector0	Total sector for SD0 disk.
SDsector1	Total sector for SD1 disk.
SDsector2	Total sector for SD2 disk.
RamSize	MSC_RAMDISK_1M~ MSC_RAMDISK_64M

Return Value

0	- Fail
1	- Success

Example

```
NDISK_T MassNDisk, MassNDisk1, MassNDisk2;
INT SDsector;
SDsector = sicSdOpen0();
mscdFlashInitNAND (&MassNDisk, &MassNDisk1, &MassNDisk2, SDsector);
```

Note

1. User can assign the export NAND flash (CS0/CS1/CS2) by mscdNandEnable.(Default is CS0 if user doesn't use mscdNandEnable)
2. User can assign the export SD card (Port0/Port1/Port2) by mscdSdEnable. (Default is Port0 if user doesn't use mscdSdEnable)
3. If user wants to export only SD, please link w55fa95_MSC_ SD.a
4. If user wants to export only NAND, please link w55fa95_MSC_ NAND.a
5. If user wants to export both SD and NAND, please link w55fa95_MSC_NAND_SD.a
6. If user wants to export only all flash, please link w55fa95_MSC_All.a

mscdSdEnable

Synopsis

```
VOID mscdSdEnable (UINT32 u32Enable)
```

Description

This function enables the SD port for MSC.

Parameter

u32Enable	MSC_SD_MP_PORT0~ MSC_SD_MP_PORT2
	MSC_SD_PORT0~ MSC_SD_PORT2

Return Value

None

Example

/* Export two SD ports (Multiple partition) */

mscdSdEnable (MSC_SD_MP_PORT0| MSC_SD_MP_PORT1);

/* Export one SD port (Single partition) */

mscdSdEnable (MSC_SD_ PORT0);

mscdNandEnable

Synopsis

VOID mscdNandEnable (UINT32 u32Enable)

Description

This function enables the NAND CS for MSC.

Parameter

u32Enable	MSC_NAND_CS0~ MSC_ NAND _CS2

Return Value

None

Example

/* Export two NAND CS */

mscdSdEnable (MSC_NAND_CS0| MSC_NAND_CS2);

/* Export one NAND CS */

mscdSdEnable (MSC_NAND_CS0);

mscdSdUserWriteProtectPin

Synopsis

```
VOID mscdSdUserWriteProtectPin (
        UINT32          u32SdPort,
        BOOL            bEnable,
        UINT32          u32GpioPort,
        UINT32          u32GpioPin
)
```

Description

This function enables/disables the SD write protect function and SD write protect pin for MSC.

Parameter

u32SdPort	MSC_SD_PORT0~ MSC_SD_PORT2
bEnable	Enable or Disable Write Protection function (TRUE/FALSE)
u32GpioPort	MSC_SD_GPIO_PORTA~ MSC_SD_GPIO_PORTE, MSC_SD_GPIO_PORTG, MSC_SD_GPIO_PORTH
u32GpioPin	GPIO pin number : 0~15

Return Value

None

Example

```
/* Set GPIOA Pin 2 for SD port0 Write Protect Pin */
mscdSdUserWriteProtectPin (MSC_SD_PORT0, TRUE, MSC_SD_GPIO_PORTA, 2);

/* Disable SD Port 0 Write Protect Pin function*/
mscdSdUserWriteProtectPin (MSC_SD_PORT0, FALSE, 0, 0);
```

Note

Only SD Port0 has default Write Protection pin and Write Protection function is default enable and use the default pin (GPA0).

mscdSdUserCardDetectPin

Synopsis

```
VOID mscdSdUserCardDetectPin (
        UINT32      u32SdPort,
        BOOL        bEnable,
        UINT32      u32GpioPort,
        UINT32      u32GpioPin
)
```

Description

This function enables/disables the SD card detection funvyion for MSC.

Parameter

u32SdPort	MSC_SD_PORT0~ MSC_SD_PORT2
bEnable	Enable or Disable card detection (TRUE/FALSE)

u32GpioPort	MSC_SD_GPIO_PORTA~ MSC_SD_GPIO_PORTE, MSC_SD_GPIO_PORTG, MSC_SD_GPIO_PORTH
u32GpioPin	GPIO pin number : 0~15

Return Value

None

Example

/* Set GPIOA Pin 2 for SD port0 Card detect Pin */

mscdSdUserCardDetectPin (MSC_SD_PORT0, TRUE, MSC_SD_GPIO_PORTA, 2);

/* Disable SD Port 0 Card detect function*/

mscdSdUserCardDetectPin (MSC_SD_PORT0, FALSE, 0, 0);

Note

1. Only SD Port0/2 has default Card detect pin and Card detect function is default enable and use the default pin (GPA1 for Port 0 and GPE11for Port2).
2. If user disable the Card detect function, MSC will consider that the SD card is always exist.

mscdMassEvent

Synopsis

VOID mscdMassEvent (PFN_USBD_EXIT_CALLBACK* callback_fun)

Description

This function processes all the mass storage class commands such as read, write, inquiry, etc. The function has loop in it and it exits the loop according to the return value of the callback function.

Parameter

callback_fun The callback function for the Mass Event Exit condition. If it returns FALSE, the mass event service is disabled.

Return Value

None

Example

mscdMassEvent(udcIsAttached);

Note

The API must be called when all APIs about MSC is completed.

1.4. USB Video Class (UVCD) API

uvcdInit

Synopsis

VOID uvcdInit(PFN_UVCD_PUCONTROL_CALLBACK* callback_func)

Description

This function initializes software source and install the Process Unit Callback function.

Parameter

callback_func	Process Uint Call back function pointer

Return Value

None

Example

```
/* Initial UVC and install Process Uint Call back function */

uvcdInit(ProcessUnitControl);

/* Process Uint Call back function */
UINT32 ProcessUnitControl(UINT32 u32ItemSelect,UINT32 u32Value)
{
        switch(u32ItemSelect)
        {
                case PU_BACKLIGHT_COMPENSATION_CONTROL:
                        sysprintf("Set Backlight -> %d\n",u32Value);
                        break;
                case PU_BRIGHTNESS_CONTROL:
                        sysprintf("Set Brightness -> %d\n",u32Value);
                        break;
                case PU_CONTRAST_CONTROL:
                        sysprintf("Set Contrast -> %d\n",u32Value);
                         break;
                case PU_HUE_CONTROL:
                        sysprintf("Set Hue -> %d\n",u32Value);
                         break;
                case PU_SATURATION_CONTROL:
                        sysprintf("Set Saturation -> %d\n",u32Value);
                         break;
                case PU_SHARPNESS_CONTROL:
                        sysprintf("Set Sharpness -> %d\n",u32Value);
                         break;
                case PU_GAMMA_CONTROL:
                        sysprintf("Set Gamma -> %d\n",u32Value);
                         break;
```

```
                    case PU_POWER_LINE_FREQUENCY_CONTROL:
                            sysprintf("Set Power Line Frequency -> %d\n",u32Value);
                            break;
            }
            return 0;

    }
```

uvcdSendImage

Synopsis

BOOL uvcdSendImage(UINT32 u32Addr, UINT32 u32transferSize, BOOL bStillImage)

Description

This function is to send preview or snapshot image to USB Host.

Parameter

u32Addr	Image data address
u32transferSize	Image data size
bStillImage	TRUE (Snapshot) / FALSE (Preview)

Return Value

None

Example

```
/* Send Image */
uvcdSendImage(u32Addr, u32transferSize, uvcStatus.StillImage);
```

uvcdIsReady

Synopsis

BOOL uvcdIsReady(VOID)

Description

This function is to check UVC is ready to send image or not.

Parameter

None

Return Value

TRUE	Ready
FALSE	Busy

Example

```
/* Wait for Complete */
while(!uvcdIsReady());
```

1.5. Example code

This demo code has sample code for MSC (Mass Storage Class) and UVC (USB Video Class) Please refer to the mass_storage & video_class sample codes of SDK Non-OS.

2. Revision History

Version	Date	Description
V1.1	Jun. 19, 2012	● Add description for the following APIs ■ mscdFlashInitNAND ■ mscdFlashInitExtend ■ mscdSdEnable ■ mscdNandEnable ■ mscdSdUserWriteProtectPin ■ mscdSdUserCardDetectPin
V1	Feb. 25, 2011	● Created