



U-boot Automatic Upgrade and Porting

## **User Guide**

<b>Issue</b>	<b>08</b>
<b>Date</b>	<b>2016-08-31</b>

**Copyright © HiSilicon Technologies Co., Ltd. 2013-2016. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of HiSilicon Technologies Co., Ltd.

## **Trademarks and Permissions**



**HISILICON**, and other HiSilicon icons are trademarks of HiSilicon Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

The purchased products, services and features are stipulated by the contract made between HiSilicon and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## **HiSilicon Technologies Co., Ltd.**

Address: Huawei Industrial Base  
Bantian, Longgang  
Shenzhen 518129  
People's Republic of China

Website: <http://www.hisilicon.com>

Email: [support@hisilicon.com](mailto:support@hisilicon.com)



# About This Document

## Purpose

This document describes how to upgrade images and port code by using a USB flash drive or SD card, including creating upgrade packages and automatically or manually upgrade images. This document also describes the precautions to be taken when you upgrade images.



### NOTE

The Hi3520D, Hi3515A, Hi3536, Hi3521A, Hi3520DV300 and Hi3535 support the image upgrade by using only the USB flash drive. Unless otherwise specified, this document applies to the Hi3516A and Hi3516D.

Unless otherwise specified, this document applies to the Hi3521A and Hi3520DV300. Unless otherwise specified, this document applies to the Hi3518E V201, Hi3516C V200 and Hi3518E V200.

## Related Versions

The following table lists the product versions related to this document.

Product Name	Version	Remarks
Hi3516C	V100	None
Hi3518	V100	The Hi3518 indicates the Hi3518A or Hi3518C.
Hi3518E	V200	The Hi3518E indicates the Hi3518E V200, Hi3518E V201 or Hi3516C V200.
Hi3520A	V100	None
Hi3521	V100	None
Hi3531	V100	None
Hi3531A	V100	Hi3531A does not support the image upgrade by using the SD card.
Hi3520D	V100	The Hi3520D does not support the image upgrade by using the SD card.
Hi3515A	V100	The Hi3515A does not support the image upgrade by using the SD card.
Hi3535	V100	The Hi3535 does not support the image upgrade by using the SD card.



Product Name	Version	Remarks
Hi3516A	V100	None
Hi3516D	V100	None
Hi3536	V100	The Hi3536 does not support the image upgrade by using the SD card.
Hi3521A	V100	The Hi3521A does not support the image upgrade by using the SD card.
Hi3520D	V300	The Hi3520DV300 does not support the image upgrade by using the SD card.
Hi3519	V100	None
Hi3519	V101	None
Hi3516C	V300	None






## Intended Audience

This document is intended for:

- Technical support personnel
- Software development engineers

## Symbol Conventions

The symbols that may be found in this document are defined as follows.

Symbol	Description
 <b>DANGER</b>	Alerts you to a high risk hazard that could, if not avoided, result in serious injury or death.
 <b>WARNING</b>	Alerts you to a medium or low risk hazard that could, if not avoided, result in moderate or minor injury.
 <b>CAUTION</b>	Alerts you to a potentially hazardous situation that could, if not avoided, result in equipment damage, data loss, performance deterioration, or unanticipated results.
 <b>TIP</b>	Provides a tip that may help you solve a problem or save time.
 <b>NOTE</b>	Provides additional information to emphasize or supplement important points in the main text.



## Change History

Changes between document issues are cumulative. Therefore, the latest document issue contains all changes made in previous issues.

### Issue 08 (2016-08-31)

This issue is the eighth official release, which incorporates the following changes:

The description of the Hi3516C V300 is added.

### Issue 07 (2016-08-10)

This issue is the seventh official release, which incorporates the following changes:

The description of the Hi3519 V101 is added.

### Issue 06 (2015-10-31)

This issue is the sixth official release. The description of the Hi3519 V100 is added.

### Issue 05 (2015-09-01)

This issue is the fifth official release. The description of the Hi3531A is added.

### Issue 04 (2015-08-05)

This issue is the fourth official release. The description of the Hi3518E V200, Hi3518E V201, and Hi3516C V201 are added.

### Issue 03 (2015-06-26)

This issue is the third official release. The description of the Hi3521A and Hi3520D V300 are added.

### Issue 02 (2015-04-01)

This issue is the second official release. The description of the Hi3536 is added.

### Issue 01 (2014-12-30)

This issue is the sixth draft release. The description of the Hi3516D is added.

### Issue 00B40 (2014-10-19)

This issue is the fifth draft release. The description of the Hi3516A is added.

### Issue 00B30 (2013-11-18)

This issue is the fourth draft release.

### Issue 00B20 (2013-05-09)

This issue is the third draft release.



# Contents

<b>About This Document.....</b>	<b>i</b>
<b>1 Upgrading Images Under the U-boot.....</b>	<b>1</b>
1.1 Preparations.....	1
1.2 Upgrade Workflow .....	1
1.3 Procedure .....	3
1.4 Operation Instances .....	4
1.4.1 Compiling the U-boot Image Supporting the System Upgrade.....	4
1.4.2 Creating Upgrade Packages .....	4
1.4.3 Upgrading the System .....	5
1.5 Precautions .....	6
<b>2 Upgrading Applications by Porting Code.....</b>	<b>7</b>
2.1 Upgrading Applications by Porting Code.....	7
2.1.1 Procedure .....	7
2.1.2 Instances.....	7
2.2 Porting the Code of the Upgraded System .....	10
2.2.1 Procedure .....	10
2.2.2 Instances.....	10



## Figures

---

<b>Figure 1-1</b> U-boot upgrade workflow.....	<b>2</b>
--	----------



# 1 Upgrading Images Under the U-boot

---

## 1.1 Preparations

Do as follows before upgrading images by using an SD card or USB flash drive:

- Enable the upgrade switch in the code to compile the U-boot image that supports the system upgrade.
- Create images for upgrading.
- Obtain the storage medium such as the USD flash drive or SD card in FAT32 format.

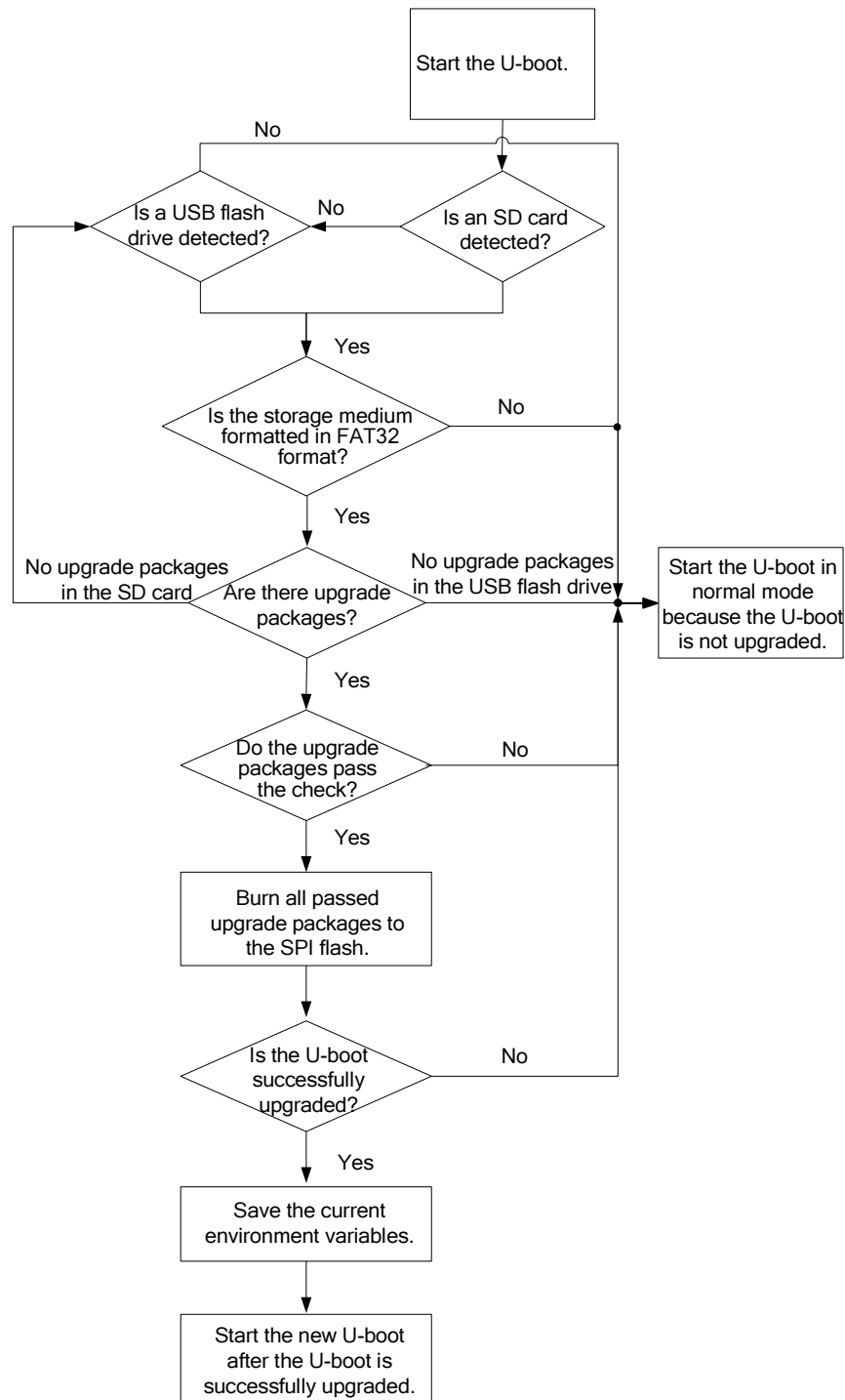
## 1.2 Upgrade Workflow

[Figure 1-1](#) shows the U-boot upgrade workflow.





**Figure 1-1** U-boot upgrade workflow



**NOTE**

When an SD card and USB flash drive are inserted, if there is no valid upgrade package in the SD card, the system automatically upgrades images by using the USB flash drive; if there is a valid upgrade package in the SD card, the system upgrades images by using the upgrade package without scanning the USB flash drive.



## 1.3 Procedure

The upgrade procedure is as follows:

**Step 1** Compile the U-boot image that supports the system upgrade.

1. Modify the chip configuration file to enable the upgrade switch.
  - include/configs/hi3516c.h [Hi3516C]
  - include/configs/hi3518a.h [Hi3518A]
  - include/configs/hi3518c.h [Hi3518C]
  - include/configs/hi3518ev200.h [Hi3518E V200]
  - include/configs/hi3518ev201.h [Hi3518E V201]
  - include/configs/hi3516cv200.h [Hi3516C V200]
  - include/configs/hi3520d.h [Hi3520D/Hi3515A]
  - include/configs/godcare.h [Hi3520A]
  - include/configs/godarm.h [Hi3521]
  - include/configs/godnet.h [Hi3531]
  - include/configs/hi3535.h [Hi3535]
  - include/configs/hi3516a.h and include/configs/hi3516a\_spinand.h [Hi3516A]
  - include/configs/hi3536.h and include/configs/hi3536\_spinand.h [Hi3536]
  - include/configs/hi3521a.h [Hi3521A]
  - include/configs/hi3531a.h [Hi3531A]
  - include/configs/hi3519.h and include/configs/hi3519\_nand.h [Hi3519 V100]
  - include/configs/hi3519v101.h and include/configs/hi3519v101\_nand.h [Hi3519 V101]
  - include/configs/hi3516cv300.h [Hi3516C V300]
2. Compile the U-boot image.
3. Burn the U-boot image to a development board.

**Step 2** Create upgrade packages.

1. To create a U-boot upgrade package, generate the U-boot image through compilation, and run **mkimage**.
2. To create a kernel upgrade package, generate the **uImage** kernel image through compilation, and rename the **uImage** kernel image **kernel**.
3. To create the file system upgrade package rootfs, generate a file system image through compilation, and run **mkimage**.

**Step 3** Insert the SD card or USB flash drive in FAT32 format that stores the upgrade package, and start the board to enable the automatic system upgrade.

----End



## 1.4 Operation Instances

This section uses the Hi3516A as an example to describe the upgrade by using an SD card. The process for upgrading images by using a USB flash drive is similar to that by using an SD card. The difference is that you need to store upgrade packages in the USB flash drive.

### 1.4.1 Compiling the U-boot Image Supporting the System Upgrade

Perform the following steps:

1. Modify the chip configuration file to enable the automatic upgrade function.

```
/*-----  
* sdcard/usb storage system update  
* -----*/  
  
/* #define CONFIG_AUTO_UPDATE 1 */  
#ifdef CONFIG_AUTO_UPDATE  
    #define CONFIG_AUTO_SD_UPDATE 1  
    #define CONFIG_AUTO_USB_UPDATE 1  
#endif
```

2. Compile the U-boot image that supports the automatic upgrade function by running the following commands:

```
make ARCH=arm CROSS_COMPILE=arm-hisiv300-linux- hi3516a_config  
make ARCH=arm CROSS_COMPILE=arm-hisiv300-linux- -j
```

3. Create the final U-boot image that can be burnt to the flash memory by using **mkboot.sh** and **reg\_info.bin**.

```
./mkboot.sh reg_info.bin u-boot-3516a-update.bin
```



#### NOTE

- The Hi3516A provides two cross compilation tool chains (arm-hisiv300-linux- and arm-hisiv400-linux-). Select one of the chains as required.
- **mkboot.sh** and **reg\_info.bin** are stored in **/Hi3516A\_SDK\_V1.0.1.0/package/osdrv/tools/pc/u-boot\_tools/**. For details about the U-boot compilation, see the *Hi3516A/Hi3516D U-boot Porting Application Notes*.
- The final U-boot image **u-boot-3516a-update.bin** supports the automatic upgrade function, and its name can be customized.

### 1.4.2 Creating Upgrade Packages

Compared with the images generated during compilation, the upgrade package images are processed by running **mkimage** for checking the validity of upgrade packages.

- Step 1** Create the U-boot upgrade package.

```
mkimage -A arm -T firmware -C none -n hiboot -d u-boot-hi3516a.bin u-boot
```



**NOTE**

- **u-boot-3516a.bin** is the Hi3516A u-boot image that can be directly burnt or started and may not support the automatic upgrade function.
- The final u-boot upgrade package must be named **u-boot**.

**Step 2** Create the kernel upgrade package by renaming the **uImage** kernel image **kernel**, because the **uImage** kernel image is created by running **mkimage**.

**Step 3** Create the file system upgrade package **rootfs** by running the following command:

```
mkimage -A arm -T filesystem -C none -n hirootfs -d rootfs_256k.jffs2  
rootfs
```



**NOTE**

The final file system upgrade package must be named **rootfs**.

----End

## 1.4.3 Upgrading the System

Perform the following steps:

**Step 1** Burn the U-boot image supporting the automatic upgrade function obtained in section 1.4.1 "[Compiling the U-boot Image Supporting the System Upgrade](#)" to the SPI flash or prepare an SD card or USB flash drive and upgrade packages and download the upgrade packages to the board memory.

**Step 2** Format the SD card or USB flash drive in FAT format.

For details about how to format the SD card or USB flash drive, see the section "Appendix" in the *Peripheral Driver Operation Guide*.

**Step 3** Copy the upgrade packages created in section 1.4.2 "[Creating Upgrade Packages](#)" to the formatted SD card or USB flash drive.

**Step 4** Power on the development board and start the U-boot to enable the images in the upgrade packages in the SD card or USD flash drive to be automatically upgraded.

The following information is displayed during the upgrade:

```
//Upgrade the U-boot.  
reading u-boot      //Scan the U-boot upgrade package.  
reading u-boot      //Copy the U-boot upgrade package to the memory.  
flash erase...      //Erase the area of the SPI flash for storing the U-boot  
image to be upgraded.  
flash write...      //Copy the new U-boot data to the specified area in the  
SPI flash.  
//Upgrade the kernel.  
reading kernel  
reading kernel  
flash erase...  
flash write...  
//Upgrade the file system.  
reading rootfs
```



```
reading rootfs
flash erase...
flash write...
//Save the current environment variables.
Erasing SPI flash, offset 0x00080000 size 256K ...done
Writing to SPI flash, offset 0x00080000 size 256K ...done
//The new system automatically starts.
```

**----End**

## 1.5 Precautions

Note the following:

- When you create the file system upgrade package, the block size of the file system must be the same as the size of the block to be erased in the SPI flash.
- The SD card or USB flash drive must be formatted in FAT format.
- If the SD card or USB flash drive has multiple partitions, upgrade packages must be stored in the first partition. Otherwise, the upgrade program cannot scan the upgrade packages.
- If the SD card and USB flash drive are inserted to the development board at the same time and both the SD card and USB flash drive have valid upgrade packages, the system will upgrade the system images in the SD card but not the USB flash drive. If the SD card has no upgrade packages or the upgrade packages in the SD card are invalid, the system will automatically scan the upgrade packages in the USB flash drive and then upgrade the system.
- The old environment variables are automatically saved after the system is upgraded. If the system is manually restarted, you need to manually set and save environment variables based on the burning position of the new system, ensuring that the system properly starts.



# 2 Upgrading Applications by Porting Code Under the U-boot

---



## NOTE

This section uses upgrading an application (APP) file as an example. The actual address and file name vary according to the actual condition.

## 2.1 Upgrading Applications by Porting Code

### 2.1.1 Procedure

The following operations are involved when you upgrade an APP file by using the network interface under the kernel:

- Adding a file name.
- Adding a file index number.
- Adding the storage position of the new file in the flash memory.
- Adding the process of checking the new file.

### 2.1.2 Instances

The following modifications for porting are made to the **auto\_update.c** file in **./product/hiupdate/**.

#### 2.1.2.1 Adding a File Name

```
/* possible names of files on the medium. */
#define AU_FIRMWARE    "u-boot"
#define AU_KERNEL      "kernel"
#define AU_ROOTFS      "rootfs"
#define AU_APP          "app"

/* pointers to file names */
char *aufile[AU_MAXFILES] = {
    AU_FIRMWARE,
```



```
AU_KERNEL,  
AU_ROOTFS  
AU_APP  
};
```

### 2.1.2.2 Adding a File Index Number

```
/* index of each file in the following arrays */  
#define IDX_FIRMWARE 0  
#define IDX_KERNEL 1  
#define IDX_ROOTFS 2  
#define IDX_APP 3  
/* max. number of files which could interest us */  
#define AU_MAXFILES 4
```

### 2.1.2.3 Adding the Storage Position of the New File in the Flash Memory

The image storage position in the flash memory during the upgrade can be specified in the code or environment variables. If a parameter in the environment variables specifies the storage position, environment variables are preferred. You are advised to specify the burning position by using environment variables because the burning position needs to be frequently changed during debugging.

- Specify the storage position in the code as follows:

```
/*Layout of the flash memory. ST = start address, ND = end address. */  
#define AU_FL_FIRMWARE_ST 0x0  
#define AU_FL_FIRMWARE_ND 0x7FFFF  
#define AU_FL_KERNEL_ST 0x100000  
#define AU_FL_KERNEL_ND 0x5FFFFFF  
#define AU_FL_ROOTFS_ST 0x600000  
#define AU_FL_ROOTFS_ND 0xbFFFFFF  
#define AU_FL_APP_ST 0x..... //Start position of the flash memory  
#define AU_FL_APP_ND 0x..... //End position of the flash memory  
  
/*Sizes of flash areas for each file */  
long ausize[AU_MAXFILES] = {  
    (AU_FL_FIRMWARE_ND + 1) - AU_FL_FIRMWARE_ST,  
    (AU_FL_KERNEL_ND + 1) - AU_FL_KERNEL_ST,  
    (AU_FL_ROOTFS_ND + 1) - AU_FL_ROOTFS_ST,  
    (AU_FL_APP_ND + 1) - AU_FL_APP_ST,  
};  
  
/*Array of flash areas start and end addresses */  
struct flash_layout aufl_layout[AU_MAXFILES] = {  
    { AU_FL_FIRMWARE_ST, AU_FL_FIRMWARE_ND, },  
    { AU_FL_KERNEL_ST, AU_FL_KERNEL_ND, },  
    { AU_FL_ROOTFS_ST, AU_FL_ROOTFS_ND, },  
};
```



```
{ AU_FL_APP_ST, AU_FL_APP_ND,  },  
};
```

- Specify the storage position by assigning values to environment variables as follows:

Search for the `do_auto_update` (void) function. The statement in red needs to be modified.

```
/*  
 * Get image layout from environment.  
 * If the start address and the end address  
 * were not defined in environment variables,  
 * use the default value  
 */  
get_update_env("firmware_st", "firmware_nd");  
get_update_env("kernel_st", "kernel_nd");  
get_update_env("rootfs_st", "rootfs_nd");  
get_update_env("app_st", "app_nd");
```

When **app\_st** and **app\_nd** are set in the environment variables, the position specified in environment variables is used to upgrade the system. Otherwise, the position specified in the code is used.

#### 2.1.2.4 Adding the Process of Checking the New File

Search for the `au_check_header_valid(int idx, long nbytes)` function. The statement in red needs to be modified.

```
/* check the type - could do this all in one gigantic if() */  
if ((idx == IDX_FIRMWARE) && (hdr->ih_type != IH_TYPE_FIRMWARE)) {  
    printf("Image %s wrong type\n", aufile[idx]);  
    return -1;  
}  
  
if ((idx == IDX_KERNEL) && (hdr->ih_type != IH_TYPE_KERNEL)) {  
    printf("Image %s wrong type\n", aufile[idx]);  
    return -1;  
}  
  
if ((idx == IDX_ROOTFS) &&  
    (hdr->ih_type != IH_TYPE_RAMDISK) &&  
    (hdr->ih_type != IH_TYPE_FILESYSTEM)) {  
    printf("Image %s wrong type\n", aufile[idx]);  
    ausize[idx] = 0;  
    return -1;  
}  
  
if ((idx == IDX_APP) && (hdr->ih_type != IH_TYPE_FILESYSTEM)) {  
    printf("Image %s wrong type\n", aufile[idx]);  
    ausize[idx] = 0;  
    return -1;  
}
```



**NOTE**

When you create the APP upgrade image, you are advised to use the **-T filesystem** parameter if you add the check information by running **mkimage**.

## 2.2 Porting the Code of the Upgraded System

### 2.2.1 Procedure

The following operations are involved when you port the code of the upgraded system:

- Adding compilation control switches for drivers and modules to be upgraded in the chip configuration file.
- Completing the MMC driver for the new chip.
- Completing the USB OHCI driver for the new chip.
- Adding automatic upgrade entry functions under the U-boot.

### 2.2.2 Instances

This section uses the Hi3516A as an example to describe how to port the automatic upgrade function to a new platform.

**NOTE**

This instance assumes that the FAT file system, MMC driver, and USB OHCI driver are ported. Therefore, ensure that you use the latest U-boot released by HiSilicon or the U-boot in which the FAT and EXT2 file systems, MMC driver, and USB OHCI driver are ported. For details about how to port the FAT and EXT2 file systems, MMC driver, and USB OHCI driver, see the *U-boot Automatic Upgrade Design Guide*.

- Add the compilation control switch.

Add the following macro definitions to the end of the chip configuration file:

```
/*-----
 * sdcard/usb storage system update
 * -----*/
#define CONFIG_AUTO_UPDATE 1 /*Main switch for
the automatic upgrade function. If the statement is commented out, the
automatic upgrade function is disabled.*/
#ifdef CONFIG_AUTO_UPDATE
    #define CONFIG_AUTO_SD_UPDATE 1
    #define CONFIG_AUTO_USB_UPDATE 1
#endif

#define __LITTLE_ENDIAN 1
#define CONFIG_DOS_PARTITION 1

#define CONFIG_CMD_FAT 1

/*-----
 * sdcard
```



```

* ----- */
#ifdef CONFIG_AUTO_SD_UPDATE
    #define CONFIG_HIMCI_HI3516a    //MMC driver compilation switch
    #define REG_BASE_MCI            0x206e0000
    #define CONFIG_HIMCI_V100
    #define CONFIG_GENERIC_MMC
    #define CONFIG_MMC              1
    #define CONFIG_CMD_MMC

#endif

```

Note that the Hi3516A has two SD card controllers. The base addresses of the registers for two SD card controllers are 0x206e0000 and 0x206f0000 respectively. Configure the macro definition of REG\_BASE\_MCI as required.

```

/* -----
*  usb
* ----- */
#define CONFIG_USB_OHCI            1
#define CONFIG_CMD_USB            1
#define CONFIG_USB_STORAGE        1
#define CONFIG_LEGACY_USB_INIT_SEQ

```

Note: As the Hi3536 supports the USB 3.0 automatic upgrade, the following macros need to be added:

```

#define CONFIG_USB_XHCI            1
#define CONFIG_SYS_USB_XHCI_MAX_ROOT_PORTS 2

```

- Complete the MMC driver.

- Add the **himci100\_3516a.c** file in **drivers/mmc/** to implement the functions of the Hi3516A MMC driver.

```

#define PERI_CRG49                (CRG_REG_BASE + 0xC4)
#define SDIO0_CLK_PCTRL           (0x1 << 4)
#define SDIO0_CLK_BIT_HIGH       (1U << 3)
#define SDIO0_CLK_BIT_LOW        (1U << 2)
#define SDIO0_CKEN                (0x1 << 1)
#define SDIO0_RESET              (0x1 << 0)
#define SYS_PERIPHCTRL4          (0x20120004)
#define SDIO0_DET_MODE           (0x1 << 2)
#define REG_UPDATE_MCI_BASE      REG_BASE_MCI
#define MMC_UHS_REG_EXT          0x108

static void hi_mci_sys_init(void)
{
    unsigned int reg_value;
    unsigned int value;
    himci_writel(0x1010000, REG_UPDATE_MCI_BASE +
MMC_UHS_REG_EXT);
    /* set detect polarity */

```



```

        reg_value = himci_readl(SYS_PERIPHCTRL4);
        reg_value &= ~SDIO0_DET_MODE;
        himci_writel(reg_value, SYS_PERIPHCTRL4);
        /* set clk polarity, mmc clk */
        reg_value = 0;
        reg_value = himci_readl(PERI_CRG49);
        reg_value &= ~(SDIO0CLK_PCTRL);
        reg_value &= ~(SDIO0_CLK_BIT_HIGH);
        reg_value &= ~(SDIO0_CLK_BIT_LOW);
        reg_value |= SDIO0_CKEN;
        himci_writel(reg_value, PERI_CRG49);
    }

static void hi_mci_ctr_reset(void)
{
    unsigned int reg_value;
    reg_value = himci_readl(PERI_CRG49);
    reg_value |= SDIO_RESET;
    himci_writel(reg_value, PERI_CRG49);
}

static void hi_mci_ctr_undo_reset(void)
{
    unsigned int reg_value;
    reg_value = himci_readl(PERI_CRG49);
    reg_value &= ~(SDIO_RESET);
    himci_writel(reg_value, PERI_CRG49);
}

```

- Add the following compilation control code in red to **himciv100.c** in **./drivers/mmc/**:

```

/*****
#ifdef CONFIG_HIMCI_HI3518
#include "himciv100_3518.c"
#endif
*****/

#ifdef CONFIG_HIMCI_HI3516a
#include "himciv100_3516a.c"
#endif
/*****

```

- Complete the USB OHCI driver.
  - Add the **hiusb-3516a.c** file in **drivers/usb/host/hiusb/** to implement the functions of the Hi3516A USB OHCI driver.

```

#define HIUSB_OHCI_BASE        0x100a0000
#define HIUSB_OHCI_DEV_NAME    "hiusb-ohci"

```



```
#define PERI_CRG46          (CRG_REG_BASE + 0xb8)
#define USB_CKEN            (1 << 7)
#define USB_CTRL_UTMI1_REG  (1 << 6)
#define USB_CTRL_UTMI0_REG  (1 << 5)
#define USB_CTRL_HUB_REG    (1 << 4)
#define USBPHY_PORT1_TREQ   (1 << 3)
#define USBPHY_PORT0_TREQ   (1 << 2)
#define USBPHY_REQ          (1 << 1)
#define USB_AHB_SRST_REQ    (1 << 0)
#define PERI_USB            (0x20120078)
#define WORDINTERFACE       (1 << 0)
#define SS_BURST16_EN       (1 << 9)
#define USBOVR_P_CTRL       (1 << 17)
#define MISC_USB            (0x20120080)
static void hiusb_ohci_enable_clk(void)
{
    int reg;

    /* enable clock to EHCI block and HS PHY PLL */
    reg = readl(PERI_CRG46);
    reg |= USB_CKEN;
    reg &= ~(USB_CTRL_UTMI1_REG);
    reg &= ~(USB_CTRL_UTMI0_REG);
    reg &= ~(USB_CTRL_HUB_REG);
    reg &= ~(USBPHY_PORT1_TREQ);
    reg &= ~(USBPHY_PORT0_TREQ);
    reg &= ~(USBPHY_REQ);
    reg &= ~(USB_AHB_SRST_REQ);
    writel(reg, PERI_CRG46);
    udelay(100);

    /* open phy clk */
    writel(0xc06, MISC_USB);
    udelay(10);
    writel(0xc26, MISC_USB);
    udelay(100);

    /* enable phy */
    reg = readl(PERI_USB);
    reg &= ~(WORDINTERFACE);
    /* disable ehci burst16 mode */
    reg &= ~(SS_BURST16_EN);
    reg |= USBOVR_P_CTRL;
```



```
writel(reg, PERI_USB);
udelay(100);
}

static void hiusb_ohci_disable_clk(void)
{
    int reg;

    /* Disable EHCI clock.
    If the HS PHY is unused disable it too. */
    reg = readl(PERI_CRG46);
    reg &= ~(USB_CKEN);
    reg |= (USB_CTRL_UTMI1_REG);
    reg |= (USB_CTRL_UTMI0_REG);
    reg |= (USB_CTRL_HUB_REG);
    reg |= (USBPHY_PORT1_TREQ);
    reg |= (USBPHY_PORT0_TREQ);
    reg |= (USBPHY_REQ);
    reg |= (USB_AHB_SRST_REQ);
    writel(reg, PERI_CRG46);
    udelay(100);

    /* enable phy */
    reg = readl(PERI_USB);
    reg |= (WORDINTERFACE);
    reg |= (SS_BURST16_EN);
    reg |= (USBOVR_P_CTRL);
    writel(reg, PERI_USB);
    udelay(100);
}
```

- Add the following compilation control code in red to **hiusb-ohci.c** in **drivers/usb/host/hiusb/**:

```
/*
 * low level initialization routine, called from usb.c
 */
static char ohci_initd;

#ifdef CONFIG_HI3535
#include "hiusb-3535.c"
#endif
#ifdef CONFIG_HI3516A
#include "hiusb-3516a.c"
#endif
```



- In the **board.c** file in **board/hi3516a**, add the entry function for using the automatic upgrade module. The following texts in red need to be added:

```
int misc_init_r(void)
{
#ifdef CONFIG_RANDOM_ETHADDR
    random_init_r();
#endif
    setenv("verify", "n");
#ifdef CONFIG_AUTO_UPDATE           //System upgrade main switch
    extern int do_auto_update(void);
#ifdef CFG_MMU_HANDLEOK
        dcache_stop();
#endif
        do_auto_update();           //Upgrade function entry
#ifdef CFG_MMU_HANDLEOK
        dcache_start();
#endif
#endif /* CONFIG_AUTO_UPDATE */
    return 0;
}
```