Freescale Semiconductor, Inc.
Application Note

Document Number: AN5243
Rev. 0, 01/2016

System Wakeup From LLS By USB Device

1. Introduction

This application note shows how to wakeup the system from LLS3 mode when the USB acts as HOST.

The USB FS/LS module in the Kinetis MCU is put into static status when the SoC enters VLPR, VLPW, STOP, VLPS and LLSx low power modes, be OFF in the VLLSx low power modes. In static or OFF status, the USB still can wakeup the system from low power to RUN or HSRUN.

In VLPR, VLPW, STOP, and VLPS modes, the USB can wake up the system from low power mode by an asynchronous interrupt triggered to NVIC (in run and wait mode) or AWIC (in stop mode) when there is activity on the USB bus or VBUS detect on USBVDD pin. Some Kinetis SoCs such as KL27, which has the USB Keep Alive (Device-Only) feature, can keep USB alive in STOP/VLPS mode. The host does not need to re-enumerate when exiting the stop modes. The KS22 MCU has all of these features except the Keep Alive. Another new feature in the LLS/VLLS modes is that when the USB power is off, the USBVDD/DP/DM pin can be configured to connect to LLWU and wakeup the system from low power modes by any activity on the three pins.

Contents

1. Introduction 1
2. Overview 2
3. Wakeup system from LLS3 by U-Disk plugin 3
   3.1. Requirements 3
   3.2. Hardware setup 3
   3.3. Software implementation 4
4. Running the demos 7
5. Conclusion 7
6. References 8
7. Revision history 8

2. Overview

The USBVDD/DP/DM pins are connected to the LLWU in the KS22 MCU, this provide the ability to wakeup the system from LLS or VLLS power mode. In both of these low power modes, the USB controller is powered off, it cannot generate any wakeup interrupt to the core when there are events on the USB bus. Only LLWU modules can help to generate wake up interrupt to core.

Figure 1 shows a SoC level overview of the USB wakeup.

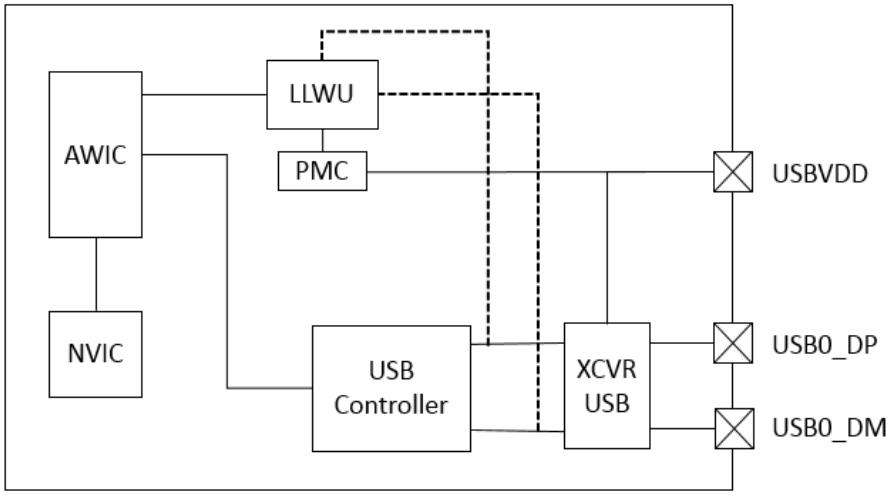

Figure 1. USB wakeup block diagram

Several IP modules work together to implement this feature:

- NVIC: Nested-Vectored Interrupt Controller
- AWIC: Asynchronous Wakeup Interrupt Controller, detects asynchronous wakeup events in stop modes and signals to the clock control logic to resume system clocking
- LLWU: Low Leakage Wakeup Controller, enables the user to select up to 32 external pins and up to 8 internal modules as interrupt wake-up sources from low-leakage power modes
- PMC: Power Management Controller, contains the internal voltage regulator, power on reset (POR), Low Voltage Detect system (LVD), and High Voltage Detect (HVD) system.
- USB Controller: USB full speed OTG controller
- USB XCVR: USB LS/FS transceiver

The USB0_DP/DM is an analog pin connected to the USB XCVR, which converts analog signal to digital. The digital pins connect to both the USB controller and the LLWU module, which means even when the USB controller is powered off, the LLWU can capture the voltage level change on the USB0_DP/DM pin, and trigger an interrupt event to the AWIC, and finally trigger the NVIC to generate a wakeup interrupt. For the USB_VDD pin, this is also an analog pin used to provide power supply to the USB XCVR and it also connects to the PMC module. The PMC module can detect the USB_VDD power change and notify LLWU to trigger an interrupt in the USB Device mode. The USB controller can also notify the AWIC to trigger interrupts to the core when its power is on. But in LLS/VLLS mode, as the USB controller is powered off, the user must use LLWU for system wakeup.

The use case introduced here is to use only the USB0_DP pin in USB Host Full Speed mode to wake up the system from LLS3 low power mode by U-Disk plugin. Other use cases such as using USB0_DM in Host Low Speed mode or USB_VDD as VBUS detect have very similar IP modules configurations.

3. Wakeup system from LLS3 by U-Disk plugin

This section describes how to implement this use case based on the usb_host_msd_fatfs demo application provided in the KS22 SDK. The usb_host_msd_fatfs demonstrates how to use the USB API to access a MSD device, it will print the attached device information when U-disk device is attached, then execute some FATFS APIs to test the FAT Filesystem on the U-Disk, such as reading the directory and files, and creating the directory.

3.1 Requirements

When you initially open a document from the template, you are prompted to enter the values for the document's metadata.

To implement and run this use case, there are some requirements for both software and hardware:

Software Requirement

- SDK 2.0 KS22 release package (SDK_2.0_MAPS-KS22)
- IAR Embedded Workbench 7.50.2

Hardware Requirement

- MAPS-KS22F256 Development Kit, including MCU board and Dock board. This is hown below in Figure 2.
- U-Disk with FAT32 formatted

3.2 Hardware setup

Before implementing the software codes, the KS22 hardware boards for USB Host function must be setup. The following jumpers must be set:

- MAPS-KS22F256 MCU board
  - M1 (1-2), VDD power for KS22
  - JP10 (1-2), 3V3 for board power supply
- MAPS-Dock board

3.3 JP15, JP17 for on board debugger

  - JP7 for debug UART
  - JP10 for USB host power and ID

You can use a USB cable connecting to CN14 from your PC for the total boards power supply. This CN14 acts as a power supply, OpenSDA JLINK debugger, and USB serial debug port. The USB host

port is CN8, the U-Disk can be plugged in to this port. For further details of the boards, refer to the MAPS-KS22F256 Users Guide.

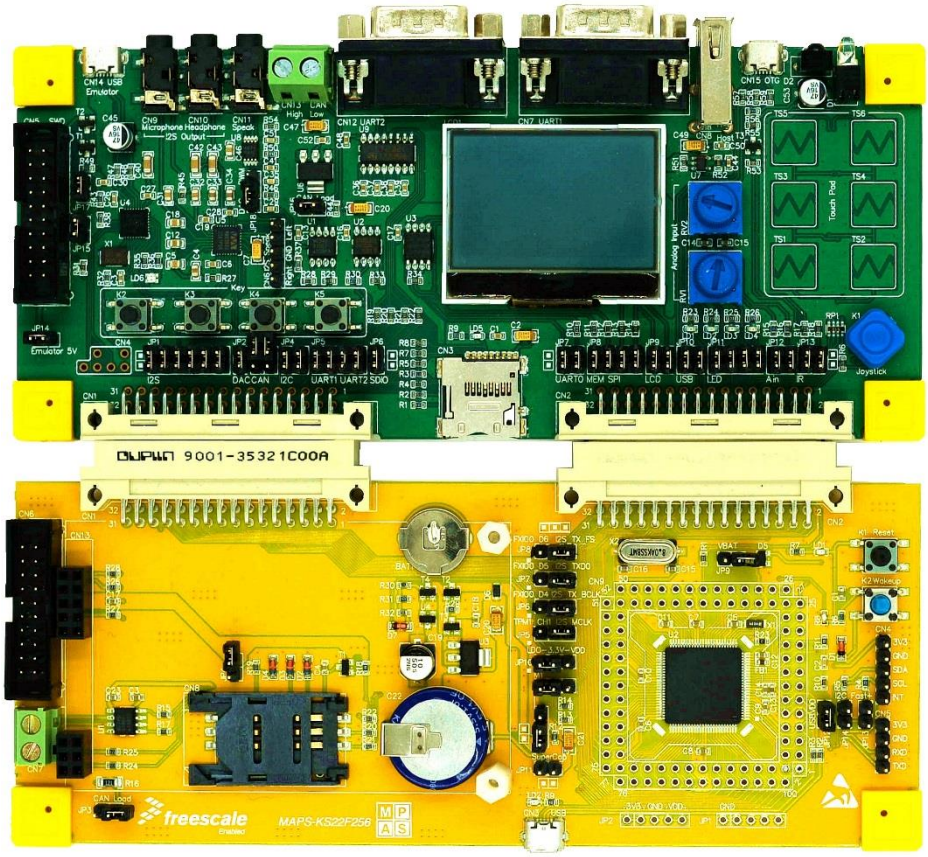

Figure 2. MAPS-KS22F256 Development Kit

3.4 Software implementation

This section describes the software implementation, which is based on usb_host_msd_fatfs demo example in the KS22 SDK (Kinetis Software Development Kit) v2.0 GA Release.

3.4.1 Overview of the USB MSD demo

The demo usb_host_msd_fatfs shows how to call the USB Stack API to implement a USB host accessing the Mass Storage Device such as U-Disk. The source code is in the location: *boards/mapsks22/usb/usb_host_msd_fatfs/bm* in the SDK package.

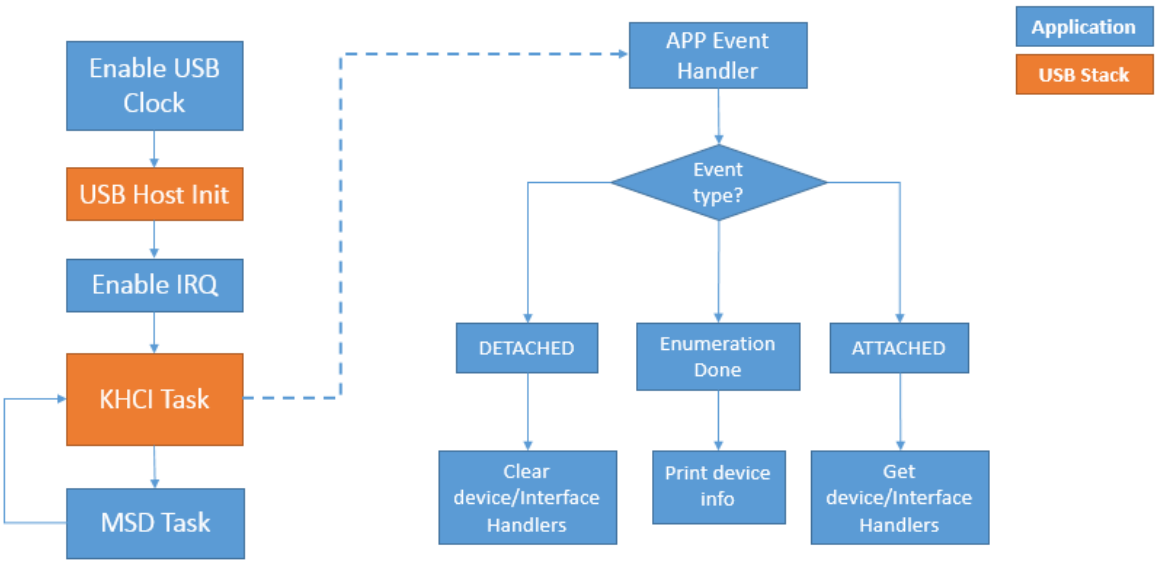

Figure 3. Demo main flow

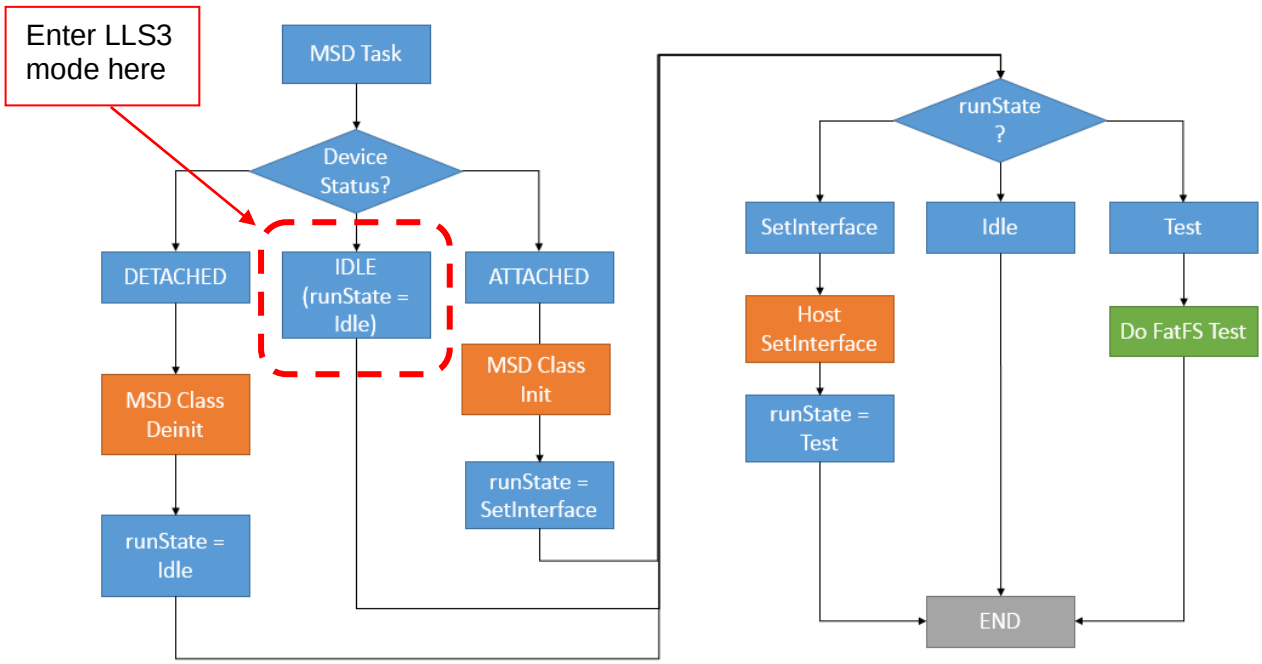

Figure 4. Demo MSD Task flow

Figure 3 shows the main () function software flow.

The major operations in the main () is to initialize the USB controller as HOST, then in an infinite loop, handle the host controller events in application.

The host event handler sets the correct USB device and Interface handler, then pass these handler to MSD Task.

The MSD task initializes the MSD class with handlers, and handles the device status change. Finally it does the U-Disk test when device is connected and enumerated. To do the U-Disk FAT filesystem test, the application mounts it to a logic device, then gets it FAT type, free size. Test the directory creation, list and rename operations, then file creation/read/write/truncate/delete operations.

3.4.2 Modify the demo

To implement how to wake up the system from LLS3 mode by a U-Disk plugin, some changes would be applied to this demo:

- Enable USB_DP pin as a LLWU wakeup external source, with raising edge detection.[1]
- Add LLWU IRQ handler to clear the wakeup flag.
- Enter LLS3 low power mode when USB HOST is in IDLE status.[2]

For KS22F256, the USB_DP pin is the LLWU_P27 wake up source input, so configure the LLWU IP module for the P27 in registers. To enter LLS3 STOP mode which is controlled by SMC, configure the SMC, then call WFI ARM instruction to enter LLS3. Code added in BLUE.

Code snippet for app.c:

```
void enter_lls3(void)
{
    uint32_t dummy;

    MCG->C6 &= ~MCG_C6_CME0_MASK;  // disable clock monitor
    SMC->PMPROT = 0xAA;           // Enable VLPS mode
    dummy = SMC->PMCTRL & ~0x07;  // Clear STOP MODE Control
    SMC->PMCTRL = dummy | 0x3;    // Set STOP MODE to LLSx
    dummy = SMC->STOPCTRL & ~0x07;
    SMC->STOPCTRL = dummy | 0x3;  // Set LLS level to LLS3
    dummy = SMC->PMCTRL;
    SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;
    asm("WFI");                   // Enter LLS3 mode
}

// LLWU IRQ Handler
void LLWU_IRQHandler(void)
{
    LLWU->PF4 = 0x08; // Clear LLWU_P27 wakeup flag
}

int main(void)
{
    BOARD_InitHardware();

    USB_HostApplicationInit();

    LLWU->PF4 = 0x08; // Clear LLWU_P27 wakeup flag
    LLWU->PE7 = 0x40; // Set LLWU_P27 to detect raising edge

    NVIC_EnableIRQ(LLWU_IRQn); // Enable the LLWU IRQ
```

[1] For USB LS device, please use USB_DM as LLWU wakeup source.

[2] Please see the Figure 4 for the LLS3 entering point in software flow

Code snippet for host_msd_fatfs.c:

```
extern void enter_lls3(void);

void USB_HostMsdTask(void *arg)
 {
     usb_status_t status;
……
         switch (msdFatfsInstance->deviceState)
         {
             case kStatus_DEV_Idle:
                usb_echo("Entering LLS3\r\n");
                enter_lls3();                    // call function to enter LLS3 mode
                usb_echo("Leaveing LLS3\r\n");
                break;
```

4 Running the demos

You can download a program image to the microcontroller through CMSIS-DAP or OpenSDA JLINK, depending on which on-board debugger MAPS-Dock installed. The PC host obtains a serial port after a USB cable is connected between the PC host and the on-board debugger USB socket (CN14) on MAPS-Dock. Run a serial terminal tool like Putty or Terminal on PC host and open the USB serial port the debugger provided with speed of 115200bps and format of 8in1.

The project and workspace files of the demo are located at:

boards/mapsks22/usb/usb_host_msd_fatfs/bm/<ide>

<ide>: iar, mdk, kds and etc.

The source file is located in:

boards/mapsks22/usb/usb_host_msd_fatfs/bm

Open the workspace file, e.g. IAR *.eww*, and then build the demo project. Download and run the demo. Then plug the U-Disk into the CN8 USB Host port. After KS22 detects the USB plug-in, print out the basic U-Disk information and running test. After the test is completed, unplug the U-Disk. The system will go into LLS3 mode. You can plug in the U-Disk to wake it up again.

5 Conclusion

With the KS22 MCU acting as a USB Host to communicate with the USB Device, it can go into different low power modes even if there is no power to the USB Controller under some modes like LLSx and VLLSx. By using LLWU, the system can be woken up by any actions on the USB bus with DP/DM signal change by the USB device. This ensures that the KS22 MCU is always staying in the lowest power consumption status when IDLE according to different use cases. This can help you to save power, especially when using mobility devices which have a battery as power supply.

6 References

1. Kinetis KS22 SoC Reference Manual and Data Sheet
2. MAPS-KS22F256 and MAPS-Dock: Freescale MAPS Platform for Kinetis MCUs
3. MAPS-KS22F256 Users Guide
4. KINETIS_SDK: Software Development Kit for Kinetis MCUs

7 Revision history

Table 1. Revision history

Revision number	Date	Substantive changes
0	01/2016	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN5243
Rev. 0
01/2016