Freescale MQX™ I/O Drivers
Users Guide

Document Number:
Rev. 0
4/2011

freescale™
semiconductor

Contents

Chapter 1 USB DCD Driver

1.1 Overview

The USB DCD (USB Device Charger Detection) module is used to detect that the USB device is attached to a charging port (stand-host, charging port or charging host). The chapter describes USB DCD driver. The driver includes:

- USB DCD interrupt-driver I/O
- USB DCD polled I/O

1.2 Source Code Location

Driver	Location
USB DCD interrupt-driven	Source\io\usb_dcd\int
USB DCD polled	Source\io\usb_dcd\polled

1.3 Header Files

To use USB DCD device driver, include the header file usbdcd.h and device-specific usbdcd_xx.h (for example: usbdcd_kn.h for Kinetis family) from source\io\usbdcd in your application or in the BSP file bsp.h. Use the header files according to the following table

Driver	Header file
USBDCD interrupt-driver	• Usb_dcd.h • Usb_dcd_xx.h
USBDCD polled	• Usb_dcd.h • Usb_dcd_xx.h

The files usbdcd_xx_prv.h, usbdcd_pol_prv.h and usbdcd_int_prv.h contain private data structure that USB DCD device driver uses. You must include these files if you recompile the driver. These file are useful for debugging your application.

1.4 Installing Drivers

Each USB DCD device driver provides an installation function that either the BSP or the application calls. The function the calls _io_dev_install() internally. Different installation functions exist for different USB DCD hardware modules. Please see the BSP initialization code in init_bsp.c for functions suitable for your hardware. The function is listed as following table:

Driver	Installation function
Interrupt-driven	_kusbdcd_int_install()
Polled	_kusbdcd_polled_install()

1.5 Initialization Records

When installing the USBDCD device driver, the pointer to initialization record is passed. The following code is an example for the Kinetis micro-controller family as it can be found in the appropriate BSP code (see the init_usb_dcd.c file for details)

```
const KUSB_DCD_INIT_STRUCT _bsp_usb_dcd_init = {
            USBDCD_CLOCK_SPEED_RESET_VALUE,
            USBDCD_TSEQ_INIT_RESET_VALUE,
            USB_DCD_TDCD_DBNC_RESET_VALUE,
            USB_DCD_TVDPSRC_ON_RESET_VALUE,
            USB_DCD_TVDPSRC_CON_RESET_VALUE,
            USB_DCD_CHECK_DM_RESET_VALUE,
            USB_DCD_LEVEL_RESET_VALUE
};
```

1.6 Driver Services

These services, which are supported, are listed below

API	Calls	
	Interrupt-driven	**Polled**
_io_fopen()	_io_usb_dcd_int_open()	_io_usb_dcd_polled_open()
_io_fclose()	_io_usb_dcd_int_close()	_io_usb_dcd_polled_close()
_io_read()	_io_usb_dcd_int_read()	_io_usb_dcd_polled_read()
_io_write()	_io_usb_dcd_int_write()	_io_usb_dcd_polled_write()
_io_ioctl()	_io_usb_dcd_int_ioctl()	_io_usb_dcd_polled_ioctl()

1.7 I/O Control Commands

This section describes the I/O control commands that are used when _io_ioctl() is called for a particular interrupt-driven or polled driver. They are defined in usb_dcd.h

Command	Description
IO_IOCTL_USB_DCD_SET_TSEQ_INIT	Sets sequence initiation time value
IO_IOCTL_USB_DCD_GET_TSEQ_INIT	Gets sequence initiation time value
IO_IOCTL_USB_DCD_GET_TUNITCON	Gets unit connection timer elapse (in ms)
IO_IOCTL_USB_DCD_SET_TDCD_DBNC	Sets Time period to Debounce D+ Signal
IO_IOCTL_USB_DCD_GET_TDCD_DBNC	Gets Time period to Debounce D+ Signal
IO_IOCTL_USB_DCD_SET_TVDPSRC_ON	Sets Time period comparator enabled
IO_IOCTL_USB_DCD_GET_TVDPSRC_ON	Gets Time period comparator enabled
IO_IOCTL_USB_DCD_SET_TVDPSRC_CON	Sets Time period Before enabling D+ Pullup
IO_IOCTL_USB_DCD_GET_TVDPSRC_CON	Gets Time period Before enabling D+ Pullup
IO_IOCTL_USB_DCD_SET_CHECK_DM	Sets Time before check of D- Line
IO_IOCTL_USB_DCD_GET_CHECK_DM	Gets Time before check of D- Line
IO_IOCTL_USB_DCD_SET_CLOCK_SPEED	Sets value of Clock speed (in Khz)
IO_IOCTL_USB_DCD_GET_CLOCK_SPEED	Gets value of Clock speed (in Khz)
IO_IOCTL_USB_DCD_GET_STATUS	Gets status register’s value of USD DCD device
IO_IOCTL_USB_DCD_RESET	Reset(/Stop) USB DCD module
IO_IOCTL_USB_DCD_START	Start USB DCD module
IO_IOCTL_USB_DCD_GET_STATE	Gets USB DCD state

IO_IOCTL_USB_DCD_GET_CHARGER_TYPE	Gets charger type of USB DCD module

1.8 Device States

This section describes the device state values which can be gotten when the _io_ioctl() is called with the IO_IOCTL_USB_DCD_GET_STATE command. They are defined in usb_dcd.h

State	**Description**
USB_DCD_DISABLE	USB DCD module is disable
USB_DCD_ENABLE	USB DCD module is enable
USB_DCD_SEQ_COMPLETE	USB DCD sequence is completed

Chapter 2 CRC Driver

2.1 Overview

A cyclic redundancy check (CRC) module is used to generate 16/32-bit CRC code for error-detection. This section describes the CRC driver that accompanies the MQX release.

The driver implements custom APIs and does not follow the standard driver interface (I/O Subsystem)

2.2 Source Code Location

The source files for the driver are located in source\io\crc directory.

2.3 Header File

To use the CRC driver with the CRC peripheral module, include the header file name crc_kn.h into the application (or into the BSP header file bsp.h)

2.4 API Function Reference – CRC Module Related Functions

This section describes in detail all functions which are used in CRC driver.

2.4.1 CRC_init()

The function (re)initializes the CRC module

Synopsis

```
uint_32 CRC_init(void)
```

Parameters

none

Return Value

MQX_OK

2.4.2 CRC_Config()

The function configures the CRC module

Synopsis

```
uint_32 CRC_Config(uint_32 poly,uint_32 tot,uint_32 totr,
                   uint_32 fxor,uint_32 tcrc)
```

Parameters

Name	Data direction	Length [bit]	Description
Poly	Input	32	The value of the polynomial for the CRC calculation
Tot	Input	2	Type of Transpose for Writes 00 No transposition. 01 Bits in bytes are transposed; bytes are not transposed 10 Both bits in bytes and bytes are transposed 11 Only bytes are transposed; no bits in a byte are transposed
Totr	Input	2	Type of Transpose for Read 00 No transposition. 01 Bits in bytes are transposed; bytes are not transposed

			10 Both bits in bytes and bytes are transposed 11 Only bytes are transposed; no bits in a byte are transposed
Fxor	Input	1	Complement Read of CRC data register
Tcrc	Input	1	Width of CRC protocol 0 16-bit CRC protocol. 1 32-bit CRC protocol.

Description

The function is used to configure the CRC module

Return Value

CRC_ERR_SUCCESS is returned if configuration is successful; otherwise error code is returned.

2.4.3 CRC_Cal_16()

Synopsis

```
uint_32 CRC_Cal_16(uint_32 seed,uint_8 *msg, uint_32 sizeBytes)
```

Parameters

Name	Data Direction	Description
seed	Input	Seed value for CRC module.
Msg	Input	The pointer which points to input data.
sizeBytes	Input	The length of input data [in bytes].

Description

This function is used to calculate the CRC code in 16-bit mode.

Return Value

CRC result.

2.4.4 CRC_Cal_32()

Synopsis

```
uint_32 CRC_Cal_16(uint_32 seed,uint_8 *msg, uint_32 sizeBytes)
```

Parameters

Name	Data Direction	Description
seed	Input	Seed value for CRC module.
Msg	Input	The pointer which point to input data.
sizeBytes	Input	The length of input data [in bytes].

Description

This function is used to calculate the CRC code in 16-bit mode.

Return Value

CRC result.

Note:

The input data is pushed into data register each four bytes. If input data length is not divisible by 4, the remaining bytes are pushed according to three types as following:

- Write single bytes
- Write two bytes
- Write three bytes

In default, the mode “Write single bytes” is used, if you want to change another mode, please add define in crc_kn.c as below, and re-build library:

```
#define BYTE_ENABLES_7_E         if use “write three bytes” mode
#define BYTE_ENABLES_3_6_C       if use “write two bytes” mode
#define BYTE_ENABLES_1_2_4_8     if use “write single bytes” mode
```