smxUSBH™

USB Host Stack

smxUSBH is a USB host stack for embedded systems. It is written in C, and can be ported to any hardware platform. smxUSBH is optimized for SMX®, but can be ported to other RTOSs or run stand alone. It is modularized so that only what is needed will be linked into the final application.

Layers

- **Class Driver Layer** provides USB device support such as mouse, keyboard, hub, printer, mass storage, and serial.
- **USB Driver Layer, or Core,** provides the common USB device framework functionality.
- **Host Controller Driver Layer** provides host controller driver functionality and contains root hub support.
- **Porting Layer** provides service functions related to the hardware, OS, and compiler.

Supported USB Host Controllers

The following drivers are available for smxUSBH and help to provide an out-of-the-box solution for many SoCs and external USB host controllers.

- EHCI
- OHCI
- UHCI
- Analog Dev Blackfin
- Atmel AT91
- Atmel AT91 HS
- Cirrus Logic EP93
- Freescale i.MX31
- Freescale Kxx
- Freescale MCF522xx
- Freescale MCF525x, 532x/7x, 544xx
- Maxim MAX3421
- NXP ISP1160/1
- NXP ISP1362
- NXP ISP1760/1/3
- NXP LPC1/2/3
- NXP LPC HS
- NXP LH7A4xx
- PPC 405 GP EHCI
- Renesas SH
- Renesas uPD720150
- STMicro STM32
- Synopsys DWC
- TI AM1x, AM335x
- TI AM35x
- TI LM3S

Features

- Class drivers are available for audio devices, Ethernet adapters, HIDs, hubs, keyboards, mass storage devices, mice, modems, printers, serial adapters/devices, video, and WiFi devices.
- Compatibility with ARM/Cortex, Blackfin, ColdFire, PowerPC, SH, x86, and other CPUs.
- Supports 16-bit addressing CPUs such as TI TMS320C55xx DSPs.
- Cascading hub support for up to 127 devices. (Hub driver is extra cost option.)
- Compliant with USB Specification 2.0.
- Compliant with EHCI 1.0, OHCI 1.0a, and UHCI 1.1 Specifications.
- Analog Devices Blackfin, Atmel AT91, Freescale ColdFire, Kinetis, Maxim MAX3421, NXP ISP176x, LPC1/2/3, Renesas SH, uPD720150, STM32, Synopsys DWC, TI AM1x, AM3x, and other USB host controller support.
- Supports all four USB data transfers (control, bulk, isochronous, and interrupt).
- Written entirely in ANSI-C.
- Typical code footprint 40 KB.
- Optimized for SMX® RTOS.
- Integrated with smxFS and smxFile for USB disk support.
- OHCI and UHCI support in real mode and under DOS.

Class Drivers

smxUSBH has extensive class driver support, further enabling us to provide an out-of-the-box solution to your USB requirements.

Audio

Allows connecting a USB headset, speaker, or microphone. Many functions are provided for playback and record. See the smxUSBH User's Guide for the API. Supports audio chips such as the Conexant CX20562 and implements feedback for audio output.

CDC ACM (Modem)

Allows connecting most USB modems and some mobile phones if they follow the USB CDC ACM specification. It allows you to use a modem to connect to the Internet or sync and exchange data with a mobile phone.

```
su_CDCACMOpen(id)
su_CDCACMClose(id)
su_CDCACMInserted(id)
su_CDCACMRead(id, pdata, len)
su_CDCACMWrite(id, pdata, len)
su_CDCACMGetLineState(port, pstate)
su_CDCACMGetLineCoding(port, rate, parity, databits,
    stopbits)
su_CDCACMSetLineState(port, state)
su_CDCACMSetLineBreak(port)
su_CDCACMSetCommFeature(port)
su_CDCACMGetCommFeature(port)
su_CDCACMRegisterStateChangeNotify()
```

Ethernet

Allows connecting USB-to-Ethernet adapters that use the ASIX 88772 chip. A TCP/IP stack such as smxNS is also required.

```
su_NetInserted(iID)
su_NetOpen(iID)
su_NetClose(iID)
su_NetWriteData(iID, pData,len)
su_NetGetNodeID(port, pData)
su_NetRegisterPortNotify(port, handler)
```

HID (Generic)

This is the basis to support human interface devices such as a joystick.

```
su_HIDInserted()
su_HIDSetCallback(handler)
```

Hub

```
su_HubInit()
su_HubRelease()
su_HubClearExtFlag()
su_HubGetExtFlag()
```

Keyboard (Included)

Allows connecting a USB keyboard.

```
su_KbdInit()
su_KbdInserted()
su_KbdRelease()
su_KbdSetCallback(handler)
```

Mass Storage

Allows connecting a USB flash disk (thumb drive), USB hard disk, USB card reader, or USB floppy drive. It also supports devices with multiple logical units, such as some USB card readers and thumb drives. smxFS is already interfaced to this driver. It is very easy to interface another file system to it, if desired.

```
su_MStorIO(buf_ptr, first_sector, num_sectors, reading)
su_MStorMaxLUN()
su_MStorMediaChanged()
su_MStorMediaInserted()
su_MStorMediaProtected()
su_MStorMediaRemoved()
su_MStorSectorNum()
su_MStorSectorSize()
```

Mouse (Included)

Allows connecting a USB mouse.

```
su_MouseInit()
su_MouseInserted()
su_MouseRelease()
su_MouseSetCallback(handler)
```

Printer

Allows connecting a USB printer and print to it or read data from it. You need to implement the Print Control Language for your specific printer.

su_PrnID(pdata, len)
su_PrnInit()
su_PrnInserted()
su_PrnRead(pdata, len)
su_PrnRelease()
su_PrnReset()
su_PrnStatus()
su_PrnWrite(pdata, len)

Serial

Allows connecting any serial device that Windows XP or 2000 can support without a custom driver, such as a Windows Mobile 5 device. Unfortunately, most serial adapters do require installation of a custom driver. Additional code must be developed to support such an adapter, which could require significant effort. We have already created a driver for USB to serial converters that use the FTDI FT232 and Prolific PL2303 chips. Please see the smxUSBH User's Guide for details, and discuss your requirements with us.

Windows
su_SerialOpen(id)
su_SerialClose(id)
su_SerialInserted(id)
su_SerialRead(id, pdata, len)
su_SerialWrite(id, pdata, len)
su_SerialGetLineState(port, pstate)
su_SerialGetLineCoding(port, rate, parity, databits, stopbits)

FT232 and PL2303
su_xxxOpen(id)
su_xxxClose(id)
su_xxxInserted(id)
su_xxxRead(id, pdata, len)
su_xxxWrite(id, pdata, len)
su_xxxSetModemCtrl (port, data)
su_xxxSetFlowCtrl (port, data)
su_xxxSetLineCoding(port, rate, parity, databits, stopbits)
su_xxxGetModemStatus (port)
su_xxxGetStatus (port)
su_xxxSetEventChar(id, data)
su_xxxSetErrorChar(id, data)
su_xxxSetLatencyTimer(id, ms)
su_xxxGetLatencyTimer(id)

xxx = FTDI or PL2303

Video

Allows connecting a USB video camera. See the smxUSBH User's Guide for limitations.

su_VideoCameraGetStillImageFormatNum(id)
su_VideoCameraGetStillImageFormat(id, index, pformat)
su_VideoCameraGetCaptureFormatNum(id)
su_VideoCameraGetCaptureFormat(id, index, pformat)
su_VideoCameraGetCurrent(id, pcurrent)
su_VideoCameraGetDefault(id, pdefault)
su_VideoCameraGetMin(id, pmin)
su_VideoCameraGetMax(id, pmax)
su_VideoCameraGetInfo(id, pinfo)
su_VideoCameraSetCaptureFrame(id, iformat, width, height, frameinterval)
su_VideoCameraSetStillImageFormat(id, format, width, height)
su_VideoCameraSetCurrent(id, pnewcurrent)
su_VideoCameraGetCaptureSize(id, pmaxvidframesz, pmaxpayloadbufsz)
su_VideoCameraOpen(id)
su_VideoCameraClose(id)
su_VideoCameraCapture(id, pdata, ilen, pbinclnewframe, pnewframeoffset)
su_VideoInserted(id)

WiFi

su_RTxxxxInit()
su_RTxxxxRelease()
su_RTxxxxGetOper()
su_RTxxxxInserted()

Wireless

Huawei K4510 3G Modem and Sierra Wireless Dongle are supported. See the smxUSBH User's Guide for information.

Writing New Drivers

Contact us first to make sure we are not already working on the driver you need.

smxUSBH provides a class driver template and a section in the manual to help you write a new class driver, if needed.

smxUSBH also provides a USB host controller driver template and a section in the manual, to help you write a new driver in case it does not support your USB host controller.

Porting

smxUSBH was developed for use with SMX®, but it can be ported to any RTOS or run stand-alone. The RTOS porting layer consists of two files, uosport.h and uosport.c. These files contain definitions, macros, and functions to port to a new RTOS. smxUSBH works best in a multitasking environment. However, it can also be ported to a non-multitasking stand-alone environment.

Due to SMX's extensive processor support, little or no porting is necessary when smxUSBH is used with it.

The hardware porting layer consists of two files, uhdwport.h and uhdwport.c. These files contain definitions, macros, and functions to port smxUSBH to a new processor. In addition, if the USB host controller is not among those already supported, a new driver will need to be written. smxUSBH also provides class driver and host controller driver template so you can add new class or host controller to your platform easily. Those templates are also used by us to support new hardware and classes.

Multiple Ports and Controllers

smxUSBH supports multiple ports on a controller and multiple controllers of different types, but not multiple controllers of the same type.

16-Bit Addressing Support

smxUSBH supports processors that can only do 16-bit memory addressing (not byte addressing) such as the TI TMS320C55xx DSPs. These processors are difficult to support for typical communication protocols because of byte data and byte fields in standard protocol data structures. This support is enabled by a configuration option in smxUSBH.

Real Mode and DOS Support

There is a need for legacy x86 systems to add USB support, especially for flash disks. Because of this, smxUSBH supports ISP116x, ISP1362, OHCI and UHCI in real mode. OHCI uses memory mapped I/O, and the PCI BIOS assigns a high address near the top of the 4GB memory space, which is not accessible in real mode. We provide two solutions for this. A 386 or better may be required. See the OHCI section of the smxUSBH User's Guide for details. UHCI uses x86 I/O space for access to UHCI registers, but a 386 or better is required for the 32-bit I/O instructions.

Testing

Unlike USB Device, there is no protocol compliance testing for software, for USB Host. Instead, we test the host stack and class drivers using multiple devices of each class, as listed in Appendix C of the smxUSBH User's Guide.

Code Size

Code size can vary greatly depending upon the processor, compiler, and optimization level. Below, *Core* includes USB Core and Porting Layer.

Component	ARM Thm IAR (KB)	ARM IAR (KB)	BF VDSP (KB)	CF CW (KB)
Core	8	10	13	11
EHCI				14
OHCI	6.5	9		11
UHCI				16
NXP ISP116x	4.5	6		8
NXP ISP1362	5			8.5
NXP ISP176x				15
Analog Dev BF5xx	—	—	5	—
Freescale CF522xx	—	—	—	7
Luminary LM3S	4	—	—	—
Maxim MAX3421	3	4		
Audio	8	11		12
CDC ACM (Modem)	2	3	2	4
HID Mouse & Kbd	2.5	3.5	2	4
HID Generic	4.5	6.5	3	7
Hub	2	3	2	3
Mass Storage	5	6.5	6.5	7.5
Printer	1.5	2	2	3
Serial (FTDI)	1.5	2.5	1.5	4
Serial (Prolific)	3	4	1	4.5
Serial (Windows)	1.5	2.5	1	3

AT91, EP93xx, LPC24xx, LPC3xxx, and LH7A404: See OHCI entry.
MCF5227x, MCF525x, MCF532x/7x, and MCF5445x: See EHCI entry.
IAR = IAR EWARM; CW = CodeWarrior; VDSP = Visual DSP

Data Size

All RAM that smxUSBH uses for data is pre-allocated from the heap when smxUSBH is first initialized. Following is a table of RAM usage:

Component	Size (KB)
Core	2
EHCI	6
OHCI	4
UHCI	70
NXP ISP116x	2
NXP ISP1362	2
NXP ISP176x	2
Analog Devices BF5xx	1
Freescale MCF522xx	1
Luminary	1
Maxim MAX3421	1
Audio	6
CDC ACM (Modem)	1
HID Mouse and Kbd	0.5
HID Generic	4
Hub	1
Mass Storage	2
Printer	2
Serial Converter (FTDI)	2
Serial Converter (Prolific)	2
Serial (Windows)	1

AT91, EP93xx, LPC24xx, LPC3xxx, and LH7A404: See OHCI entry.
MCF5227x, MCF525x, MCF532x/7x, and MCF5445x: See EHCI entry.

UHCI requires much more memory than OHCI because the hardware is more rudimentary and the software must do more work. The UHCI RAM requirements include 1024 *Transfer Descriptors* (TDs) of 64 bytes each (64KB total). The number of TDs can be reduced, but performance suffers. For example with 128 TDs, performance is reduced by a factor of 10. For OHCI, there is no RAM vs. performance tradeoff. OHCI is obviously preferable to UHCI for limited RAM systems.

Performance for Mass Storage

The following table shows **raw** transfer speed from and to a USB flash disk. 20MB total transfers are done 4KB at a time.

The following table shows **smxFS** read/write performance for the same USB flash disk. Total file size is 20MB with 4KB transferred, at a time.

Host Controller	File Read	File Write
EHCI (NEC)	10556 KB/sec	7787 KB/sec
OHCI (NEC)	885 KB/sec	817 KB/sec
UHCI (VIA)	611 KB/sec	590 KB/sec
ISP116x (NXP)	336 KB/sec	328 KB/sec
ISP1362 (NXP)	591 KB/s	478 KB/s
ISP176x (NXP)	7023 KB/s	3072 KB/s
BF5xx (ADI)	9500 KB/s	7500 KB/s

The following table shows **raw** data transfer speed between EHCI and LACIE USB 2.0 40GB hard disk.

Host Controller	Raw Reading	Raw Writing
EHCI (VIA)	24966 KB/sec	19784 KB/sec

Performance for Serial

The following table shows serial read/write performance. The device driver reads/writes 256 bytes of data at a time from/to the USB serial device (not connected to a real RS232 device).

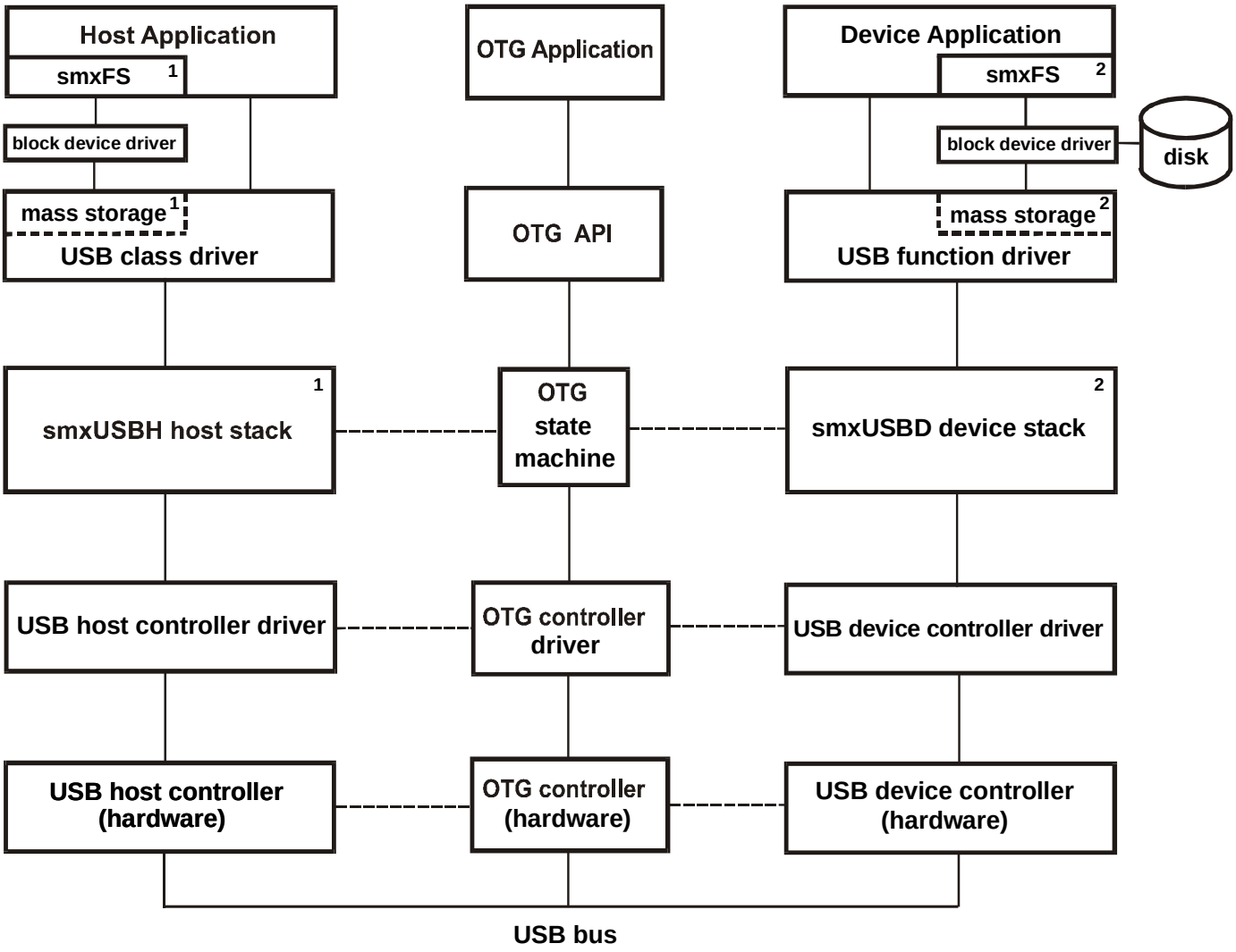
Host Controller	Raw Reading	Raw Writing
EHCI (NEC)	12684 KB/sec	8320 KB/sec
OHCI (NEC)	891 KB/sec	832 KB/sec
UHCI (VIA)	639 KB/sec	611 KB/sec
ISP116x (NXP)	352 KB/sec	334 KB/sec
ISP1362 (NXP)	621 KB/s	493 KB/s
ISP176x (NXP)	7425 KB/s	3214 KB/s
BF5xx (ADI)	10000 KB/s	8000 KB/s

Host Controller	Data Reading	Data Writing
OHCI (NEC)	124 KB/sec	124 KB/sec

Notes

1. The hardware environment for this testing is: Celeron 300MHz CPU; 32MB 100M SDRAM; PC motherboard; Host Controller connects to System by 33MHz PCI bus.
2. Flash Disk is Lexar JumpDrive USB 2.0 512MB
3. CPU speed, SDRAM speed and size, and External Memory Bus speed will affect the performance.
4. DMA is not used for NXP ISPxxxx controller.

smxUSB Product Illustration



1 Included in USB Thumb Drive Bundle

2 Included in USB Disk Emulator Bundle