smxUSBO™

USB OTG Add-On

smxUSBO adds On-The-Go support to smxUSBD and/or smxUSBH. It was specifically designed and developed for embedded systems. It is written in C, and can run on any hardware platform. While optimized for SMX®, smxUSBO can be ported to another RTOS or operate in a stand-alone environment.

An OTG device is a dual-role device that can function both as a limited-capability host and as a USB peripheral. The targeted peripherals can be any combination of other OTG devices and peripheral-only devices. smxUSBO requires smxUSBD and/or smxUSBH; it adds OTG capability to them.

An OTG cable has a mini-A plug on one end and a mini-B plug on the other. A-Device or B-Device is determined by which end is plugged into the device. If mini-A is plugged into a device then it is A-Device by default, otherwise, it is B-Device. smxUSBO is used only to set up the connection. After that, smxUSBD or smxUSBH do the communication with the other device.

Layers

- **OTG Core Layer** Provides the common OTG SRP and HNP state machine and APIs used by the application to switch between Host and Device stack for the dual-role device.
- **OTG Controller Driver (OCD) Layer** Provides a unique interface for different USB OTG controllers such as ISP1362 and ISP1761.
- **Porting Layer** Provides service functions related to the hardware, OS, and compiler.

Features

- Compatibility with ARM, ColdFire, PowerPC, x86, and other CPU's.
- Compliant with the On-The-Go Supplement to the USB 2.0 Specification 1.0a.
- Philips ISP1362 and 1761 OTG controller support. Others being developed.
- Written entirely in ANSI-C.
- Optimized for SMX® RTOS.
- Easily portable to other RTOSs.
- Also runs stand-alone.

Code Size

Also see the smxUSBD and smxUSBH product briefs for their code sizes.

Component	ARM IAR (KB)	CF CW (KB)
Core	5	5
Philips ISP1362 Driver	2	2
Philips ISP1761 Driver	2	2

Data Size

smxUSBO needs less than 128 bytes for global data. It does not need to allocate dynamic memory. Also see the smxUSBD and smxUSBH product briefs for their data sizes.

Performance

Please see the smxUSBD and smxUSBH product briefs. smxUSBO is only involved to set up the OTG connection.

OTG API

suo_ARequestHost()
suo_AGoIdle()
suo_BRequestHost()
suo_BReturnDevice()
suo_GetDeviceType()
suo_GetStackType()
suo_GetIdleType()

Porting

The hardware porting layer consists of two files, uohdw.h and uohdw.c. These files contain definitions, macros, and functions to port smxUSBO to a particular target. In addition, if the USB OTG controller is not among those already supported, a new driver will need to be written.

smxUSBO was developed for use with SMX, but it can be ported to any RTOS. The RTOS porting layer consists of two files, uoosport.h and uoosport.c. These files contain definitions, macros, and functions to port to a particular RTOS.

smxUSBO works best in a multitasking environment. However, it can also be ported to a non-multitasking stand-alone environment.