Cypress EZ-USB® FX3™ SDK

Release Notes

Version 1.3.1

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone (USA): 800.858.1810
Phone (Intl): 408.943.2600
http://www.cypress.com

Copyrights

Disclaimer

CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

License Agreement

Please read the license agreement during installation.

Contents

1 FX3 SDK 1.3.1

1.1 Introduction

The EZ-USB FX3 controller is a general purpose integrated USB 3.0 SuperSpeed controller with a built-in programmable interface (GPIF™ - II) and support for accessing a set of serial peripherals.

The EZ-USB FX3S (CYUSB3035) controller is an extension to the EZ-USB FX3 device which adds the capability to control up to two SD/MMC/SDIO peripherals to the FX3 feature set.

The EZ-USB CX3 (CYUSB3065) controller is an extension to the EZ-USB FX3 device which adds the ability to interface with and perform uncompressed video transfers from Image Sensors implementing the MIPI CSI-2 interface.

The FX3 SDK enables users to make use of the device features for each of these devices effectively. Full support for the CX3 device (CYUSB3065) is added to the SDK starting with this 1.3.1 release.

FX3, FX3S and CX3 devices have an embedded ARM9 micro-controller, and the SDK contains a set of flexible software modules that enable the effective use of all device features. The primary component of the SDK is a set of API that can be used by programmers to configure and control the data paths through the device as well as peripherals connected to the device.

The FX3 SDK consists of the following major components:-

1. FX3 Firmware stack with the API library and sample firmware examples.
2. Eclipse IDE and GNU Toolchain.
3. Documentation – API documentation, Programmer's Manual and Build instructions

1.2 Release Components

This 1.3.1 release of the FX3 SDK includes:

- FX3 Firmware drivers and APIs for device initialization, configuring USB, configuring DMA, configuring the GPIF™ - II interface, debug logging and configuring the serial interfaces.
- FX3S firmware driver and API for managing SD/eMMC/SDIO peripherals connected to the device.
- CX3 APIs for interfacing with Image Sensors using the MIPI-CSI2 interface.
- FX3 Boot Firmware drivers and APIs for device initialization, configuring USB and configuring the serial interfaces.
- Sample FX3 firmware examples in the form of Eclipse projects namely USB Bulk data loopback example, USB Isochronous data loopback examples, Serial Interface Examples (I2C, UART, SPI and GPIO), USB Video Class example, USB Bulk Source Sink Example, USB Isochronous Source Sink Example, USB Bulk Streams example, Slave FIFO example,

FX3 boot source (I2C EEPROM and SPI FLASH) programming example, Data cache usage example and USB setup command handling example.

- FX3S firmware examples that demonstrate USB mass storage class implementation, GPIF to storage access, File System integration and SDIO peripheral access.
- CX3 firmware examples that demonstrate USB Video class implementation using uncompressed 1080p and 720p streams at 30 and 60 frames per second respectively.
- A preliminary release of a MIPI CSI-2 interface configuration generation tool for CX3.
- FX3/FX3S Getting Started Guide, API documentation and Programmers Manual
- GPIF II Designer
- Eclipse IDE with GNU ARM C/C++ Development support plug-in and Zylin Embedded CDT plug-in
- GNU Toolchain
- Cypress Software License Agreement

1.3 Firmware features

- Support for USB - SuperSpeed, Hi-Speed and Full Speed device operation.
- Support for USB Hi-Speed and Full Speed Host/OTG operation
- Tested with the Intel, Renesas, Fresco, Via Labs, ASMedia and TI XHCI host controllers
- Support for configuring GPIF™ - II interface
- Support for pass-through data traffic between USB host and GPIF™ - II interface at USB 2.0 and 3.0 speeds.
- Support for controlling SD/eMMC/SDIO peripherals and performing data transfers.
- Support for UART configuration and data communication.
- Support for I2C (master mode only) configuration and data communication.
- Support for SPI (master mode only) configuration and data communication.
- Support for Debug logging
- Support for simple and complex mode GPIO access
- Sample firmware examples

1.4 Development Tool features

- Full featured Eclipse IDE
- GNU ARM C/C++ Development support plug-in
- Zylin Embedded CDT plug-in
- ARM GNU Toolchain comprising GNU compiler (gcc), GNU binary utilities (assembler, linker) and GNU debugger (gdb)
- Integrated debugging (using GDB)
- Source debugging
- Set/Clear breakpoints in the code

2 Changes from FX3 SDK 1.3

This 1.3.1 version of the FX3 SDK includes the following changes on top of the 1.3 version. Please refer to the FX3 API guide for more details of the specific API level changes.

Device Support

1. Full support for the CX3 device in the firmware library. The CX3 specific features are only available in the full firmware library, and not part of the boot library.

Build Changes

1. Changed the build options for the firmware library to obtain reduced memory footprint. These changes compensate for the 10 KB of additional code (defect fixes and new device support) that was added in SDK 1.3.

DMA Driver and APIs

1. Fixed a defect in the implementation of the CyU3PDmaMultiChannelDiscardBuffer() API for Many-to-One Manual DMA channels.
2. Reverted a change made in SDK 1.3 to release the lock on a DMA channel before calling the callback function. The lock on the channel is now held while the callback function is invoked. This will allow APIs that use the same channel to work faster when called from the callback routine.
3. Added a new API called CyU3PDmaMulticastSocketSelect() to select the active consumer sockets dynamically on a multicast DMA channel.

USB Driver and APIs

1. Fixed a USB start-up defect which caused the device to re-enumerate when loading a application on top of a SDK 1.2.3 or older version of boot-loader.
2. Fixed the CyU3PUsbStall API to compulsorily send an ERDY TP so that the host can identify that there is a STALL condition. Earlier, the ERDY was only being sent if the endpoint was in a flow controlled state.

Storage Driver and APIs

1. Freed up two additional DMA sockets (CY_U3P_SIB_SOCKET_4 and CY_U3P_SIB_SOCKET_5) for applications using the FX3S part.
2. Added a new field called lvGpioState to the CyU3PSibIntfParams_t structure. This field defines the polarity of the GPIO that controls the SD port voltage switch.
3. Fixed a defect in the calculation of erase unit size for SD cards of 64 GB and greater capacities.

Debug APIs

1. Fixed a defect in the CyU3PDebugSysMemInit() API implementation, which prevented the logs from being added to the memory buffer.

Firmware Examples

1. Updated the UsbBulkSourceSink application to restart sending data on the source (IN) endpoint after completion of a CLEAR_FEATURE(EP_HALT) command.
2. Updated all firmware example projects to remove the “Create Extended Listing” option in the default build settings. This option will have to be added manually by the user, where required.
3. Added a new example (FX3SRaid0) that demonstrates the use of FX3S APIs to implement a RAID-0 disk.

CX3 APIs and Examples

1. Updated APIs as part of the CX3 firmware library (cyu3mipicsi.a) that provides the MIPI-CSI interface support. FX3/FX3S users do not need to link with this new library and can use only the changes listed in other categories.
2. Updated examples that demonstrate various features of the CX3 API using Aptina AS0260 sensor and Omnivision OV5640 sensor. Sensor functionality is implemented through sensor-specific libraries (cy_as0260.a and cy_ov5640.a).
3. The CyU3PMipicsiCfg_t structure used for configuring the MIPI-CSI interface has changed. The order of parameters in the structure has changed and one additional parameter has been added.
4. A preliminary release, for a CX3 MIPI CSI-2 interface configuration generation tool, has been provided within the EZ-USB Suite Eclipse IDE. This tool simplifies creation of the CyU3PMipicsiCfg_t structure, based on image stream and sensor settings. Instructions for use of the tool are provided in the Getting Started with FX3 guide.
5. The fixed function GPIF-2 interface has been updated for improved stability.

3 Changes from FX3 SDK 1.2.3

The 1.3 release of the FX3 SDK has the following changes from the 1.2.3 version.

Device Support

1. Added support for the EZ-USB FX3S, SD3 and SD2 device families in the SDK libraries. The FX3S family adds the capability to communicate with SD/MMC/SDIO peripherals to the FX3 device. The SD3 and SD2 families are USB to peripheral bridge devices that support the SD/MMC/SDIO devices in addition to the serial peripheral (UART, SPI, I2C and I2S) interfaces.
2. Added BETA support for the EZ-USB CX3 device family in the SDK libraries. The CX3 family adds the capability to interface with Image Sensors using the MIPI-CSI2 interface.

Device APIs

1. Added new fields called s0Mode and s1Mode to the CyU3PIoMatrixConfig_t structure passed to the CyU3PDeviceConfigureIoMatrix API. FX3 users should set these fields to the value CY_U3P_SPORT_INACTIVE. FX3S users can select 4 bit wide or 8 bit wide storage port operation through these parameters.
2. Added a new lppMode definition called CY_U3P_IO_MATRIX_LPP_NONE. This mode should be chosen for FX3S designs which use the S1 storage port in 8-bit wide mode.
3. Fixed an error in the CyU3PSysEnterSuspendMode() API that could get the device stuck when used with USB bus activity as the wake-up source.
4. Changes to allow the watchdog to be left running when transferring control from the 2-stage boot firmware to the full firmware application. If the watchdog is not explicitly disabled prior to initiating the control switch, it will be left running and needs to be cleared when the new firmware application starts execution.

Storage API and Examples

1. Added a new firmware library (cyu3sport.a) that provides the storage drivers and APIs for the FX3 device. FX3 users do not need to link with this new library and can use only the changes listed in other categories.
2. Added examples (FX3SMassStorage, GpifToStorage, FX3SFileSystem, FX3SSdioUart) that demonstrate various features of the FX3S API.

USB Driver and APIs

1. Fixed a driver defect that could prevent the device from staying in the USB 3.0 compliance state for doing electrical compliance testing.
2. Fixed a defect in the CyU3PUsbDoRemoteWakeup() API that would cause the device to drop off the bus after signaling remote wakeup.

3. Added support for detecting USB connections based on the VBatt supply (as a replacement to the VBus supply). Please refer to the CyU3PUsbControlVbusDetect() API for details.

Beta CX3 APIs and Examples

6. Added a new firmware library (cyu3mipicsi.a) that provides the MIPI-CSI interface support. FX3/FX3S users do not need to link with this new library and can use only the changes listed in other categories.
7. Added examples that demonstrate various features of the CX3 API.

PIB Driver and APIs

1. Added support for configuring the PIB (Processor Interface Block) port on FX3/FX3S as an MMC slave in addition to the default GPIF mode. Only one of the MMC slave or GPIF modes can be used at a time. Please refer to the CyU3PPibSelectMmcSlaveMode API for configuring the PIB as an MMC slave.
2. Added fixed function GPIF interfaces (8-bit, 16-bit and 24-bit) for the CX3 parts. The CX3 parts do not contain a user configurable GPIF interface.

2-Stage Boot APIs

1. Added the CyFx3BootGpioOverride and CyFx3BootGpioRestore APIs to allow run-time functionality overrides on the FX3/FX3S device IOs.
2. Changes to allow the Watchdog to run when switching control from the boot firmware to the full firmware or vice versa. This allows the device to recover in cases where the new firmware fails to start-up correctly (due to configuration/programming errors).
3. Fixed SPI configuration errors that restricted the interface clock to half of the user specified frequency.
4. Reduced the polling delays in the I2C/SPI WaitForXfer APIs to allow better transfer speeds.
5. Moved the run-time stacks used by the firmware to the DTCM region and allocated a larger stack (4 KB) for the default execution mode (System mode).
6. Changed the USB VID/PID used by the USB boot example to the same as that used by the main FX3 examples.

Build Scripts and Utilities

1. Provided new linker scripts (cyfx3_256k.ld and fx3_256k.ld) which can be used when building applications for FX3 parts having only 256 KB of RAM.
2. Changed the default i2cConf value used by the elf2img converter program to 0x1C. This value corresponds to the use of I2C EEPROMs with linear addressing (slave address incremented after each 64 KB) with a 400 KHz. This value works with the I2C EEPROM device that is in place on the latest FX3/FX3S DVK boards.

Example Applications

1. Added a new example application (cyfxusbuart) that demonstrates the implementation of a CDC-ACM compliant USB to UART bridge device.
2. Updated the USBBulkLoopAuto example to demonstrate the use of the CyU3PSysEnterSuspendMode() API.

Documentation

1. Updated the format of the API reference guide (FX3APIGuide.pdf). This document is now being generated using a different tool, and will be organized differently.

4 Changes from Release 1.2.2

The 1.2.3 release of the FX3 SDK has the following additional changes on top of the 1.2.2 release version. Please refer to the FX3APIGuide.pdf for more details.

Eclipse Integration

1. The FX3 API help has been integrated into the Eclipse development environment. All of the API documentation in the header files has been re-formatted to enable this change.

General Firmware Framework

1. Changed the return type of the CyU3PDmaBufferFree function from void to int, to enable the function to flag errors. This change will cause existing application to break during compilation. The CyU3PDmaBufferFree implementation from the cyfxtx.c files in the new SDK should be used to fix this.
2. Fixed a bug that could cause the FX3 device to get stuck when placed in Suspend mode through the CyU3PSysEnterSuspendMode API.

USB Driver and API

1. Updated the USB driver to re-attempt a USB 3.0 connection on the very first USB 2.0 reset, instead of waiting for a USB 2.0 reset following a control request. This enables the device to consistently enumerate on USB 3.0 when used with a host stack that does not follow the normal USB initialization sequence.

Programmer's Manual

1. Added description on how the Olimex ARM USB OCD debug probe can be used to debug FX3 firmware. See section 12.2.1 of the Programmer's Manual for details.

5 Changes from Release 1.2.1

The 1.2.2 release of the FX3 SDK has the following changes on top of the 1.2.1 release version. Please refer to the FX3APIGuide.pdf for more details.

SDK Installation Path

2. The SDK installer has been updated to use the "**C:\Program Files**" directory as the default installation path, instead of "**C:\Cypress**". This change is only applicable for a fresh SDK installation and not for updates from a previous version of the SDK.

Boot Firmware Library and API

1. Updated the USB driver to handle aborted USB control requests cleanly.
2. Added a new parameter to the ***CyFx3BootUsbSetClrFeature()*** API to indicate whether the USB device has been configured or not.
3. Fixed a couple of control request handling errors in the Fx3BootAppGcc example application.

General Firmware Framework

3. Added a new API called CyU3PSysEnterStandbyMode() to place the FX3 device in low power standby mode.
4. Included more error checks around all ThreadX OS services that are used by the FX3 library and by user applications.

USB Driver and API

2. Updated the USB block clocking scheme to fix intermittent failures in USB Hi-Speed enumeration, particularly when FX3 is clocked using a 26 MHz or 52 MHz clock input.
3. Fixed a USB driver issue which could cause the device to re-enumerate occasionally due to USB 3.0 link errors that accumulate over a long period of time.
4. Enabled new USB events (***CY_U3P_USB_EVENT_VBUS_VALID*** and ***CY_U3P_USB_EVENT_VBUS_REMOVED***) to provide notification of VBus signal being turned on or off.
5. Added a new API called ***CyU3PUsbEPSetBurstMode()*** to allow the FX3 device to combine data from multiple DMA buffers into a single transfer burst on the USB IN endpoint. Using this API can improve the data transfer performance on burst enabled endpoints.
6. Added new APIs (***CyU3PUsbGetEpSeqNum*** and ***CyU3PUsbSetEpSeqNum***) to save and restore the sequence number on USB 3.0 endpoints.
7. Added a new API called ***CyU3PUsbControlVBusDetect*** to select whether VBus detection should be done by the FX3's USB driver or not. If VBus detection in the driver is turned off, it is expected that the CyU3PConnectState() API will be called at appropriate times based on an external means of VBus detection.

8. Added a new API called ***CyU3PUsbControlUsb2Support*** to disable the USB 2.0 device PHY on the FX3 device. This API is only useful in designs where the FX3 device is being used only in SuperSpeed mode, and the USB 2.0 operation is handled by an external controller. In such a case, the ***CY_U3P_USB_EVENT_USB3_LNKFAIL*** event will provide notification that the 3.0 connection has failed.

GPIF Driver and API

1. Added support for early notification of GPIF state machine interrupts in the interrupt context itself. Added a new API called ***CyU3PGpifRegisterSMIntrCallback*** to register the callback function to be called for this notification.

Example Build Settings

1. The build settings for the API libraries have been updated to place each function in a different code section. The linker parameters in all the Eclipse projects have been updated to add the “-Wl,--gc-sections” parameter to take advantage of this change. The use of this option can help reduce the memory footprint of FX3 applications.

6 Changes from Release 1.2

The 1.2.1 release of the FX3 SDK has the following additional changes on top of the 1.2 release version. Please refer to the FX3APIGuide.pdf for more details.

General Firmware Framework

1. Re-organized the firmware modules so as to reduce the memory footprint of common applications (not using USB OTG or multicast DMA channels) by another 20 KB.

USB Drivers and APIs

2. Added a new API called CyU3PUsbEnableEPPrefetch() to enable the DMA prefetch settings that are required to prevent data corruption due to endpoint underrun errors. This API needs to be called after CyU3PUsbStart().
3. Added a new API called CyU3PUsbResetEndpointMemories() to reset the endpoint memory block on the FX3 as a whole. This API should only be used to recover from error conditions that cause data transfers through FX3 to freeze.
4. Fixed race conditions in the drivers that could cause EP0 transfers (CyU3PUsbSendEP0Data or CyU3PUsbGetEP0Data) to be failed with the CY_U3P_ERROR_ABORTED code, even if the actual transfer on the bus has passed.
5. Fixed a driver defect that prevented the generation of the CY_U3P_USB_EVENT_CONNECT event when the firmware is configured to enable USB 3.0 connections, and the device is connected to a USB 2.0 host.
6. Updated the USB driver to improve CPU responsiveness when there is a high frequency of U1 power mode entry requests from the host. The earlier implementation was leading to some failures when testing against the host drivers on Windows 8.
7. Added a new USB event (CY_U3P_USB_EVENT_LNK_RECOVERY) for notification of cases where the FX3 device is entering USB 3.0 link recovery.
8. Added new APIs called CyU3PUsbSetEpPktMode(), CyU3PUsbGetBooterVersion() and CyU3PUsbGetErrorCounts(). Please see the API guide for details on these functions.

DMA Driver and APIs

1. Added a new API called to CyU3PDmaEnableMulticast(). It is required to call this API before creating any multicast DMA channels.

GPIO Driver and APIs

1. Fixed a GPIO driver defect that caused the ARM CPU to get stuck when a GPIO interrupt is raised.
2. Modified the GPIO APIs to allow the state of FX3 GPIOs to be retained when switching from the boot firmware to the full firmware and back.

3. Updated the driver code to keep all FX3 GPIOs unaffected when the device is placed in suspend mode. Earlier, all GPIOs would have been tri-stated when the device was placed in suspend mode.

2-Stage Boot Firmware Library

1. Added a new API (CyFx3BootRetainGpioState) to allow the state of FX3 GPIOs to be retained when switching control from the boot firmware to the full firmware.
2. Added new APIs (CyFx3BootUsbLPMDisable and CyFx3BootUsbLPMEnable) to dynamically enable/disable U1/U2 power mode acceptance by the boot firmware.
3. Updated the USB driver to handle U1/U2 mode transitions in the middle of control endpoint transfers more efficiently.
4. Updated the build settings for the boot application example to reduce memory footprint of the binary.

Firmware Examples

1. Updated the FX3 Boot App example and the USBBulkSourceSink examples to demonstrate switching between the boot firmware and the full firmware.
2. Added a new example called USBIsoSource that demonstrates a high performance ISO IN pipe in parallel with USB control transfers.
3. Enabled the data cache in all firmware examples.

Firmware Binary Converter

1. Updated the elf2img converter to support output images with section sizes broken down to 2KB and 1KB. This can be used as a work-around for firmware download failures on the Etron USB 3.0 host controller. The Etron host controller (driver) has a known error in the handling of Control OUT transfers with data length of 4 KB, and this leads to firmware download errors. These errors can be worked around by using the elf2img converter with the argument “-i2cconf 0x02” or “-i2cconf 0x00”. This change forces the Control Center to use smaller bursts (2KB or 1KB) during the firmware download process.

SDK Documentation

1. Added a new section on FX3 Programming Guidelines to the “Getting Started with FX3 SDK” document.

7 Changes from Release 1.1.1

The 1.2 release of the FX3 SDK has the following additional changes on top of the 1.1.1 patch version. Please refer to the FX3APIGuide.pdf for more details.

Serial Peripheral Drivers

1. The serial peripheral drivers have been made open source in response to several requests to allow customization of these drivers. The output from these drivers will be part of a different library which will be linked into the firmware application. Full documentation for all serial peripheral interface registers is also provided as part of the Programmer's Manual.

   Please see the <INSTALL_ROOT>/firmware/lpp_source folder for the source code for these drivers and APIs. An eclipse project that builds these files into a library (cyu3lpp.a) is also provided in the package.

Device APIs

9. Added support in the firmware to jump back to the boot firmware image, without disconnecting the USB connection. This feature allows users to modify the application code without breaking the USB connection.
10. Added support for all devices (CYUSB3011, CYUSB3012, CYUSB3013 and CYUSB3014) in the FX3 device family. The firmware drivers detect the device part number and return appropriate error codes when unsupported functionality is exercised.

GPIF Driver and APIs

1. Added separate types for P-port interface errors and GPIF specific errors reported through the CYU3P_PIB_INTR_ERROR callback. Macros to retrieve the relevant error information from the callback argument have been added.

USB Driver and APIs

1. Reorganized the USB driver code so that only applications that make use of the USB host and/or OTG features will include the corresponding driver functions. This change reduces the memory footprint of applications that do not use USB host or OTG features.
2. Updated the USB 3.0 driver to configure the USB block to make endpoint under-run errors on the IN path unlikely. Also added a new USB event type (CY_U3P_USB_EVENT_EP_UNDERRUN) which is sent to the application if an under-run condition is detected.
3. Added the capability to log USB related events into a user provided buffer. Please see the CyU3PUsbInitEventLog() and CyU3PUsbGetEventLogIndex() APIs for details.
4. Fixed a FX3 device issue which caused the device to re-enumerate when doing high throughput data transfers with some USB 3.0 hosts. This error was seen most often when testing with the Intel XHCI host controllers.

Build Settings

5. Updated the FX3 example makefiles to function on non-Windows platforms like Linux and Mac. This change allows the examples to be built in batch mode without invoking the Eclipse IDE.
6. Modified the cyfxtx.c file in the SDK to reserve the last 32 KB of RAM for the boot firmware application. This change reduces the amount of memory available for DMA buffers. If the application does not use the 2-stage boot feature, the upper limit of the buffer heap can be restored to the address 0x80000.

2-Stage Boot Firmware Library

7. Added code to support switch back to the boot application from the main firmware application.
8. Added watchdog timer support. Please see the CyFx3BootWatchdogConfigure() API for details.
9. Added a parameter to the CyFx3BootDeviceInit API to select between a system clock setting of 384 MHz and 403.2 MHz.
10. Updated the USB boot application code to handle requests for non-existent descriptors cleanly.

Firmware Examples

4. Added new Slave FIFO examples in both Synchronous and Asynchronous modes, that makes use of all the USB endpoints supported by the FX3 device. The 5-bit addressed mode of the Slave FIFO protocol is used so that a separate pipe and DMA channel on the GPIF-II side can be associated with each USB endpoint.
5. Added a new example that implements a mass storage device using a portion of the FX3 device RAM.
6. Added a new example that implements a USB Audio Class (UAC) microphone device and streams audio data that it reads from a SPI flash device connected to FX3. This implementation only works at USB 2.0 speeds, as the UAC class does not require the bandwidth provided by USB 3.0.
7. Updated the UsbBulkSourceSink example to make use of the USB event log support.

8 Changes from Release 1.1

In addition to the above changes, the SDK 1.1.1 patch version had the following changes on top of the SDK 1.1 release.

General Changes

1. A new member called ***setSysClk400*** has been added to the ***CyU3PSysClockConfig_t*** structure passed to the ***CyU3PDeviceInit()*** function. This parameter is used to select whether the FX3 system clock should be running at 384 MHz or 403.2 MHz. The system clock frequency was compulsorily set to 403.2 MHz in previous SDK releases.

USB Driver and API changes

1. Added a new USB event called ***CY_U3P_USB_EVENT_EP0_STAT_CPLT*** to provide notification of the completion of the status handshake for USB control requests that are handled by the user's callback function.
2. Added a new USB API called ***CyU3PUsbDoRemoteWakeup()*** to cause FX3 to perform USB 2.0 remote wakeup signaling.
3. Added a new USB API called ***CyU3PUsbForceFullSpeed()*** to force the FX3 device to function at USB full speed instead of high speed.
4. Added a new USB API called ***CyU3PUsbSendDevNotification()*** to allow sending of Device Notification Transaction Packets (TP) such as Function Wake and Latency Tolerance Message.
5. Added new USB APIs called ***CyU3PUsbLPMDisable()*** and ***CyU3PUsbLPMEnable()*** to disable/enable U1/U2 power mode entry at runtime. Please refer to the API description in the API guide for usage restrictions.
6. Fixed a defect in the USB driver that could cause control transfers without a data phase to fail if the FX3 device was placed in U1/U2 before the status handshake is completed.
7. Made USB driver changes to fix USB 3.0 hotplug errors and TD 9.23 (Reset Device Test) failures in the USBCV suite. These errors had been seen only on some FX3 boards, and were associated with the device failing to go through USB 3.0 link training.
8. Made USB driver changes to improve firmware responsiveness while working under aggressive USB link power management.
9. Updated the USB driver to ensure that the sequence numbers (or data toggles) on all endpoints are cleared when handling a SET_CONFIGURATION request.

I2C Driver Changes

1. Fixed a defect in the I2C driver which could cause the ***CyU3PI2cTransmitBytes()*** API to incorrectly return ***CY_U3P_ERROR_LOST_ARBITRATION*** even in some success cases.

GPIF Driver and API changes

1. Fixed a defect in the ***CyU3PGpifSMSwitch()*** API which would cause state machine switches to fail if the GPIF designs use mirror states.

Debug API changes

1. Fixed a defect in the CyU3PDebugPrint() API when trying to print messages whose total length exceeds 128 characters. The new implementation is limited to a total length of 256 characters, and will return an error if longer messages are passed in as parameters.

Example Application Changes

1. Updated all example applications to use 32 byte aligned buffers for descriptors and other data that is transferred via DMA to the USB host. This is required for the applications to work when the Data cache is enabled.
2. Updated the descriptors and callback function implementations in all USB device mode examples to pass USB chapter 9 compliance tests.
3. Updated the cyfxbulksrcsink example application with additional code demonstrating new features such as link power management control.
4. Updated the GPIF descriptors used in the Slave FIFO examples to the latest version generated by the GPIF II Designer tool.

Boot Firmware Changes

1. Updated boot firmware library and USB boot example to pass USB Chapter 9 compliance tests and Physical/Link Layer tests.

9 Changes from Release 1.0.1

This release of the FX3 SDK has the following changes from the 1.0.1 version. Please refer to the FX3APIGuide.pdf for details.

General Changes

1. Macro definition for version information has also been added.
2. FX3 release configuration has reduced error checks for better performance.
3. CyU3PSysEnterSuspendMode API signature has been changed to accommodate the polarity of the wakeup source.
4. CyU3PDebugSysMemInit and CyU3PDebugSysMemDeInit APIs have been added to do debug logging into system memory buffer.
5. CyU3PDebugEnable and CyU3PDebugDisable API signature has been modified to take the thread/module information as a bit mask. CyU3PDebugIsEnabled API has been removed as this is redundant.

USB Driver and APIs

1. USB host (cyu3usbhost.h) and OTG (cyu3usbotg.h) APIs have been added.
2. CY_U3P_USB_EVENT_SOF_ITP event has been added to track SOF/ITP USB events. CyU3PUsbEnableITPEvent API has been added to control this.
3. OTG descriptor, CY_U3P_USB_SET_OTG_DESCR, has been added to the list of supported standard USB descriptors.

I2C Driver and APIs

1. Error return codes for I2C APIs have been modified to return detailed error information.
2. CyU3PI2cGetErrorCode API has been added to retrieve the error information when an I2C API returns CY_U3P_ERROR_FAILURE.

GPIO

1. GPIF CTL4 is now made available as GPIO and need not be overridden.
2. GPIO register definition (gpio_regs.h) has been included with the release

2-Stage Boot Firmware Library

1. The timeout value for I2C and SPI DMA transfer functions have been reduced to multiples of 10usec (from 100usec).

Firmware Examples

1. cyfxbulklpautoenum and cyfxbulksrcsink examples have been modified to pass USB compliance. Other examples do not have these changes. The changes required are:
    a. USB version field in CyFxUSB20DeviceDscr should be 2.10
    b. LPM support needs to be enabled in supported device level features field in CyFxUSBBOSDscr.
    c. Handle SET_FEATURE(FUNCTION_SUSPEND) and CLEAR_FEATURE(FUNCTION_SUSPEND) USB setup requests. These should be allowed to pass if the device is in configured state and failed otherwise.
    d. Register a callback to handle LPM requests and return CyTrue for all requests.
2. Performance optimization changes for cyfxbulksrcsink and cyfxisosrcsink examples have been limited to few macro changes.
3. All ISO examples have been modified to use EP3 IN and EP3 OUT so that mult field can be used.
4. cyfxbulklpmanual example has been modified to invert all bits on the received data.
5. cyfxbulklpauto_cpp, Cyfxbulklplowlevel, cyfxbulklpman_addition, cyfxbulklpman_removal, cyfxbulklpotg, cyfxusbdebug, cyfxusbhost, cyfxusbotg, cyfxusbspigpiomode, cyfxusbi2sdmamode and cyfxuvcinmem_bulk examples have been added.

10 Changes from Release 1.0

This patch to the FX3 SDK Release 1.0 has the following changes from the 1.0 version. Please refer to the FX3APIGuide.pdf for details.

Linux Support

1. Linux is now supported as a development and debug platform for FX3 firmware. Please refer to the FX3_SDK_Linux_Support.pdf document in the doc folder for installation and usage instructions on a Linux platform.

USB Driver

2. Included USB driver changes to fix rare USB 3.0 failures when operating behind a Via SuperSpeed hub. No API interface changes are involved in this.
3. Fixed a defect in the CyU3PUsbStart() API to reset the signature and length field used for the USB No Re-enumeration feature of the 2-Stage boot firmware

GPIF Driver and APIs

5. Fixed a defect in the CyU3PGpifDisable() API when the forceReload parameter is set to True.
6. All optional PIB interrupts are kept disabled by default and are only turned on when the CyU3PPibRegisterCallback() function is used to enable callbacks for these events.
7. Added the CyU3PPibSelectIntSources() API to select the sources that can drive the output INT pin from the FX3 device to the external processor/device.

I2C Driver and APIs

1. Fixed a problem in the CyU3PI2cTransmitBytes() API where no error is reported for a single byte transfer even if the target device is missing.

UART Driver and APIs

1. Fixed a defect in the UART driver that led to UART callbacks never being invoked.

ELF to Boot Image Converter

1. Fixed a defect that caused boot failure when the boot image was stored across multiple I2C EEPROM devices.

2-Stage Boot Firmware Library

1. Fixed a defect in the boot support firmware library which caused the no re-enumeration after firmware load feature to intermittently fail.
2. Reduced internal processing delays in the SPI driver to get better SPI transfer performance.
3. Added Full Speed support in the usb_boot.c
4. Fixed a defect in usb_boot.c which caused the descriptor length to be calculated incorrectly.

Firmware Examples

1. Fixed un-initialized variable errors in the cyfxflashprog, cyfxusbi2cdmamode and cyfxusbspidmamode examples that would cause write transfers to fail on some builds.
2. Updated the cyfxusbi2cregmode example to enable reading/writing arbitrary counts of data.
3. Fixed a condition check in the cyfxbulklpautoenum example that caused errors in retrieving string descriptors.
4. Fixed errors in the build settings for the cyfxbulklpmanual_rvds example, and added a functional release configuration.

11 Known Issues & Solutions

This 1.3.1 release of the FX3S SDK has the following known issues.

1. The FX3 firmware libraries and applications can be compiled using most standard ARM tool-chains (gcc, Keil MDK or RVDS, IAR etc.) The libraries have been tested using the Sourcery G++ Lite tool-chain that is shipped along with the SDK. While the ThreadX OS is embedded in the FX3 SDK, Cypress does not provide any OS aware debugging capabilities.
2. The settings for DMA channels associated with USB sockets are likely to be invalidated as part of the USB device mode connection startup. Therefore, it is required that all USB related DMA channels be reset (CyU3PDmaChannelReset) and the re-enabled after each enumeration cycle. This can be done on a USB reset event, or on a Set Configuration event; as none of the USB DMA channels are expected to be used before SET_CONFIGURATION is completed.
3. The USB bulk stream mappings for USB 3.0 endpoints are also likely to be invalidated as part of the USB connection startup. Therefore, it is required that the bulk stream mappings are setup on receiving a SET_CONFIGURATION event.
4. When there is a failure in an I2C transaction (a DMA failure or an API return of CY_U3P_ERROR_BLOCK_FAILURE), the I2C block needs to be reset. This can be done by invoking CyU3PI2cDeInit(), followed by a CyU3PI2cInit() and a CyU3PI2cSetConfig().
5. After a CyU3PI2cReceiveBytes()/CyU3PI2cTransmitBytes() API call, sufficient time must be given for the slave to complete the operation before the next transaction (read/write) is issued. This can be achieved by using the CyU3PI2cWaitForAck() or by providing a sufficient delay.
6. When the warm reset functionality of the CyU3PDeviceReset() API is used, any global variables used by the application will not be properly re-initialized. This is because the startup code that initializes these variables would have been lost and cannot be executed again without loading the application again. If warm reset is needed, the application code must ensure that all necessary data is saved and restored or re-initialized as required.
7. The CyU3PSpiWaitForBlockXfer() function will wait forever if there are no pending transfers at the time of calling this API.
8. Any UART/I2C/SPI read transfers in DMA mode that do not fill up the entire DMA buffer will not trigger a DMA callback or a transfer complete event. The application needs to check for transfer completion based on the UART/I2C/SPI events and then invoke the CyU3PDmaChannelSetWrapUp() API on the DMA channel.
9. ARM GCC tool chain, if any, installed on the host system needs to be completely removed before installing the ARM GCC toolchain supplied with this package. If the older installation is not removed, it could lead to firmware build issues.

10. Some data transfer errors may be seen if a Zero Length Packet is followed very quickly (within one microframe or 125 us) by another data packet on a burst enabled USB IN endpoint operating at super speed. The solution is to ensure that some time is allowed to elapse between a ZLP and the next data packet on burst enabled USB IN endpoints. If this cannot be ensured at the data source, the CyU3PDmaChannelSetSuspend() API can be used to suspend the corresponding USB DMA socket on seeing the EOP condition. The channel operation can then be resumed as soon as the suspend callback is received.
11. The RTOS porting layer code that is part of the cyfxtx.c file in the firmware examples has a tight correlation with the memory map information specified in the linker script file (fx3.ld). When moving to a new version of the FX3 SDK using an example that was previously created, it is required to copy the new version of the cyfxtx.c file (from the <FX3_INSTALL_ROOT>/firmware/common folder) into the project.
12. The CyU3PGpifSMSwitch API may not be able to trigger a desired state machine switch if the state machine in use makes use of mirror states. This is because the state machine may actually be stuck in a mirror of the user specified start state instead of in the start state itself. In this release, the API has been updated to trigger an immediate switch if the state machine is in the specified start state or one of its mirrors. This will not be sufficient if the state machine reaches a mirror of the start state only after the Switch API has been called.

    If the CyU3PGpifSMSwitch API needs to be used in systems which make use of state machines with mirror states, it is recommended that the application specify a timeout value in the CyU3PGpifSMSwitch() API call and retry the switch operation periodically until it succeeds.
13. USB Compliance Requirements: Both the full-featured firmware library and the boot firmware library require some support from the firmware application implementation to pass all relevant USB compliance tests. Please refer to the example application source code to understand these requirements.
14. The gcc linker script provided with the FX3 SDK (firmware/common/fx3.ld) does not initialize a runtime heap that is required for standard C library functions such as sprintf. If these functions are to be used, you need to provide an implementation for standard system calls and update the fx3.ld file to create a heap. Please refer to the cyfxbulklpauto_cpp example for a sample implementation of the required system calls, and the fx3cpp.ld file for a script sample with heap initialization.
15. When the FX3 device is functioning as a high speed USB device with high bandwidth isochronous endpoints, the PID sequence of the ISO data packets is governed solely by the isomult setting. The length of the data packet is not considered while generating the PID sequence during each microframe. For example, even if a short packet is being sent on an endpoint with MULT set to 2; the PID used will be DATA2.

    This problem can be worked around by reconfiguring the endpoint with a lower isomult setting prior to sending short packets, and then switching back to the original value.
16. The 5-bit Slave FIFO descriptors provided in the FX3 SDK and the GPIF II Designer cause the FX3 device to flag a number of GPIF errors during data transfers. These errors are false errors and can safely be ignored. Application can avoid registering for the CYU3P_PIB_INTR_ERROR callback if these interrupts and callbacks are found to be slowing down the system.
17. If the clkCfg->setSysClk400 parameter to the CyU3PDeviceInit API is set to true, the operating clock frequency of the FX3 device will be changed at runtime. This can cause a JTAG debug session using the JLINK debugger to break. This problem can be prevented by setting up the FX3 device clocks to the desired value during JTAG initialization itself. If

the clocks have already been setup to the desired value, the FX3 API does not make any further changes. Please add the following code fragment to the set of Initialize commands used in Eclipse based debug configuration.

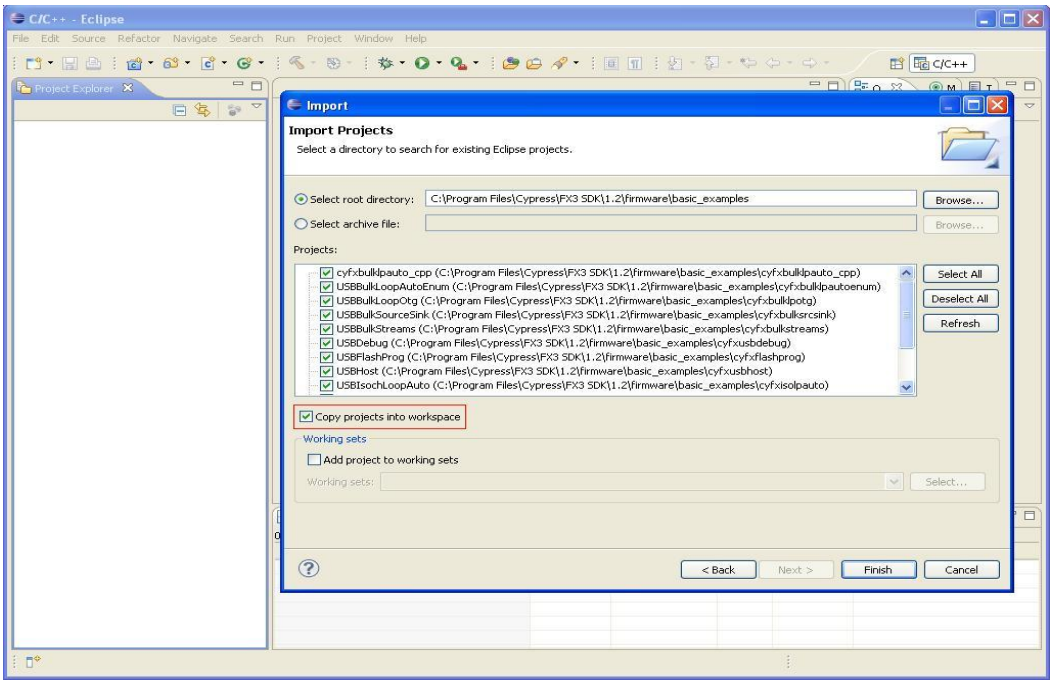
```
# If required, set the FX3 system clock to faster than 400 MHz.
# This is done here to prevent debug session errors due to the
# clock being changed at runtime.
# Update with the correct value from list below.
# Clock input is 19.2 MHz: Value = 0x00080015
# Clock input is 38.4 MHz: Value = 0x00080115
monitor memU32 0xE0052000 = 0x00080015
# Add a delay to let the clock stabilize.
monitor sleep 1000
```

18. USB data transfers through the FX3 device may get stuck if the DMA channels corresponding to the endpoints are unexpectedly reset. This condition can be detected by checking for a CY_U3P_USB_EVENT_EP_UNDERRUN event. The under-run error can be avoided by ensuring that data transfer through the endpoint is disabled before the DMA channel is reset. This is done by using the CyU3PUsbSetEpNak() to place the endpoint in NAK state for about 100us, before resetting the DMA channel.

    ```
    e.g.: CyU3PUsbSetEpNak (CY_FX_EP_CONSUMER, CyTrue);
          CyU3PBusyWait (100);
          CyU3PDmaChannelReset (&glEpSourceChannel);
          CyU3PUsbSetEpNak (CY_FX_EP_CONSUMER, CyFalse);
    ```

19. Compilation of FX3 example projects may fail in some cases if the user account does not have Administrator rights on the PC. This happens because some versions of Windows do not allow non-Administrator users to update files under the “Program Files” folder where the SDK is installed.

    This issue can be avoided by copying the project contents into the Eclipse workspace while importing the project. The following Eclipse UI screenshot shows the checkbox that needs to be selected to perform this copy operation.



20. The return type of the CyU3PDmaBufferFree function has been modified to allow this function to indicate errors during the free operation. The implementation of this function needs to be updated as shown in the cyfxtx.c files in the new SDK. If not, existing applications will return compile errors such as:

    ```
    cyfxtx.c:474:1: error: conflicting types for 'CyU3PDmaBufferFree'
    ```

21. As the runtime stacks in the boot firmware library have been moved to the DTCM region, stack variables can no longer be used for DMA transfers. Any buffers used for DMA transfers have to be made global buffers placed into the DATA region or explicitly placed at a valid SYSMEM address.

22. The FX3 (FX3S and CX3 as well) device is only capable to responding to the USB 2.1 LPM-L1 request with an ACK. The device is not capable of rejecting L1 entry by returning a NYET handshake. Using the CyU3PUsbLPMDisable() API will cause the device to reject LPM-L1 requests by ignoring them. As this behavior is not compliant with the USB specification, the CyU3PUsbLPMDisable function should not be used when working on a Hi-Speed link.

23. The cycx3_rgb24_as0260 example fails on Windows 8 with the Renesas µPD720200 host controller when using the 1920x1080 resolution. The issue is due to the host not reading data fast enough to read a complete frame (1920x1080x3 Bytes = 6220800 Bytes per frame) within the required time period. This issue is not seen on other USB 3.0 hosts or with the same host on Windows 7.