chipKIT™ USB Host Classes

Introduction

The chipKIT™ USB host classes are fundamentally direct mappings of the underlying Microchip USB Host Interface Routines. Buy understanding the Microchip USB Stack, the usage of the chipKIT™ USB host classes will become clear. Only the chipKIT™ "ChipKITUSBHost" class implements a few additional methods to initialize the chipKIT hardware and to provide a default event handler. Above and beyond that, all of the other methods are direct thunks to the underlying Microchip USB Host Interface Routines. The chipKIT™ USB host class methods will have the same base API names and parameter arguments as the underlying Microchip USB Host Interface Routines.

For more information on how to use the Microchip USB Host APIs, refer to http://ww1.microchip.com/downloads/en/DeviceDoc/MCHPFSUSB_FW_UG_51679a.pdf and "*Library Interface (API) > Embedded Host*" section in the Milcrochip's "*MCHPFSUSB Library Help.chm*" helpfile.

The chipKIT™ USB host architecture is slightly different than the chipKIT™ ChipKITUSBDevice class. There is only 1 chipKIT™ ChipKITUSBDevice class, however there are multiple chipKIT™ USB host classes. The host class architecture is designed with a common host ChipKITUSBHost class and chipKITUSBHost library to perform all of the common host USB functions with additional classes supporting each "*USB device class*"[1], such as HID or MSD. The common chipKITUSBHost library must be included in every USB host sketch along with at least 1 "*USB device class*" host class library to implemented the desired USB device (i.e. HID, CDC, MSD….).

In general, it is up to the implementation of the specific "*USB device class*" to implement the host class library for that "*USB device class*". However, Microchip has provided and documented support for HID, CDC, Charger, Generic, HID, Mass Storage (MSD), and Printer host "*USB device classes*". ChipKIT™ has provided class library implementations for HID and MSD host "*USB device classes*". If chipKIT™ support for one of the other Microchip supported "*USB device classes*" is needed, it is a relatively mechanical process to copy the chipKITUSBHIDHost library and modify it to call the underlying Microchip APIs for the desired "*USB device class*".

In addition to the 2 chipKIT™ supported "*USB device classes*", chipKIT™ also provides a chipKITMDDFS library to support Microchip's MDD File System. The MDD File System is not a "*USB device class*", instead this supports the FAT file system which can be used with the MSD (Mass Storage Device) "*USB device class*" to manipulate files and directories on a Mass Storage device such as a USB flash drive / thumb drive. This is documented in Microchip's AN1045 document 01045b.pdf, "*Implementing File I/O Functions Using Microchip's Memory Disk Drive File System Library*". While the Microchip's MDD File System can be used over a

[1] Sorry for the confusion with the overloading of the term "*device class*". When the term "*USB device class*" is used, this refers to a specific USB kind of device like HID, or MSD, not the chipKIT™ C++ ChipKITUSBDevice class; which is referred to as the "*ChipKITUSBDevice class*". The confusion comes in that HID or MSD is a USB device classification, and historically has been called a "*USB device class*" by the USB specification. This document is mostly about USB host implementations, so for the most part when we talk about a "*device class*" we are talking about a USB device classification such as HID or MSD.

variety of transports other than USB (such as SPI), the chipKIT™ ChipKITMDDFS class is specifically tailored to the USB transport and the MSD "*USB device class*". As a result, if you wish to use the ChipKITMDDFS class, you must first import both of the chipKITUSBHost, and chipKITUSBMSDHost libraries. For convenience the mapping of the ChipKITMDDFS class methods to the Microchip's MDD File System APIs is provided in this document.

ChipKITUSBHost Class

This class implements the common USB host functions. This class is implemented in the chipKITUSBHost library and must be imported into your sketch before any other USB host "*USB device class*" library. The class is pre-instantiated as the USBHost instance and you should not instantiate another instance. This is the only chipKIT™ USB class that implements any methods beyond that supported by the underlying Microchip USB Host APIs. Two new methods are implemented, USBHost.Begin(), and the USBHost.DefaultEventHandler() methods. USBHost.Begin initiates the chipKIT™ USB hardware and calls the underlying Microchip USBHostInit() API. You should not call both USBHost.Begin() and USBHost.Init() in the same sketch as this would cause the USBHostInit() API to be called twice.

The USBHost.DefaultEventHandler() method provides the sketch access to a default event handler that supports VBus power ON/OFF to the chipKIT™ USB port, as well as device detect, and initial endpoint 0 setup and configuration. While the event handler may seem somewhat mystical, on closer inspection of the example sketches you will be less intimidated by its implementation. However, to utilize the full capabilities of the event handler will require more detailed knowledge of the USB specification.

It is not required to call either USBHost.Begin() or use the USBHost.DefaultEventHander() if you wish to implement these yourself; however this would require more detailed knowledge of both the chipKIT™ hardware and the USB specification.

ChipKITUSBHost Methods

void USBHost.Begin()
void USBHost.Begin(USB_CLIENT_EVENT_HANDLER FEventHandler)

Parameters:

FEventHandler – A pointer to the event handler that will be called when the USB host receives events from the USB device.

Returns:

Nothing.

Initializes the chipKIT™ hardware and sets the event callback handler. If no event callback handler is specified, USBHost.DefaultEventHandler() will be used. Please make note that USBHOst.Begin() will call USBHost.Init() and you should not call it again. USBHost.Init() is provided for completeness should you want to initialize the hardware and USB Stack yourself.

*boolean DefaultEventHandler(uint8_t address, USB_EVENT event, void * data, DWORD size)*

Parameters:

Address - Address of the USB device generating the event
event - Event that occurred
data - Optional pointer to data for the event
size - Size of the data pointed to by *data

Returns:
Returns TRUE if the event was handled, FALSE otherwise

This gives you access to the default event handler. This handler processes common USB host events including VBUS (5v) power to the USB host port. It will return TRUE if it handles the event, FALSE otherwise. You may monitor any event and provide additional functionality even if the default handler processed it.

ChipKITUSBHost Class Method Mappings to the Underlying Microchip USB Host Interface Routines

Refer to the "*Library Interface (API) > Embedded Host->Embedded Host Stack->Interface Routines*" section in the "*MCHPFSUSB Library Help.chm*" for documentation of these methods.

chipKIT™ USBHost Methods	Microchip USB Host Interface Routines
`void Tasks(void)`	`void USBHostTasks(void)`
`uint8_t ClearEndpointErrors( uint8_t deviceAddress, uint8_t endpoint)`	`BYTE USBHostClearEndpointErrors( BYTE deviceAddress, BYTE endpoint)`
`BOOL DeviceSpecificClientDriver( uint8_t deviceAddress)`	`BOOL USBHostDeviceSpecificClientDriver( BYTE deviceAddress)`
`uint8_t DeviceStatus( uint8_t deviceAddress)`	`BYTE USBHostDeviceStatus( BYTE deviceAddress)`
`uint8_t * GetCurrentConfigurationDescriptor( uint8_t deviceAddress)`	`#define USBHostGetCurrentConfigurationDescriptor( deviceAddress) ( pCurrentConfigurationDescriptor )`
`uint8_t * GetDeviceDescriptor( uint8_t deviceAddress)`	`#define USBHostGetDeviceDescriptor( deviceAddress) ( pDeviceDescriptor )`
`uint8_t GetStringDescriptor( uint8_t deviceAddress, WORD stringNumber, WORD LangID, uint8_t * stringDescriptor, WORD stringLength, uint8_t clientDriverID)`	`#define USBHostGetStringDescriptor( deviceAddress, stringNumber, LangID, stringDescriptor, stringLength, clientDriverID ) USBHostIssueDeviceRequest( deviceAddress, USB_SETUP_DEVICE_TO_HOST | USB_SETUP_TYPE_STANDARD | USB_SETUP_RECIPIENT_DEVICE,`

	USB_REQUEST_GET_DESCRIPTOR, (USB_DESCRIPTOR_STRING << 8) | stringNumber, LangID, stringLength, stringDescriptor, USB_DEVICE_REQUEST_GET, clientDriverID)
BOOL Init(unsigned long flags)	BOOL **USBHostInit**(**unsigned long flags**)
uint8_t Read(uint8_t deviceAddress, uint8_t endpoint, uint8_t * data, DWORD size)	BYTE **USBHostRead**(BYTE **deviceAddress**, BYTE **endpoint**, BYTE * **data**, DWORD **size**)
uint8_t ResetDevice(uint8_t deviceAddress)	BYTE **USBHostResetDevice**(BYTE **deviceAddress**)
uint8_t ResumeDevice(uint8_t deviceAddress)	BYTE **USBHostResumeDevice**(BYTE **deviceAddress**)
uint8_t SetDeviceConfiguration(uint8_t deviceAddress, uint8_t configuration)	BYTE **USBHostSetDeviceConfiguration**(BYTE **deviceAddress**, BYTE **configuration**)
uint8_t SetNAKTimeout(uint8_t deviceAddress, uint8_t endpoint, WORD flags, WORD timeoutCount)	BYTE **USBHostSetNAKTimeout**(BYTE **deviceAddress**, BYTE **endpoint**, WORD **flags**, WORD **timeoutCount**)
uint8_t SuspendDevice(uint8_t deviceAddress)	BYTE **USBHostSuspendDevice**(BYTE **deviceAddress**)
void TerminateTransfer(uint8_t deviceAddress, uint8_t endpoint)	**void** **USBHostTerminateTransfer**(BYTE **deviceAddress**, BYTE **endpoint**)
BOOL TransferIsComplete(uint8_t deviceAddress, uint8_t endpoint, uint8_t * errorCode, DWORD * byteCount)	BOOL **USBHostTransferIsComplete**(BYTE **deviceAddress**, BYTE **endpoint**, BYTE * **errorCode**, DWORD * **byteCount**)
uint8_t VbusEvent(USB_EVENT vbusEvent, uint8_t hubAddress, uint8_t portNumber)	BYTE **USBHostVbusEvent**(USB_EVENT **vbusEvent**, BYTE **hubAddress**, BYTE **portNumber**)
uint8_t Write(uint8_t deviceAddress, uint8_t endpoint,	BYTE **USBHostWrite**(BYTE **deviceAddress**, BYTE **endpoint**,

uint8_t * data, DWORD size)	BYTE * **data**, DWORD **size**)

ChipKITUSBHIDHost Class

This class implements the USB HID (Human Interface Device) host functions. This class is implemented in the chipKITUSBHIDHost library and must be imported after the chipKITUSBHost library in your sketch. The class is pre-instantiated as the USBHIDHost instance and you should not instantiate another instance.

ChipKITUSBHIDHost Class Method Mappings to the Underlying Microchip USB HID Host Interface Routines

Refer to the "*Library Interface (API) > Embedded Host->HID Client Driver->Interface Routines*" section in the "*MCHPFSUSB Library Help.chm*" for documentation of these methods.

chipKIT™ USBHIDHost Methods	Microchip USB HID Host Interface Routines
void Tasks(void)	void USBHostHIDTasks(void)
BOOL ApiFindBit(WORD usagePage, WORD usage, HIDReportTypeEnum type, uint8_t* Report_ID, uint8_t* Report_Length, uint8_t* Start_Bit)	BOOL **USBHostHID_ApiFindBit**(WORD **usagePage**, WORD **usage**, HIDReportTypeEnum **type**, BYTE* **Report_ID**, BYTE* **Report_Length**, BYTE* **Start_Bit**)
BOOL ApiFindValue(WORD usagePage, WORD usage, HIDReportTypeEnum type, uint8_t* Report_ID, uint8_t* Report_Length, uint8_t* Start_Bit, uint8_t* Bit_Length)	BOOL **USBHostHID_ApiFindValue**(WORD **usagePage**, WORD **usage**, HIDReportTypeEnum **type**, BYTE* **Report_ID**, BYTE* **Report_Length**, BYTE* **Start_Bit**, BYTE* **Bit_Length**)
uint8_t ApiGetCurrentInterfaceNum(void);	BYTE **USBHostHID_ApiGetCurrentInterfaceNum(void)**
BOOL ApiImportData(uint8_t * report, WORD reportLength, HID_USER_DATA_SIZE * buffer, HID_DATA_DETAILS * pDataDetails)	BOOL **USBHostHID_ApiImportData**(BYTE * **report**, WORD **reportLength**, HID_USER_DATA_SIZE * **buffer**, HID_DATA_DETAILS * **pDataDetails**)
BOOL HasUsage(HID_REPORTITEM * reportItem, WORD usagePage, WORD usage, WORD * pindex, uint8_t* count)	BOOL **USBHostHID_HasUsage**(HID_REPORTITEM * **reportItem**, WORD **usagePage**, WORD **usage**, WORD * **pindex**, BYTE* **count**)
BOOL	BOOL

DeviceDetect(
  uint8_t deviceAddress)
	USBHostHIDDeviceDetect(
	  BYTE **deviceAddress**)

uint8_t
DeviceStatus(
  uint8_t deviceAddress)
	BYTE
	USBHostHIDDeviceStatus(
	  BYTE **deviceAddress**)

BOOL
Initialize(
  uint8_t address,
  DWORD flags,
  uint8_t clientDriverID)
	BOOL **USBHostHIDInitialize**(
	  BYTE **address**,
	  DWORD **flags**,
	  BYTE **clientDriverID**)

uint8_t
ResetDevice(
  uint8_t deviceAddress)
	BYTE
	USBHostHIDResetDevice(
	  BYTE **deviceAddress**)

uint8_t
TerminateTransfer(
  uint8_t deviceAddress,
  uint8_t direction,
  uint8_t interfaceNum)
	BYTE
	USBHostHIDTerminateTransfer(
	  BYTE **deviceAddress**,
	  BYTE **direction**,
	  BYTE **interfaceNum**)

uint8_t
Transfer(
  uint8_t deviceAddress,
  uint8_t direction,
  uint8_t interfaceNum,
  WORD reportid,
  WORD size,
  uint8_t * data)
	BYTE
	USBHostHIDTransfer(
	  BYTE **deviceAddress**,
	  BYTE **direction**,
	  BYTE **interfaceNum**,
	  WORD **reportid**,
	  WORD **size**,
	  BYTE * **data**)

BOOL
TransferIsComplete(
  uint8_t deviceAddress,
  uint8_t * errorCode,
  uint8_t * byteCount)
	BOOL
	USBHostHIDTransferIsComplete(
	  BYTE **deviceAddress**,
	  BYTE * **errorCode**,
	  BYTE * **byteCount**)

uint8_t
Read(
  uint8_t deviceAddress,
  WORD reportid,
  uint8_t interfaceNum,
  WORD size,
  uint8_t * data)
	#define
	USBHostHIDRead(
	  deviceAddress,
	  reportid,
	  interface,
	  size,
	  data)
	USBHostHIDTransfer(
	  deviceAddress,
	  1,
	  interface,
	  reportid,
	  size,
	  data)

uint8_t
Write(
  uint8_t deviceAddress,
  WORD reportid,
  uint8_t interfaceNum,
  WORD size,
  uint8_t * data)
	#define
	USBHostHIDWrite(
	  address,
	  reportid,
	  interface,
	  size,
	  data)
	USBHostHIDTransfer(
	  address,
	  0,
	  interface,

	reportid, size, data)
USB_HID_DEVICE_RPT_INFO * GetCurrentReportInfo(void)	**#define** **USBHostHID_GetCurrentReportInfo** (&deviceRptInfo)
USB_HID_ITEM_LIST * GetItemListPointers(void);	**#define** **USBHostHID_GetItemListPointers** (&itemListPtrs)

ChipKITUSBMSDHost Class

This class implements the USB MSD (Mass Storage Device) host functions. This class is implemented in the chipKITUSBMSDHost library and must be imported after the chipKITUSBHost library in your sketch. The class is pre-instantiated as the USBMSDHost instance and you should not instantiate another instance.

ChipKITUSBMSDHost Class Method Mappings to the Underlying Microchip USB MSD Host Interface Routines

Refer to the "*Library Interface (API) > Embedded Host->Mass Storage Client Driver->Interface Routines*" section in the "*MCHPFSUSB Library Help.chm*" for documentation of these methods.

chipKIT™ USBMSDHost Methods	Microchip USB MSD Host Interface Routines
void Tasks(void)	void USBHostMSDTasks(void)
uint8_t DeviceStatus(uint8_t deviceAddress)	BYTE **USBHostMSDDeviceStatus**(BYTE **deviceAddress**)
BOOL EventHandler(uint8_t address, USB_EVENT event, void * data, DWORD size)	BOOL **USBHostMSDEventHandler**(BYTE **address**, USB_EVENT **event**, **void** * **data**, DWORD **size**)
BOOL Initialize(uint8_t address, DWORD flags, uint8_t clientDriverID)	BOOL **USBHostMSDInitialize**(BYTE **address**, DWORD **flags**, BYTE **clientDriverID**)
uint8_t ResetDevice(uint8_t deviceAddress)	BYTE **USBHostMSDResetDevice**(BYTE **deviceAddress**)
BOOL SCSIEventHandler(uint8_t address, USB_EVENT event, void * data, DWORD size)	BOOL **USBHostMSDSCSIEventHandler**(BYTE **address**, USB_EVENT **event**, **void** * **data**, DWORD **size**)
BOOL SCSIInitialize(uint8_t address, DWORD flags,	BOOL **USBHostMSDSCSIInitialize**(BYTE **address**, DWORD **flags**,

uint8_t clientDriverID)	BYTE **clientDriverID**)
uint8_t SCSISectorRead(DWORD sectorAddress, uint8_t * dataBuffer)	BYTE **USBHostMSDSCSISectorRead**(DWORD **sectorAddress**, BYTE * **dataBuffer**)
uint8_t SCSISectorWrite(DWORD sectorAddress, uint8_t * dataBuffer, uint8_t allowWriteToZero)	BYTE **USBHostMSDSCSISectorWrite**(DWORD **sectorAddress**, BYTE * **dataBuffer**, BYTE **allowWriteToZero**)
void TerminateTransfer(uint8_t deviceAddress)	**void** **USBHostMSDTerminateTransfer**(BYTE **deviceAddress**)
BOOL TransferIsComplete(uint8_t deviceAddress, uint8_t * errorCode, DWORD * byteCount)	BOOL **USBHostMSDTransferIsComplete**(BYTE **deviceAddress**, BYTE * **errorCode**, DWORD * **byteCount**)
uint8_t Transfer(uint8_t deviceAddress, uint8_t deviceLUN, uint8_t direction, uint8_t * commandBlock, uint8_t commandBlockLength, uint8_t * data, DWORD dataLength)	BYTE **USBHostMSDTransfer**(BYTE **deviceAddress**, BYTE **deviceLUN**, BYTE **direction**, BYTE * **commandBlock**, BYTE **commandBlockLength**, BYTE * **data**, DWORD **dataLength**)
uint8_t Read(uint8_t deviceAddress, uint8_t deviceLUN, uint8_t * commandBlock, uint8_t commandBlockLength, uint8_t * data, DWORD dataLength)	**#define** **USBHostMSDRead**(deviceAddress, deviceLUN, commandBlock, commandBlockLength, data, dataLength) USBHostMSDTransfer(deviceAddress, deviceLUN, 1, commandBlock, commandBlockLength, data, dataLength)
uint8_t Write(uint8_t deviceAddress, uint8_t deviceLUN, uint8_t * commandBlock, uint8_t commandBlockLength, uint8_t * data, DWORD dataLength)	**#define** **USBHostMSDWrite**(deviceAddress, deviceLUN, commandBlock, commandBlockLength, data, dataLength) USBHostMSDTransfer(deviceAddress, deviceLUN, 0, commandBlock, commandBlockLength,

	data, dataLength)
uint8_t SCSIWriteProtectState(void)	BYTE USBHostMSDSCSIWriteProtectState(void)
MEDIA_INFORMATION * SCSIMediaInitialize(void)	MEDIA_INFORMATION * USBHostMSDSCSIMediaInitialize(void)
uint8_t SCSIMediaDetect(void)	BYTE USBHostMSDSCSIMediaDetect(void)

ChipKITMDDFS Class

This class implements the Microchip MDD (FAT) File System. This class is implemented in the chipKITMDDFS library and must be imported after the chipKITUSBHost and chipKITUSBMSDHost libraries in your sketch. The class is pre-instantiated as the MDDFS instance and you should not instantiate another instance.

ChipKITMDDFS Class Method Mappings to the Underlying Microchip USB MDD File System Routines

Refer to Microchip's AN1045 document "*01045b.pdf*" for documentation of these methods.

chipKIT™ MDDFS Methods	Microchip MDD File System APIs
int Init(void)	int FSInit(void)
FSFILE * fopen(const char * fileName, const char *mode)	FSFILE * FSfopen(const char * fileName, const char *mode)
int fclose(FSFILE *fo)	int FSfclose(FSFILE *fo)
void rewind(FSFILE *fo)	void FSrewind(FSFILE *fo)
size_t fread(void *ptr, size_t size, size_t n, FSFILE *stream)	size_t FSfread(void *ptr, size_t size, size_t n, FSFILE *stream)
int fseek(FSFILE *stream, long offset, int whence)	int FSfseek(FSFILE *stream, long offset, int whence)
long ftell(FSFILE *fo)	long FSftell(FSFILE *fo)
int	int

fEOF(FSFILE * stream)[2]	FSfeof(FSFILE * stream)
int format(char mode, long int serialNumber, char * volumeID)	int FSformat(char mode, long int serialNumber, char * volumeID)
int attrib(FSFILE * file, unsigned char attributes)	int FSattrib(FSFILE * file, unsigned char attributes)
int rename(const char * fileName, FSFILE * fo)	int FSrename(const char * fileName, FSFILE * fo)
int remove(const char * fileName)	int FSremove(const char * fileName)
size_t fwrite(const void *ptr, size_t size, size_t n, FSFILE *stream)	size_t FSfwrite(const void *ptr, size_t size, size_t n, FSFILE *stream)
int chdir(char * path)	int FSchdir(char * path)
char * getcwd(char * path, int numbchars)	char * FSgetcwd(char * path, int numbchars)
int mkdir(char * path)	int FSmkdir(char * path)
int rmdir(char * path, unsigned char rmsubdirs)	int FSrmdir(char * path, unsigned char rmsubdirs)
int SetClockVars(unsigned int year, unsigned char month, unsigned char day, unsigned char hour, unsigned char minute, unsigned char second)	int SetClockVars(unsigned int year, unsigned char month, unsigned char day, unsigned char hour, unsigned char minute, unsigned char second)
int FindFirst(const char * fileName, unsigned int attr, SearchRec * rec)	int FindFirst(const char * fileName, unsigned int attr, SearchRec * rec)
int FindNext(SearchRec * rec)	int FindNext(SearchRec * rec)
int	int

[2] fEOF was used instead of feof as there was a name conflict when compiling from within MPIDE.

error(void)	FSerror(void)
int CreateMBR(unsigned long firstSector, unsigned long numSectors)	int FSCreateMBR(unsigned long firstSector, unsigned long numSectors)
Void GetDiskProperties(FS_DISK_PROPERTIES* properties)	Void FSGetDiskProperties(FS_DISK_PROPERTIES* properties)