Application Note

AN_226

FT313H Programming Guide

Document Reference No.: FT_000764

Version 1.1

Issue Date: 2012-11-01

This document provides a detailed description on how to develop software for the FT313H host controller

Future Technology Devices International Limited (FTDI)

Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom
Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758
E-Mail (Support): **support1@ftdichip.com** Web: http://www.ftdichip.com

Table of Contents

1 Introduction

1.1 Overview

FT313H is a single port Hi-Speed Universal Serial Bus (USB) Host Controller which is compliant with Universal Serial Bus Specification Rev 2.0 [1] and compatible with Enhanced Host Controller Interface Specification Rev 1.0 [2].

In this document, software development and integration for the FT313H is described, which covers the following topics:

1. USB host software and FT313H chip architecture
2. FT313H hardware access method
3. FT313H initialization sequence
4. FT313H root hub operations
5. Brief introduction of USB traffic scheduling for FT313H
6. FT313H power management
7. FT313H interrupt handling

1.2 USB host software architecture

USB host software architecture is shown in Figure 1:

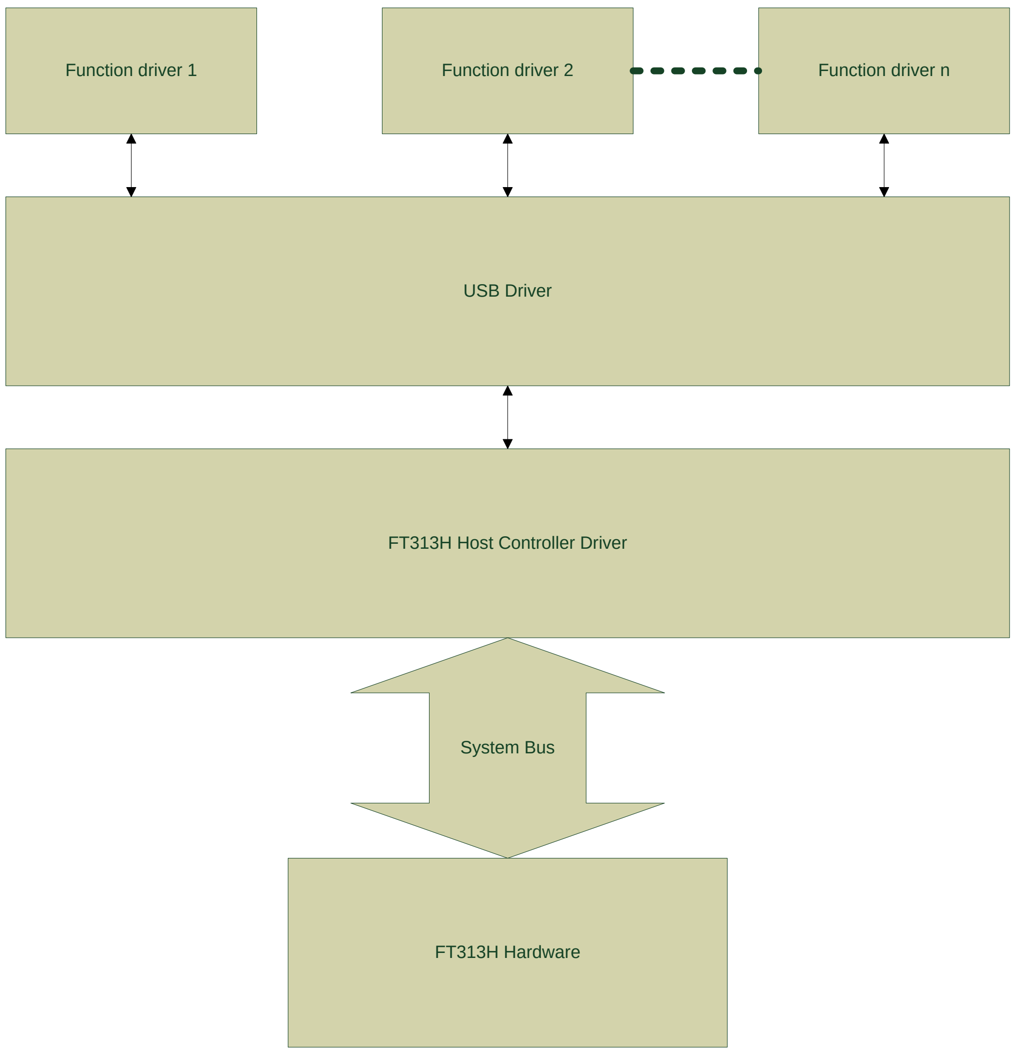

Figure 1: USB host software architecture

The highest layer is the function driver(s) which is used to serve certain USB peripherals, such as Mass storage or HID; the second highest layer is USB driver (USBD), which provides general USB operations and interface to function drivers, this part is normally provided by the operating system; the lowest layer of software is the host controller driver (HCD), which is hardware dependent and provides services to USBD by the actual hardware. For a detailed description on the USB host software architecture refer to chapter 10 of the Universal Serial Bus Specification Rev 2.0 [1].

In this document, we focus on the lowest layer, that is, host controller driver development for FT313H host controller hardware.

1.3 FT313H Chip Archiecture

The rough block diagram of the FT313H is shown in Figure 2 from the software perspective.

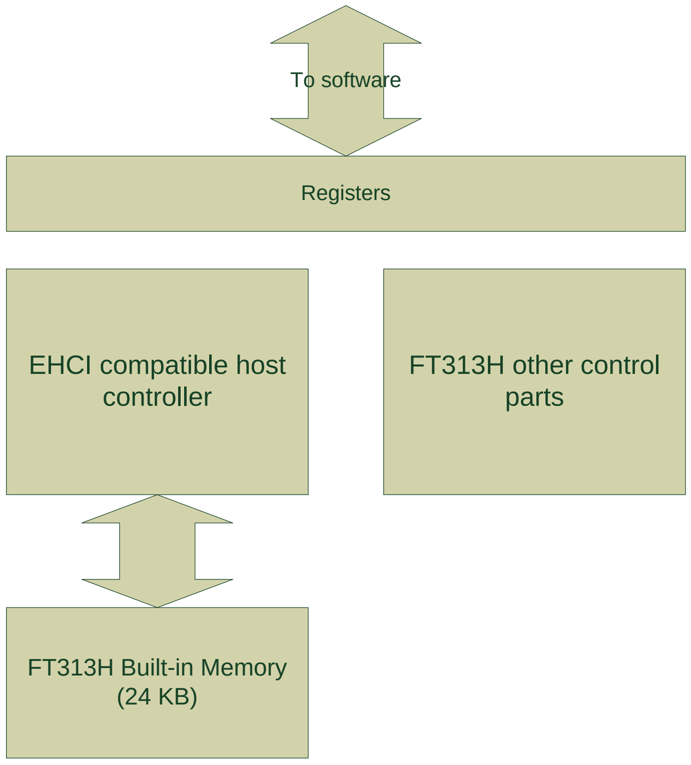
To software
Registers
EHCI compatible host controller
FT313H other control parts
FT313H Built-in Memory (24 KB)

Figure 2: Block diagram of FT313H

From the software perspective, FT313H is roughly an EHCI host controller with its own built-in memory (not using system memory) plus some other control functions not provided by EHCI. All of these resources are accessible through the FT313H's register interface which occupies 256 bytes of address space. For a detailed structure of the FT313H register map, refer to FT313H Datasheet [3]. The built-in memory of the FT313H houses the EHCI periodic frame list, compatible transfer descriptors and associated USB payload data. It is the host controller software's responsibility to manage all these hardware resources so as to complete the requests from upper layer software (USBD).

2 FT313H hardware access

All FT313H hardware access is through register access, this is true also for built-in memory access. The register map of the FT313H is shown in section 2.1.

The first set of registers contains the EHCI operational registers set which is located from address 0 to 30h. This register set is a customized version of the standard EHCI register set (support one downstream port and remove 64 bit support, etc.) with register's offset unchanged.

The second set of registers is the configuration registers which contain functions like hardware reset, hardware mode setup, internal memory access etc. It also contains registers for power management support.

The third set of registers are FT313H specific interrupt registers; please note that EHCI operational register has its own interrupt control registers.

Finally, there are also USB testing registers which are used for supporting USB compliance testing.

2.1 FT313H register map

FT313H Register Map is shown in Table 1, for detailed description of each register; please refer to chapter 5 of the FT313H Datasheet [3].

Table 1: FT313H Registers Map

Address	Register	Reset value	Description
EHCI operational register			
00h	HCCAPLENGTH	01000010h	Capability register
04h	HCSPARAMS	00000001h	Structural parameter register
08h	HCCPARAMS	00000006h	Capability parameter register
10h	USBCMD	00080B00h	USB command register
14h	USBSTS	00001000h	USB status register
18h	USBINTR	00000000h	USB interrupt enable register
1Ch	FRINDEX	00000000h	Frame index register
24h	PERIODICLISTADDR	00000000h	Periodic frame list base address register
28h	ASYNCLISTADDR	00000000h	Current asynchronous list address register
30h	POSTSC	00000000h	Port status and control register
Configuration register			
34h	EOFTIME	00000041h	EOF time and asynchronous schedule sleep timer register
80h	CHIPID	03130001h	Chip ID register
84h	HWMODE	00000000h	HW mode control register
88h	EDGEINTC	0000001Fh	Edge interrupt control register
8Ch	SWRESET	00000000h	SW reset register
90h	MEMADDR	0000h	Memory address register
92h	DATAPORT	0000h	Data port register
94h	DATASESSION	0000h	Data session length register
96h	CONFIG	1FA0h	Configuration register
98h	AUX_MEMADDR	0000h	Auxiliary memory address register

Address	Register	Reset value	Description
9Ah	AUX_DATAPORT	0000h	Auxiliary data port register
9Ch	SLEEPTIMER	0400h	Sleep timer register
Interrupt register			
A0h	HCINTSTS	0000h	Host controller interrupt status register
A4h	HCINTEN	0000h	Host controller interrupt enable register
USB testing register			
50h	TESTMODE	00000000h	Test mode register
70h	TESTPMSET1	00000000h	Test parameter setting 1 register
74h	TESTPMSET2	00000000h	Test parameter setting 2 register

2.2 Chip Access Mode Select

FT313H can function in either 16-bit mode or 8-bit mode; the default mode when FT313H powered up is 16-bit mode.

The 16-bit or 8-bit mode selection is completed by clearing or setting DATA_BUS_WIDTH bit (bit 4) in SWRESET register (8Ch). This register's valid bits are from bit 0 to bit 7, thus it could be accessed correctly either from a 16-bit system or 8-bit system.

Chip data bus width selection should be done during the initialization phase, and should not change during the software execution.

2.3 Register Access

After the chip access mode is set as described in section 2.2, the application software can access other registers to complete further chip operations.

Accessing registers with data wider than the mode selected (8/16 bit data) will require multiple accesses starting at the lowest address.

For example, if the chip is in 8-bit mode and a 32 bit long register read is needed, the register should be read 4 times from the lowest address to the highest address of the register offset.

2.4 Memory Access

FT313H built-in memory access is completed by manipulating three registers, i.e. MEMADDR (90h) register, DATAPORT (92h) register and DATASESSION (94h) register. Actual procedures are in Figure 3.

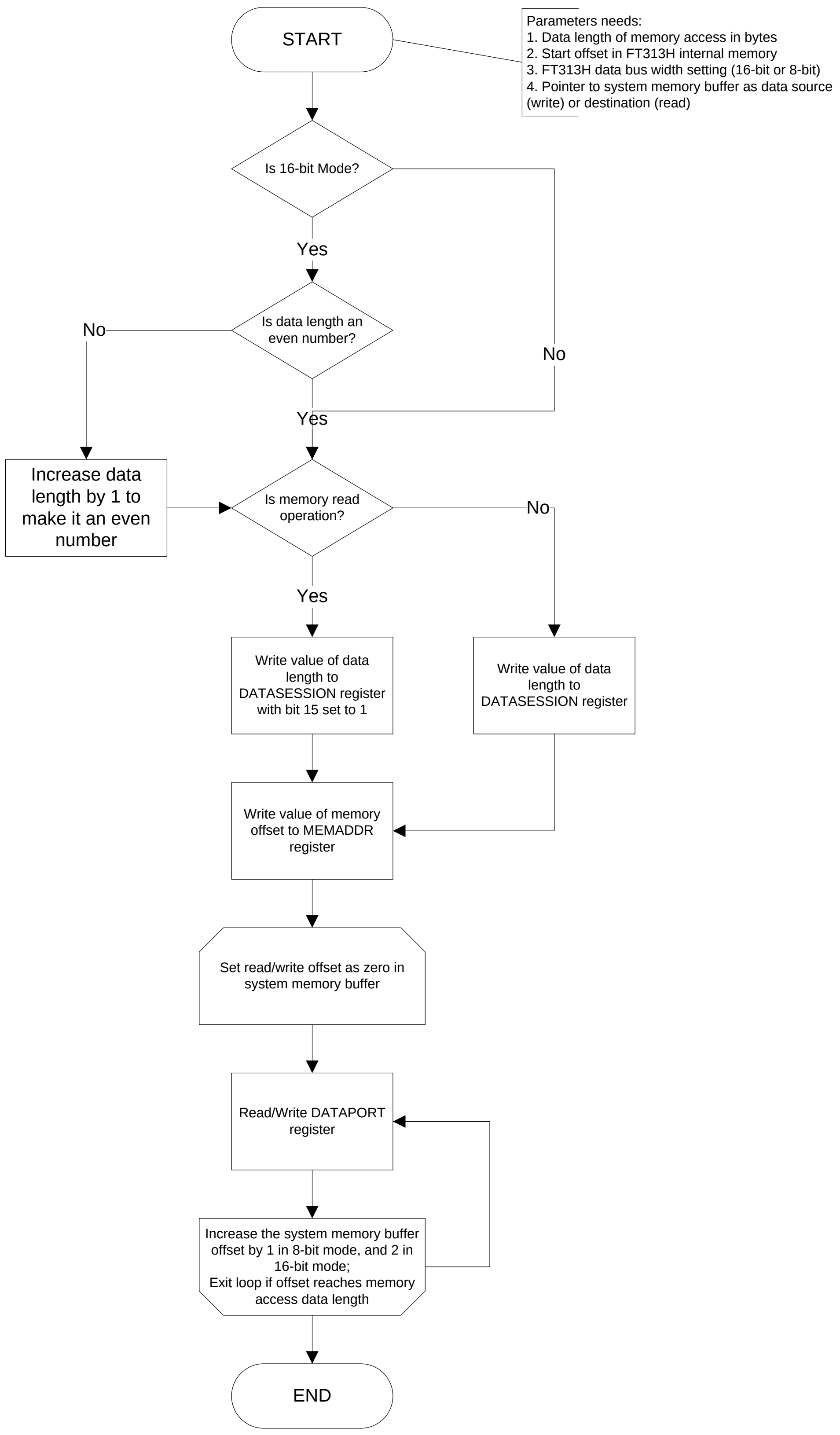
START
Parameters needs:
1. Data length of memory access in bytes
2. Start offset in FT313H internal memory
3. FT313H data bus width setting (16-bit or 8-bit)
4. Pointer to system memory buffer as data source (write) or destination (read)
Is 16-bit Mode?
Yes
No
Is data length an even number?
No
Yes
Increase data length by 1 to make it an even number
Is memory read operation?
No
Yes
Write value of data length to DATASESSION register with bit 15 set to 1
Write value of data length to DATASESSION register
Write value of memory offset to MEMADDR register
Set read/write offset as zero in system memory buffer
Read/Write DATAPORT register
Increase the system memory buffer offset by 1 in 8-bit mode, and 2 in 16-bit mode;
Exit loop if offset reaches memory access data length
END

Figure 3: FT313H Memory Access Flowchart

First, set the length and direction (read or write) of the data transmission session by writing the length value to the DATASESSION register, and if data session is read, bit 15 must be set to 1, otherwise 0; As FT313H has 24 kilobytes built-in memory, the range of data length is from 0 to 6000h.

Second, set the address where memory access should start by writing the address to MEMADDR register. Please note that addresses start from offset 0000h and up to 5FFFh. It is software's responsibility to make sure that the address plus the data length in the previous step will NOT exceed the total memory range of the FT313H's built-in memory.

Third, read or write the actual payload by accessing the DATAPORT register repeatedly, until all data is read out or written into the FT313H built-in memory. The amount of data that each read or write operation to the DATAPORT register can convey depends on the chip access mode, that is, if chip is in 16-bit mode, 2 bytes will be read or written each time, while 1 byte will be read or written if the chip is in 8-bit mode. Thus in 16 bit mode, the software needs to make sure that the data session length is even and starts from an even number offset.

One important point to document: one memory access session cannot be interrupted by another memory access session; access control mechanism (such as mutex) should be used to make the memory access atomic.

Also, both the DATASESSION register and MEMADDR register are 16 bit registers, and must follow the register access method as described in section 2.3; while DATAPORT register's width depends on the chip access mode.

2.5 Summary

As both register and built-in memory access are fundamental operations, it is a good practise to implement a set of API functions which are capable of reading/writing different length registers as well as memory sessions by calling one of these API functions.

All descriptions from this point assume that such a set of API functions are properly implemented, and low level access details will not be repeated.

3 Chip initialization sequence

After the chip is powered, software needs to initialize FT313H so that it will be ready for new insertions and other services. The general initialization sequence of the FT313H is shown in

Figure 4: .

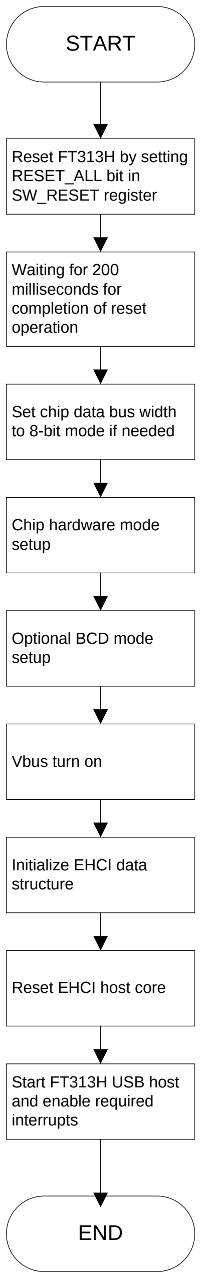
START
Reset FT313H by setting RESET_ALL bit in SW_RESET register
Waiting for 200 milliseconds for completion of reset operation
Set chip data bus width to 8-bit mode if needed
Chip hardware mode setup
Optional BCD mode setup
Vbus turn on
Initialize EHCI data structure
Reset EHCI host core
Start FT313H USB host and enable required interrupts
END

Figure 4: FT313H Initialization Sequence Flowchart

The initialization sequence is described as following.

3.1 Set chip access mode and reset the chip

Upon system power on, software needs to reset the FT313H chip, so that chip will be in a known state, this is done by setting bit RESET_ALL (bit 0) to 1 in the SWRESET register. A 200 milliseconds delay must be applied after register writing to make sure that the chip reset is complete.

Based on platform requirements, software then needs to set the chip access mode correctly (refer to section 2.2), and complete the first step of chip initialization.

3.2 Chip hardware mode setup and turn on VBus

Software must then set the hardware mode, such as interrupt type, i.e. edge trigger or level trigger; interrupt polarity etc. Software should also set the interface lock and enable the global interrupt. All of the above operations are implemented by setting corresponding bits in the HWMODE register (84h). For details of related register settings refer to section 5.3.3 and section 5.3.4 of the FT313H datasheet [3].

The software should now turn on VBus, as VBus is off by default. The method to turn on the VBus is by clearing the VBUS_OFF bit (bit 7) in CONFIG register (96h). Software could also, optionally set the BCD module to the expected mode when turning on VBus. Details in next section.

Next, the software could read the chip ID register (80h) to verify that low level access is ok as well as getting the FT313H hardware revision information.

3.3 Optional battery charging function initialization

Battery charging function initialization is accomplished by programming certain bits in CONFIG register (96h) before turning on VBus.

This feature depends on system requirements. If the BCD function is not needed, software should disable the function by clearing the BCD_EN (bit 5) in CONFIG (96h) register, please note that the BCD function is enabled by default.

If the BCD function is needed, software should keep the BCD_EN bit as one and, based on system requirements, BCD mode could be set either by hardware pins (CPE0 and CPE1) or software. If the BCD mode is hardware or pin configured, software should set the BCD_MODE_CTRL (bit 15) in CONFIG register as 0; otherwise, software should set the BCD_MODE_CTRL bit as 1 and further set the BCD mode by programming BCD_MODE field (bit 14:13) in CONFIG register.

Supported modes are SDP, DCP or CDP, and value is 0h, 1h or 3h respectively.

After the BCD function related values are determined, software could turn on the VBus together with all the BCD function related bits by writing the appropriate value to CONFIG register.

3.4 Initiailze data structure for USB host controller

To prepare the FT313H working as a USB host controller, the EHCI related data structure must be initialized. Firstly, based on system decisions, software needs to allocate one segment of 4096 bytes aligned memory as a periodic list in the FT313H build-in memory. As FT313H has a limited number of 4096 bytes aligned memory addresses, it is a good idea to use memory offset 0 as the start address for the periodic list. Software also needs to initialize all the vectors in the periodic list as 00000001h (the EHCI null pointer) to show that there is no active periodic transfer so far.

Software also needs to allocate one queue head (qH) and its associate queue element transfer descriptor (qTD) and mark it as the head for the asynchronous schedule list.

Software must then send a software reset to the FT313H by setting HC_RESET bit (bit 1) to 1 in the USBCMD (10h) register, and then poll this bit until its value is cleared by hardware.

After that, based on the system requirements, software must also set a desired interrupt threshold and a periodic frame list size in the USBCMD (10h) register and make sure that INT_OAAD, PSCH_EN, ASCH_EN and HC_RESET bits are all cleared. Software can then start the host controller by setting the RS bit to 1 in the USBCMD register.

Software also needs to enable required interrupts, such as the port change interrupt enable and USB interrupt enable, by setting corresponding bits in the USBINTR (18h) register.

At this time, the USB host is already in a working state; SOF packets will be getting sent out and the FT313H is able to detect peripheral insertion.

Further software processes serviced by the FT313H are described in the next section.

4 USB host operations

This section mainly describes the FT313H's behaviour as a USB host. As the FT313H is compatible with EHCI, most of the behaviour is quite similar to that of an EHCI host. In this section, it is assumed that readers are familiar with the EHCI specification and this section will provide an overview of the FT313H's behaviour with an emphasis on important points as well as the unique parts of the FT313H.

4.1 Root hub control

4.1.1 USB peripheral insert and remove

After the FT313H completes the initialization sequence, a USB peripheral device insertion will generate a port change interrupt which is indicated by PO_CHG_DET bit (bit 2) in USBSTS register (14h) becoming one and bit CONN_STS (bit 0) in PORTSC register (30h) also becoming one. Software can detect that a new device is inserted from these two bits changing via an interrupt (if interrupt is enabled) or polling mechanism.

After software detects a peripheral insertion, it should clear the PO_CHG_DET bit in USBSTS register by writing a one into the same bit in the register. Further processing of the peripheral insertion event such as port reset and speed negotiation will be discussed in section 4.1.2.

When a USB peripheral is removed from the FT313H downstream port, a port change interrupt will be generated. This time the CONN_STS bit in PORTSC register will become zero.

4.1.2 USB port reset and speed negotiation

After software detects the new USB peripheral insertion and before a full USB enumeration starts, software has to enable the port and determine the speed of the inserted peripheral first.

This is done by the port reset operation; the actual procedure is as follows:

1. Stop the host controller driver by clearing RS bit in USBCMD register (10h) and make sure that HCHALTED bit (bit 12) is set in USBSTS (14h)register.

2. The software must write a one to PO_RESET bit (bit 8) in PORTSC register(30h) with PO_EN bit cleared (as software cannot forcefully enable a port) to start the port reset operation.

3. After that, a 50 milliseconds delay is expected, before software stops the port reset operation by clearing the PO_RESET bit in PORTSC register. Software needs to further check that PO_RESET bit is cleared which may take a few hundred microseconds.

4. Software should check whether PO_EN bit (bit 2) in PORTSC register (30h) is set by hardware; if not, this means something wrong has happened and port reset failed. Otherwise, software should restore the RS bit to 1 and make sure that HCHALTED bit is cleared.

5. Software should now clear the PO_EN_CHG bit (bit 3) if the port is enabled successfully. Next, software should check what speed class the newly inserted USB peripheral belongs to. This is done by checking the HOST_SPD_TYP field (bit 6 and 7) of HWMODE register (84h). The value "2", "0" and "1" represents "high speed", "full speed" and "low speed" respectively.

6. If the speed value is read successfully, port reset is complete, and software can start the enumeration operation based on information received from the port reset operation.

4.2 USB transfer schedule

4.2.1 Asynchronous transfer schedule

4.2.1.1 General information on asynchronous transfer

USB control transfer and bulk transfer belong to the asynchronous transfer type. The actual timing of the transfers being scheduled is determined by the host controller hardware. The software's job is to submit the requirements of the asynchronous transfer and make sure that dynamic USB transfer request submissions will not interfere with host controller operation. .

In general, software submits request for asynchronous transfer by adding items to the asynchronous list which resides in the FT313H's built-in memory. The general format for the asynchronous schedule list is shown in Figure 5.

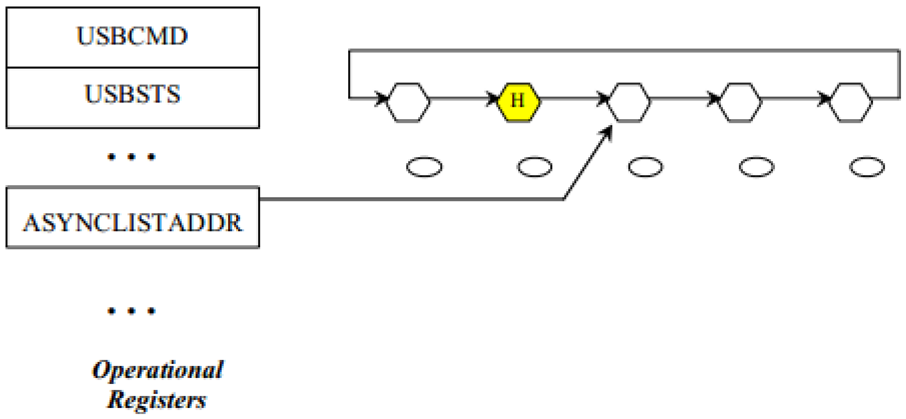

Figure 5: General Format of Asynchronous Schedule List

The Asynchronous schedule traversal is enabled or disabled via the Asynchronous Schedule Enable bit in the USBCMD register (10h). If the Asynchronous Schedule Enable bit is set to a zero, then the host controller simply does not try to access the asynchronous schedule via the ASYNCLISTADDR register (28h). Likewise, when the Asynchronous Schedule Enable bit is a one, then the host controller does use the ASYNCLISTADDR register (28h) to traverse the asynchronous schedule. Modifications to the Asynchronous Schedule Enable bit are not necessarily immediate. Rather the new value of the bit will only be taken into consideration the next time the host controller needs to use the value of the ASYNCLISTADDR register (28h) to get the next queue head.

The Asynchronous Schedule Status bit in the USBSTS register (14h) indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing a one (or zero) to the Asynchronous Schedule Enable bit in the USBCMD register (10h). Software then can poll the Asynchronous Schedule Status bit to determine when the asynchronous schedule has made the desired transition. Software must not modify the Asynchronous Schedule Enable bit unless the value of the Asynchronous Schedule Enable bit equals that of the Asynchronous Schedule Status bit.

4.2.1.2 qH and qTD handling

Asynchronous transfers are carried by qH and qTD only. Practically, each (control or bulk) end point on a USB peripheral is represented in the FT313H by a qH and the payload of each end point is carried by qTD. qHs are linked through the Queue Head Horizontal Link Pointe to form a link list and there is one special qH with its H bit (bit 15) in queue head Dword 1 set as 1 to indicate that it serves as the head of the asynchronous list, thus software must be sure that there should be only one qH with its H bit set as 1.

One qH could have several qTDs, and this is most commonly used for control transfer. Practically speaking, a control transfer is served by 3 cascaded qTDs, first one carries SETUP token, second one carries data phase, while last one carries status phase. If there is no data phase for this control transfer, the second qTD is not needed.

One important point to consider: any qH needs at least one additional qTD, normally referred to as a dummy qTD. This qTD does not carry any actual payload and is identified by the Halted status bit (bit 6) in its token field being set to 1, which serves as an indicator to the host controller hardware as the end of the qTD list. Its main usage is for adding more qTDs to the qH safely. When the software needs to add more qTD(s) to a given qH, it must add the new qTD(s) by following this procedure (assume the qTD(s) list that needs to be added is already linked together by the next qTD pointer field):

1. Save the content of the token field of the first qTD to be added somewhere that software could retrieve back later. E.g. to a temporary variable.

2. Change the token of the first qTD so its Halted bit is set as 1 and all other bits are zero, and copy the content of the first qTD to the dummy qTD, thus this qTD is still considered as dummy and hardware will NOT access this partially updated qTD and qTD(s) after it.

3. Link the first qTD to the last of the qTD of the newly qTD list, if only one qTD is added, this qTD will be added after the dummy qTD that has just been updated from the previous step. Re-initialize this qTD so that it will become the new dummy qTD.

4. Restore the token value store in step 1 to the previous dummy qTD's token field to complete the qTDs appending operation, and hardware will start processing the newly added qTD when its qH is scheduled.

After qTD is executed by hardware, it could be unlinked from the qH that it belongs to, and makes it possible to be reused by the same or different qH again, however, a dummy qTD must be kept all time until its qH is freed (mainly due to the peripheral being unplugged and the end point no longer exists).

4.2.2 Periodic transfer schedule

USB interrupt and isochronous transfers belongs to the periodic transfer type which is characterised as transfers occurring repeatedly based on a certain period. Periodic transfers are indicated by different transfer descriptors, high speed isochronous transfer is indicated by iTD, while full and low speed isochronous transfer are indicated by siTD. Interrupt transfers of all speeds are carried-out by qH and qTD which are also used by asynchronous transfers as described in prior sections.

For FT313H, the transaction length field in iTD can be modified by both software and hardware. , Software writes a value in this field to indicate the maximal length possible for the isochronous transaction. After this transaction is scheduled, the hardware will modify the value of the transaction length field to report the actual bytes of the payload received. Per the EHCI specification, the hardware will update this field to the actual number of bytes received, however, with the FT313H, hardware will update this field to the remaining value (i.e. how many bytes that this transaction is still able to receive). The actual number of bytes received is the difference of this number and previous software programmed value. This behaviour is the same as qTD's process for similar fields.

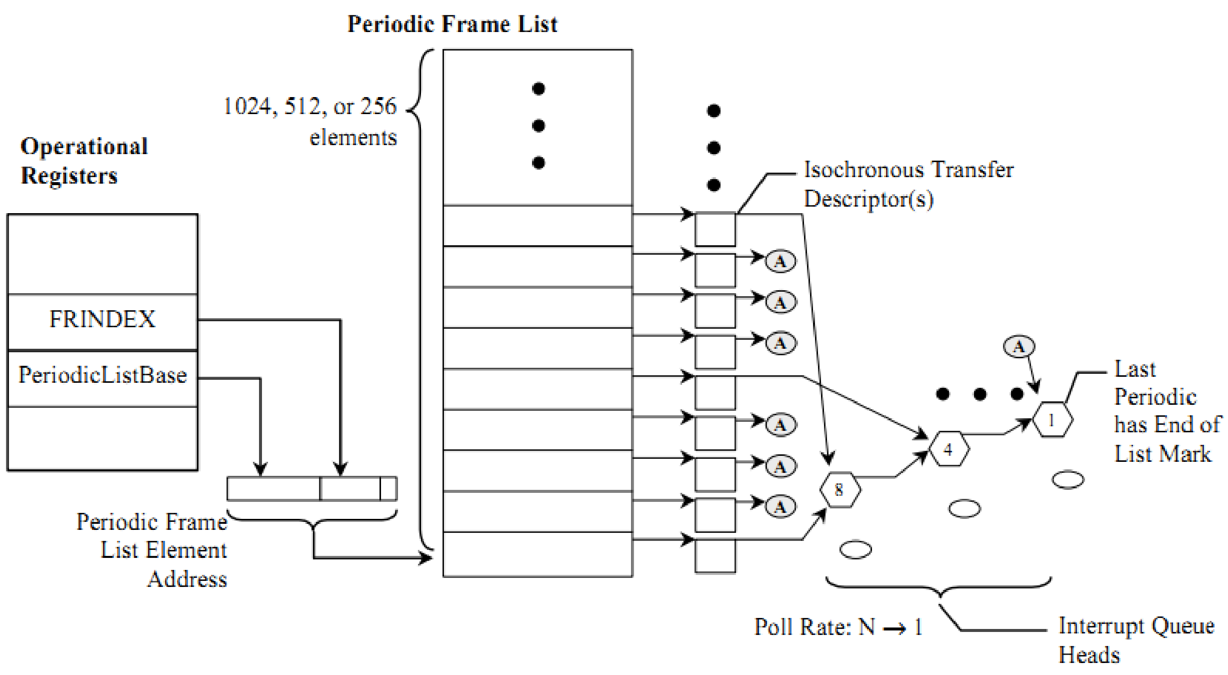
Periodic Frame List
1024, 512, or 256 elements
Operational Registers
FRINDEX
PeriodicListBase
Periodic Frame List Element Address
Isochronous Transfer Descriptor(s)
A
Last Periodic has End of List Mark
1
4
8
Poll Rate: N → 1
Interrupt Queue Heads

Figure 6: General format for periodic schedule list

Unlike asynchronous transfer, periodic transfer is software scheduled. Software needs to control the exact timing of the periodic transfer based on timing requirements of the interrupt or isochronous transfer properties.

The actual scheduling is driven through the periodic frame list, which is a pointer table of 1024, 512 or 256 elements based on the configuration determined by software as shown in Figure 6. Each entry in the periodic frame list represents one USB frame (in the timing perspective). The content is a pointer to a transfer descriptor (iTD etc.) list that the hardware should execute at the given frame (and also micro-frame for high speed transfers). If the entry in the periodic frame list points to an EHCI NULL pointer, it means that there is no periodic transfer needs scheduling at this USB frame.

Periodic schedule traversal is enabled or disabled by the Periodic Schedule Enable bit (bit 4) in USBCMD register. If Periodic Schedule Enable bit is set to a zero, then the host controller simply does not try to access the periodic schedule via the PERIODICLISTADDR register (24h). Likewise, when the Periodic Schedule Enable bit is a one, then the host controller does use the PERIODICLISTADDR register (24h) to traverse the periodic schedule. When both periodic and asynchronous schedule are enabled, periodic schedule will always happen first in any micro-frame. It is the software's responsibility to ensure that there is no over allocation of periodic transfer.

As software will take the full control of the periodic transfer timing, software needs to reserve enough time before the actual scheduling occurs. This is performed by a read to the FRINDEX register to determine the current frame and micro-frame the host controller is currently executing. Exactly how long the reserved time should be depends on the specific platform, such as processing speed and interrupt latency as well as the periodic transfer's timing and amount of communication buffer requirements.

4.2.3 Summary and further readings

USB transfer scheduling for EHCI-compatible host controller FT313H involves many additional elements. For more details, please refer to the EHCI specification [2] and reference source code.

4.3 Power management

FT313H supports power management features so that power consumption can be optimized.

FT313H power management function consists of two parts, chip level power management and port level power management.

4.3.1 FT313H chip level power management

FT313H chip level power management function is used when the higher levels of software determine that the FT313H should be put into suspend mode, to save power, and when higher levels of software determine the FT313H should resume operation.

Chip level power management is comprised of three main functions: chip suspend, chip resume, and (external) wakeup functions.

4.3.1.1 Chip suspend

Chip suspend is triggered by upper layer software when it believes that the FT313H should be put into suspend mode to save power. The actual steps to put FT313H into suspend mode are as follows:

1. Stop the asynchronous schedule as well as periodic schedule by clearing ASCH_EN and PSCH_EN bits in command register.
2. Complete all the pending activities in FT313H's local memory
3. Stop FT313H by clearing the RS bit in command register
4. If the USB downstream port is enabled, suspend the port by setting the PO_SUSP bit in PORT_SC register to 1 and wait for 5 milliseconds. This ensures USB peripherals being able to detect the suspend signal from the FT313H
5. Switch off the FT313H's oscillator, host controller clock and internal PLL by clearing the OSC_EN, HC_CLK_EN and PLL_EN in CONFIG register.
6. Set the corresponding bits in HCINTEN register according to the resume or wakeup events that are needed, if OC is also needed as wakeup events, PORT_OC_EN (bit 6) in CONFIG register must also be set as well as OCINT_EN (bit 6) in HCINTEN register
7. Finally, clear the U_SUSP_N bit in the EOTTIME register to put the chip into suspend mode

After that, unless chip resume is needed, no register read operation should be done to FT313H any more, as this will wake up the chip from suspendmode.

4.3.1.2 Chip resume

Chip resume is also triggered by upper layer software when it believes that FT313H should be restored to active mode; for example, when a user presses a button to wake up the system.

The resume operation starts with a dummy read of the FT313H, the register dummy read serves as a resume signal to the FT313H that is in suspend mode. It is recommended to use a **single** read of the software reset register (Read only lower 8 bits in 8-bit mode) to generate the resume signal and apply a 10 milliseconds delay after the register dummy read.

Note: If the clock ready interrupt is enabled, this interrupt could trigger within 10 milliseconds. The software needs to ensure that this interrupt will not interfere with the resume operation.

Upon the dummy read operation, FT313H's clock ready interrupt will be triggered. Upon receiving this interrupt, software should turn on transceiver by set U_SUSP_N bit in EOTTIME register to ONE. Software should read back the content of the register after writing the appropriate value to the register to make sure that U_SUSP_N bit is set already.

At this time, the FT313H hardware is active again, and the software needs to restore it to the proper status prior to the suspend event.

Actual procedure is as follows:

1. Disable EHCI interrupt so that restore operation will not be interrupted unnecessarily
2. Restore register PERIODICLISTADDR and ASYNCLISTADDR to their original value
3. Restore previous command register value
4. Resume port if the port has been suspend before by set the F_PO_RESM bit in port status register and clear it after 20 milliseconds
5. Set the ASCH_EN and/or PSCH_EN bits if there is active transfer before suspend

4.3.1.3 External wakeup

The FT313H may return to a working or active mode as a result of external events, which we refer to as wakeup.

FT313H support several kinds of events as wakeup events, they are:

- USB peripheral connection/disconnection
- USB peripheral remote wakeup
- OC (over current)

If the FT313H is to woken up from external events, the software must set the corresponding bit(s) in the HCINTEN register. Bit WAKEUPINT_EN governs wakeup on connection/disconnection; while bit REMOTEWKINT_EN governs remote wakeup, and bit OCINT_EN governs over current wakeup. The software must set any or all of these three bits according to the system configuration before putting the FT313H to suspend.

Unlike resume operation, the wakeup operation is started from an interrupt service routine (ISR). When a hardware wakeup event happens, FT313H's interrupt handler will be called. In the FT313H interrupt handler, the following operation should be executed:

1. Read the value of register HCINTSTS, and mask it with the value of register HCINTEN
2. If result is non-zero, clear the interrupt status by writing the masked result back to HCINTSTS register
3. Check and make sure CLKREADY bit is also set in HCINTSTS register
4. After that, execute the same sequence as the resume operation

For OC triggered wakeup, special processing is needed as OC is considered an abnormal case and requires shutting down the USB bus. Upon getting the OC event, software should set bit VBUS_OFF to 1 in CONFIG register to turn off the VBUS and produce some alert message if possible.

4.3.2 FT313H port level power management

In certain circumstances it is not necessary to suspend the FT313H chip, instead it is only required to suspend or resume the USB downstream port of the FT313H, e.g. when executing USB compliance tests. With port suspend only, the power consumption will not reduce much (as chip is still active), however, port resume will take much less time than the chip suspend case.

Following sections will describe the downstream port power management related operations.

4.3.2.1 Port suspend

To suspend the downstream port only, software should first check whether the port is enabled, and not currently under reset operation.

If the above check passed, software could suspend the port by setting PO_SUSP bit in the port status register with the PO_EN_CHG and CONN_CHG bits cleared.

4.3.2.2 Port resume

To resume the downstream port, software should first check whether the port is enabled, not in reset operation and is in suspend mode.

If the above check passed, the software could start the port resume operation by setting F_PO_RESM bit in the port status register with the PO_EN_CHG and CONN_CHG bits cleared.

Port resume signal needs to last at least 20 milliseconds and software has to guarantee the timing. After 20 milliseconds have passed, software will complete the port resume by clearing F_PO_RESM bit with PO_EN_CHG and CONN_CHG bits set as zero in the port status register. Software must then read out the value of port status register to make sure that F_PO_RESM bit is really cleared.

4.4 Interrupt handling

The FT313H host controller supports a few interrupt sources which could be categorized into two categories, EHCI related interrupt and FT313H specific interrupt.

4.4.1 EHCI related interrupts

EHCI related interrupt is mainly about USB transfer scheduling status, such as one previously scheduled transfer has completed or new USB peripheral is inserted or unplugged. Software needs to take appropriate actions based on the nature of the interrupt. For example, a USB peripheral insertion interrupt (PO_CHG_DET) should further trigger port reset and speed negotiation operations; and a USB_INT interrupt will lead to a full scanning of all previously scheduled descriptors to find out which descriptor has been completed and appropriate follow up actions (e.g. continue programming next segment) should be applied also.

In the USBCMD register, there is a field named the interrupt threshold control, which is bit 16 to bit 23. When the value of this field is set as 01h, the minimal possible value, in EHCI spec, this means the interrupt threshold is 1 micro-frame or 125 μS; however, for FT313H, this value means there is no limitation to the interrupt interval, hardware will generate interrupt as soon as any interrupt event happens.

Due to this, when software selects the minimal value, software must prepare for the scenario that another interrupt may happen during the execution of the interrupt handler for the previous interrupt, which may cause an interrupt missing under certain conditions. The solution to this is: after processing the current interrupt, the software interrupt handler should read the EHCI interrupt status and enable register again to find out whether there were new interrupts received during the previous process. If an interrupt was received, the software must continue to process the new interrupt until there are no more interrupts pending.

4.4.2 FT313H interrupts

FT313H has its own interrupts which are mainly related to FT313H power management and those that could help software achieve some special goals. There are two registers related to FT313H interrupt, i.e. HCINTSTS (A0h) and HCINTEN (A4h), the former contains the interrupt status bits while the latter contains the corresponding enable bits.

Important ones are described as following:

4.4.2.1 Power management related interrupts

Bits 3, 5, 6, and 7 of both HCINTSTS and HCINTEN registers control the power management behaviour of the FT313H such as which event could be used as a wakeup event. For detailed description on how to use these bits could refer to section 4.3.

4.4.2.2 SOF and uSOF interrupts

Bits 0 and 1 control the SOF and uSOF interrupt respectively. These two interrupts will be triggered periodically if enabled, which is good for initial software debugging. Software will always have an access point to hardware through the interrupt handler.

4.4.2.3 BUSINACTIVE interrupt

Bit 4 of HCINTSTS and HCINTEN is used for reporting USB bus activities. This interrupt will be triggered if FT313H's USB bus is in idle mode for more than certain amount of time, software could use this feature to have a smarter way of controlling FT313H's power state to achieve better power efficiency. The bus inactive time is configured though SLEEP_TIMER register (9Ch).

5 Reference Source Code

Author	Reference Code
Yang Chang	FT313H-HCD_Linux-1.0.tar.gz (must align with actual file name)

6 FTDI Chip Contact Information

Head Office – Glasgow, UK

Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales)	sales1@ftdichip.com
E-mail (Support)	support1@ftdichip.com
E-mail (General Enquiries)	admin1@ftdichip.com
Web Site URL	http://www.ftdichip.com
Web Shop URL	http://www.ftdichip.com

Branch Office – Taipei, Taiwan

2F, No. 516, Sec. 1, NeiHu Road
Taipei 114
Taiwan , R.O.C.
Tel: +886 (2) 8797 1330
Fax: +886 (2) 8751 9737

E-mail (Sales)	tw.sales1@ftdichip.com
E-mail (Support)	tw.support1@ftdichip.com
E-mail (General Enquiries)	tw.admin1@ftdichip.com
Web Site URL	http://www.ftdichip.com

Branch Office – Oregon, USA

7130 SW Fir Loop
Tigard, OR, USA 97223-8160
Tel: +1 (503) 547 0988
Fax: +1 (503) 547 0987

E-Mail (Sales)	us.sales@ftdichip.com
E-Mail (Support)	us.support@ftdichip.com
E-Mail (General Enquiries)	us.admin@ftdichip.com
Web Site URL	http://www.ftdichip.com

Branch Office – Shanghai, China

Room 1103, No666 West Huahai Road,
Shanghai, 200052
China
Tel: +86 21 62351596
Fax: +86 21 62351595

E-mail (Sales)	cn.sales@ftdichip.com
E-mail (Support)	cn.support@ftdichip.com
E-mail (General Enquiries)	cn.admin@ftdichip.com
Web Site URL	http://www.ftdichip.com

Distributor and Sales Representatives

Please visit the Sales Network page of the FTDI Web site for the contact details of our distributor(s) and sales representative(s) in your country.

7 Appendix A – References

Document References

[1] Universe Serial Bus Specification, Revision 2.0
[2] Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 1.0
[3] FTDI FT313H Datasheet

Acronyms and Abbreviations

Terms	Description
USB	Universe Serial Bus
EHCI	Enhanced Host Controller Interface
BCD	Battery Charging Device
SDP	Standard Downstream Port
DCP	Dedicated Charging Port
CDP	Charging Downstream Port

8 Appendix B – List of Tables & Figures

List of Tables

List of Figures

9 Appendix C – Revision History

Document Title:	FT313H Programming Guide
Document Reference No.:	FT_000764
Clearance No.:	FTDI# 319
Product Page:	http://www.ftdichip.com/FTProducts.htm
Document Feedback:	AN_226_FT313H_Programming_Guide

Version 1.0	Initial Release	OCT 2012
Version 1.1	Formatting tidy up/added flow charts	NOV 2012