presented by

American Megatrends

Implementing Advanced USB Interrupt Transfers

UEFI Summerfest – July 15-19, 2013
Presented by Zachary Bobroff (AMI)

Updated 2011-06-01

Agenda

- Introduction
- USB Background
- Isochronous Transfers
- Connection with UEFI Specification
- Potential Use Cases
- Demonstration
- Further Thoughts
- Questions?

Section Heading

Introduction

Introduction

- USB is a highly utilized bus in all computer systems today
- Typically UEFI USB transfers are done through control and bulk transfers
- These types of transfers limit the devices that can be used by UEFI to simple devices like:
  - Mass storage
  - Mice
  - Keyboards
  - Pointers
  - USB->Serial Adapters
  - CCIDs
- Other device types exist and can be utilized in new ways if their UEFI interfaces are properly implemented

Section Heading

USB Background

Basics of USB Operation

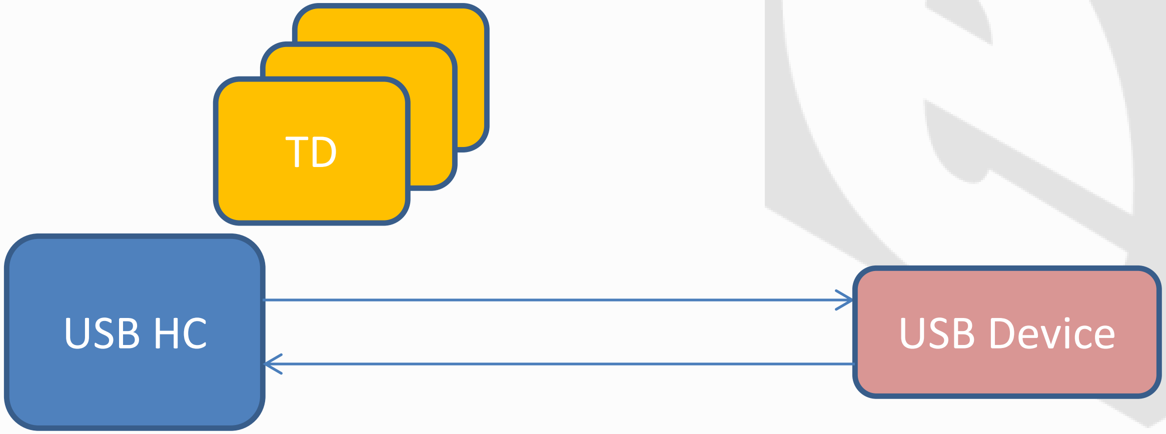
- USB is a serial bus that transfers data one bit at a time at a high clock rate
- USB Devices are connected to USB controllers that preform data transactions to communicate with connected devices
- USB is a polled bus
  - No device initiates a transaction
- When a USB device is connected, the software stack uses the host controller to read device capabilities and initialize the device

Basics of USB Transactions

- Depending on the type of the device, different transaction types are used
- For Mass Storage, the transaction is a one time transfer that goes until completion
- For Input devices like keyboard and mice, devices are checked periodically to see if new data is available
- Transactions are scheduled and completed through something called a Transfer Descriptor (TD)

Transfer Descriptors

- TDs schedule a transfer to complete with a specific device
- TDs are prepared for one time transfers and devices that need to be checked for data again need to setup a new TD when earlier ones complete



Isochronous Transfers

Isochronous Transfers

- Some classes of USB devices transfer large amounts of data at a defined schedule and the previously mentioned transfers do not fit
- Isochronous transfers ensure that data flows at a defined rate so that applications can process it when time is available
- Devices requiring this type of transfer tend to be Audio or Video related

USB Video Class

- USB Video Class is industry standard defined by usb.org
- UVC devices are USB video cameras
- UVC Specification defines the interface to communicate with UVC device
  - Set camera streaming parameters
  - Get video stream

UEFI USB Interface

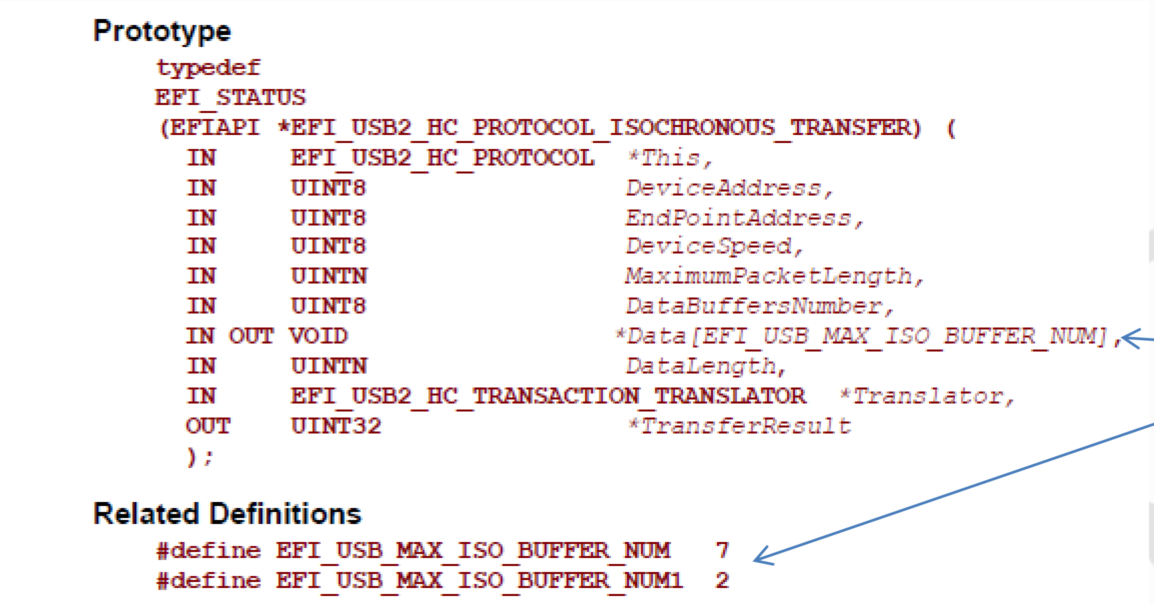
USB IO Interface

```
typedef struct _EFI_USB_IO_PROTOCOL {
    EFI_USB_IO_CONTROL_TRANSFER           UsbControlTransfer;
    EFI_USB_IO_BULK_TRANSFER              UsbBulkTransfer;
    EFI_USB_IO_ASYNC_INTERRUPT_TRANSFER   UsbAsyncInterruptTransfer;
    EFI_USB_IO_SYNC_INTERRPUT_TRANSFER    UsbSyncInterruptTransfer
    EFI_USB_IO_ISOCHRONOUS_TRANSFER       UsbIsochronousTransfer;
    EFI_USB_IO_ASYNC_ISOCHRONOUS_TRANSFER UsbAsyncIsochronousTransfer;
    EFI_USB_IO_GET_DEVICE_DESCRIPTOR      UsbGetDeviceDescriptor;
    EFI_USB_IO_GET_CONFIG_DESCRIPTOR      UsbGetConfigDescriptor;
    EFI_USB_IO_GET_INTERFACE_DESCRIPTOR   UsbGetInterfaceDescriptor;
    EFI_USB_IO_GET_ENDPOINT_DESCRIPTOR    UsbGetEndpointDescriptor;
    EFI_USB_IO_GET_STRING_DESCRIPTOR      UsbGetStringDescriptor;
    EFI_USB_IO_GET_SUPPORTED_LANGUAGES    UsbGetSupportedLanguages;
    EFI_USB_IO_PORT_RESET                 UsbPortReset;
} EFI_USB_IO_PROTOCOL;
```

UEFI API Usage by UVC

- UsbIo ControlTransfer to set up the parameters
  - Focus
  - Brightness
  - Gamma saturation
  - Etc…
- UsbIo IsochronousTransfer to stream the data
- Data output
  - GOP can be used to display the image on the monitor
  - Network stack can be used to transfer the video stream over the network

EHCI Specific Concern



Prototype

```
typedef
EFI_STATUS
(EFIAPI *EFI_USB2_HC_PROTOCOL_ISOCHRONOUS_TRANSFER) (
  IN      EFI_USB2_HC_PROTOCOL   *This,
  IN      UINT8                  DeviceAddress,
  IN      UINT8                  EndPointAddress,
  IN      UINT8                  DeviceSpeed,
  IN      UINTN                  MaximumPacketLength,
  IN      UINT8                  DataBuffersNumber,
  IN OUT  VOID                  *Data[EFI_USB_MAX_ISO_BUFFER_NUM],
  IN      UINTN                  DataLength,
  IN      EFI_USB2_HC_TRANSACTION_TRANSLATOR  *Translator,
  OUT     UINT32                 *TransferResult
  );
```

Related Definitions

```
#define EFI_USB_MAX_ISO_BUFFER_NUM   7
#define EFI_USB_MAX_ISO_BUFFER_NUM1  2
```

Very EHCI Specific

- The specification should make assumptions about controller architecture
- These sorts of information can be hidden from the caller and a simple buffer can be provided with length
  - The implementation can make use of hardware specifics on the back end

Potential Use Cases

Use cases in BIOS?

- Camera can be used to:
  - Check for user presence
  - Do facial recognition for a password
  - Take a photograph if your laptop has been reported stolen

Demonstration

Demonstration

- USB video camera running:
  - In background in Shell
  - In background of Post Screen
  - In background of Setup
- The camera display can also change:
  - Resolution within video screen
  - Location within the video screen

Further Thoughts

Further Development

- USB video cameras do not transfer data in raw BLT format GOP is expecting
  - Video protocols could be created to accept different video buffers types
- Video cards could publish helper protocols to convert video data buffers between formats more quickly than the CPU alone
- Develop useful image processing libraries for user presence and facial recognition

Questions?

Thanks for attending the UEFI Summerfest 2013

For more information on the Unified EFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by

American Megatrends