Introduction to MTP: Media Transfer Protocol

by Steve Kolokowsky and Trevor Davis, Cypress Semiconductor

Moving secure digital content is no simple matter; creating a technology that hides the underlying complexity from the user is even harder. Companies involved in the creation and distribution of digital audio and video content are today being faced with this challenge. In an effort to make development of a secure, feature-rich, user interface on next-generation handheld devices, Microsoft has proposed a new protocol: Media Transfer Protocol, or MTP.

Digital Rights Management (DRM), the protection of licensed or purchased audio or video content, is extremely important in the handheld consumer market. With uncontrolled ability to transport digital content, providers will remain reluctant to release digital material such as movies. With the right protocol protection in place, however, content providers will become comfortable with making their Intellectual Property available in digital form. Microsoft is convinced that MTP is that protocol.

MTP in Detail – Investigating the Layers

Like most modern protocol stacks, MTP can be described in terms of layers. We have broken up the MTP protocol into three layers: 1. Physical, 2. Transport, and 3. Command. These are not the same as the first three layers of the OSI model, but the concepts are the same. Interestingly, the physical transport is not mentioned in the specifications at all. For our purposes, it will be USB. The data transport layer used for MTP is the Picture Transfer Protocol (PTP) specification. Many of the commands are also PTP commands, extended via the vendor-specific extension capability of PTP.

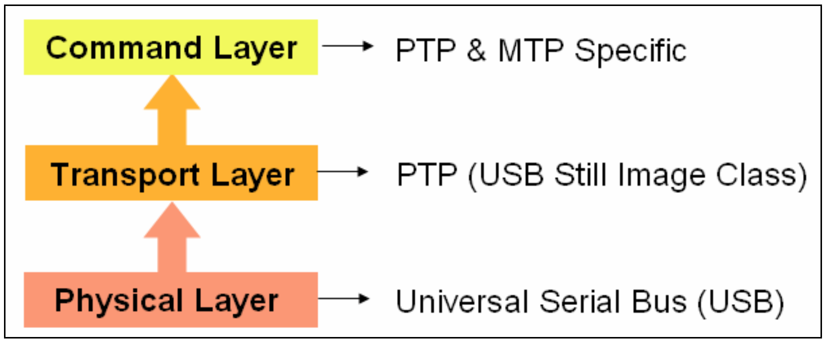


Fig. 1: Three Layers Of MTP

1. Physical Layer

Note: Since PTP and MTP are identical at the physical layer, PTP and MTP can be used interchangeably here. We will use MTP throughout this section for clarity.

USB devices communicate to the host via endpoints. A USB endpoint is an independent communication channel with the host. Each device is required to have one special bi-directional endpoint, the control endpoint, which is labeled endpoint 0. This endpoint is used by the host to control the device and to determine the characteristics of the device. All other endpoints are unidirectional channels with independent ordering and flow control. MTP devices have three endpoints that provide the following communication channels:

- Data OUT of the host (bulk data endpoint)
- Data IN to the host (bulk IN data endpoint)
- Commands (requests) OUT of the host (shares the control endpoint: EP 0)
- Events IN to the host (Interrupt IN endpoint)

The words IN and OUT describe data flow directions in USB. USB is a host-centric protocol: any data flowing in the IN direction is going to the host, data flowing in the OUT direction is going to the device.

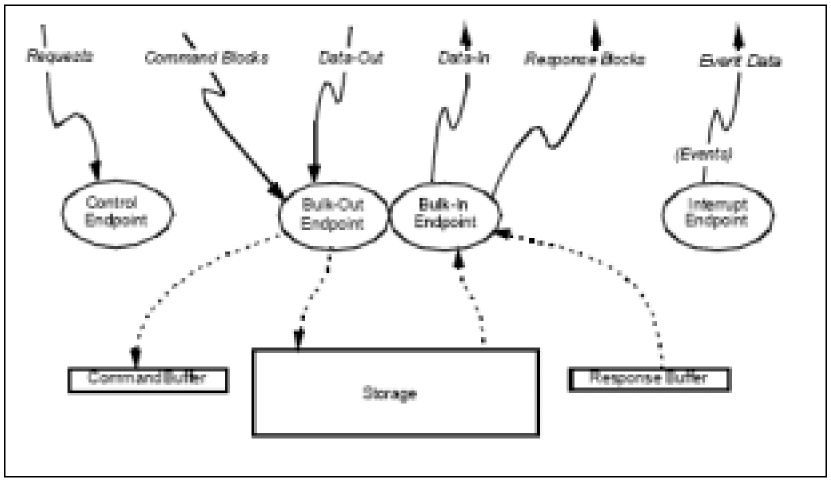


Fig. 2: Endpoint Use In PTP / MTP (Source: PTP Specifications)

2. Transport Layer

The data transport layer of MTP is taken directly from PTP (USB Still Image Class). The USB Still Image Class defines how commands are sent over PTP, how to respond to commands, how to stop a transfer, and how commands are formatted. This specification does not describe the commands, which will be described in the next section.

The USB transport state machine has three main phases: command, data and response with the host always initiating the command phase via the data pipe (could last several packets, especially on older USB 1.1 devices) and all defined commands are less than 64 byte.

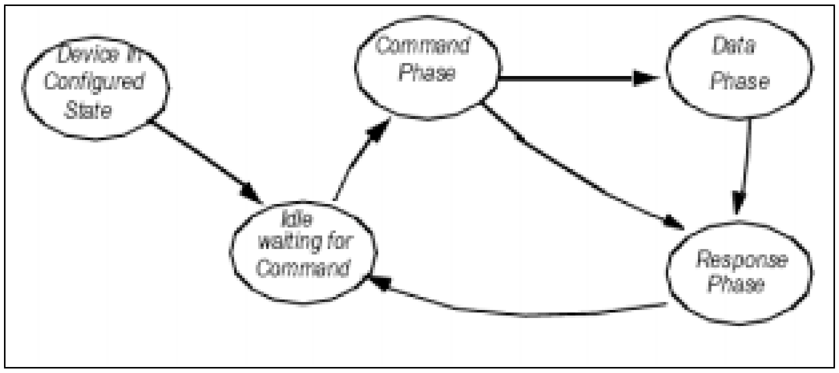


Fig. 3: Data Transport State Machine (Source: PTP specifications)

During the command phase the host will send a single command to the device in containers that always have a fixed format:

Byte Offset	Length (Byte)	Field Name	Description
0	4	Container Length	This field encodes as an unsigned integer the number of bytes in this container. A Still Image Capture Device uses this field to determine the size of the container.
4	2	Container Type	This field describes the type of the container: 0 undefined 1 Command Block 2 Data Block 3 Response Block 4 Event Block
6	2	Code	This field contains the PIMA 15740 OperationCode, ResponseCode, or EventCode. The Data Block will use the OperationCode from the Command Block
8	4	TransactionID	This is a host generated number that associates all phases of an PIMA15740 operation.
12	Command specific	Payload	The contents of this field depend on the operation and phase of the PIMA15740 operation. Because the spec is extensible, this payload could be large..

Table 1: Container Structure (Source PTP specifications)

If a command has a data phase, the IN or OUT data phase follows the command phase. The data is also sent in containers with a Container Type of 2 instead of Container Type 1 for Command Blocks. Finally, the device sends a response indicating the result of the command. The response codes are command specific: they generally include all of the expected outcomes of the command. For example, the allowed response for GetObjectHandles has over ten possible ResponseCodes including OK and Store_Not_Available.

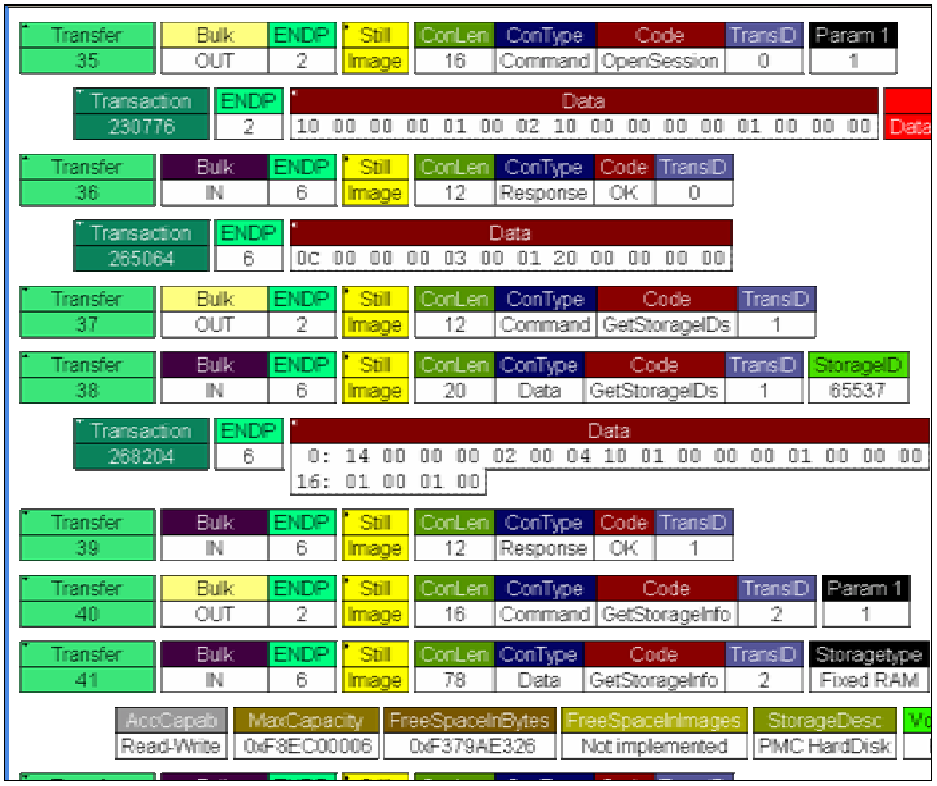
3. Commands

PTP and MTP treat most things on your device as Objects. Most of the commands used in the day-to-day life of your device deal with Objects.

Object command	Purpose
GetNumObjects	Returns a count of the number of objects (songs, files, video clips) on the device
GetObjectHandles	Returns pointers to some or all of the objects
GetObjectInfo	Returns object type, size, format
GetObject	Returns object dataset
GetThumb	Returns summary (thumbnail, audio sample, video clip or frame)
DeleteObject	Bam! It's gone.
SendObjectInfo	Prepare to write object. Allocates space for the object.
SendObject	Host sends data to the device
SetObjectProtection	Write protect enable / disable

Sample Data

Looking at sample traffic is often a very instructive way to get a feel for a protocol. The CATC trace below shows some of the initial communication between a host and device.



```
Transfer | Bulk | ENDP | Still | ConLen | ConType | Code        | TransID | Param 1
35       | OUT  | 2    | Image | 16     | Command | OpenSession | 0       | 1

    Transaction | ENDP | Data                                               | Data
    230776      | 2    | 10 00 00 00 01 00 02 10 00 00 00 00 01 00 00 00

Transfer | Bulk | ENDP | Still | ConLen | ConType  | Code | TransID
36       | IN   | 6    | Image | 12     | Response | OK   | 0

    Transaction | ENDP | Data
    265064      | 6    | 0C 00 00 00 03 00 01 20 00 00 00 00

Transfer | Bulk | ENDP | Still | ConLen | ConType | Code          | TransID
37       | OUT  | 2    | Image | 12     | Command | GetStorageIDs | 1

Transfer | Bulk | ENDP | Still | ConLen | ConType | Code          | TransID | StorageID
38       | IN   | 6    | Image | 20     | Data    | GetStorageIDs | 1       | 65537

    Transaction | ENDP | Data
    268204      | 6    |  0: 14 00 00 00 02 00 04 10 01 00 00 00 01 00 00 00
                         16: 01 00 01 00

Transfer | Bulk | ENDP | Still | ConLen | ConType  | Code | TransID
39       | IN   | 6    | Image | 12     | Response | OK   | 1

Transfer | Bulk | ENDP | Still | ConLen | ConType | Code           | TransID | Param 1
40       | OUT  | 2    | Image | 16     | Command | GetStorageInfo | 2       | 1

Transfer | Bulk | ENDP | Still | ConLen | ConType | Code           | TransID | Storagetype
41       | IN   | 6    | Image | 78     | Data    | GetStorageInfo | 2       | Fixed RAM

        AccCapab   | MaxCapacity | FreeSpaceInBytes | FreeSpaceInImages | StorageDesc   | Vo
        Read-Write | 0xF8EC00006 | 0xF379AE326      | Not implemented   | PMC HardDisk  |
```

Fig. 4: Example MTP Data Traffic

The host is initiating all of the transfers labeled OUT and the device is sending all of the IN traffic. The Transfers are decodes of the data in the Transactions. For example, transfer 35's data is contained in transaction 230776.

- Transfer 35: Here we see the container as it is actually used in practice. The length field (0x10 / 16 decimal) is transmitted LSB to MSB, followed by the Type (Command = 1) and the command code (OpenSession). The MTP transaction ID is 0, indicating that this is the first transaction in the connection
- Transfer 36: This closes the first transaction, indicating to the host that the transaction completed successfully (Code = OK)
- Transfer 37: The host asks for the list of storage elements in the device (GetStorageIDs). For example, a Palm Pilot with an expansion card would probably have two elements, one for main memory and another for the expansion card
- Transfer 38: This is our first example of a Data Container. The device responds that it has a single storage element, numbered 0x1 (logical) and 0x1 (physical). This command is fully decoded in the table below
- Transfer 39: Closes Transaction 1 with an OK code
- Transfer 40: Starts the next MTP transaction

Byte Offset	Length (Byte)	Field Name	Value
0	4	Container Length	0x14 / 20 Decimal
4	2	Container Type	2 = Data
6	2	Code	0x1004 = GetStorageIDs
8	4	TransactionID	1 = This is the second transaction in the session
12		Payload	In this command, the payload is an array of storageIDs. PTP/MTP arrays always start with a 32-bit number representing the size of the array.
12	4	NumElements	1 = Array size is one element
16	4	StorageID	0x00010001 = Physical storage element 1, logical storage element 1.

Understanding the Differences Between MTP Vs PTP

We've spent a lot of time describing the common portion of MTP and PTP because it is 70 - 80% of the implementation effort needed for MTP, but MTP enhances PTP in major and minor ways. Some of the minor enhancements to the protocol enable large changes to functionality.

MTP adds new objects that define collections of objects in new ways. This enables support for playlists, which are essential in any MP3 or media player. MTP also adds support for Palm-type devices by adding calendar, contact and group objects. Of course MTP also adds support for video object types as well.

MTP makes Digital Rights Management (DRM) support easier to implement by adding specific DRM properties including DRM status, URL (to allow DRM to ask for money when the license rights expire), use count and other properties.

MTP also adds extensions to make data handling easier. It extends the PTP transport to permit transfer of large (>4 Gbyte) items. It also allows the container and data to be broken into different USB packets. This allows USB chips (including the Cypress EZ-USB FX2LPTM and EZ-USB SX2TM families) to process the packets more efficiently.

Basic MTP Vs Enhanced MTP

Microsoft has added an additional set of optimizations on top of the basic MTP structure to improve the experience of using MTP devices with Windows. If you're a device (Responder), these functions are included in the royalty-free license from Microsoft. If you're a host (Initiator), these "Enhanced Initiator" functions are licensed separately from the Basic functions under terms that you must negotiate with Microsoft.

The purpose of Enhanced MTP is to allow a host to tightly integrate with an MTP-enabled peripheral for a better end-user experience. Many of the enhanced MTP functions are intended to allow manipulation of multiple objects at once, so that sync performance can be improved.

Legal Issues

If you download the MTP spec from the Microsoft web site, you will find that the first three pages of the specifications are actually an EULA (End User License Agreement). We recommend that you have your lawyers read this license before starting any MTP development. Some highlights of the EULA:

- Basic MTP is available without paying a license fee: "… Microsoft grants you the following limited, non-exclusive, world-wide, <u>royalty-free</u>, non-assignable, nontransferable, non-sublicenseable license" (emphasis added)
- "Enhanced Initiator" functions are not covered by this license. "This Agreement does not grant you a license to build Solutions that implement any of the "Enhanced Initiator" features or functions described in the Specification"
- You must implement the whole specification: "Your Implementations as incorporated into your Solutions must implement the Specification in its entirety"

Conclusion

Ultimately the sole reason for the creation of yet another transport specification is the end user. With increased pressure from the consumer market to make simple, feature-rich handheld devices, and equal or greater pressure from the content creation industry to properly protect the digital content they release, Microsoft remains caught in the middle.

In an attempt to make both ends of the spectrum happy, Microsoft has proposed MTP Understanding the underlying technology is key for the designer to understand the benefits their customers will ultimately enjoy. So, while learning a new protocol might not be the most fun thing you do this year, rest assured that consumers will thank you.

To go beyond the scope of this article and start implementing your own MTP device, you will need several specifications:	
Universal Serial Bus Specification Revision 2.0 http://www.usb.org/developers/docs/.	This is the basic USB specification. It governs the connectors, cables, electrical characteristics and basic communication structure of USB.
USB Still Image Capture Device Definition http://www.usb.org/developers/docs/	This spec defines the transport layer for USB.
PIMA 15740:2000: Photography – Electronic still picture imaging - Picture Transfer Protocol (PTP) www.pima.net	PIMA merged with the DIG (Digital Imaging Group) of i3a (International Imaging Industry Association) in early 2001. The www.pima.net website is no longer associated with the DIG or PIMA. The PTP specs are available for a fee from www.i3a.org. The fee is waived for .edu and .gov individuals. This spec describes the basic command set.
Media Transfer Protocol http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwmt/html/mtp_spec.asp	Describes the commands added to PTP to create MTP

References

Media Transfer Protocol; Manders & Rosenbloom, Microsoft Corp. WinHEC 2004
All specifications listed above

About The Authors

Trevor Davis is currently the Director of Marketing for Cypress's Consumer and Computation Division focused on USB in consumer products. Trevor received his undergraduate degree from the US Air Force Academy and also has an MBA. He served as an Air Force Officer for 5 years before joining Cypress. Trevor lives with his wife, Yvonne, and two children, Morgan and Ryann, in San Diego, CA. tmd@cypress.com

Steve Kolokowsky is a Member of the Technical Staff in Cypress Semiconductor's Consumer and Computation Division. His focus is high-speed USB peripheral products and he has participated in the design of many of today's highest-selling MP3 players. Steve earned a BS in Systems Engineering from Rensselaer Polytechnic Institute in Troy, NY. He is based in sunny San Diego, California, USA. syk@cypress.com

as published in...
analogZONE