MCCI Corporation
3520 Krums Corners Road
Ithaca, New York 14850 USA
Phone +1-607-277-1029
Fax +1-607-277-6844
www.mcci.com

MCCI Composite Device User's Guide

Engineering Report 950000684
Rev. B
Date: 2011/09/30

PROPRIETARY NOTICE AND DISCLAIMER

Unless noted otherwise, this document and the information herein disclosed are proprietary to MCCI Corporation, 3520 Krums Corners Road, Ithaca, New York 14850 ("MCCI"). Any person or entity to whom this document is furnished or having possession thereof, by acceptance, assumes custody thereof and agrees that the document is given in confidence and will not be copied or reproduced in whole or in part, nor used or revealed to any person in any manner except to meet the purposes for which it was delivered. Additional rights and obligations regarding this document and its contents may be defined by a separate written agreement with MCCI, and if so, such separate written agreement shall be controlling.

The information in this document is subject to change without notice, and should not be construed as a commitment by MCCI. Although MCCI will make every effort to inform users of substantive errors, MCCI disclaims all liability for any loss or damage resulting from the use of this manual or any software described herein, including without limitation contingent, special, or incidental liability.

MCCI, TrueCard, TrueTask, MCCI Catena, and MCCI USB DataPump are registered trademarks of MCCI Corporation.

MCCI Instant RS-232, MCCI Wombat and InstallRight Pro are trademarks of MCCI Corporation.

All other trademarks and registered trademarks are owned by the respective holders of the trademarks or registered trademarks.

NOTE: The code sections presented in this document are intended to be a facilitator in understanding the technical details. They are for illustration purposes only, the actual source code may differ from the one presented in this document.

Document Release History

Rev. A	2011/03/25	Original release
Rev. B	2011/09/30	Added source code disclaimer.

TABLE OF CONTENTS

1 Introduction **1**

1.1 Purpose **1**

1.2 Referenced Documents **1**

2 Composite Device Driver Overview **1**

3 Initializing the Composite Device Driver **2**

3.1 Initializing the Composite Device Driver in your application **2**

2 Composite Device Driver Memory Requirements **3**

LIST OF TABLES

Table 1. Composite Device Driver Memory Requirements 4

1 Introduction

1.1 Purpose

The purpose of this document is to describe the functionality of the Composite Device Driver, how to initialize it and what the memory requirements are for various Composite Device configurations.

1.2 Referenced Documents

[EUSBDI] *MCCI USB Datapump Embedded USBDI*, MCCI Engineering report 950000325

[USBCORE] *Universal Serial Bus Specification*, version 2.0/3.0 (also referred to as the USB Specification), with published erratas and ECOs. This specification is available on the World Wide Web site http://www.usb.org/.

[DPUSERGUIDE] *MCCI USB Datapump User's Guide,* MCCI Engineering Report 950000066

[TTUSERGUIDE] *MCCI Transaction Translator User's Guide,* MCCI Engineering Report 950000548

2 Composite Device Driver Overview

The MCCI Composite Device Driver adds support for composite (multi-function) USB devices to the MCCI USB Datapump Embedded Host stack.

As defined by the USB core specification [USBCORE] section 5.2.3, a composite device is a "device that has multiple interfaces controlled independently of each other". In the MCCI Embedded Host stack, each composite device is divided up into groups of interfaces related by function, and then the usual class drivers are loaded on a function-by-function basis.

MCCI's composite driver has the following characteristics:

- Supports composite devices that use the Interface Association Descriptor for functional grouping of interfaces
- Supports composite devices that use Audio Class 1.0 descriptors for functional grouping of audio-related interfaces
- Supports composite devices that use Communication Device Class UNION descriptors for functional grouping of interfaces
- Supports composite devices that use MCPC GL-004/005 UNION descriptors for functional grouping of interfaces (not yet implemented)

- Supports composite devices that assign separate functions to each interface
- Allows use of properly-coded standard class drivers without modification. For example, the MCCI Mass Storage and HID class drivers can be used with Composite devices without modification
- Allows the system integrator to choose how much memory is to be reserved for supporting composite devices
- Operates automatically; after initialization, the composite driver requires no run-time interaction with other user-supplied modules

3 Initializing the Composite Device Driver

Before initializing the Composite Device Driver, you must first ask yourself the following questions:

1. How many Composite Devices do I wish to support?
2. How many Composite Functions across all Composite Devices do I wish to support?
3. What is the maximum number of Sub-Functions per Function I wish to support?
4. What is the maximum number of pipes in a supported Composite Device?
5. What is the maximum size of the configuration descriptor bundle in a supported Composite Device?

When a Composite Device is attached, a Composite Device Driver instance is started. The Composite Device Driver instance reads the configuration bundle and identifies Composite Functions. A Function Class Driver instance is started for each Composite Function identified. A Composite Function maps one-to-one to an interface descriptor or collection of interface descriptors.

If a Composite Function maps to a collection of interface descriptors, the subordinate interfaces are referred to as Sub-Functions. For example, for a Communications function, the configuration bundle contains two interfaces, Communcation Control and Communication Data. In this case, one Composite Function and one Composite Sub-Function would be required.

To initialize the Composite Device Driver you just need to modify your application as described in the next section.

3.1 Initializing the Composite Device Driver in your application

To initialize the Composite Device Driver you must define the configuration in the DataPump initialization code of your application. (This is the code that is normally launched by the "finish

function" in your application's appinit.c file.) For more details on how to define device configuration please refer to [DPUSERGUIDE]. Note that the Composite Device Driver must be initialized from within the same task as the rest of the Embedded Host Stack. Initialization of the Transaction Translator and the memory requirements for various configurations are described in [TTUSERGUIDE].

```
static CONST USBPUMP_USBDI_CLASS_COMPOSITE_CONFIG sk_UsbPumpUsbdCD_Config =
USBPUMP_USBDI_CLASS_COMPOSITE_CONFIG_INIT_V1(
            &gk_UsbPumpUsbdiClassComposite_InitMatchList,
            /* driver class name */ NULL, /* use implementation default */
            /* function instance name */ NULL, /* use default */
            /* number of instances */ 8,
            /* number of Composite Functions Total */ 64,
            /* number of Composite Sub Function per Function */ 32,
            /* number of Pipes */ 12,
            /* size of Configuration Desc */ 512
            );
```

Once you have defined the configuration parameters, you must call the API to initialize the Composite Device Driver. This API must be called after the API to initialize the USBD (after UsbPumpUsbd_Initialize()). It can be called before or after you initialize the Transaction Translator.

```
if (! UsbPumpUsbdiClassComposite_Initialize(
          &ErrorCode,
          &pUsbdObject->ObjectHeader,
          &sk_UsbPumpUsbdCD_Config.ClassConfig,
          &sk_UsbPumpUsbdCD_Config.ClassPrivateConfig
          ))
          {
          TTUSB_OBJPRINTF((
              &pHcd->ObjectHeader, UDMASK_ERRORS,
              "-Combo_InitFinish: "
              "UsbPumpUsbdCD_Initialize failed to initialize Composite Device support.\n"
              ));
          }
```

2 Composite Device Driver Memory Requirements

Below is the equation to calculate the memory requirements. The equation returns the number of bytes required. The equation is specific to the Catena platform and the Microsoft Visual C 6.0 compiler, but is typical of memory use on 32-bit platforms.

RequiredMemory =
[20 + **/* USBD CD overhead */**
114 + (NumInstances * (2156 + (2 *SizeConfigDesc) + (84 * NumberPipes))) + **/* CD instances */**
(NumberCFs * 336) + **/* Composite Ports */**
(((NumberCFs * (NumberCSFs + 1)) + 1) * 68)] **/* Function Nodes */**

The table below shows the memory requirements for some possible configurations of the Composite Driver. The first configuration could be used to support one Composite Device containing a keyboard and mouse function. The second provides more general Composite Device support.

In this release, it is not possible to configure the Composite driver to request and release memory dynamically – all required memory is allocated during initialization.

Table 1. Composite Device Driver Memory Requirements

Configuration	Variable	Comment	Required Memory
Single keyboard+mouse	NumInstances = 1	Only want to support one instance	3,462 bytes
	NumberCFs = 2	Max number of functions across all composite devices is 2	
	NumberCSFs = 0	No multi-interface functions	
	NumberPipes = 2	Two pipes per instance	
	SizeConfigDesc = 64	We don't expect any configuration descriptors that are larger than 64 bytes.	
General-purpose	NumInstances = 8	Support 8 instances of composite device at the same time	64,046 bytes
	NumberCFs = 32	Max number of functions across all composite devices is 32	
	NumberCSFs = 8	Each function could have up to 8 extra interfaces	
	NumberPipes = 12	Up to twelve pipes per instance	
	SizeConfigDesc = 512	We don't expect any configuration descriptors that are larger than 512 bytes.	