

Medical Connectivity Library Users Guide

Document Number:MEDCONLIBUG Rev. 5 05/2012



NP

How to Reach Us:

Home Page: www.freescale.com

E-mail: support@freescale.com

USA/Europe or Locations Not Listed:

Freescale Semiconductor Technical Information Center, CH370 1300 N. Alma School Road Chandler, Arizona 85224 +1-800-521-6274 or +1-480-768-2130 support@freescale.com

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH Technical Information Center Schatzbogen 7 81829 Muenchen, Germany +44 1296 380 456 (English) +46 8 52200080 (English) +49 89 92103 559 (German) +33 1 69 35 48 48 (French) support@freescale.com

Japan:

Freescale Semiconductor Japan Ltd. Headquarters ARCO Tower 15F 1-8-1, Shimo-Meguro, Meguro-ku, Tokyo 153-0064, Japan 0120 191014 or +81 3 5437 9125 support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor China Ltd. Exchange Building 23F No. 118 Jianguo Road Chaoyang District Beijing 100022 China +86 10 5879 8000 support.asia@freescale.com

For Literature Requests Only:

Freescale Semiconductor Literature Distribution Center 1-800-441-2447 or 303-675-2140 Fax: 303-675-2150 LDCForFreescaleSemiconductor@hibbertgroup.com Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the



Freescale[™] and the

Freescale logo are trademarks of Freescale Semiconductor, Inc. ARC, the ARC logo, ARCangel, ARCform, ARChitect, ARCompact, ARCtangent, BlueForm, CASSEIA, High C/C++, High C++, iCon186, MetaDeveloper, MQX, Precise Solution, Precise/BazeNet, Precise/EDS, Precise/MFS, Precise/MQX, Precise/MQX Test Suites, Precise/RTCS, RTCS, SeeCode, TotalCore, Turbo186, Turbo86, V8 μ RISC, V8 microRISC, and VAutomation are trademarks of ARC International. High C and MetaWare are registered under ARC International.

The PowerPC name is a trademark of IBM Corp. and is used under license. The described product contains a PowerPC processor core. The PowerPC name is a trademark of IBM Corp. and used under license. The described product is a PowerPC microprocessor. The PowerPC name is a trademark of IBM Corp. and is used under license. The described product is a PowerPC

microprocessor core. The PowerPC name is a trademark of IBM Corp. and is used under license. All other product or service names are the property of their respective owners.

© 1994-2008 ARC[™] International. All rights reserved.

© Freescale Semiconductor, Inc. 2010. All rights reserved.

Document Number: MEDCONLIBUG Rev. 5



Revision History

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://www.freescale.com

The following revision history table summarizes changes contained in this document.

Revision Number	Revision Date	Description of Changes	
Rev. 1 10/2009 Initial release.		Initial release.	
Rev. 2	04/2010	Updated Figure 2-1, Figure 4-1, and Section A.1.1.1, "Software Setup."	
Rev. 3	06/2010	Added support for CFV2 devices	
Rev. 4	07/2011	Updated images in Appendix A	
Rev. 5	05/2012	Added chapters IEEE 11073 Manager PHDC Manager Demo Applications 	

Freescale[™] and the Freescale logo are trademarks of Freescale Semiconductor, Inc. © Freescale Semiconductor, Inc., 2010. All rights reserved.



Chapter 1 Before You Begin

1.1	About Medical Connectivity Library	1-1
1.2	About This Book	1-1
1.3	Reference Material	1-2
1.4	Acronyms and Abbreviations	1-3
1.5	Important Terms	1-3

Chapter 2 Getting Familiar

2.1	Introduction	. 2.	-1
2.2	Software Suite	. 2.	-1
2.3	Directory Structure	. 2.	-1

Chapter 3 Medical Connectivity Library Architecture

3.1	Architecture Overview
3.2	Software Flows
	3.2.1 Initialization Flow
	3.2.2 Association Flow
	3.2.3 Send Measurement Data Flow
	3.2.4 Send Person Measurement Flow

Chapter 4 Developing New Device Specialization

4.1	Introduction
4.2	Directory Structure
4.3	Medical Connectivity Library Interfaces
	4.3.1 Data Proto List
	4.3.2 DIM Variable
	4.3.3 MDS Object
	4.3.4 Device Configuration
	4.3.5 Metric Object
	4.3.6 Numeric Object
	4.3.7 Real-Time Sample Array (RT-SA) Object 4-9
	4.3.8 Enumeration Object 4-9
	4.3.9 Scanner
	4.3.10 Configurable Scanner 4-10
	4.3.11 Episodic Configurable Scanner
	4.3.12 Periodic Configurable Scanner 4-11
	4.3.13 PM Store Object
	4.3.14 PM Segment Object 4-13
4.4	Application Design
	4.4.1 Main Application Function 4-14
	4.4.2 Application Task
	4.4.3 Callback Function

Chapter 5 IEEE 11073 Manager

5.1	Introduction						5-1
-----	--------------	--	--	--	--	--	-----

MEDCONLIB Users Guide, Rev. 5



5.2	Folder structure	5-1
5.3	Module Usage/Aim	5-2
5.4	Functional description	5-2
	5.4.1 MQX and Bare-Metal System Interraction	5-2
	5.4.2 MQX Host API	5-3
	5.4.3 MQX Chapter 9 Services	5-4
	5.4.4 MQX Common-class Services	5-4
5.5	System Decomposition	5-5

Appendix A Working with the Software

A.1	Introduction
	A.1.1 Preparing the Setup A-1
	A.1.1.1Software SetupA-1
	A.1.1.2Hardware SetupA-5
	A.1.2 Building the Application
	A.1.3 Running the Application
A.2	Uninstall Freescale Medical Connectivity Library 1.0 Software
A.3	Important Files

Appendix B PAN USB Agent Demo

B.1	Setting Up the Demo	. B-1
B.2	Running the Demo	. B-1

Appendix C PAN Serial Bridge Demo

C.1	Setting Up the Demo	C-1
C.2	Running the Demo	C-2

Appendix D PHDC Manager Demo Application

D.1	Setting up the demo	C-1
	D.1.1 Hardware setup	
	D.1.2 Set up HyperTerminal to get log	C-1
	D.1.3 Running the demo	C-5



Chapter 1 Before You Begin

1.1 About Medical Connectivity Library

The increased use of electronic devices in the medical domain has created a need for the development of common protocol for data interchange between various devices. This problem is addressed by IEEE-11073 Part 20601 (Optimized Exchange) specification. Freescale Medical Connectivity Library is based on this specification. This library follows Continua design guidelines for implementation.

Freescale Medical Connectivity Library allows customers to implement various device specializations defined under this standard without worrying about details of implementation of this protocol. The library is transport independent, allowing users to use different transport technologies like Ethernet, USB, and so on.

1.2 About This Book

This book describes the Freescale Medical Connectivity Library architecture and also explains you how to create customized applications using this library. Table 1-1 shows the summary of chapters included in this book.

Chapter Title	Description
Before you begin	This chapter provides the prerequisites of reading this book.
Getting Familiar	This chapter provides the information about the Freescale Medical Connectivity Library software suite.
Medical Connectivity Library Architecture	This chapter discusses the architecture design of Freescale Medical Connectivity Library software suite.
Developing New Device Specialization	This chapter discusses the steps a developer must take to develop new device specialization applications using Freescale Medical Connectivity Library.
Working with Software	This chapter provides information on how to build, run, and debug drivers and applications.
PAN USB Agent Demo	This chapter provides the setup and running PAN USB Agent demo for MC9S08JM60 and MCF51JM128 devices.

Table 1-1. MEDCONLIBUG Summary



Before You Begin

1.3 Reference Material

Use this book in conjunction with:

- Medical Connectivity Library API Reference Manual (document MEDCONLIBAPIRM)
- *Freescale USB Stack with PHDC API Reference Manual* (document MEDUSBAPIRM)
- *Freescale USB Stack with PHDC Users Guide* (document MEDUSBUG)

For better understanding, also refer to the following documents.

- Continua Design Guidelines (document ContinuaV1_DG_HL7_R1.pdf)
- IEEE Std 11073-20601TM-2008, Health informatics Personal health device communication Part 20601: Application profile Optimized Exchange Protocol.
- IEEE P11073-10441TM, Health informatics Personal health device communication Part 10441: Device specialization Cardiovascular fitness and activity monitor.
- IEEE P11073-10442TM, Health informatics Personal health device communication Part 10442: Device specialization Strength fitness equipment.
- IEEE Std 11073-10408TM, Health informatics Personal health device communication Part 10408: Device specialization Thermometer.
- IEEE Std 11073-10415TM, Health informatics Personal health device communication Part 10415: Device specialization Weighing scale.
- IEEE Std 11073-10471TM, Health informatics Personal health device communication Part 10471: Device specialization Independent living activity hub.
- ISO/IEEE P11073-10404, Health informatics Personal health device communication Part 10404: Device specialization Pulse oximeter.
- ISO/IEEE P11073-10407, Health informatics Personal health device communication Part 10407: Device specialization Blood pressure monitor.
- ISO/IEEE P11073-10417, Health informatics Personal health device communication Part 10417: Device specialization Glucose meter.
- ISO/IEEE 11073-10101, Health informatics Point-of-care medical device communication Part 10101: Nomenclature.
- ISO/IEEE 11073-10201:2004, Health informatics Point-of-care medical device communication Part 10201: Domain information model.



1.4 Acronyms and Abbreviations

CFV1	ColdFire V1 (MCF51JM128 CFV1 device is used in this document)
CFV2	ColdFire V2 (MCF52221and MCF52259 CFV2 devices are used in this document)
DIM	Domain Information Model
IDE	Integrated Development Environment
IEEE	The Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
JM60	MC9S08JM60 Device
MDS	Medical Device System
PHD	Personal Healthcare Device
PHDC	Personal Healthcare Device Class
RTSA	Real Time Sample Array
TIL	Transport Independent Layer
USB	Universal Serial Bus

1.5 Important Terms

Table 1-2 shows the terms used throughout the book.

Table 1-2. Important Terms

Term	Description
Continua Alliance	This is a consortium of companies to establish standards for the medical segment devices.
DemoJM	This is the physical hardware where the expansion card with the silicon is mounted.
Expansion Card	This is the card where the silicon is embedded and can be loaded on to the hardware board.
Shim	Transport used for Data Send/Receive.
PM Store	Persistent Metric Store
PM Segment	Persistent Metric Segment



Chapter 2 Getting Familiar

2.1 Introduction

The Freescale Medical Connectivity Library software suite contains Medical Connectivity Library (designed based on IEEE 11073-20601 specification and Continua Design guideline), USB Transport Layer (based on USB-PHDC Specification), and a sample application. This section intends to help you develop an understanding of the Medical Connectivity Library and to assist you in developing more device specializations as defined by the standard. The document is targeted for firmware application developers who would like to develop the applications using this library.

2.2 Software Suite

This suite contains Freescale Medical Connectivity Library, USB Transport Layer, and a sample application for JM60, CFV1, and CFV2 devices.

2.3 Directory Structure

The software suite has a standard directory structure. You can extend it easily to accommodate more applications and Transport Layers.



Getting Familiar

The following figure shows the directory structure.

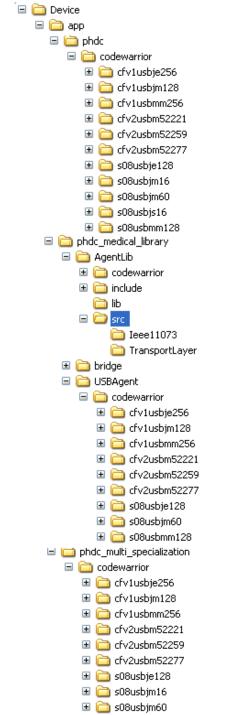


Figure 2-1. Freescale Medical Connectivity Library Directory Structure



Chapter 3 Medical Connectivity Library Architecture

3.1 Architecture Overview

Figure 3-1 shows the Freescale Medical Connectivity Library architecture.

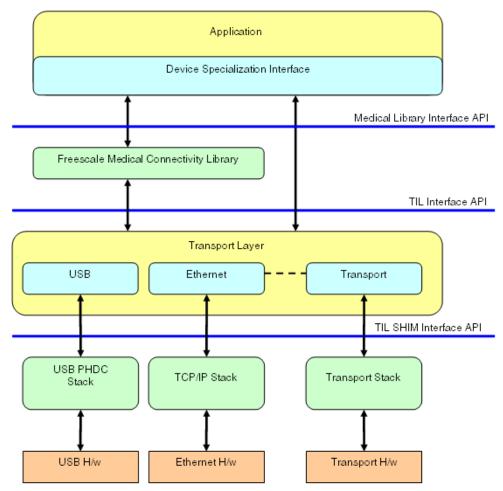


Figure 3-1. Freescale USB Stack with PHDC Architecture

Medical Connectivity Library is divided into two layers with application developed on top of them. The layered architecture helps the application developers concentrate on developing the application without being concerned about the other layers.

Transport Layer interacts with underlying Shim to provide reliable data communication to upper layer. This layer abstracts application and Medical Connectivity Library from details of how the data is sent or received through Shim. This layer interacts with underlying Shims with its defined interfaces, therefore allowing users to add different transport technologies as required.



Medical Connectivity Library Architecture

Freescale Medical Connectivity Library Layer implements IEEE11073-20601 Standard (based on Continua guidelines). Freescale Medical Connectivity Library Layer defines interfaces for Device Specialization (part of Application Software) layer to communicate with Continua Manager. Device Specialization layer uses interfaces provided by this layer to send measurement and configuration data to Continua Manager.

As stated earlier, the layered architecture helps the application developers to develop applications. However, it does not limit the developer to interface lower layer APIs if they prefer to.

CAUTION

Simultaneous use of driver APIs and class APIs may have undefined behavior. In this case, the driver functionality will not work as defined in this document. Functioning of class drivers depends upon lower layer USB device controller state and USB bus state machine. If application invokes lower layer functions directly, then class driver state machine might get affected, leading to undefined behavior. However, this does not necessitates application to use class driver only. You can develop its application directly using lower layer driver APIs. In such a case, application needs to take care of class layer functionality.

3.2 Software Flows

This section describes the execution flow of the stack across various layers and modules.

3.2.1 Initialization Flow

Figure 3-2 describes application initialization flow.

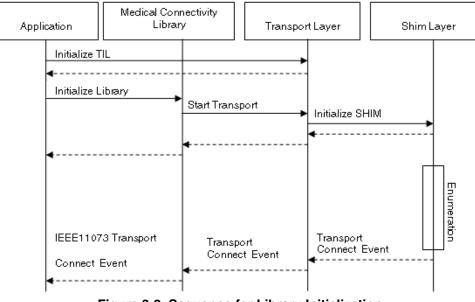


Figure 3-2. Sequence for Library Initialization



The application initialization sequence starts by initializing Transport Independent Layer (TIL). Using this interface, application registers various Shims that can be used as transport. After successful TIL initialization, application initializes Medical Connectivity Library with a callback and Shim Identification Code. This internally initializes Shim to send and receive data.

After Shim is initialized, enumeration process begins. After successful initialization, "Transport Connect" event is sent to TIL, which is passed to the Medical Connectivity Library and then to the application.

3.2.2 Association Flow

Figure 3-3 describes association process flow.

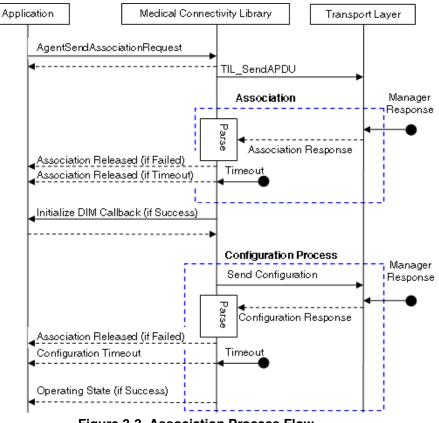


Figure 3-3. Association Process Flow

After receiving Transport Connect Event, application can start Association Process. Application uses AgentSendAssociationRequest interface to send Association Request to Continua Manager. If the request is successful, application receives a callback to initialize DIM. Once DIM is initialized, configuration process begins. This process is initiated by Medical Connectivity Library. After configuration process is successfully completed, application is given a callback Operating State. After this callback event is received, application can now send measurement data to Continua Manager.

MEDCONLIB Users Guide, Rev. 5



Medical Connectivity Library Architecture

3.2.3 Send Measurement Data Flow

Figure 3-4 and Figure 3-5 below describes how application can send confirmed/un-confirmed measurement data to Continua Manager.

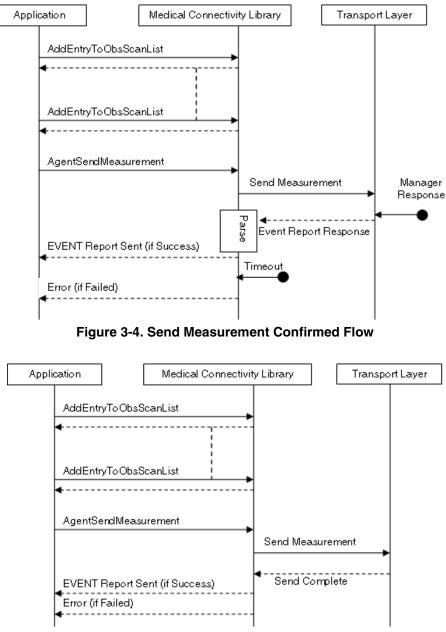


Figure 3-5. Send Measurement Un-Confirmed Flow

To send measurement data to Continua Manager, application creates an ObservationScanList. Application then calls AgentSendMeasurements interface to send data to Continua Manager. Application is informed by a callback event (EVENT Report Sent) if successful, otherwise an error is reported in the callback. Application should wait for these callback events before attempting to send another measurement.



To prepare ObservationScanList, AddEntryToObsScanList interface is provided to the application. This abstracts the internal details of this structure from the application.

3.2.4 Send Person Measurement Flow

Figure 3-6 and Figure 3-7 below describes how application can send confirmed/un-confirmed person measurement data to Continua Manager.

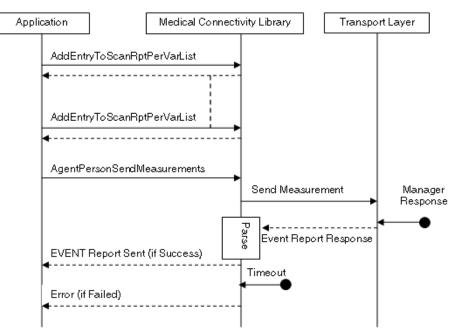


Figure 3-6. Send Person Measurement Confirmed Flow

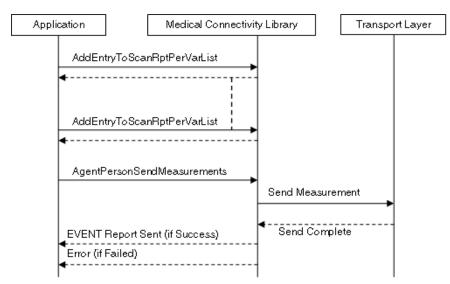


Figure 3-7. Send Person Measurement Un-Confirmed Flow

To send measurement data to Continua Manager, application creates a ScanReportPerVarList. Application then calls AgentSendPersonMeasurements interface to send data to Continua Manager. Application is



Medical Connectivity Library Architecture

informed by a callback event (EVENT Report Sent) if successful, otherwise an error is reported in the callback. Application should wait for these callback events before attempting to send another measurement.

To prepare ScanReportPerVarList, AddEntryToScanRptPerVarList interface is provided to the application. This abstracts the internal details of this structure from the application.



Chapter 4 Developing New Device Specialization

4.1 Introduction

This chapter discusses how you can create new device specialization applications based on Medical Connectivity Library.

4.2 Directory Structure

The following figure shows the directory structure of the PAN USB Agent application.

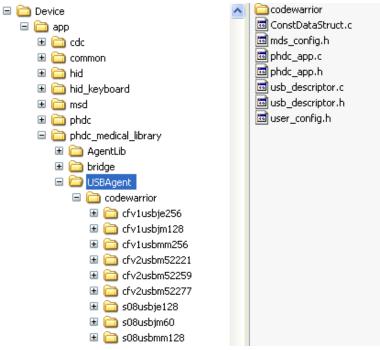


Figure 4-1. PAN USB Agent Application Directory Structure

For developing a new device specialization, change the following pre-existing files:

- ConstDataStruct.c
- phdc_app.c
- phdc_app.h



Developing New Device Specialization

4.3 Medical Connectivity Library Interfaces

This section discusses the structures of the device specialization defined in the file ConstDataStruct.c.

4.3.1 Data Proto List

Sample Structure:

```
/* Data Proto List */
const intu8 g DataProtoList[] =
{
      0x00, 0x01,
      0x00, 0x30,
      /* Data Proto ID = 20601 */
      0x50, 0x79,
      /* Data Proto Length */
      0x00, 0x2c,
      /* PROTOCOL_VERSION1 */
      0x80, 0x00, 0x00, 0x00,
      /* Encoding rules */
      0x80, 0x00,
      /* NOM VERSION1 */
      0x80, 0x00, 0x00, 0x00,
      /* Functional units */
      0x00, 0x00, 0x00, 0x00,
                               /* Device Cannot Enter Test Association */
      /* SYS TYPE AGENT */
      0x00, 0x80, 0x00, 0x00,
      /* system id */
      0x00, 0x0e,
      'F','S','L','M','E','D','I','C','A','L','O','O','1', 0x0,
      /* dev config id */
      0x40, 0x00, /* --- user modifiable */
      /* data_req_mode_capab */
      0x08, 0x81, /* --- user modifiable */
      /* maximum number of parallel agent initiated data requests*/
      MAX AGENT DATA COUNT, /* --- user modifiable */
      /* maximum number of parallel manager initiated data requests*/
      MAX MANAGER DATA COUNT,
      /* optionList */
      0x00, 0x00, 0x00, 0x00
};
```

In the above structure, the user modifiable parameters are marked in red color. The device configuration Id for standard configuration is in the range 0x0001 - 0x3FFF (both inclusive). These configurations are fixed. If you want to create a configuration apart from the standard configuration, the configuration Id should be in the range 0x4000 - 0x7FFF (both inclusive). These configurations are called extended configurations. You can change data request mode capability flags depending on the requirement. The maximum number of parallel agent initiated data requests can be either 0 or 1.

NOTE

• Library does not support manager initiated data transmission (as per Continua Design guidelines).



• If agent initiated data transmission is not supported, at least one Scanner should be a part of the configuration.

4.3.2 DIM Variable

Sample Structure:

```
/* DIM variable */
DIM g_DIM =
{
    /* Pointer to MDS */
    &g_Mds, /* --- user modifiable */
    /* Device Configuration ID */
    EXTENDED_CONFIG_START /* --- user modifiable */
};
```

In the above structure, the user modifiable parameters are marked in red color. &g_Mds is a pointer to MDS which is the top-most object. The device configuration Id for standard configuration is in the range 0x0001 - 0x3FFF (both inclusive). These configurations are fixed. If you want to create a configuration apart from the standard configuration, the configuration Id should be in the range 0x4000 - 0x7FFF (both inclusive). These configurations are called extended configurations.

4.3.3 MDS Object

Medical Device System (MDS) represents the topmost object. Each device has one MDS class. The MDS represents the identification and status of the device through its attributes. Each class has a nomenclature code for its identification and some set of attributes. Out of all the attributes, some attributes are mandatory to implement (if that class is instantiated), and some are conditional that is, they should be present if the required condition is met. The rest of the attributes are optional. Each object has a "Handle" attribute that is used to identify the object for operations. The value of this attribute is unique. For MDS object, the handle is fixed to zero.

Sample Structure:

```
/* MDS Object */
MDS g Mds =
{
      /* MDS Handle */
      0x0000, /* --- not modifiable */
      /* Svs type */
      #ifdef MDS ATTR_SYS_TYPE
      {
          4224,
          8136
      },
      #else
          /* Type Per Var List */
          TypeVerList*)&g MdcAttrSysTypeSpecList,
      #endif /* MDS ATTR SYS TYPE */
          /* System Model */
          (SystemModel*)&g MdcAttrSysModel,
          /* System Id */
```

```
NP
```

Developing New Device Specialization

```
(octet_string*)&g_MdcAttrSysId,
    /* device Configuration ID */
    0x4000,
    /* pointer to attribute val map */
#ifdef MDC FIXED DATA FORMAT
    (AttrValMap*)&g_MdcAttrValMap,
#endif /* MDC FIXED DATA FORMAT */
/* pointer to production spec */
#ifdef MDS PROD SPEC
    (ProductionSpec*) &g_MdcAttrIdProdSpecn,
#endif /* MDS ABSOLUTE TIME */
/* Mds Time Info */
#ifdef MDS SETTABLE TIME
{
    #ifdef MDS MGR SET TIME
        MDS_TIME_MGR_SET_TIME |
    #endif
    #ifdef MDS HIRES TIME
        MDS TIME CAPAB HIGH RES RELATIVE TIME |
    #endif
    #ifdef MDS RELATIVE TIME
        MDS TIME CAPAB RELATIVE TIME |
    #endif
    #ifdef MDS SET CLOCK
        MDS TIME CAPAB SET CLOCK |
    #endif
    #ifdef MDS RTC
        MDS_TIME_CAPAB_REAL_TIME_CLOCK |
    #endif
    0),
    MDC TIME SYNC NONE,
    TIME SYNC ACCURACY UNKNOWN,
    TIME RESOLUTION UNKNOWN,
                                /* Absolute Time Resolution */
    TIME RESOLUTION UNKNOWN,
                                /* Relative Time Resolution */
                                 /* High Res Time Resolution */
    TIME RESOLUTION UNKNOWN
},
#endif /* MDS SETTABLE TIME */
/* Absolute Time Stamp */
#ifdef MDS ABSOLUTE TIME
{
    0x20,
            /* Century */
            /* Year */
    0x09,
    0x07,
            /* Month */
    0x22,
            /* Day */
    0x19,
            /* Hour */
    0x30,
            /* Minute */
    0x10,
            /* Second */
    0x00
            /* Second Fraction */
},
#endif /* MDS_ABSOLUTE_TIME */
/* Relative Time */
#ifdef MDS RELATIVE TIME
```



```
0x0000000,
#endif /* MDS ABSOLUTE TIME */
/* Hi Resolution Relative Time */
#ifdef MDS HIRES TIME
{
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
},
#endif /* MDS ABSOLUTE TIME */
/* Absolute Date and Time Adjust */
#ifdef MDS ADJUST DATE TIME
{
    0x00, 0x00, 0x00, 0x00, 0x00,0x00
},
#endif /* MDS ABSOLUTE TIME */
/* Power Status */
#ifdef MDS POWER STATUS
    ON MAINS,
#endif /* MDS POWER STATUS */
/* Battery Charge */
#ifdef MDS BATT CHARGE
    /* 100% Charge */
    100,
#endif /* MDS BATT CHARGE */
/* Battery Measure */
#ifdef MDS BATT REMAIN
{
    /* 1 Day of Battery Remaining */
    0x1,
    MDC DIM DAY
},
#endif /* MDS BATT REMAIN */
    /* pointer to Reg Certified Data List */
    (RegCertDataList *)&g MdcAttrRegCertDataList,
/* Confirm Timeout - 3 secs */
#ifdef MDS CONFIRM TIMEOUT
    0x00005dc0,
#endif
    /* Configuration Selected */
    Ο,
            /* --- not modifiable */
    /* Configuration Count */
    2,
    /* pointer to an array of configurations */
    (CONFIGURATION (*)[])&g Configuration
```

};

The user modifiable attribute values are marked in red color. You can modify all the attributes, except for Handle attribute. The configuration selected parameter, in the above structure, is modified in the library. The device can support multiple configurations whose number is given by the configuration count



Developing New Device Specialization

followed by an array of configurations. The index of the configuration accepted by the manager is updated in the configuration selected parameter in the library.

4.3.4 Device Configuration

There are zero or more Numeric, Real-Time Sample Array, Enumeration, Scanner, or PM Store objects associated with an MDS object. There are zero or more PM Segments that contain persistent metrics associated with a PM Store. Numeric, Real-Time Sample Array, and Enumeration are derived from a common Metric base class that contains common and shared attributes. In general, Numeric objects represent episodic measurements, Real-Time Sample Array objects represent continuous samples or wave forms, Enumeration objects represent event annotations, and PM Stores along with PM Segments provide a persistent storage mechanism for metrics that are accessed by the Manager at a later time. In addition, a Scanner object facilitates the reporting of agent initiated data transfers. All these objects form a part of the device configuration.

Sample Configuration Structure:

```
/* Device Configuration */
const CONFIGURATION g Configuration[] =
{
      /* Device Configuration ID */
      0x4000,
      #ifdef MDS NUMERIC
          /* Numeric object count */
          2,
          /* Pointer to an array of Numeric classes */
          (NUMERIC (*)[])&g_Numeric,
      #endif
      #ifdef MDS RTSA
          /* RTSA object count */
          1,
          /* Pointer to an array of RTSA classes */
          (RTSA (*)[])&g Rtsa,
      #endif
      #ifdef MDS ENUMERATION
          /* Enumeration object count */
          1,
          /* Pointer to an array of Enumeration classes */
          (ENUMERATION (*)[])&g Enum,
      #endif
      #ifdef MDS EPISOIDIC SCANNER
          /* Episoidic Scanner object count */
          1,
          /* Pointer to an array of Episoidic Scanner classes */
          (EPICFGSCANNER (*)[])&g EpiScanner,
      #endif
      #ifdef MDS PERIODIC SCANNER
          /* Periodic Scanner object count */
          1,
          /* Pointer to an array of Periodic Scanner classes */
          (PERICFGSCANNER (*)[])&g PeriScanner,
      #endif
      #ifdef MDS PMSTORE
```



```
/* PM Store object count */
1,
    /* Pointer to an array of PM Store classes */
    (PMSTORE (*)[])&g_PMStore
#endif
};
```

The whole configuration structure is user modifiable. The configuration structure sets the number of Numeric, RT-SA, Enumeration, Episodic Scanner, Periodic Scanner, PM Store classes and the pointers to these classes. It also provides the device configuration Id.

NOTE

- The configuration Id of the first configuration should be the one specified in the Data Proto List.
- If agent initiated data transmission is not supported (specified in data request mode flags in Data Proto List), at least one Scanner should be a part of the configuration.

4.3.5 Metric Object

The Metric class is the base class for all objects representing measurements, status, and context data. The Metric class is not instantiated. As a base class, it defines all common attributes, methods, events, and services that are common for all objects representing measurements.

Sample Structure:

```
#ifdef MDS METRIC
METRIC g Metric[] =
{
        /* Numeric(Temperature) Class Handle */
        (intu16)0x0002,
        /* TYPE */
        {
            MDC PART SCADA,
            MDC TEMP BODY
        },
        /* Metric Spec Small */
        (intu16) (
        MSS AVAIL INTERMITTENT | MSS AVAIL STORED DATA |
        MSS UPD APERIODIC | MSS MSMT APERIODIC | MSS ACC AGENT INITIATED
        ),
        /* Optional Attributes */
        (intu16) (OPT MET UNIT CODE | OPT MET ATTRVALMAP),
        /* SupplementalTypeList */
        (SupplementalTypeList*)&g_Supplemental_type_list,
        /* Metric Struct Small */
        {
            MS_STRUCT_SIMPLE, 0
        },
        /* Measurement Status */
        MS VALIDATED DATA,
        /* MetricIdPhysioList */
        {
```

MEDCONLIB Users Guide, Rev. 5



Developing New Device Specialization

```
0,
            (OID Type *) & gMetricId
        },
        /* Metric ID Part */
        NOM PART OBJ,
        /* Unit Code */
        MDC DIM DEGC,
        /* Attribute Val Map */
        (AttrValMap*)&g_AttrValMap,
        /* source handle reference*/
        0x0001,
        /* Label String */
        (octet string*)&g label string,
        /* Unit Label String */
        (octet string*) &g unit label string
};
#endif
```

The user modifiable attribute values are marked in red color. You can modify all the attributes. The operational attribute flag tells which conditional and optional attributes of this class are implemented by the user.

4.3.6 Numeric Object

An instance of a Numeric class represents a numerical measurement. This class is derived from the Metric base class.

Sample Structure:

The user modifiable attribute values are marked in red color. &g_Metric[0] sets the pointer to the metric base class. The operational attribute flag tells which conditional and optional attributes of this class are implemented by the user.



4.3.7 Real-Time Sample Array (RT-SA) Object

An instance of the Real-Time Sample Array (RT-SA) class represents a wave form measurement. This class is derived from the Metric base class.

Sample Structure:

```
const RTSA g Rtsa[] =
      /* Metric class Pointer */
      &g Metric[3],
      /* sample period */
     0x00000002,
      /* Simple Sa Observed Value */
      (octet string*) &g RtsaSimpSaObsVal,
      /* Scale Range Spec */
      0x00,
      (ScaleRangeSpec8 *)&g scale8,
      /* Sa Spec */
      0x00a,
                     /*array size */
      0x08, 31,
                    /* sample type */
      SMOOTH CURVE
                     /* sa flags */
};
```

The user modifiable attribute values are marked in red color. &g_Metric[3] sets the pointer to the Metric base class. All attributes of the RT-SA object class are mandatory.

4.3.8 Enumeration Object

An instance of the Enumeration class represents status information and/or annotation information. The values of the Enumeration object are coded in the form of normative codes (as defined in ISO/IEEE Std 11073-10101) or in the form of free text. This class is derived from the Metric base class.

Sample Structure:

The user modifiable attribute values are marked in red color. &g_Metric[2] sets the pointer to the Metric base class. The operational attribute flag tells which conditional and optional attributes of this class are implemented by the user.



4.3.9 Scanner

A Scanner object is an observer and 'summarizer' of object attribute values. It observes attributes of Metric objects (for example, Numeric objects) and generates summaries. The Scanner class is an abstract class defining attributes, methods, events, and services that are common for its subclasses. As such, it cannot be instantiated. More specialized Scanner classes are derived from the Scanner base class.

Sample Structure:

```
#ifdef MDS_SCANNER
SCANNER g_Scanner[] =
{
    /* Episoidic class handle */
    5,
    /* Operational Stat */
    0, /* --- not modifiable */
    /* Handle Attribute Val Map */
    (HandleAttrValMap*)&g_HandleAttributeValMap,
};
#endif
```

The user modifiable attribute values are marked in red color.

4.3.10 Configurable Scanner

The Configurable Scanner class is an abstract class defining attributes, methods, events, and services that are common for its subclasses (Episodic and Periodic Configurable Scanner objects). In particular, it defines the communication behavior of a configurable Scanner object. As such, it cannot be instantiated.

Sample Structure:

```
/* Configuration scanner class */
#ifdef MDS CFGSCANNER
CFGSCANNER g CfgScanner[] =
{
      /* Optional Attributes */
      З,
      /* pointer to scanner class */
      (SCANNER*) &g Scanner[0],
      /* confirm mode */
      0x0001,
      /* confirm timeout - 2 secs */
      16000,
      /* transmit window */
      0x0001
};
#endif
```

The user modifiable attribute values are marked in red color. &g_Scanner[0] sets the pointer to the Scanner base class. The operational attribute flag tells which conditional and optional attributes of this class are implemented by the user.



4.3.11 Episodic Configurable Scanner

The Episodic Configurable Scanner objects are used to send reports containing episodic data, that is, data that does not have a fixed period between each data value. A report is sent whenever one of the observed attributes changes value. However, two consecutive event reports shall not have a time interval less than Min-Reporting-Interval.

Sample Structure:

The user modifiable attribute values are marked in red color. &g_CfgScanner[0] sets the pointer to the Configurable Scanner base class. The operational attribute flag tells which conditional and optional attributes of this class are implemented by the user.

4.3.12 Periodic Configurable Scanner

Periodic Configurable Scanner objects are used to send reports containing periodic data, that is, data sampled during fixed periods. It buffers any data value changes to be sent as part of a periodic report. Event reports shall be sent with a time interval of Reporting-Interval. The number of observations for each Metric object is dependent on the Metric object's update interval and the Scanner's Reporting-Interval.

Sample Structure:

The user modifiable attribute values are marked in red color. &g_CfgScanner[1] sets the pointer to the Configurable Scanner base class. The scan reporting period sets the Scanner's reporting interval.



Developing New Device Specialization

4.3.13 PM Store Object

An instance of the PM Store class provides long-term storage capabilities for Metric data. The data is stored in a variable number of PM Segment objects.

Sample Structure:

```
/* PM Store Object Class */
PMSTORE g PMStore[] =
{
      /* optinal attribute flag */
      0x0f,
      /* PM Store Object handle */
      0x04,
      /* PM Store Capability */
      (
      #ifdef MULTI PERSON SUPPORT
          PMSC MULTI PERSON |
      #endif
      #ifdef SAMPLE PERIOD
          PMSC PERI SEG ENTRIES | /* if sample period attr present in PM Store or in all PM
          Segments, then this shud be set ^{\star/}
      #endif
      PMSC CLEAR SEGM REMOVE | PMSC CLEAR SEGM BY TIME SUP |
      PMSC EPI SEG ENTRIES | PMSC CLEAR SEGM BY LIST SUP),
      /* sampling algorithm */
      ST ALG NO DOWNSAMPLING,
      /* max PM Segments entries */
      0x04,
      /* actual num of segment entries presently used */
      0x02,
      /* operational state, if data is actively added it should be 0x01 */
      0x00.
      /* PM Store Label String */
      (octet string*) &g pm label string,
      /* Sample Period */
      0x00000ff,
      /* num of PM Segments instantiated */
      0x02,
      /* Clear segment timeout */
      OxFFFFFFF,
      /* PM Segment Count */
      0x02,
      /* Pointer to array of PM Segments */
      (PMSEGMENT (*)[])&g PmSegment
};
```

The user modifiable attribute values are marked in red color. You can modify all the attributes. The PM Store object structure also sets the PM Segments instantiated followed by the pointer to an array of PM Segment structures.



{

PM Segment Object 4.3.14

An instance of the PM Segment class represents a persistently stored episode of measurement data.

Sample Structure:

```
PMSEGMENT g PmSegment[] =
      /* optional attribute flag */
      0x01FF,
      /* instance number */
     0x0001,
      /* PM Segment entry map */
      (PmSegmentEntryMap*)&g_PmSegEntryMap,
      /* Person ID */
      #ifdef MULTI PERSON SUPPORT
          0x0001,
      #endif
      /* operational state */
      0x0000,
      /* Sample period */
      0x05,
      /* segment label string */
      (octet string*)&g PmSeg label string1,
      /* Segment Start Time */
      0x20, 0x09, 0x09, 0x13, 0x02, 0x00, 0x00, 0x00,
      /* Segment End Time */
      0x20, 0x09, 0x09, 0x13, 0x04, 0x00, 0x00, 0x00,
      /* Date and time adjustment */
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
      /* usage count */
      0x00,
      /* Segment Statistics */
      (SegmentStatistics*)&g SegStat,
      /* pointer to Segment data */
     NULL,
      /* Confirm timeout = 4 secs */
      0x00007D00,
      /* Transfer timeout = 4 secs */
      0x00007D00
};
```

The user modifiable attribute values are marked in red color. You can modify all the attributes. Whenever an entry is added to the PM Segment, the library updates the Segment Start Time (if implemented), Segment End Time (if implemented), and Usage Count (if implemented).



Developing New Device Specialization

4.4 Application Design

This section discusses the application design. The application is made up of the main application function, application task, and the callback function.

4.4.1 Main Application Function

The main application function initializes Transport Independent Layer (TIL) and Medical Connectivity Library. It also registers the application callback function. The main application function uses the following C code:

```
void TestApp_Init(void)
      DisableInterrupts;
      <Application Buffers Initialization Code>
      <Application Specific Initialization Code goes here>
      /* Initialize TIL */
      TIL_Initialize((PTIL)&g_Til);
      /* Initialize IEEE11073 and start Transport */
      (void)Ieee11073Initialize((PTIL)&g Til, SHIMID, MedAppCallback);
      EnableInterrupts;
      while(TRUE)
      {
            RESET WATCHDOG();
          <Application Specific Code goes here>
          new app task();
      }
}
```

4.4.2 Application Task

The application task performs application specific functionality. Sample application code below uses Keyboard inputs to send measurement data to Continua Manager. The application task uses the following C code:

```
static void new_app_task(void)
{
      ERR CODE err = ERROR SUCCESS;
      if(kbi stat > 0)
      {
          switch(kbi stat & KBI STAT MASK)
           {
               case SEND BPM MSR : /* PTG1 is pressed */
               {
                    <Application Specific Code goes here>
               }
               break;
               case SEND TEMPERATURE MSR : /* PTG2 is pressed */
               {
                    <Application Specific Code goes here>
               }
               break;
```



4.4.3 Callback Function

This is the application callback function. This callback function is registered during initialization in the main application function. This function performs various tasks depending upon the event received from the Medical Connectivity Library. The callback function uses the following C code:

```
static void MedAppCallback(IEEE11073 EVENT event id, void *pvoid)
{
      switch (event id)
      {
          case IEEE11073 ASSOCIATION RELEASING:
              <Application Specific Code goes here>
          break;
          case IEEE11073 ASSOCIATION RELEASED:
              <Application Specific Code goes here>
          break;
          case IEEE11073_TRANSPORT_CONNECT:
              <Application Specific Code goes here>
          break;
          case IEEE11073 TRANSPORT DISCONNECT:
              <Application Specific Code goes here>
          break;
          case IEEE11073 GET DATAPROTO:
              <Application Specific Code goes here>
          break;
          case IEEE11073 OPERATING:
              <Application Specific Code goes here>
          break;
          case IEEE11073 EVNTRPT SENT:
              <Application Specific Code goes here>
          break;
          case IEEE11073 CLEAR PMSEGMENT:
               <Application Specific Code goes here>
          break;
          default:
               <Application Specific Code goes here>
          break;
         }
        return;
```

}



Developing New Device Specialization



Chapter 5 IEEE 11073 Manager

5.1 Introduction

This chapter describes the IEEE 11073-based PHDC Manager implementation using USB host stack functionality for transport purposes.

The IEEE 11073 Manager's purpose is to enable seamless interoperability between personal healthcare devices (such as glucose meters, pulse oximeters, thermometers, etc) and USB hosts. The USB Class definition for personal healthcare devices provides a generic mechanism by which standardized messages can be sent over USB

5.2 Folder structure

The PHDC manager source code includes:

- Include sub-folder:
 - ieee11073.h: contains agent structure definition ieee11073_comm.h: contains definition of application events, application Callback function pointer and function prototypes for communication layer
 - ieee11073_dec.h: describes function prototypes for event report decoding functionality
 - ieee11073_phd_types.h: defines all phdc structures for PHDC manager library
 ieee11073_non_codes.h: contains the nomenclature codes used by IEEE_11073ieee11073_sl.h:
 describes function prototypes for service layer
 - type.h: define basic data type used for PHDC manager libraryieee11073_timer.h: defines time object structure and function prototype serving time out functionality
 - TransportLayer/TIL.h: defines communication callback function pointer and function prototype for TIL layer
 - TransportLayer/UsbShimManager.h: defines constants, structures, function prototypes used for SHIM layer
- "Src sub-folder:
 - Ieee11073/ieee11073_comm.c: implements all functions of communication layer. It deals with analyzing APDU data from receive and send callback function called in lower level layer.
 - Ieee11073/ieee11073_dec.c: implements all decoding functions: It supports to decode:
 - Fix format
 - Variable format
 - Grouped format

For

- Numeric class



IEEE 11073 Manager

- Real time sample array class
- Enumeration class
- MDS class
- ieee11073/ieee11073_sl.c: implement all service layer's functions which can be called by applications
- TransportLayer/TIL.c: implements TIL functions
- TransportLayer/USbShimManager.c: implements all functions of SHIM layer.

5.3 Module Usage/Aim

The PHDC Host class driver provides an interface to the USB Host controller, allowing the application layer to handle the data exchange with the Agent using standard PHDC commands in the scope of gathering the personal healthcare data.

The PHDC Host class provides the following functions:

- Manages a class interface with the connected device consisting in 3 communication pipes corresponding to the attached device endpoints (1 Bulk IN, 1 Bulk OUT endpoint and 1 Interrupt IN Endpoint)
- PHDC data sending with Metadata support.
- PHDC data receiving with Metadata support
- PHDC Send Class Request function with SET_FEATURE, CLEAR_FEATURE, GET_STATUS requests supportSend Complete Event indication to the application layer
- Receive Complete Event indication to the application layer
- Send Control Requests Complete Event indication to the application layer

5.4 Functional description

5.4.1 MQX and Bare-Metal System Interraction

The PHDC Host class driver uses the MQX or bare-metal stack provided Host common API services in order to access the USB bus and manage USB device communication. The Freescale MQX ™ USB Host Stack provides an abstraction of the USB hardware controller consisting in:

- Host API
- Chapter9 API
- Common-Class API



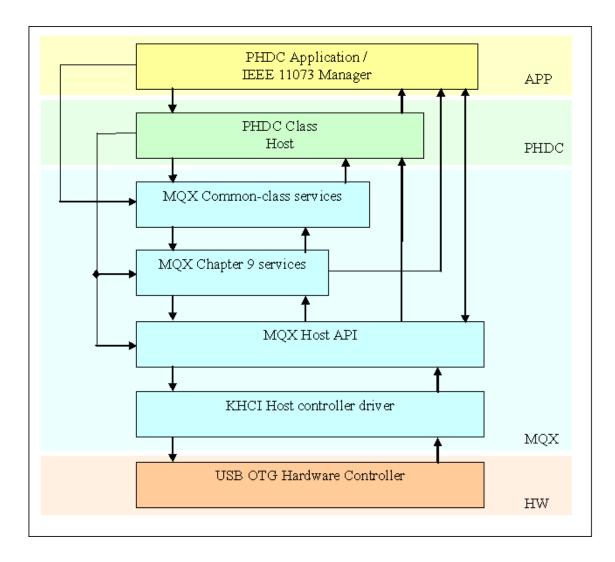


Figure 5-1. PHDC Host class interaction with the MQX USB services

The following sub-chapters present the interaction between the Application / PHDC Class driver and the used MQX USB layers.

5.4.2 MQX Host API

The MQX Host API is an abstraction of the host controller driver, providing interfaces independent of the underlying USB controllers.

The PHDC Application / IEEE 11073 Manager is responsible of initializing the Host controller and handle the USB attach and detach events issued by this layer.

The following MQX Host API services are used by the Application layer:

• _usb_host_init this function will initialize the USB Host stack. This function is called under application control as the application can manage multiple interfaces.

MEDCONLIB Users Guide, Rev. 5



IEEE 11073 Manager

The PHDC Host class is responsible of handle the Bulk / Interrupt data transfers with the connected device using the MQX Host API pipe send / receive services:

- **_usb_host_send_data** this is a non blocking function that schedules a block of data for USB transmission and registers a callback for the send data complete event. The PHDC class calls this function following the Application request of sending a PHDC data (metadata or opaque data) to the connected agent.
- **_usb_host_recv_data** this is a non blocking function that schedules an USB data reception and registers a callback for the data received complete event. The PHDC class calls this function following the Application request of interrogating the connected Agent for measurement data.

5.4.3 MQX Chapter 9 Services

The MQX Chapter 9 layer implements the dedicated services specific to the USB Ch9 commands.

The Application / IEEE 11073 Manager is responsible of handling the relevant Chapter 9 indication events:

- USB_ATTACH_EVENT this event is sent to the application as the result of a PHDC device connection to the Host. The application uses this event to select the PHDC class interface on the Host (See the MQX Common Class)
- USB_DETACH_EVENT this event is sent to the application when the Device has disconnected from the host.

The PHDC Host class driver uses the MQX Chapter 9 services to transmit PHDC Specific requests (SET_FEATURE, CLEAR_FEATURE, GET_STATUS) using the control pipe:

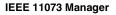
• _usb_hostdev_cntrl_request this function is used by the MQX Ch9 Layer to process standard USB, class or vendor specific control pipe device requests. The PHDC class uses this service to send the PHDC class specific requests to the device and also USB standard requests as CLEAR_FEATURE (ENDPOINT_HALT)

5.4.4 MQX Common-class Services

The MQX Common-Class layer implements the common-class USB specifications. It interacts with the Host API as well as with the Chapter 9 layer in order to enumerate the attached device. After the device descriptors are identified, the common class layer searches for applications that are registered for the class or device plugged in.

The Application / IEEE 11073 Manager is responsible of handling the PHDC interface selection after the device has attached and getting the full descriptors of the PHDC device for a proper identification of the connected agent and QoS managing.

• _usb_hostdev_select_interface this function will select a new interface on the attached device, open the pipes associated to that interface and create the pipe bundle. This function also initializes the corresponding class driver by calling its registered initialization function. The application calls this interface after receiving the USB_ATTACH_EVENT from the Host API in order to set the PHDC interface and to initialize the PHDC class.





The PHDC Host class driver uses the Common-Class services to collect the PHDC class specific descriptors (QoS descriptor and the optional Metadata descriptor) for all the endpoints opened by the attached device:

• _usb_hostdev_get_descriptor this function returns the requested descriptor based on the provided descriptor type and descriptor index. The PHDC class calls this function in order to get the identification information from the connected device, after receiving the USB_ATTACH event from the Host API. The PHDC uses the PHDC specific descriptors in order to collect the data from the connected agent with regards to the supported QoS.

5.5 System Decomposition

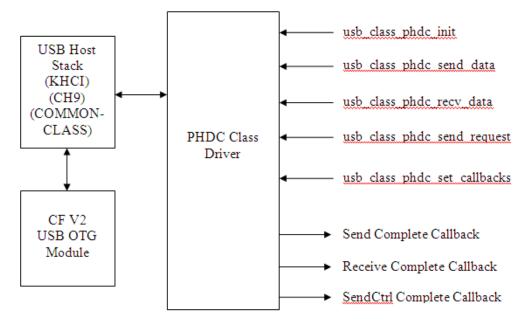


Figure 5-2. PHDC class module decomposition



Appendix A Working with the Software

A.1 Introduction

This chapter gives you insight on how to use the Medical Connectivity Library software. The following sections are described in this chapter:

- Preparing the setup
- Building the application
- Running the application

Knowledge of CodeWarrior IDE will be helpful to understand this section. While reading this chapter, practice the steps mentioned.

To take you through this chapter, USB Agent Demo for the MCS08JM60 is used as an example.

A.1.1 Preparing the Setup

A.1.1.1 Software Setup

Refer to Readme file for compatibility with Freescale USB Stack with PHDC.

To install Medical Connectivity Library software setup:

- 1. Double-click MEDCONLIB_SW.exe file..
- 2. The Freescale Medical Connectivity Library Setup window appears. The following example shows the demonstration for Medical Connectivity Library 1.0 installation. You can follow the same instructions for new versions.

Example:

1. Click on the Next button to continue with Medical Connectivity Library 1.0 installation.

Working with the Software





Figure A-1. Freescale Medical Connectivity Library 1.0 Setup Wizard

2. In Figure A-2, click on the I Agree button to accept the license agreement.

🏓 Freescale Medical Connectivity Library 1.4 Setup	X
License Agreement Please review the license terms before installing Freescale Medical Connectivity Library 1.4.	*
Press Page Down to see the rest of the agreement.	
MPORTANT. Read the following Freescale Semiconductor Software License Agreement ("Agreement") completely. By selecting the "I Accept" button below, you indicate that you accept the terms of this Agreement. You may then install the software.	ļ
FREESCALE SEMICONDUCTOR SOFTWARE LICENSE AGREEMENT [SOFTWARE FOR: Freescale MQX]	
This is a legal agreement between you (either as an individual or as an authorized representative of your employer) and Freescale Semiconductor, Inc. ("Freescale"). It concerns your rights to use this file and any accompanying written materials (the	
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Freescale Medical Connectivity Library 1.4.	
Nullsoft Install System v2.45	

Figure A-2. Freescale Medical Connectivity Library 1.0 Setup License Agreement

3. In Figure A-3, select Medical Connectivity Library and PAN USB Agent demo application to install and click on the **Next** button.

Working with the Software

🏓 Freescale Medical Conne	ctivity Library 1.4 Setup	
Choose Components Choose which features of Frees install.	scale Medical Connectivity Library	1.4 you want to 🔅
Check the components you war install. Click Next to continue.	nt to install and uncheck the comp	onents you don't want to
Select components to install:	 Medical Connectivity Libra PAN USB Agent PAN Serial Bridge Demo PAN USB Agent Blood Pre 	Description Position your mouse over a component to see its description,
Space required: 18.0MB		
Nullsoft Install System v2.45 ——	< <u>B</u> ack	Next > Cancel

Figure A-3. Freescale Medical Connectivity Library 1.0 Components

4. In Figure A-4, select the location of the folder where you require to install the Freescale Medical Connectivity Library 1.0 software and click on the **Install** button.

Freescale Medical Connectivity Library 1.4 Setup	
Choose Install Location Choose the folder in which to install Freescale Medical Connectivity Library 1.4.	
Setup will install Freescale Medical Connectivity Library 1.4 in the following folder. T a different folder, click Browse and select another folder. Click Install to start the in	
Destination Folder [\Program Files\Freescale\Freescale USB Stack with PHDC v3.1] Browse	<u>}</u>
Space required: 18.0MB Space available: 73.9GB Nullsoft Install System v2.45	
< Back Install	Cancel

Figure A-4. Freescale Medical Connectivity Library 1.0 Installation Folder Location



CAUTION

Use the same destination folder where Freescale USB Stack with PHDC is installed.

5. Click on the **Finish** button to successfully complete the Freescale Medical Connectivity Library 1.0 Setup Wizard.

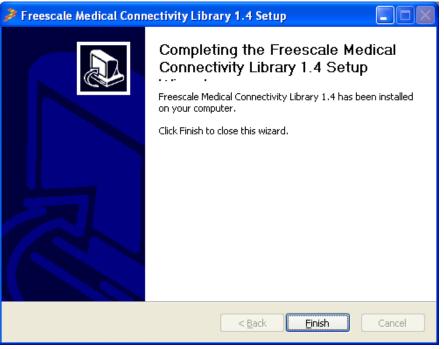


Figure A-5. Freescale Medical Connectivity Library 1.0 Installation Finish

To launch the Medical Connectivity Library project:

1. Click Start > Programs > Freescale Medical Connectivity Library > Source.

NP

Working with the Software

	•	Set Program Access and Defaults						
	1	Programs	•	Freescale Medical Connectivity Library v1.2		Documentation Source	_	a PAN - Medical Connectivity Library
	*	Favorites	×		۲	Uninstall		DAN - Serial Bridge Demo
	3	Documents	۲		Γ			눰 PAN - USB Agent Demo
	<u>v</u> -	Settings	×					
onal	P	Search	×					
Professional	?	Help and Support						
	2	Run						
/s XP	P							
Windows	≜	Undock Computer						
Wir	0	Shut Down						
4	🖌 sta	nrt 🛛 🕫 🌢 🗮 🔌 👋						
					-			

Figure A-6. Freescale Medical Connectivity Library Source Program for Launch

A.1.1.2 Hardware Setup

• Make the connections as shown in Figure A-7.

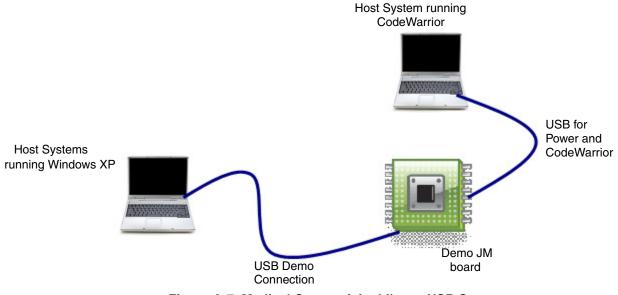


Figure A-7. Medical Connectivity Library USB Setup

- Make the first USB connection between the personal computer where the software is installed and the DemoJM board where the silicon is mounted. This connection is required to provide power to the board and downloading image to the flash.
- Make the second connection between the DemoJM board and the personal computer where the demo is run.



NOTE

Although, we have used two personal computers in Figure A-7, in reality you may achieve the same result by a single personal computer with two or more USB ports.

A.1.2 Building the Application

The software for CFV1 is built using CodeWarrior 6.3. In addition, the software for CFV2 is built with CodeWarrior 7.2. Therefore, it contains application project files that can be used to build the project.

Before starting the process of building the project, make sure CodeWarrior 6.3 is installed on your computer.

To build the MC9S08JM60 project:

1. Navigate to the project file and open the s08usbjm60.mcp project file in CodeWarrior IDE.

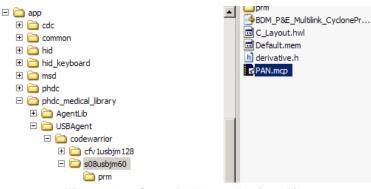


Figure A-8. Open PAN.mcp Project File

2. After you have opened the project, the following window appears. To build the project, click the button as shown in Figure A-9.



Working with the Software

@Freescale CodeWarrior		_8×
File Edit View Search Project Proc	rocessor Expert Device Initialization Window Help	
🐚 🖂 📽 🔠 🗠 🗠 🐂	🛯 🖷 🖀 🖀 🍽 🖕 🖀 🕞 📷 🖉 Click to build the project	
PAN.mcp		
P&E Multilink/Cyclone Pro 🔹	- 🛱 🖻 🗸 🔇 💺	
Files Link Order Targets		
✓ File	Code Data 🕊 /	
Bit Includes Bit Includes		
40 files	0 0 0	
<[2	

Figure A-9. Build PAN.mcp Project

3. After the project is built, the code and data columns must appear filled across the files.

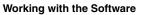
NOTE

You must follow the above procedure to build CFV1 and CFV2 projects also.

A.1.3 Running the Application

Refer to the board documentation and CodeWarrior manual for details on how to program the flash memory on the evaluation board used. The following steps are presented as an example about how to run the application with DemoJM60 board using a P&E-micro debugger.

1. To run the application, click the button as shown in Figure A-10.



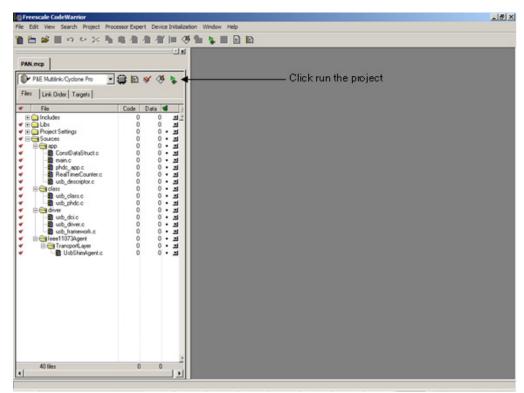
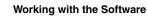


Figure A-10. Running the Application

2. The dialog box in Figure A-11 appears. Click on the **Connect (Reset)** button to connect to hardware as shown in Figure A-11.



Connection port and Interface Type		Add LPT Port
Interface: USB HCS08/HCS12/CFV1 Multilink - USB Port	-	Refresh List
Port DEMOJM on USB1 (Name=PE5038205) (Autodete	cted) *	
* Contains Embedded Multilink. Click for details.	Socket Programming Ad	apter Settings
Target CPU Information CPU: HCS08 Processor - Autodetect MCU reset line: MCU Voltage:		
Reset Options Delay after Reset and before communicating to target for	0 miliseco	nds (decimal).
Cyclone Pro Power Control (Voltage -> Power-Out Jack)		
Provide power to target Regulator Output Volta Power off target upon software exit	ge Power Down Dela Power Up Dela	
Trim Control Default trim reference frequency is : 31250.00 Hz. (Valid Rang	ge: 31250.00 to 39062.5 Hz Click for trim details.	0 Hz)

Show this dialog before attempting to contact target (Otherwise only display on Error)

Figure A-11. Connection Manager

3. The pop-up in Figure A-12 appears. Click on the Yes button to load the built image to the JM60 flash.

á.	Load image contains flash memory data. Era	ase and Program flash
1	[
	✓ Yes	

Figure A-12. Erase and Program Flash Pop-Up

4. The pop-up in Figure A-13 appears to erase and program the built image to the JM60 flash.



CPROGHCS08 Programmer - Version 1.56.00.01 - [Status Window]	- I I X
X Abort	http://www.pemicro.com
Windows NT detected.	
Copyright 1999,2002 P&E Microcomputer Systems, Inc.	
CMD>RE	
USB HCS08/HCS12 MULTILINK detected - Flash Version 5.77	
Initializing. Target has been RESET and is active.	
CMD>CM C:\Program Files\Freescale\CodeWarrior for Microcontrollers V6.: USB HCS08/HCS12 MULTILINK detected - Flash Version 5.77	2\prog\P&E\9S08JM60.S8P
Initializing. (Recommended TRIM=\$00A5,FTRIM=1) (Bus Freq = 15937RHz)	Initialized.
]	
Ranning programming script	

Figure A-13. Image Programmed in Flash

5. After the image is programmed in the flash, the debugger window as shown in Figure A-14 appears. Click on the Green Arrow as shown in Figure A-14 to run the programmed image.

	≧ <u>१₩</u> →२⊽४२-	L 🔍		True-Time Simulator & Real-Time	and the second second
Source			_OX	Assembly	
D:\CW_Projects\JM_USB\	phod\device_19hApr09\device\app\hid\m	ain.c Line	× 44	main	
Init_Sys(); RICSC_RIIE=1; #ifdef _MCFS1JM12 KBI1SC_KBIE=1		(for CFv1) */	-	1275 JSR 0x127F 1278 BSET 4,0x40 1272 BSET 1,0x10 1270 JNP 0x1000 1277 JSR 0x128	
telse			1	Register	
P Procedure			-0×	HC508 A 0 HX 100 SP 2C4 SR 6A Status VHIH2C	Auto
ata:1			_O×	🖼 Hemory	
	main.c	Auto Symb	Global	[·	Auto
E KBISC <1>	volatile RICSCSIR volatile XBISCSIR volatile SOPTISIR		1 1	0080 00 24 00 00 00 00 00 06 00 0088 04 FB 33 00 00 00 00 00 0090 20 00 00 00 00 00 00 00	
Data:2			_OX	I Command	
	main	Auto Symb	Local	in>	-

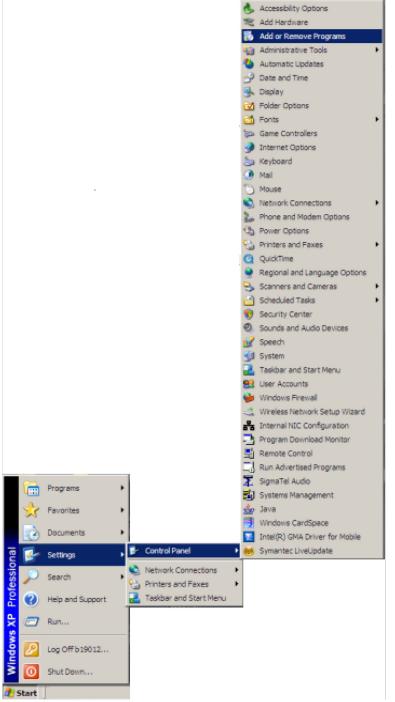
Figure A-14. Simulator and Real-Time Debugger



Working with the Software

A.2 Uninstall Freescale Medical Connectivity Library 1.0 Software

1. From your computer, click Start > Settings > Control Panel > Add or Remove Programs.







2. In Figure A-16, select Freescale Medical Connectivity Library 1.0 and click on the Change/Remove button.

🐞 Add or Ren	nove Programs			_ 0	×
5	Currently installed programs:	Show up <u>d</u> ates	Sort by: Name	•	1
C <u>h</u> ange or Remove	🗭 Adobe Flash Player 10 Plugin			-	
Programs	🔂 Adobe Flash Player 9 ActiveX				
- 🥵	🗾 Adobe Flash Player 9 ActiveX		Size	3.05MB	
Add <u>N</u> ew	Adobe Reader 9.2		Size	141.00MB	
Programs	🔂 ALPS Touch Pad Driver		Size	1.36MB	
6	🔁 ATI Display Driver				
Add/Remove	Beyond Compare Version 3.0.15		Size	12.75MB	
<u>W</u> indows Components	🔁 CESL Reference Design Applications		Size	54.34MB	
	CodeWarrior Development Studio for Microcontrollers V6.2		Size	955.00MB	
	Compatibility Pack for the 2007 Office system		Size	129.00MB	
Set Pr <u>o</u> gram Access and	🛃 EditPlus 3		Size	2.05MB	
Defaults	🔂 FastLane ActiveRoles Extension Package		Size	2.01MB	
	Freescale Medical Applications USB Stack 1.1		Size	8.95MB	
	Freescale Medical Connectivity Library 1.0		Size	12.72MB	
	Click here for support information.				
	To change this program or remove it from your computer, dick	Change/Remove.	Chang	je/Remove	
	Freescale Medical Library 1.0		Size	8.95MB	
	靜 Freescale Medical Library for MQX 1.0		Size	0.05MB	-

Figure A-16. Add or Remove Programs

3. The uninstall confirmation message appears. Click on the Yes button to uninstall.

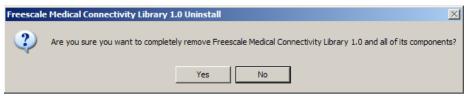


Figure A-17. Freescale Medical Connectivity Library 1.0 Uninstall Confirmation Message

4. A message box appears. Click on the **Ok** button to complete the uninstall operation.



Figure A-18. Freescale Medical Connectivity Library 1.0 Uninstall Completion Message



Working with the Software

A.3 Important Files

Table A-1 shows the programming files that contain source code and header files.

Table A-1. Important Files

Files	Description
device\app\phdc_medical_library\AgentLi b\include\error.h	This header file contains error codes.
device\app\phdc_medical_library\AgentLi b\include\ieee11073_dimstruct.h	This header file defines DIM structures to be used by device specialization.
device\app\phdc_medical_library\AgentLi b\include\ieee11073_nom_codes.h	This header file contains IEEE11073 Nomenclature Codes.
device\app\phdc_medical_library\AgentLi b\include\ieee11073_phd_types.h	This header file contains IEEE11073 Structure definitions.
device\app\phdc_medical_library\AgentLi b\include\mds_config.h	This header file contains Medical Connectivity Library configuration (DO NOT CHANGE THIS FILE).
device\app\phdc_medical_library\AgentLi b\include\MedAgentLibInterface.h	This header file contains Medical Connectivity Library Interface definitions.
device\app\phdc_medical_library\AgentLi b\include\mempool.h	This header file contains Memory Management Interface.
device\app\phdc_medical_library\AgentLi b\include\stack.h	This header file contains Buffer Stack Interface definitions.
device\app\phdc_medical_library\AgentLi b\include\type.h	This header file contains basic data type definitions.
device\app\phdc_medical_library\AgentLi b\include\TransportLayer\til.h	This header file contains TIL Interface definitions.
device\app\phdc_medical_library\AgentLi b\include\TransportLayer\UsbShimAgent. h	This header file contains USB Shim Agent Interface definitions.
device\app\phdc_medical_library\AgentLi b\src\TransportLayer\UsbShimAgent.c	This source file contains USB Shim Agent Source.
device\app\phdc_medical_library\AgentLi b\lib\leee11073AgentLibCfv1.lib	This is Medical Connectivity Library for CFV1 devices.
device\app\phdc_medical_library\AgentLi b\lib\ leee11073AgentLibJm60.lib	This is Medical Connectivity Library for JM60 device.



Appendix B PAN USB Agent Demo

Personal healthcare application interacts with the host system using IEEE-11073 – 20601 and (IEEE-11073 – 10415 (Weigh Scale), IEEE-11073 – 10407 (Blood Pressure Monitor), IEEE-11073 – 10417 (Glucose Meter), and IEEE-11073 – 10408 (Thermometer) protocol. To run this demo, a host system is required that runs the same IEEE-11073 protocols. One example of such implementation is done by Continua Alliance. In this demo, Continua Manager is used on the host system.

B.1 Setting Up the Demo

- 1. Set the systems as described in Section A.1.1.2, "Hardware Setup."
- 2. Get the Continua Alliance (www.continuaalliance.org) CESL Reference Software V1.0 RC2.
- 3. Install the software on a host system
- 4. Program the JM60 flash with the PAN USB Agent Demo using CodeWarrior IDE. Refer to Section A.1.2, "Building the Application" for more details.

NOTE

CESL reference software is not provided as part of the suite. You will have to get this software independently from Continua Alliance.

B.2 Running the Demo

After the system has been set, you must follow these steps to run the demo:

1. Turn on the DemoJM board. Found New Hardware window appears.



PAN USB Agent Demo



Figure B-1. Found New Hardware Window

2. Select **Install from a list or specific location (Advanced)** option as shown in Figure B-1, and click on the **Next** button. Search and installation options window appears as shown in Figure B-2.

Found New Hardware Wizard
Please choose your search and installation options.
 Search for the best driver in these locations.
Use the check boxes below to limit or expand the default search, which includes local paths and removable media. The best driver found will be installed.
Search removable media (floppy, CD-ROM)
Include this location in the search:
D:\program files\CESL_Binary\usb Browse
O Don't search. I will choose the driver to install.
Choose this option to select the device driver from a list. Windows does not guarantee that the driver you choose will be the best match for your hardware.
< Back Next > Cancel

Figure B-2. Search and Installation Options

3. Point the search path to the bin directory where the Continua CESL software was installed and click on the **Next** button. The driver for the device will get installed.



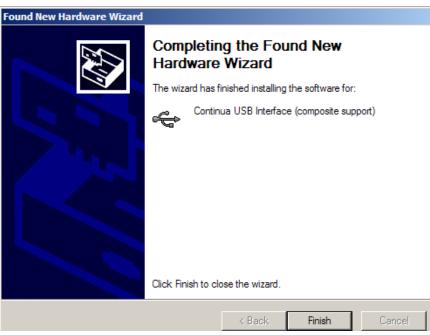
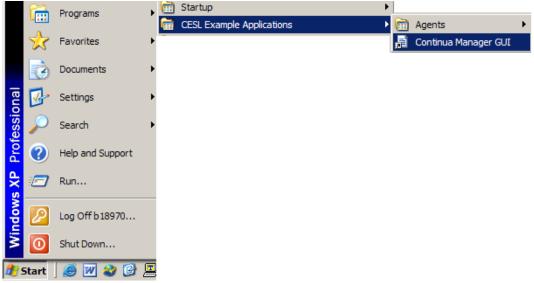
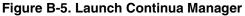


Figure B-3. Installation Complete window

4. To verify the installation, open the device manager. You must see the Continua USB Interface device entries.

- Figure B-4. Continua USB Interface Device Entry in Device Manager
- 5. Launch the Continua Manager from Start > All Programs menu as shown in Figure B-5.







PAN USB Agent Demo

6. The Continua Manager GUI opens as shown in Figure B-5. Enter the name of the skim directory and click on the **Start Transport** button.

Continua Manager	
File Edit Help	Click Start Transport
Current State: Disconnected	
Select Shim Directory: D:/Program Files/CESL_Binary/bin/	Browse Start Transport
Device List	Device Information
Device Name Transport Device Specializatio Addr	Waiting for Agent Device
	Waiting for Agent Device
	. /≫ ? T
Discover Connect Unassociate Abort	
	<u>/</u> / / → 伊
Output	
0x3841b8: WASCManagerGUI.cpp(1286)VASCManagerGUI::loadAppD	ta The APDU Dump Directory is set to: D:,Program Files/CESL_Binary,bin/apdu_dump.txt
Continua	Transport: Disabled
HEALIN ALLIANCE	USB Bluetooth TCP/IP

Figure B-6. Host Entering Operating State

7. The Continua Manager now enters the Operating State using Extended Configuration specialization. The Continua application window appears as shown in Figure B-7.

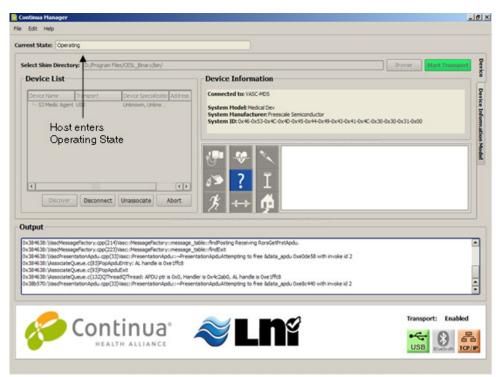


Figure B-7. Host Entering Operating State

8. After the host device is in operating state, **Push** Buttons on the device can be used to send weight measurements to the host. Figure B-8 shows the function of these buttons.

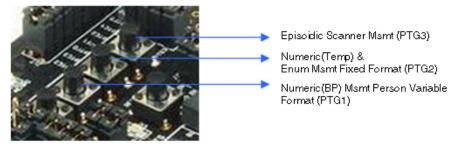


Figure B-8. DemoJM Push Button Panel

9. Figure B-9 shows Domain Information Model screen of Manager. This screen describes Agent Configuration.



PAN USB Agent Demo

ent State: Operating				
08H - Objects				
Metric Enumerated Episodic Scanner Periodic Scanner temperature	196 MCC, MCC, MCC, MCHERIS, SU MCC, MCC, MCO, METRIS, SU MCC, MCC, MCO, METRIS, SU MCC, MCC, SMC, MCO, METRIS, MCC, MCC, MCO, METRIS, SU MCC, MCC, MCO, METRIS, SU MCC, MCC, MCO, MCO, MCO, MCO, MCO, MCO,	Hendle 2 1 8 7 5 3 4 0 0	No object selected.	
	ontinua	. 🝣		ransport: Enabled

Figure B-9. Domain Information Model screen

10. When push button **PTG1** is pressed, Blood Pressure measurement in Person Variable Format is sent to host. Figure B-10 shows measurement data on host.

nt State: Opera	ting						
ect Shim Directo	D:/Program Fil	es/CESL_Binary/bin/				Browse Start Trans	port
evice List			Devic	e Information			
Device Name	Transport.	Device Specializatio A	ddress Conner	ted to: VASC-MDS			
- S3 Medic Agen	t U58	Unknown, Unkno	System	Model: Medical Dev Manufacturer: Free ID: 0x46-0x53-0x4C-	0x4D-0x45-0x44-0x49-0x43-0x41-0x40		
				*	- Tmestamp: 8:57:00 am on Sun Aug - Systolic: 83 mmHg - Diastolic: 103 mmHg - Hean: 768 mmHg	uet 9 2009	-
Discover	Disconnect	(at K	⊥ 1 ↔ 1¢	Timestamp: 8:58:00 am on Sun Aug Systolic: 84 mmHg Diastolic: 104 mmHg Hean: 768 mmHg	ist 9 2009	
184638: (Associate 184638: (VascPrese	.c(354)AL_SendAPD ntationApdu.cpp(33		f (0x0) Success	tempting to free &data	.apdu 0x3540510 with invoke id 37 .apdu 0x3540560 with invoke id 37		
84638: 'Associate' 84638: 'Associate'	Queue.c(33)PopApd Queue.c(93)PopApd	uEntry: AL handle is Oxe1 uExit	uffic8				
	Cont	cinua [.]	-			Transport: Enab	bled

Figure B-10. Blood Preasure measurement in Person Variable Format

11. When push button **PTG2** is pressed, Temperature measurement and Enumeration class measurement in Fixed Format is sent to host. Figure B-11 shows measurement data on host.

lect Shim Directory: D:,Pro; Device List	gram Files/CESL_Binary/bin/	Device Information	Start Transpor
Device Name Transport	Device Specializatio Address	Connected to: VASC-MDS	
- S3 Medic Agent US8	Unknown, Unkno	System Hodek Medical Dev System Hanufacturer: Prescale Semiconductor System ID: 0x46-0x53-0x40-0x45-0x44-0x49-0x43-0x41-0x4C-0x30-0x30-0 U	x31-0x00
		- Timestang: - H&F Set: 532	-
		-Timestamp: 8:59:00 am on Sun August 9 2009 -Body Temp: -30.5021 °C	
		- Timestanp: - H&F Set: 532	
		- Trestanc: - Body Temp: - 29.8467 °C	
		- Timestamp: 9:00:00 am on Sun August 9 2009 - H&F Set: 532	
4			
Discover Discon	10	- Timestamp: 9:02:00 am on Sun August 9 2009	-
	2011/201	≈Ln ĩ	Transport: Enable

Figure B-11. Temperature measurement and Enumeration class measurement in Fixed Format

12. To enable Episoidic Scanner Measurement, go to **DIM** tab and select **Episoidic Scanner** on Continua Host. Click on **Enable Scanning** button.

Continua Hanager e Edit Help	,			
urrent State: Ope	rating			
DIM - Objects				Derice
Hetic Numeri Real Time Sa. Metric Brune Epsedic Sam. Periodic Scan. Imperature Medical Devic.	Type Type MOC_MOC_VMO. MOC_MOC_VMO. MOC_MOC_VMO. MOC_MOC_VMO. MOC_MOC_SCAN. MOC_MOC_SCAN. MOC_MOC_VMO. MOC_MOC_VMO.	1 8 7 Type 5 4 Enabl	HDC_HOC_SCAN_CFG_EPI d: OperationalState_disabled Enable Scanning	Device Informat
P	Con	tinua	≈LN ĩ	Transport: Enabled

Figure B-12. Enabling Episoidic Scanner

MEDCONLIB Users Guide, Rev. 5



13. Verify Episoidic Scanner is successfully enabled.

ent State: Operating					
III - Objects					
bject	Type	Handle			
 Metric Enumerated 	MDC_MOC_VMO_METRIC_NU MDC_MOC_VMO_METRIC_NU MDC_MOC_VMO_METRIC_SA_RT MDC_MOC_VMO_METRIC_ENUM	7	Туре:	MDC_MOC_SCAN_CFG_EPI	
Episodic Scanner Periodic Scanner bemperature Medical Device (MDS)	MDC_MOC_SCAN_OPG_ERI MDC_MOC_SCAN_OPG_PERI MDC_MOC_VMO_PHSTORE MDC_MOC_VMS_MDS_SIMP	5 3 4 0	Enabled:	OperationalState_enabled	
				Disable Scanning	1
itput					
	ontinua	. 🝣			Transport: Enabled

Figure B-13. Episoidic Scanner enabled

14. When push button **PTG3** is pressed, Episoidic Scanner measurement is sent in Grouped Format to host. Click on **Device** tab on Continua Host to see Episoidic Scanner data. Figure B-14 below shows measurement data on the host. If Episoidic Scanner is not enabled, then no data is sent to the host.

elect Shim Directory: D:/Progra	n Electro Boardon/	Browse Shart Transport
Device List	an unalizzazione kineri	Device Information
Device Name Transport	Device Specializatio Address Unknown, Unkno	Connected to: VASC-MDS System Hodel: Medical Dev System Handfacturer: Presscale Semiconductor System ID: 0x46-0x33-0x40-0x40-0x44-0x49-0x43-0x41-0x4C-0x30-0x31-0x00
		Image: State State - Timestamp: 6:16:00 am on Sun August 9 2009 Image: State State - Sody Temp:: 33.6214 *C Image: State State - Timestamp: 6:15:00 am on Sun August 9 2009 Image: State State - Timestamp: 6:15:00 am on Sun August 9 2009 Image: State State - Timestamp: 6:15:00 am on Sun August 9 2009 Image: State State State - Timestamp: 6:15:00 am on Sun August 9 2009 Image: State St
	ct Unassociate Abort	
Discover Discover Discover	t Unassociate Abort	<i>℁</i> ↔∲
Discover	t Unassociate Abort	

Figure B-14. Episoidic Scanner measurement

15. To disable Episoidic Scanner, go to **DIM** tab and select **Episoidic Scanner** on Continua Host. Click on **Disable Scanning** button.

ent State: Operating				
DIH - Objects				
Metric Enumerated Episodic Scanner Periodic Scanner temperature	Tee Tee, Hoc, Yoo, Yen Methic, NJ Hoc, Sooc, Yen Settinic, NJ Hoc, Hoc, Yen Settinic, NJ Hoc, Hoc, Yen Methic, Paul Hoc, Hoc, Carlo, Methic, Paul Hoc, Hoc, Carlo, Gel, She Hoc, Hoc, Wei, Mos, She	7 5 3 4 0	Type: HDC_HOC_SCAL_CPC Enabled: OperationalState_en Disable Scanning	
	ontinuo	. 🤏		Transport: Enabled

Figure B-15. Disabling Episoidic Scanner

MEDCONLIB Users Guide, Rev. 5

N



PAN USB Agent Demo

16. To enable Periodic Scanner Measurement, go to **DIM** tab and select **Periodic Scanner** on Continua Host. Click on **Enable Scanning** button.

e Edt Help	
arrent State: Operating	
DIM - Objects	Device
Object Type Handle - Netric Numeric MDC_MOC_VMO 2	â
Metric Numeric MDC_MOC_VMO 1 Real Time Sa MOC_MOC_VMO 8 Type: MDC_MOC_SCAN_CFG_PERI	Per
- Metric Enume MDC_MOC_VMO 7 - Episode Scan MDC_MOC_SCAN 5	Device Information Model
Periode Scan. MDC_MOC_SCAN. 3 Enabled: OperationalState_disabled	1
Medical Devic MDC_MOC_VMS 0 Period: 0	ation
	Mod
	2
Enable Scanning	
<u>د ا</u>	
Dutput	
	Transport: Enabled
🎸 Continua 🛛 🝣 🖬 🗳	mansport. Enabled
	·~ 🛞 🖧
	USB Bluetosth TCP/IP

Figure B-16. Enabling Periodic Scanner

17. Verify Periodic Scanner is successfully enabled.

ent State: Operating					
0IM - Objects					
 Metric Numeric Real Time Sample Array Metric Enumerated Episodic Scanner Periodic Scanner 	MDC_MOC_VMO_METRIC_NU MDC_MOC_VMO_METRIC_NU MDC_MOC_VMO_METRIC_SA_RT MDC_MOC_VMO_METRIC_SNAM MDC_MOC_SCAN_CPG_EPI MDC_MOC_SCAN_CPG_EPI MDC_MOC_SCAN_CPG_EPI	2 1 8 7 5 3 4	Type: Enabled:	MDC_MOC_SCAN_CFG_PERI OperationalState_enabled	
- Medical Device (MDS)	MDC_MOC_INS_MDS_SRP	0	Period:	0 Disable Scanning	
384638: \Associate.c(354)/ 384638: \VascPresentation/ 384638: \VascPresentation/ 384638: \AssociateQueue.c 384638: \AssociateQueue.c	es.c(395)Filter_TxAPOLExit with eff kc_sendAPOLExit with effeasilCod ApoL.gop(33)Vasc: PresentationAp (33)PopApoLExit y: AL handle is 0 (33)PopApoLExit (33)OpApoLExit (33)QThreadQThread: APOL ptr	tesultCode of [0x0] Success of [0x0] Success duc:-PresentationApduAttemptin duc:-PresentationApduAttemptin duc:-PresentationApduAttemptin er Iffc8	to free &data_apdu 0xd	31c88 with invoke id 3	

Figure B-17. Periodic Scanner enabled



 Periodic Scanner data is automatically sent by agent. Click on Device tab on Continua Host. Measurement data gets updated on screen periodically. Figure B-18 below shows sample Periodic Scanner data.

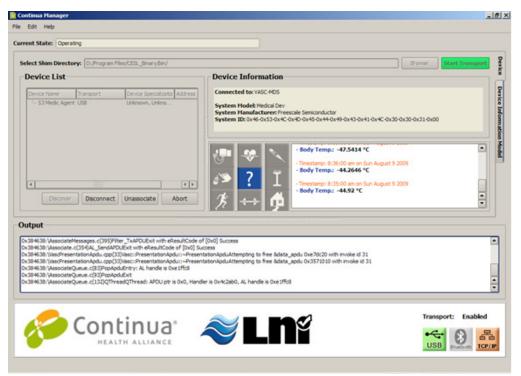


Figure B-18. Periodic Scanner measurement

19. To disable Periodic Scanner, go to **DIM** tab and select **Periodic Scanner** on Continua Host. Click on **Disable Scanning** button.



PAN USB Agent Demo

DIH - Objects Type Handle Metric Nameric MOC_MOC_IMO_VETRIC_UAL 2 Metric Nameric MOC_MOC_IMO_VETRIC_UAL 2 Metric Nameric MOC_MOC_IMO_VETRIC_UAL 1 Real Time Sample Array MOC_MOC_IMO_VETRIC_UAL 1 Real Time Sample Array MOC_MOC_IMO_VETRIC_UAL 1 Metric Numeric MOC_MOC_IMO_VETRIC_UAL Metric Numeric MOC_MOC_IMO_VETRIC_UAL Previote Samerie MOC_MOC_IMO_VETRIC_UAL Previote Samerie MOC_MOC_IMO_VETRIC_UAL Imperature MoC_SCARL_OFG_IVEL Medical Device (MOS) MOC_MOC_IMO_VETRICE Medical Device (MOS) MOC_MOC_IMO_VETRICE	
Metric Numeric NOC_MOC_WO_METRIC_NU 2 Rest: Numeric NOC_MOC_WO_METRIC_NU 1 Rest: Numeric NOC_MOC_WO_METRIC_SA_RT 1 Rest: Numeric NOC_MOC_WO_METRIC_SA_RT 1 Metric Numeric NOC_MOC_WO_METRIC_SA_RT 1 Metric Numeric NOC_MOC_WO_METRIC_SA_RT 1 Periods Source NOC_MOC_SOURCE_REST 5 Previde Source NOC_MOC_SOURCE_REST 5 Image/Device NOC_MOC_SOURCE_REST 5 Image/Device NOC_MOC_SOURCE_REST 6 Image/Device NOC_MOC_WO_METRIC_SOURCE_REST 4	
Disable Scanning	:Here
atput 134638: MascateMessages.c(3915)Pfiler_TxAPOUENt with eReal/Code of [0x0] Success 234538: Mascate.c135404, SecAPOUENt with eReal/Code of [0x0] Success 234538: MascherstatonLodu.cogO13/Masc.ThesentatonApdu:-PresentatonApduAttempting to firet Bdata_apdu Dx831c88 with invoke id 3 234538: MascherstatonLodu.cogO13/Wasc.ThesentatonApdu:-PresentatonApduAttempting to firet Bdata_apdu Dx831c88 with invoke id 3 234538: MascherstatonLodu.cogO13/Wasc.ThesentatonApdu:-PresentatonApduAttempting to firet Bdata_apdu Dx831c88 with invoke id 3 234538: MascateQueue.cl;359/Packd.thtrr X. Annuel is Dxe1f68 234538: MascateQueue.cl;359/Packd.thtr	

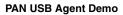
Figure B-19. Disabling Periodic Scanner

20. To fetch PM Store information, go to **DIM** tab and select **Temperature** (Name of PM Store) as shown in Figure B-20.



Figure B-20. PM Store Segment information

21. To fetch "seg2" data, click on seg2 and press Get Segment Data button.





DIM - Objects					
Object Type H		HOC_VHO_PHSTO	RE		
Metro Numeric NOC_VIOC_14/00_2 Metro Numeric NOC_VIOC_14/00_1 Real Time Sample Array NOC_VIOC_14/00_8 Metro Enumerated NOC_VIOC_VIOC_9 Periode Scarmer NOC_VIOC_SCA14_5 Periode Scarmer NOC_VIOC_SCA14_5 Imperature NOC_VIOC_SCA14_5 Medical Device (HOS) MDC_VIOC_VINS_0_0	seg2 seg1	Instance #	Op State 0 0	Segment Data	
()			Click Here		
atput 38458: VascMessageFactory.cpp(214)Vasci Mess 38458: VascMessageFactory.cpp(221)Vasci Mess 38458: VascMessageFactory.cpp(221)Vasci Mess 38458: VascMessageFactory.cpp(231)Vasci Mess	sageFactory::messag sageFactory::messag	e_table::findPosting R e_table::findExit	leceiving RorsGetSegment[InfoActionPristApdu.	
(384658: \AssociateQueue.c(83)PopApduEntry: AL (384658: \AssociateQueue.c(93)PopApduEntr (384658: \AssociateQueue.c(132)QThreadQThread (38b578: \\AssociateQueue.c(132)QThreadQThread (38b578: \\AssociateQueue.c(132)QThreadQThread (38b578: \\AssociateQueue.c(132)QThreadQThread (38b578: \\AssociateQueue.c(132)QThreadQThread (38b578: \\AssociateQueue.c(132)QThreadQThreadQThread (38b578: \\AssociateQUeue.c(132)QThreadQThreadQThread (38b578: \\AssociateQUeue.c(132)QThreadQThre	handle is Oxde+640 9: APOU ptr is Ox0, Ha	nder is 0x4c2ab0, AL	handle is 0xde4640		
			Lní		Transport: Enabled

Figure B-21. Fetch PM Segment data

22. PM Segment "seg2" data is displayed on Continua Host as shown in Figure B-22.

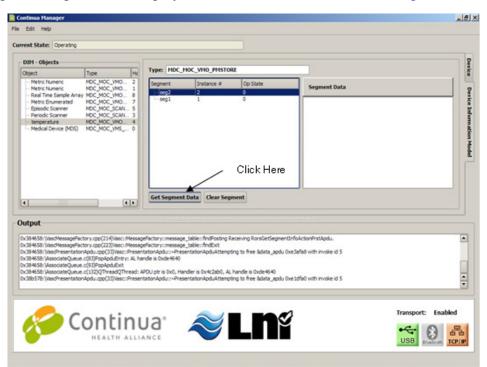


Figure B-22. PM Segment data

23. To delete PM Segment ("seg1"), click on **seg1** and then press **Clear Segment** button on Continua Host.



PAN USB Agent Demo

DIH - Objects Object Object Type Metric Nameric MCC_MOC_MOD_1 Metric Nameric MCC_MOC_MOD_1 Metric Drumerated MCC_MOC_MOD_0 Metric Drumerated MCC_MOC_SCAL_3 Bemerative MCC_MOC_SCAL_3 Metric Scamer MCC_MOC_SCAL_3 Metric Scamer MCC_MOC_SCAL_3 Metric Scamer MCC_MOC_SCAL_3 Metric Scamer MCC_MOC_SCAL_3 Get Segment Data Click Here	
Opect Instance © Opect Segment Segment Data Metric Numeric MOC_MOC_MOD_1 Eggment Instance # Op State Metric Numeric MOC_MOC_MOD_1 Eggment Instance # Op State Metric Summer MOC_MOC_MOD_7 Eggment Instance # Op State Specific Summer MOC_MOC_MOD_7 Eggment Instance # Op State Segment MOC_MOC_MOD_7 Eggment Instance Click Here Eggment Segment MOC_MOC_MOD_6 MOC_MOC_MOD_7 Eggment Click Here Eggment Eggment	
Metric Numeric MOC_MOC_WO_ 1 100 Motion 100 Motion	
Intel Sampe Array MSC MCL MOL 10 Method Damene MPC MOC SCAN_3 Bronds Sammer MPC MOC SCAN_3 Brenderstare Medical Device (MOS) MCC MOC (MOL 4) Click Here	
Method Dame and MC_MC_MC_MC_7 Product Scame MC_MCC_SCAN_3 Product Scame MC_MCC_SCAN_3 Method Device (MCG) MCC_MC_4 Method Device (MCG) MCC_MC_4	
Periode Seamerer MOC MOC SEAM. 3 Benerentare MOC MOC MOC 4 Medical Device (MDS) MDC MOC (MS_ 0 Click Here	
Medcal Device (MDS) MDC_MOC_UMS 0 Click Here	
Click Here	
Get Segment Data Clear Segment	
put	
84653: WasdMessageFactory.cpp(214)Wasc::MessageFactory::message_table::findPosting Receiving RorsGetSegmentInfoActionPrstApdu. 84658: WasdMessageFactory.cpp(223)Wasc::MessageFactory::message_table::rfindExit	
94658: WasdPresentationApdu.cpp(33)Wasc::PresentationApdu::-PresentationApduAttempting to free 8data_apdu 0xe3afa8 with invoke id 5	
84658: VAssociateQueue.c(83)PopApduEntry: AL handle is 0xde4640 84658: VAssociateQueue.c(93)PopApduEntr	
HeS3: (AssociateQueue.c)(32)QThread(Thread: APDU ptr is 0x0, Handler is 0x4c2ab0, AL handle is 0x4c4640	
86578: (Vas:PresentationApdu.cpp(33)Vasc::PresentationApdu::-+PresentationApduAttempting to free 8data_apdu 0xe18fa0 with invoke id 5	

Figure B-23. PM Segment delete

24. After PM Segment is deleted, the Continua Manager is updated and "seg1" entry is removed.

Objects Type: Handle Objects Type: Handle Metric Numeric MOC_NOC_XMO_NETRIC_NU 1 Segment Dratance # Op State Metric Numeric MOC_NOC_XMO_NETRIC_SAUR # 1 Metric Numeric MOC_NOC_XMO_NETRIC_SAUR # 1 Metric Numeric MOC_NOC_XMO_NETRIC_SAUR # 1 Metric Drune ated MOC_NOC_SCAUR OFER PST 3 Metric Numeric MOC_NOC_SCAUR OFER PST 3 Bengerature MOC_NOC_SCAUR OFER PST 3 Medical Device (MDS) MOC_NOC_SCAUR OFER PST 3 Medical Device (MDS) MOC_NOC_SCAUR OFER PST 3 Medical Device (MDS) MOC_NOC_SCAUR OFER PST 4 Image: Image	rent State: Operating							
Wettic Numeric MCC_MOC_MO_METRIC_NU 1 Mettic Numeric MCC_MOC_MO_METRIC_NU 1 Real Time Singhe Array MCC_MOC_MO_METRIC_NU 1 Ensolis Gammer MCC_MOC_MO_METRIC_SALT 8 Mettic Drunested MCC_MOC_SCAL_OF_GIPII Segment MCC_MOC_SCAL_OF_GIPII Emperature MCC_MOC_SCAL_OF_GIPII Immediation MCC_MOC_SCAL_OF_GIPIII Medical Device (MOS) MCC_MOS_SSDPP Medical Device (MOS) MCC_MOS_SSDP MCC_MOS_SSDP MCC_MOS_SSDP MCC_MOS_MOS_SSDP MCC_MOS_SSDP	DIH - Objects		_	The Labor 14	or 1810 886708			
Metric Numeric MDC_MOC_M0_9ETRIC_MU 1 sept 2 0 segment Data Segment Data Segment Data Segment Data				Type: Plux_P	OC_VHO_PHSTOR	a		
Medical Device (NDS) MDC_NOC_UMS_NDS_SIMP 0	Metric Numeric Real Time Sample Arra Metric Enumerated Episodic Scanner	MDC_MOC_VMO_METRIC_NU yy MDC_MOC_VMO_METRIC_SA_RT MDC_MOC_VMO_METRIC_ENUM MDC_MOC_SCAN_CPG_PRI MDC_MOC_SCAN_CPG_PRI	1 8 7 5 3				Segment Data	
Get Segment Data Clear Segment								
	1		••	Get Segment		nent		

Figure B-24. PM Segment delete verification



Appendix C PAN Serial Bridge Demo

The Serial Bridge demo demonstrates the PAN device demo working on a serial agent. The setup consists of two EVB51JM128 boards. The PHDC Medical Connectivity Library runs on one board with a serial agent. This device communicates with the Continua Host via Serial Bridge which is running on the other board. Continua Host Software detects Serial Bridge Device as PHDC Medical Connectivity Library USB Agent.

C.1 Setting Up the Demo

1. Set the systems as shown below in Figure C- 1. The Serial connection between the two EVB51JM128 boards uses COM1 port.

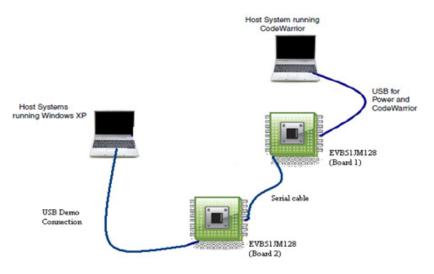


Figure C-1. Hardware Setup

- 2. Get the Continua Alliance (www.continuaalliance.org) CESL Reference Software V1.0 RC2.
- 3. Install the software on a host system
- 4. Program EVB51JM128 flash of Board 1 with the PHDC Serial Agent Demo application using CodeWarrior IDE.
- 5. Program the EVB51JM128 flash of Board 2 with the Serial Bridge demo application using CodeWarrior IDE.

NOTE

CESL reference software is not provided as part of the suite. You will have to get this software independently from Continua Alliance.



C.2 Running the Demo

After the system has been set, you must follow these steps to run the demo:

- 1. Power ON Serial Agent (Figure C-1 board 1).
- 2. Power ON Serial bridge (Figure C-1 board 2).
- 3. To run the demo, follow the steps given in Section B.2, "Running the Demo." The push buttons used to send measurement data are that of the Serial Agent (Figure C-1 board 1). The Push button panel of EVB51JM128 is shown below in Figure C-2.

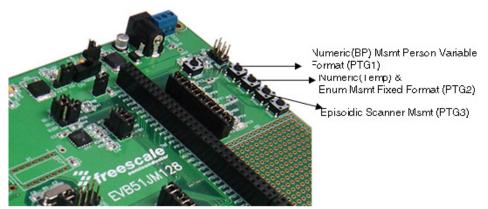


Figure C-2. EVB51JM128 Push Button Panel



Appendix D PHDC Manager Demo Application

D.1 Setting up the demo

D.1.1 Hardware setup

This demo runs on M52259DEMO ColdFire board and follows the hardware setup below:

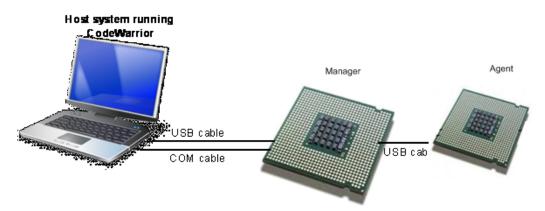


Figure D-1. PHDC Manager demo setup

D.1.2 Set up HyperTerminal to get log

To ensure that application run correctly, the HyperTerminal is used on the PC to get events from the device. These steps are used to configure HyperTerminal:

1. Open HyperTerminal application as shown in Figure D-2

All Programs 🜔	Accessories	•	Communications	HyperTerminal
Remote Desktop Cor	Microsoft Office Live Meeting	•	G WordPad	🗳 Wireless Network Setup Wizard
Remote Desktop Cor	RealVIVC	•	Windows Movie Maker	Remote Desktop Connection
Microsoft Office Exce	m Mozilla Firefox	•	Windows Explorer	New Connection Wizard
Paint			Tour Windows XP	Network Setup Wizard
Command Prompt			i Synchronize	Network Connections
😂 HyperTerminal				S HyperTerminal

Figure D-2. Launch HyperTerminal application

2. The HyperTerminal opens as shown in Figure D-3. Enter the name of the connection and click on the OK button.



New Connection - HyperTermina File Edit View Call Transfer Help ご デ 会 る 単 音 留	l						
	Enter <u>N</u> ame	_19200	e an icon for		? > > Cancel		
Disconnected Auto detect	Auto detect	SCROLL CAR	PS NUM	Capture	Print echo		

Figure D-3. Hyper Terminal



3. The window shown in the following figure appears. Select the COM port.

🗞 COM_19200 - HyperTerminal		
File Edit View Call Transfer Help		
Pie Lat View Call Iransfer Hep	Connect To COM_19200 Enter details for the phone number that you want to dial: County/region: United States (1) Area code: 123 Phone number: Connect using: COM1 COM1 COM1 COM1 COM1 COM1 COM1 COM1 COM1 COM1	
Disconnected Auto detect Auto detect	ct SCROLL CAPS NUM Capture Print echo	

Figure D-4. Connect using COM port

4. Configure the COM port baud rate and other properties as shown in Figure D-5



COM1 Properties	? 🛛
Port Settings	
<u>B</u> its per second:	19200
<u>D</u> ata bits:	8
<u>P</u> arity:	None
<u>S</u> top bits:	1
Elow control:	None
	<u>R</u> estore Defaults
	K Cancel Apply

Figure D-5. COM properties

5. The HyperTerminal is now configured as shown in Figure D-6



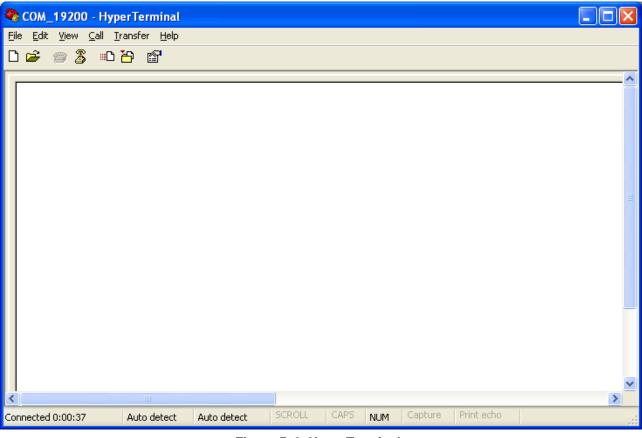


Figure D-6. HyperTerminal

D.1.3 Running the demo

Perform the following steps to run demo:

- 1. Open and load the image of PHDC manager demo application to the board.
- 2. After the image has been loaded successfully, HyperTerminal appears as shown in Figure D-7



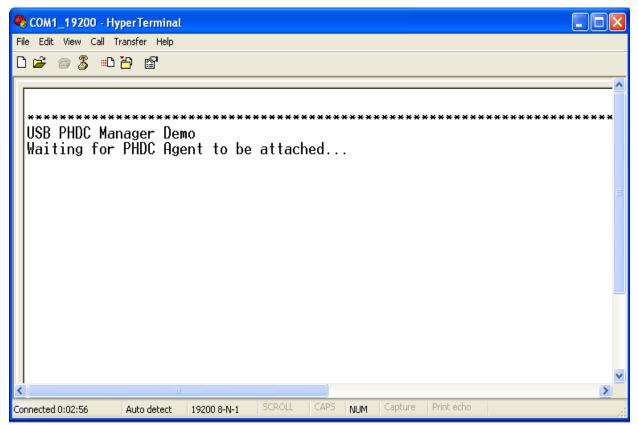


Figure D-7. Using PHDC Manager Demo attached event

3. Plug a PHDC agent device into the board. The phdc device examples which is included in this package can be used as PHDC agent device. The agent will be associated with manager and the manager enters operating state.

le Edit Yiew ⊆	jal Iransfer Help
1 🖙 👘 🏅	10 2 9 2 7 0

USB PHDC	Manager Demo
	for PHDC Agent to be attached
	tach Event PHDC manager state
11073MNG:	ENTER UNASSOCIATED state
11073MNG:	ENTER ASSOCIATING state Agent type
11073MNG:	ENTER OPERALING state
	: Type = 4104 (Thermometer) Version = 1 Agent Report
	Received a Scan Report (MPVariable) event.
	Scan Report Num: 0 Num Observations: 2
	: Person ID = 1
	=>> Object type: NOT SUPPORT NOMENCLATURE STRING, Partition: MDC_PART.
SCADA	with the second se
	: Observation value: 123.000000 83.000000 103.000000 mm Hg
	==>> Object type: NOT SUPPORT NOMENCLATURE STRING, Partition: HDC_PART.
SCADA	
	: Absolute Timestamp = 2009-08-09 06:33
	: Person ID = 2
11073MNG	:==>> Object type: NOT SUPPORT NOMENCLATURE STRING, Partition: MDC_PART.
SCADA	
	: Observation value: 124.000000 84.000000 104.000000 mm Hg
11073MN6	:==>> Object type: NOT SUPPORT NOMENCLATURE STRING, Partition: MDC_PART

Figure D-8. USB PHDC Manager Demo operating



PHDC Manager Demo Application