Omnii HDK
User Manual

(Omnii XT10)

December 8, 2010 P/N 8100210.A

ISO 9001 Certified
Quality Management System

PSION TEKLOGIX

2100 Meadowvale Boulevard, Mississauga, Ontario, Canada L5N 7J9

http://www.psionteklogix.com

Disclaimer

Every effort has been made to make this material complete, accurate, and up-to-date. In addition, changes are periodically added to the information herein; these changes will be incorporated into new editions of the publication.

Psion Teklogix Inc. reserves the right to make improvements and/or changes in the product(s) and/or the program(s) described in this document without notice, and shall not be responsible for any damages, including but not limited to consequential damages, caused by reliance on the material presented, including but not limited to typographical errors.

Omnii™ is a trademark of Psion Teklogix Inc.

Windows® and the Windows Logo are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.

*The **Bluetooth**® word mark and logos are owned by Bluetooth SIG, Inc. and any use of such marks by Psion Teklogix Inc. is under license.*

All trademarks and trade names are the property of their respective holders.

Return-To-Factory Warranty

Psion Teklogix Inc. provides a return to factory warranty on this product for a period of twelve (12) months in accordance with the Statement of Limited Warranty and Limitation of Liability provided at:

www.psionteklogix.com/warranty

The warranty on Psion Teklogix manufactured equipment does not extend to any product that has been tampered with, altered, or repaired by any person other than an employee of an authorized Psion Teklogix service organization. See Psion Teklogix terms and conditions of sale for full details.

Important: Psion Teklogix warranties take effect on the date of shipment.

Service and Information

Psion Teklogix provides a complete range of product support services and information to its customers worldwide. Services include technical support and product repairs. To locate your local support services, please go to *www.psionteklogix.com/service-and-support.htm*

To access further information on current and discontinued products, please go to *http://community.psionteklogix.com/login.aspx?ReturnUrl=%2fdefault.aspx* and log in. A section of archived product information is available online.

TABLE OF CONTENTS

Chapter 5: Mechanical Considerations

Chapter 6: Omnii Expansion Ports and Connectors

Chapter 7: Docking Stations

Chapter 8: EEPROM Specifications

Chapter 9: Breakout Board

1 INTRODUCTION

1.1 About This Manual

This manual provides guidance on creating expansion devices for Psion Teklogix Omnii devices using the Omnii HDK. The manual is organised into the following chapters:

Chapter 1: Introduction

provides an overview of the Omnii Hand-Held Computer and the Omnii HDK.

Chapter 2: Getting Started

provides at-a-glance information about the capabilities of the HDK.

Chapter 3: Hardware

describes, in general terms, the hardware of Omnii.

Chapter 4: Software

gives an overview of the registry entries and API for controlling expansion devices and the installation of device drivers.

Chapter 5: Mechanical Considerations

describes the physical aspects of designing and mounting expansion modules.

Chapter 6: Omnii Expansion Ports and Connectors

describes the connectors on Omnii, including the three standard expansion ports and docking adaptors.

Chapter 8: EEPROM Specifications

describes the details of programming I2C and 1-wire EEPROMs for expansion modules.

Chapter 9: Breakout Board

describes the features and functions of the Omnii HDK breakout board kit (sold separately).

Chapter 10: HDK Demo Application

describes the features and functions of the HDK Demo application program.

Appendix A: Resources

lists extra resources which may be of use in conjunction with the HDK.

Appendix B: Omnii Specifications

lists the specifications of Omnii.

Appendix C: HDK License Agreement

provides the license agreement that is assumed by using the Omnii HDK.

1.2 Text Conventions

The following conventions and syntax are followed throughout this document:

Note: Notes highlight additional helpful information.

Important: These statements provide important instructions or additional information that is critical to the operation of the computer or other equipment.

Warning: These statements provide important information that may prevent injury, damage to the equipment, or loss of data.

An arrow next to field description information (usually in tables) indicates a recommended or suggested configuration setting.

1.3 About the HDK

The Omnii HDK (Hardware Development Kit) provides the software tools and technical information necessary to design and integrate your own expansion modules for your Omnii hand-held computer.

Three dedicated expansion ports provide access to USB, serial and GPIO (General Purpose Input/Output) interfaces for connecting to standard devices (bar code scanners, imagers, RFID readers, etc.). An audio expansion port is also available for attaching speaker and/or microphone devices.

3D model files and schematic drawings are provided which give the precise measurements needed for designing custom back-cover, end-cap and pod expansion modules that fit and seal perfectly with the main housing.

Finally, the Omnii HDK API library provides the software tools necessary to access and control the expansion ports, and the devices attached to them.

1.4 Development Platform

The Omnii API library is designed for application development using Visual Studio 9 (2008).

1.5 Contents of the HDK

The HDK (Hardware Development Kit) for Omnii includes the following items:

- This manual.
- Installer for development files, including C header files for managing expansion devices and HDK Demo application. See Section 4.7: “Omnii HDK Application Development Software” and Chapter 10: “HDK Demo Application” for more details on these files.
- 2D drawings and 3D models of the areas of Omnii where devices and modules can be mounted.

1.5.1 Files in the HDK

The following files are included with the Omnii Hardware Development Kit:

Table 1.1 Files in the HDK

Filename	Description
OmniiHDK_Setup	Installer for the Omnii HDK development files. Installs the following files on a PC:
Hdk7545.h	C header file containing API functions for the Omnii HDK
Hdk7545Structs.h	C header file containing API structures for the Omnii HDK
Hdk7545Consts.h	C header file containing API constants for the Omnii HDK
7545HDK.dll	
7545HDK.exp	
7545HDK.lib	
7545HDK.pdb	
HDKDemo.exe	Demo application for testing and demonstrating features and components of the HDK
Audio_External_Speaker.pdf	Reference design for an external speaker audio expansion
Audio_PTT.pdf	Reference design for a push-to-talk handset audio expansion
Audio_Single-Ended_Headset.pdf	Reference design for a single-ended headset audio expansion
Back_Cover_Auto_Std_2D.dwg	2D line drawing of the auto-range standard back cover with locations of mounting points for the back cover and pistol grip
Back_Cover_Auto_Std_2D.pdf	2D line drawing of the auto-range standard back cover with locations of mounting points for the back cover and pistol grip
Back_Cover_Auto_Std_3D.igs	3D CAD model of the auto-range standard back cover
Back_Cover_Auto_Std_3D.stp	3D CAD model of the auto-range standard back cover
Back_Cover_Expan_2D.dwg	2D line drawing of the expansion back cover with locations of mounting points for the back cover, pistol grip, end-cap and pod expansion
Back_Cover_Expan_2D.pdf	2D line drawing of the expansion back cover with locations of mounting points for the back cover, pistol grip, end-cap and pod expansion
Back_Cover_Expan_3D.igs	3D CAD model of the expansion back cover, showing end-cap and pod expansion openings with sealing overmoulds, and keep-away areas for camera and speaker options
Back_Cover_Expan_3D.stp	3D CAD model of the expansion back cover, showing end-cap and pod expansion openings with sealing overmoulds, and keep-away areas for camera and speaker options
Back_Cover_Large_Std_2D.dwg	2D line drawing of the large standard back cover with locations of mounting points for the back cover and pistol grip
Back_Cover_Large_Std_2D.pdf	2D line drawing of the large standard back cover with locations of mounting points for the back cover and pistol grip

Table 1.1 Files in the HDK

Filename	Description
Back_Cover_Large_Std_3D.igs	3D CAD model of the large standard back cover
Back_Cover_Large_Std_3D.stp	3D CAD model of the large standard back cover
Breakout_Sch.pdf	Schematic diagrams of the HDK breakout board
DS2431.pdf	Data sheet for the Maxim DS2431 1-wire EEPROM
Endcap_GPS_2D.dwg	2D line drawing of the end-cap with GPS antenna
Endcap_GPS_2D.pdf	2D line drawing of the end-cap with GPS antenna
Endcap_GPS_3D.igs	3D CAD model of the end-cap with GPS antenna
Endcap_GPS_3D.stp	3D CAD model of the end-cap with GPS antenna
Endcap_Standard_2D.dwg	2D line drawing of the standard end-cap
Endcap_Standard_2D.pdf	2D line drawing of the standard end-cap
Endcap_Standard_3D.stp	3D CAD model of the standard end-cap
KB_HardCaps_36ModNumCal12.pdf	Artwork for the 36-key, alpha modified, numeric calculator, 12 Fn keyboard hard caps
KB_HardCaps_36NumTel12.pdf	Artwork for the 36-key numeric telephony, 12 Fn keyboard hard caps
KB_HardCaps_59ABCTel6.pdf	Artwork for the 59-key, alpha ABC, numeric telephony, 6 Fn keyboard hard caps
KB_Overlay_36ModNumCal12.pdf	Artwork for the 36-key, alpha modified, numeric calculator, 12 Fn keyboard overlay
KB_Overlay_36NumTel12.pdf	Artwork for the 36-key numeric telephony, 12 Fn keyboard overlay
KB_Overlay_59ABCTel6.pdf	Artwork for the 59-key, alpha ABC, numeric telephony, 6 Fn keyboard overlay
Omnii_Connectors_NoRadio_2D.dwg	2D line drawing of Omnii chassis and MLB showing locations of expansion ports with **no** GPS or WWAN radio installed
Omnii_Connectors_NoRadio_2D.pdf	2D line drawing of Omnii chassis and MLB showing locations of expansion ports with **no** GPS or WWAN radio installed
Omnii_Connectors_NoRadio_3D.stp	3D CAD model of Omnii chassis with back cover removed showing keep-away areas, with **no** GPS or WWAN radio installed
Omnii_Connectors_Radio_2D.dwg	2D line drawing of Omnii chassis and MLB showing locations of expansion ports with GPS and WWAN radio installed
Omnii_Connectors_Radio_2D.pdf	2D line drawing of Omnii chassis and MLB showing locations of expansion ports with GPS and WWAN radio installed
Omnii_Connectors_Radio_3D.stp	3D CAD model of Omnii chassis with back cover removed showing keep-away areas, with GPS and WWAN radio installed
Scanner_Pod_Std_2D.dwg	2D line drawing of the standard scanner pod with locations of mounting points for the pod and for the scanner assembly

Table 1.1 Files in the HDK

Filename	Description
Scanner_Pod_Std_2D.pdf	2D line drawing of the standard scanner pod with locations of mounting points for the pod and for the scanner assembly
Scanner_Pod_Std_3D.igs	3D CAD model of the standard scanner pod
Scanner_Pod_Std_3D.stp	3D CAD model of the standard scanner pod

1.6 Obtaining the HDK

The Omnii HDK is available for download on the Psion Teklogix Community website (http://community.psionteklogix.com). You will need an account on the website in order to download files. An account can be easily created by clicking on the **Join** link in the upper right corner of the home page.

To download the HDK:

1. Click on the **Downloads** link in the top bar of the Community home page.
2. Click on **Psion Teklogix HDK** in the list that appears.
3. Click on **Hardware Development Kit (HDK) for** Omnii.
4. Click on the link to view the license agreement and download the .zip file containing the HDK files.
5. Open the .zip file and extract the files within to a folder on your PC hard drive.

To continue with installing the HDK files required for developing applications to work with your expansion devices, see Section 4.7: “Omnii HDK Application Development Software”.

1.7 About the Omnii Hand-Held Computer

Omnii is an industrial hand-held computer. It has a modular design that allows for many variations and combinations of the component modules. Currently, Omnii XT10 is the only model available; information on future Omnii models will be added to this document as they are released.

For more information on the Omnii XT10 operation and hardware variants, refer to the Omnii XT10 Hand-Held Computer User Manual (P/N 8100190).

Omnii XT10 uses the Microsoft® Windows® CE 6.0 operating system.

GETTING STARTED 2

2.1 Overview

This section gives a brief look at what the Omnii HDK can be used for, some quick links to the relevant sections of this manual for each task, as well as some basic information on what is required to develop working devices using the HDK.

2.2 What Can I Do With the Omnii HDK?

The information provided in the Omnii HDK allows you to:

- design custom end-cap, pod and back cover expansion modules that fit and seal precisely with your Omnii hand-held computer.
  - Section 5.4: “Expansion Module and Device Design and Installation”
- install non-Psion Teklogix serial, USB or GPIO devices in existing Psion Teklogix end-cap, pod and back cover modules.
  - Section 5.5: “Installing Devices Inside Existing Modules”
  - Section 6.4: “Expansion Ports”
  - Section 4.4.1: “Registry Settings for Expansion Devices”
  - Chapter 8: “EEPROM Specifications”
- design custom keyboard overlay and hard cap artwork.
  - Section 5.4.6: “Keyboard Modules”
- design custom audio devices.
  - Section 6.3: “Audio Connector”
- design custom pistol grips.
  - Section 5.4.5: “Pistol Grip Modules”
- design custom devices that connect to Psion Teklogix desktop docking stations.
  - Chapter 7: “Docking Stations”

2.3 Expansion Areas

The following illustrations show the areas where custom expansion modules can be mounted on your Omnii hand-held computer:

End-Cap

Pod

Back Cover

2.4 Expansion Device Requirements

2.4.1 Device EEPROM

It is highly recommended that any expansion device attached to an Omnii expansion port be equipped with a Maxim DS2431 EEPROM. This 1-wire EEPROM is used to store specific data that identifies the expansion device to Omnii. On boot-up, Omnii reads the EEPROM of all devices attached to the expansion ports, and searches for corresponding entries in the registry that specify the device driver(s) and port configuration to use for that device.

Details on this EEPROM can be found in Chapter 8: “EEPROM Specifications”, and instructions on reading and writing the EEPROM using the HDK Demo application can be found in Section 10.6.2.4: “EEPROM”.

2.4.2 Device Registry Keys

For each expansion device, certain registry keys and values must be added to the Omnii registry. These registry keys provide Omnii with the information needed to load the required device driver(s) and to configure the data pins of the expansion port appropriately.

Details on these registry keys can be found in Section 4.4: “Registry Keys”, and specific examples are given in Section 10.4: “Creating Registry Keys”.

HARDWARE 3

3.1 Overview

This chapter gives an overview of the hardware of Omnii.

3.2 Hardware Variants

Omnii has variant modules for the display, keyboard, back cover and bar code scanner or imager pod attachment.

3.2.1 Display Variants

Omnii has two standard variants for the LCD touchscreen display:

High Impact Display

The high impact display withstands impacts up to 1.2 joules and consists of separate overlaid panels for the LCD display and touchscreen surface.

High Visibility Display

The high visibility display withstands impacts up to 0.4 joules provides increased visibility in sunlight and uses a single combined LCD and touchscreen panel.

3.2.2 Keyboard Variants

Omnii has three standard variants for the keyboard module: two numeric variants, and one full alphanumeric variant.

36-Key, Numeric Telephony, 12Fn Keyboard

This numeric keyboard has the number keys arranged telephone-style, with the numbers *1*,*2*,*3* along the top row. The alphabetic characters are also arranged telephone-style, in groups of 3 or 4 [FN]-shifted characters above the number keys. It has 24 function keys (12 single-press and 12 shifted), and five macro keys available.

36-Key, Alpha Modified, Numeric Calculator, 12 Fn Keyboard

This numeric keyboard has the number keys arranged calculator-style, with the numbers *7*,*8*,*9* along the top row. The alphabetic characters are located as single [FN]-shifted characters on individual keys across the entire keyboard. It has 24 function keys (12 single-press and 12 shifted) available.

59-Key, Alpha ABC, Numeric Telephony, 6 Fn Keyboard

This full alphanumeric keyboard has the numeric keys arranged telephone-style with the numbers *1*,*2*,*3* along the top row, and the alphabetic keys arranged in order from *A* to *Z*. It has 30 function keys (6 single-press and 24 shifted), and 6 macro keys available.

36-Key Numeric Telephony

36-Key Alpha Modified Numeric Calculator

59-Key Alpha ABC Numeric Telephony

3.2.3 Back Cover Variants

Omnii has three standard variants for the back cover module.

- Expansion Back Cover - a slim back cover with separate end-cap, and a back plate that can be removed for an optional pod expansion module (see Section 3.2.4: “Scanner/Imager Variants”). This back cover can also have an optional built-in speaker and/or camera, which need to be considered when designing modules.
- Large Standard Back Cover - a single-piece back cover module that also covers the end-cap area and accommodates an integrated scanner or imager (see Section 3.2.4: “Scanner/Imager Variants”).
- Auto-Range Standard Back Cover - similar to the Large Standard Back Cover but with different options for the integrated scanner or imager (see Section 3.2.4: “Scanner/Imager Variants”).

3.2.4 Scanner/Imager Variants

Omnii comes standard with no scanner or imager installed. A scanner or imager engine can be installed in a pod expansion module and mounted to the expansion back cover. Alternatively, a scanner or imager could be mounted in the end-cap of the unit, with an appropriately designed end-cap. The scanner or imager engine attaches to an expansion port on the Omnii main logic board through a flex cable.

Only one internal bar code scanner or imager can be installed in Omnii at a time. The internal scanner or imager can be activated from the trigger switch on the Omnii pistol grip (if present), from the [SCAN] buttons on the Omnii keyboard and side panels, or from another software-assigned key on the keyboard.

The following scanner/imager options are currently available to order from the factory:

Expansion Back Cover with Scanner Pod Module

- SE1223LR long range bar code scanner
- SE1224HP high performance bar code scanner
- EV15 1D imager
- 5080 2D imager

Auto-Range Standard Back Cover

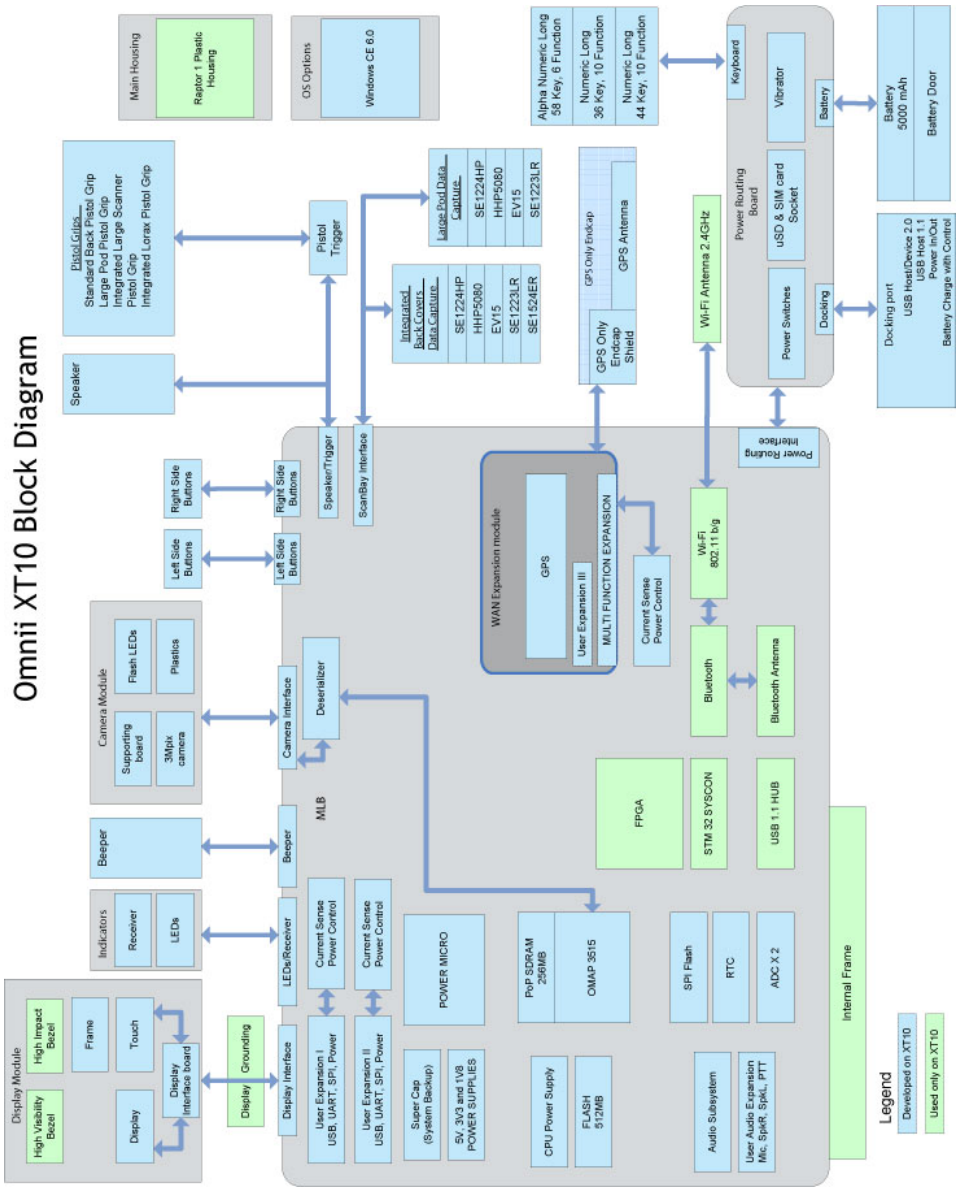
- SE1223LR long range bar code scanner
- SE1524ER auto-range bar code scanner

Large Standard Back Cover

- SE1224HP high performance bar code scanner
- EV15 1D imager
- 5080 2D imager

3.3 Processor

Omnii XT10 is built around a Texas Instruments OMAP3515 600MHz ARM Cortex-A8 dual-core processor.



3.4 Identifying Hardware

An overview of the operating system and the installed hardware on Omnii can be viewed by opening the System applet in the Windows Control Panel.

3.5 The LEDs

Omnii has four LEDs in the top left corner of the display bezel. From left to right the colors of the LEDs are green/red/yellow, yellow, blue, and green/red/yellow. The yellow LED (second from the left) can be controlled by applications with the Windows API; the name of this LED for programming purposes is “Application”.

3.6 Connectors

In addition to the external docking connector on the base of the Omnii hand-held computer, the following connectors exist on the main logic board:

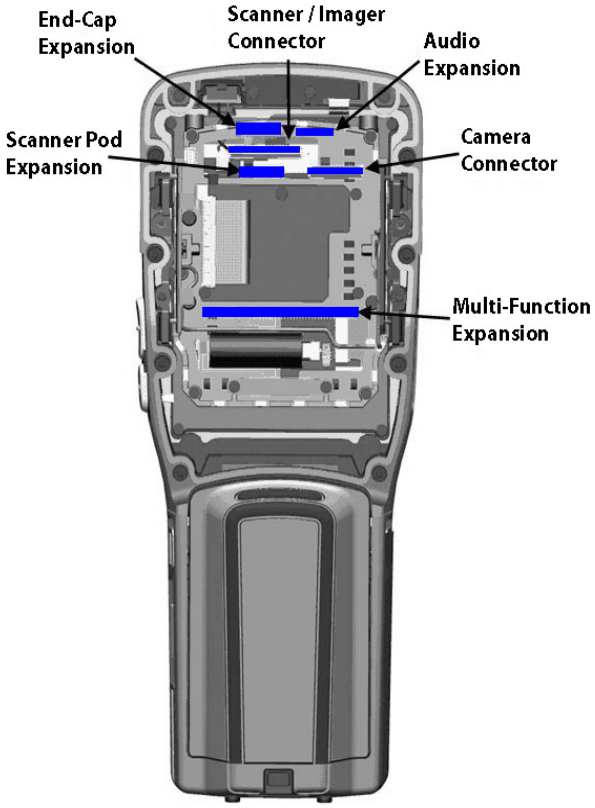
- Audio Expansion Connector
- End-cap Expansion Connector (Expansion Port 1)
- Pod Expansion Connector (Expansion Port 2)
- 100-Pin Multi-Function Connector (includes Expansion Port 3)

These connectors are described in detail in Chapter 6: “Omnii Expansion Ports and Connectors”.

3.6.1 Connector Locations

The following illustration shows the positions of the electrical connectors on the Omnii XT10 main logic board. These locations are shown in more precise detail in the drawing and model files included with the HDK. See Section 5.4.1: “Physical Space Considerations” for details.

Figure 3.1 Connector Locations of the Omnii XT10 Main Logic Board



Note: The camera and scanner/imager connectors are not intended for third-party expansion on Omnii XT10, and are shown here for reference only.

3.7 Power Management

Omnii is powered by a lithium-ion rechargeable battery pack and can also be powered from external power. When Omnii is powered from external power, the battery pack also charges.

Use only power sources recommended or sold by Psion Teklogix for Omnii.

3.7.1 Batteries

The battery is a custom 5000 mAh lithium-ion cylindrical multi-cell pack that fully implements a Smart Battery Specification and is CTIA approved (Model No. ST3000). The battery is fully sealed and designed to survive in rugged environments. For more details, refer to the Omnii Hand-Held Computer User Manual (P/N 8000190).

SOFTWARE 4

4.1 Overview

This chapter describes the software aspects of controlling expansion modules for Omnii.

4.2 Drivers

4.2.1 Windows Drivers

The Peripherals Driver

Psion Teklogix provides the peripherals driver for all expansion and docking peripherals. The peripherals driver is a stream driver activated very early at boot up time.

The Serial Port Driver

The full-function UART (Universal Asynchronous Receiver/Transmitter) serial port driver is loaded if required, as determined by the registry settings for any expansion devices detected. For details on the registry settings, see Section 4.4.1 on page 26.

4.2.2 Non-Psion Teklogix Drivers

The Psion Teklogix platform loads some standard device drivers. If the expansion module uses standard drivers such as serial or USB, there is no need to load custom drivers.

There must be a registry entry for the driver and its parameters. For details see Section 4.4: “Registry Keys”.

4.3 System Initialization

During system startup on Omnii, the following sequence occurs:

1. The expansion module EEPROMs are initialized and read.
2. If the registry entry for a detected expansion device indicates it is a serial device, the full-function UART (FFUART) serial port driver is loaded and activated.
3. If the USB flag is set in the registry entry for a detected expansion device, the USB hub is activated.
4. All drivers are identified from the EEPROM data and activated. For more details see Section 4.4.1: “Registry Settings for Expansion Devices”.

4.4 Registry Keys

4.4.1 Registry Settings for Expansion Devices

For an expansion device to be properly detected by the peripherals driver, and to have the correct drivers loaded to support the device, registry keys must be added to the Omnii registry.

Creating Registry Entries for a Device

The basic steps for creating registry entries for a device are outlined below. The individual features of each step are explained in more detail in the sections that follow.

1. Using a registry editor, locate the following key in the Omnii registry:

   ```
   [HKLM\Drivers\BuiltIn\Peripherals\devices]
   ```

2. Create a subkey for the type of connector to which the device connects.
   (0=Expansion Port, 1=WWAN (not on Omnii XT10), 2=GPS, 4=Docking)
   For example:

   ```
   [HKLM\Drivers\BuiltIn\Peripherals\devices\0]
   ```

3. Create the DeviceId subkey as a concatenation of the device manufacturer and model names, separated by a space.
   For example, for a device made by Psion Teklogix with the model name “Exp1_UART” the subkey should be:

   ```
   [HKLM\Drivers\BuiltIn\Peripherals\devices\0\Psion Teklogix Exp1_UART]
   ```

4. Program the expansion device EEPROM Manufacturer and Model fields to match the DeviceID subkey (case sensitive).
   For example:

   ```
   Manufacturer: Psion Teklogix
   Model: Exp1_UART
   ```

   See Section 8.4: “EEPROM Reading/Writing” for more details.

5. Add the following registry values under the DeviceID subkey (see “Device Registry Values” on page 28 for more details on these values):
   a. **Name**: a descriptive name for the device.
   b. **ConnectorId**: optional field restricting devices to a specific expansion port.
   c. **PinFunctions**: specify the GPIO/Serial/SPI pin functions.
   d. **PowerMode**: set power management for the device.
   e. **Notifications**: set user notification behaviour for device.
   f. **LoadFlags**: specifies default device driver(s).

Sample Device Registry Entries

Serial Device Registry Entry Sample

```
; Registry entry for a serial device
;
[HKLM\Drivers\BuiltIn\Peripherals\devices\0\Psion Teklogix Exp1_UART]
“Name”=”Exp1_UART”
“ConnectorId”=dword:0        ;ConnectorId_Expansion1: 0
                             ;ConnectorId_Expansion2: 1
                             ;ConnectorId_Expansion3: 2
“PinFunctions”=dword:0F      ;Pin GPIO0_TXD:   bit 0 [1]
                             ;Pin GPIO1_RXD:   bit 1 [1]
                             ;Pin GPIO2_CTS:   bit 2 [1]
                             ;Pin GPIO3_RTS:   bit 3 [1]
                             ;Pin GPIO4_MOSI:  bit 4 [0]
                             ;Pin GPIO5_MISO:  bit 5 [0]
                             ;Pin GPIO6_SCLK:  bit 6 [0]
                             ;Pin GPIO7_CS_N:  bit 7 [0]
“PowerMode”=dword:1          ;PowerMode_Auto:    1
                             ;PowerMode_Manual:  2
“Notifications”=dword:0
“LoadFlags”=dword:1          ;Flags_Uart:        0x01, load default UART driver
                             ;Flags_UsbHost:     0x02, load default USB host driver
                             ;Flags_UsbClient:   0x04, load default USB client driver
                             ;Flags_Spi:         0x08, load default SPI driver
                             ;Flags_UsbOtg:      0x10, load default USB OTG driver
                             ;Flags_None:        0x00, load vendor supplied driver
```

SPI Device Registry Entry Sample

(SPI not available on Omnii XT10):

```
; Registry entry for an SPI device
;
[HKLM\Drivers\BuiltIn\Peripherals\devices\0\Psion Teklogix Exp1_SPI]
“Name”=”Exp1_SPI”
“ConnectorId”=dword:0        ; ConnectorId_Expansion1: 0
                             ; ConnectorId_Expansion2: 1
                             ; ConnectorId_Expansion3: 2
“PinFunctions”=dword:F0      ; Pin GPIO0_TXD:   bit 0 [0]
                             ; Pin GPIO1_RXD:   bit 1 [0]
                             ; Pin GPIO2_CTS:   bit 2 [0]
                             ; Pin GPIO3_RTS:   bit 3 [0]
                             ; Pin GPIO4_MOSI:  bit 4 [1]
                             ; Pin GPIO5_MISO:  bit 5 [1]
                             ; Pin GPIO6_SCLK:  bit 6 [1]
                             ; Pin GPIO7_CS_N:  bit 7 [1]
“PowerMode”=dword:1          ; PowerMode_Auto:    1
                             ; PowerMode_Manual:  2
“Notifications”=dword:0
“LoadFlags”=dword:0          ; Flags_Uart:        0x01, load default UART driver
                             ; Flags_UsbHost:     0x02, load default USB host driver
                             ; Flags_UsbClient:   0x04, load default USB client driver
                             ; Flags_Spi:         0x08, load default SPI driver
                             ; Flags_UsbOtg:      0x10, load default USB OTG driver
                             ; Flags_None:        0x00, load vendor supplied driver
```

4.4.1.1 Device Information Registry Keys

This section describes the registry keys required by the peripherals driver to identify and define the behaviour of expansion devices. The parent key for all of the device-specific subkeys is:

```
[HKLM\Drivers\BuiltIn\Peripherals\devices]
```

Within that key, create a subkey (if it does not already exist) for the type of connector that the device will attach to. The connector types are defined in the following table:

Table 4.1 Connector Type Definitions

Subkey Number	Connector Type
0	Expansion Port
1	WWAN[1]
2	GPS
4	Docking

[1] WWAN is not available on Omnii XT10, and is included here for development on future products.

For example, the registry keys that describe devices connecting to the expansion port would be stored in the subkey:

```
[HKLM\Drivers\BuiltIn\Peripherals\devices\0]
```

Within the connector type subkey create a further subkey using the Device ID reported by the device. For devices that attach to the docking connector, an integer value based on a resistor ID in the device is used for identification. For example, the Device Name (resistor ID) for the desktop dock device is 18, therefore the correct registry key for parameters pertaining to that device is:

```
[HKLM\Drivers\BuiltIn\Peripherals\devices\4\18]
```

For all expansion devices with EEPROMs, the Device Name is a concatenation of the **Manufacturer** and **Model** fields in the EEPROM (see Section 8.3.1: "Common EEPROM Fields" for more details). For example, if an end-cap expansion device manufactured by Psion Teklogix with the model field defined as "Endcap" was connected to the end-cap expansion port, the correct registry key for parameters pertaining to that device would be:

```
[HKLM\Drivers\BuiltIn\Peripherals\devices\0\Psion Teklogix Endcap]
```

Note: Because the model and manufacturer fields in the EEPROM are used as part of the registry key, they cannot contain any characters that are not permitted in registry key names (null, backslash, etc.).

Device Registry Values

Within the subkey for the specific device, add the following device registry values:

- **Name** (REG_SZ): A descriptive name for the device.
- **ConnectorId** (REG_DWORD): An optional field that restricts a device to be recognized only on a specific expansion slot. The possible values are *0* (Expansion Port 1), *1* (Expansion Port 2) and *2* (Expansion Port 3).
  If this field is not specified, the device will be recognized on all logical connectors with the same Connector Type (e.g. if the Connector Type of the device is 0, it will be recognized on all expansion ports).

- **PinFunctions** (REG_DWORD): A one-byte value that configures the communication mode of the dual-purpose GPIO / UART and GPIO / SPI pins for the type of device attached.
  Each of the three major expansion ports have eight pins that can be configured to communicate to a GPIO device (General Purpose Input/Output), or to a serial RS-232/UART or SPI (not available on Omnii XT10) device. If no PinFunctions value is specified, the pins default to the non-GPIO (serial / SPI) function.
- To set the communication mode of these pins, set the PinFunctions value according to the following bit field:

Table 4.2 PinFunctions Registry Value Definitions

Bit	Pin Name	Description
0 (LSB)	EXP1_TXD_GPIO0	0 = GPIO pin 0, 1 = Serial TXD
1	EXP1_RXD_GPIO1	0 = GPIO pin 1, 1 = Serial RXD
2	EXP1_CTS_GPIO2	0 = GPIO pin 2, 1 = Serial CTS
3	EXP1_RTS_GPIO3	0 = GPIO pin 3, 1 = Serial RTS
4	EXP1_MOSI_GPIO4	0 = GPIO pin 4, 1 = SPI MOSI
5	EXP1_MISO_GPIO5	0 = GPIO pin 5, 1 = SPI MISO
6	EXP1_SCLK_GPIO6	0 = GPIO pin 6, 1 = SPI SCLK
7 (MSB)	EXP1_CS_N_GPIO7	0 = GPIO pin 7, 1 = SPI chip select

For example, to configure the pins for serial communication, set the PinFunctions value to *0F*.

- **PowerMode** (REG_DWORD): This value determines how and when the device hardware is powered by the peripherals driver. The possible values are *1* (Auto) and *2* (Manual). If the power mode is set to Auto, the device power is managed by the peripherals driver; the device is powered off when the computer enters suspend mode and powered on when the computer resumes activity.

The default setting for this value is *2*, which is the recommended setting. Under this setting, power to the device must be controlled by a loaded device driver or application.

- **Notifications** (REG_DWORD): The notifications registry value determines how the user is notified about expansion devices.
  This value is a bit field as defined in the following table:

Table 4.3 Notifications Registry Value Definitions

Bit	Functionality	Description
0 (LSB)	Show Icon	This flag displays an icon in the status bar. The **Icon** value (see page 30) must be set for this to occur.
1 (MSB)	Show Window	Setting this flag causes a “new device” window to be displayed, containing the name and status of the device. The name reported is the **DeviceNameID** registry value (see page 30). If that value does not exist, the **Name** registry value (see page 28) is used instead. If that also does not exist, the Device Name from the registry key itself is used.

The default setting for this value is *0*, which displays an icon in the taskbar. The other setting, which displays a “new device” window, is primarily intended for docking devices.

- **LoadFlags** (REG_DWORD): The load flags specify the functionality required by the attached device, and therefore the device driver (e.g. USB, UART, etc.) that needs to be loaded to support the device. The LoadFlags value is treated as a bit field, as defined in the following table:

  Table 4.4 LoadFlags Registry Value Definitions

Bit	Functionality	Description
0 (LSB)	UART	This flag indicates a serial device requiring a UART driver. The UART driver is loaded only for the specific expansion connector to which the device is attached. Multiple instances of the driver are loaded for multiple serial devices.
1	USB Host	This flag indicates a device that requires USB Host functionality. When this bit is set, the USB hub and ports are powered and enabled for the expansion connector. This bit must be set for any docking device with a USB Host connector, or for any USB device connected to an expansion port.
2	Reserved	
3	Reserved	
4 (MSB)	USB OTG	This flag is required for docking devices with USB On-The-Go functionality. It is not used by devices connecting to the expansion ports.

If this flag is not specified, any custom device drivers required by the expansion device must be specified in the driver registry subkey (see Section 4.4.1.2 on page 30).

- **Icon** (REG_DWORD): This is the Resource ID of the icon to be displayed for this device in the status bar. Currently, icons can only be loaded from Psion Teklogix DLLs.
- **DeviceNameID** (REG_DWORD): This is the Resource ID of the name string to be displayed in the "New Device" window. Currently, the name string can only be loaded from Psion Teklogix DLLs.

4.4.1.2 Device Driver Registry Keys

If the expansion device requires an additional driver to be loaded, registry keys need to be created to specify the information for the driver. As a rule, docking expansion devices do not require additional drivers, nor do many USB expansion devices. For all expansion devices that require an additional driver to be loaded, follow these steps:

Within the device registry key, add a "driver" subkey. For example:

```
[HKLM\Drivers\BuiltIn\Peripherals\devices\2\Psion Teklogix GPS\driver]
```

Under the \driver subkey, add the following standard registry values for drivers:

- **Prefix** (REG_SZ)
- **Dll** (REG_SZ)
- **Index** (REG_DWORD)
- **Flags** (REG_DWORD)
- **IClass** (REG_MULTI_SZ)

For descriptions and details of these values, consult the Microsoft documentation on developing device drivers. Note that the **Order** value is not used here.

The registry keys and values in the \driver subkey are not accessed directly, but are used as a template to create a driver entry in a different registry location. The \driver subkey and all of its entries are copied to the following registry location:

```
[HKLM\Drivers\BuiltIn\Peripherals\devices\active\[Conn Type]\[Dev Name]
```

*Note: The driver entries are only copied if the driver key is present and contains a **Dll** registry value.*

The drivers for detected peripherals are loaded from this "active" registry location. The driver is loaded through a call to **ActiveDeviceEx()** after other initialization is finished.

It may also be necessary to copy registry keys from one location to another in the registry before loading a driver. To do this, first create a "RegCopy" subkey. For example:

```
[HKLM\Drivers\BuiltIn\Peripherals\devices\2\Psion Teklogix GPS\RegCopy]
```

Within the \RegCopy subkey, add one or more entries in the form of "**source**" = "**dest**", where **source** is the source registry key and **dest** is the destination registry key.

Remember that the backslash '\' characters in the registry key strings will need to be 'escaped' with another backslash character. For example:

```
[HKLM\Drivers\BuiltIn\Peripherals\devices\0\Psion Teklogix Endcap\RegCopy]
"Drivers\\BuiltIn\\Peripherals\\devices\\0\\Psion Teklogix Endcap\\RegKeys"
        ="Software\\PsionTeklogix\\Endcap"
```

This function copies the specified source key and all subkeys underneath it to the target location.

In rare cases, multiple drivers may need to be loaded to support a single piece of hardware. In these cases, the Windows bus enumerator (see the Microsoft documentation at http://code.msdn.microsoft.com/BusEnum2) can be used. Alternatively, the driver specified in the driver key can load the other drivers.

Sample Driver Registry Entries

This is a sample of the registry entries for an Actel® IGLOO® SPI device driver (SPI not available on Omnii XT10):

```
; Registry entry for an SPI driver
;
[HKLM\Drivers\BuiltIn\Peripherals\devices\0\Psion Teklogix Exp1_SPI\driver]
     "Prefix"="SPI"
     "Dll"="iglooSpi.dll"
     "Index"=dword:5
     "Order"=dword:8
     "IoBase"=dword:28200000        ; Register base address
     "IoLen"=dword:10               ; Register length is 16 bytes
     "Irq"=dword:169                ; IRQ: PHIP42IRQ_SPI_EP1 (361)
     "Priority"=dword:50            ; IST thread priority 0x50
     "Mode"=dword:0                 ; 0: Master, 1: Slave, IP only support Master
     "Polarity"=dword:0             ; 0/1: base value of the clock is 0/1
     "Phase"=dword:0                ; 0/1: sample on the leading/trailing clock edge
     "BitSequence"=dword:0          ; 0/1: MSB/LSB
     "BitRate"=dword:1E8480         ; Baudrate 2Mb/s
     "DataWidth"=dword:8            ; Data Width 8; igloo IP only support 8 bits width
     "BufferLen"=dword:400          ; Loop Buffer Length 1024 bytes
     "Bubble"=dword:0               ; the Bubble char
```

4.4.2 Software Registry Entries

If the expansion device uses custom software, the version information for the software can be added to the System Properties of the System Control Panel applet.

Using a registry editor, create the following registry key (where <name> is the name of the software component as it will appear in the System Properties):

```
[HKLM\Software\PsionTeklogix\SystemProperties\Software\<name>]
```

Beneath that key, set the following registry values:

- **@** (REG_SZ): Default value. Set to "Components" to make the software information appear in the Components list of the System Properties.
- **Value** (REG_SZ): Enter the version of the software component here.

For example:

```
; Registry entry for a software program named Scanner Program, version 1.5.21
;
[HKLM\Software\PsionTeklogix\SystemProperties\Software\Scanner Program]
     "@"="Components"
     "Value"="1.5.21"
```

This example creates an entry in the Components list of the System Properties tab of the System Control Panel applet, which reads "Scanner Program: 1.5.21".

4.5 Device Detection and Driver Loading Sequence

With the exception of devices attached to the docking connector, device detection is only performed at startup. After the peripherals driver finishes initializing, it performs the following steps to detect connected hardware and load the appropriate drivers:

1. The EEPROM or other identifier is read to determine the hardware attached to each connector. If an EEPROM is unprogrammed, blank, or is otherwise inaccessible, the detect operation for that connector terminates.
2. The registry is searched for a matching device entry, by connector type and device ID (manufacturer and model). If a matching entry is not found, the detect operation for that connector terminates.
3. If a matching device entry is found, the registry entry for the driver (if any) is copied to the **active** registry key.
4. If one or more **RegCopy** entries are found, the source keys are copied to the destination key locations.
5. Power is enabled to the connector.
6. If USB or UART functionality is specified, the Peripherals driver enables and configures the specified interface pins accordingly.
7. The device driver is loaded.

4.6 Serial (COM) Port Assignments

The default serial port assignments for Omnii are shown in the following table. Ports not listed are unassigned.

Table 4.5 Default Omnii Serial (COM) Port Assignment

Serial Port	Default Assignment	Comments
COM2:	GPS	This COM port is opened by applications that require GPS data. This COM port may instead be opened by the GPS intermediate driver.
COM5:	External USB-serial dongle	External USB-to-serial adaptor dongle WA4015 can be plugged into USB-A port on desktop dock and snap module.
COM6:	USB port replicator	RS-232 port on portable docking module. RS-232 port on desktop docking station. RS-232 port on vehicle cradle. Internal USB-to-serial converter in the desktop dock XMOD option, snap-module, and vehicle cradle.
COM7:		Reserved for future use.
COM8:	WWAN virtual serial port	WWAN not available on the Omnii XT10.
COM18	WWAN hardware (private)	Reserved for internal use.
COM19	GPS hardware (private)	Reserved for internal use.
COM20	*Bluetooth* hardware (private)	Reserved for internal use.
COM24	GPS power (private)	Reserved for internal use.
COM30	Expansion UART1	
COM31	Expansion UART2	
COM32	Expansion UART3	
COM33	Internal scanner port	

Note: 1. The proper name for COM ports above COM9 is \$device\COMxx (no ":" following the COM port number).

2. COM ports cannot be reassigned on the Omnii.

*3. **Bluetooth** creates and destroys many virtual ports.*

4.7 Omnii HDK Application Development Software

To develop software applications for Omnii and expansion devices using the Omnii HDK, you must install the following software packages on your development system. All packages are available on the Psion Teklogix Community website (http://community.psionteklogix.com), in the **Downloads** section (free registration is required for downloading).

4.7.1 Windows Embedded CE 6.0 Development Platform for Visual Studio

This package installs a development platform in Visual Studio 2008 named *PsionTeklogixCE600*. Use this platform to develop applications for Psion Teklogix devices running Windows Embedded CE 6.0.

This package is located in the **Mobile Devices SDK** subfolder of the Community website **Downloads** section as "Windows Embedded CE 6.0 Native (C/C++) MSI Installer for Visual Studio". Download and execute the setup program, and follow the onscreen instructions to install the package.

4.7.2 Psion Teklogix Mobile Devices SDK

The Mobile Devices SDK contains many APIs designed specifically for interacting with Psion Teklogix mobile devices and peripherals. Very simple and generic applications may not require these APIs, so it may not be necessary to install this package, but it is recommended.

This package is located in the **Mobile Devices SDK** subfolder of the Community website **Downloads** section as "MDSDK [version] - Installer" (the current version at the time of this publication is 5.0). Download and execute the setup program, and follow the onscreen instructions to install the package.

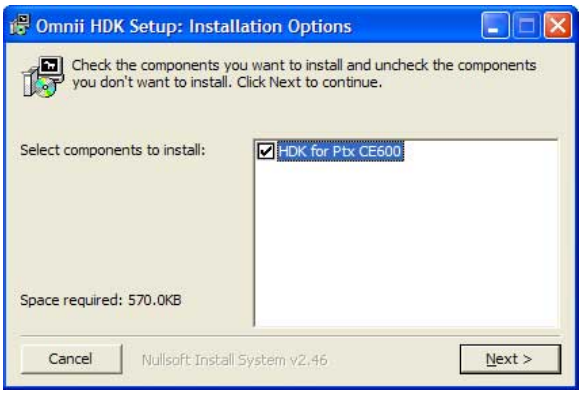
4.7.3 Omnii HDK Development Files

The Omnii HDK files provide an API library of C functions to interact with custom-built hardware connected to the Omnii expansion ports, as well as an HDK Demo application program.

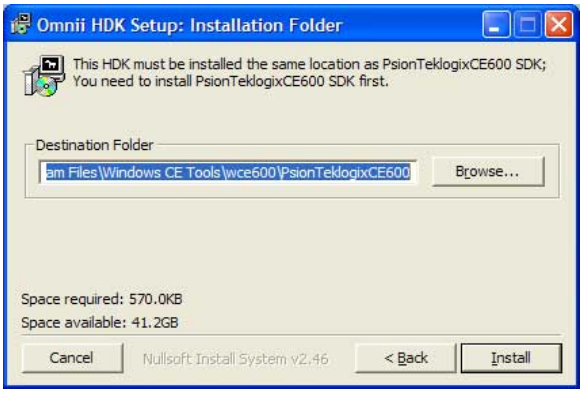
The installation program for these files is included in the Omnii HDK package. See Section 1.6: "Obtaining the HDK" for instructions on how to download this package to your computer.

Follow these instructions to install the Omnii HDK API library and HDK Demo application files:

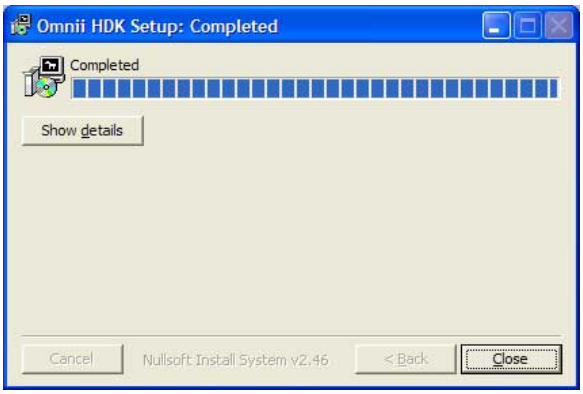
1. Navigate to the folder with the HDK files, and double-click on the file *OmniiHDK_Setup.exe* to begin the installation.
   The Installation Options *dialog box appears:*



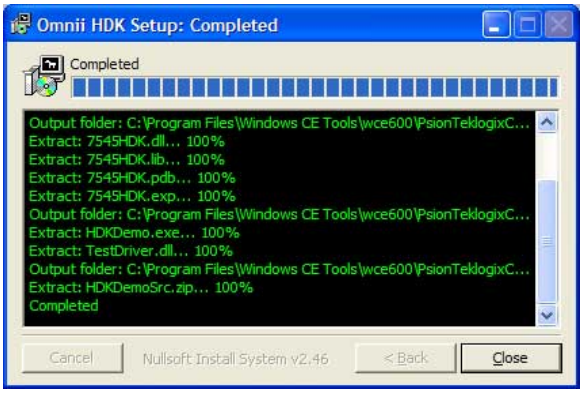
2. Select the destination platform(s) you will be developing the applications for. Omnii XT10 only supports the CE 6.0 operating system, but future Omnii models will add options to this menu. Ensure there is a check mark in the box next to *PsionTeklogixCE600*, then click **Next >**.
   The Installation Folder *dialog box appears:*



3. To change the default installation folder, type the path into the field, or click the **Browse** button and navigate to the destination folder. Click **Install** to proceed.
   The progress dialog box appears as the installer extracts and copies the files to the destination folders.



4. If you wish to see a breakdown of the installation progress, click the **Show details** button.
   The details window appears. Click and drag the scroll bar on the right to scroll the information up or down.



5. Click **Close** to end the installation.

4.7.4 Omnii HDK API Functions

The following sections describe the C functions declared in the file Hdk7545.h.

Note: HDK functions cannot be called from the xxx_Init method of a driver loaded by the peripherals driver.

4.7.4.1 Hdk7545_ExpansionSlotFromActiveRegKey

Syntax

DWORD Hdk7545_ExpansionSlotFromActiveRegKey (const wchar_t *ActiveRegKey, Hdk7545_Connector *expansionSlot);

Parameters

- ActiveRegKey – [in] pointer to a null-terminated string containing the driver active registry key of the driver. For example (under HKEY_LOCAL MACHINE) "\Drivers\Active\31". The active key is passed to the XXX_Init function of the driver as the context value, cast to a DWORD. This parameter must not be null.
- expansionSlot – [out] pointer to the expansion connector slot ID. This parameter must not be null.

Description

This function is used to retrieve the ID of the expansion slot containing the device this driver is going to control. The connector ID value returned can be used in a call to Hdk7545_Open.

This function can only be used by drivers loaded by the Peripherals Driver. Other drivers and applications must determine the expansion slot with which to communicate through other means.

Returns

- ERROR_SUCCESS – if successful. The 'expansionSlot' parameter contains a valid ID.
- ERROR_INVALID_PARAMETER – one of the pointers was null or incorrect.
- ERROR_INVALID_DATA – an exception was generated.
- ERROR_NOT_SUPPORTED – this device is not supported by the HDK.
- Other errors are possible.

4.7.4.2 Hdk7545_Open

Syntax

DWORD Hdk7545_Open(HANDLE *hdk, Hdk7545_Connector connector);

Parameters

- hdk – [out] pointer to a HANDLE. If the open call succeeds, the handle is changed to a valid handle value that can be used in other HDK operations.
- connector – [in] one of the values in the Hdk7545_Connector enumeration identifying the expansion slot (or other connector) being controlled.

Description

This function is used to open a handle to the Psion Teklogix HDK. The handle opened can then be used in other HDK functions. The handle must be closed using Hdk7545_Close(). This parameter must not be null. Each handle is tied to a single particular expansion slot or connector.

The expansion slot or other connector being controlled is determined by the 'connector' parameter. If this function is being called from a device driver loaded by the Peripherals driver, the value for this parameter can be determined by calling the Hdk7545_ExpansionSlotFromActiveRegKey

function. If this function is being called from a driver loaded by other means (installed under Drivers\Builtin) or is being called from an application, the correct expansion slot must be known by the caller in advance.

Returns

- ERROR_SUCCESS – if successful. The handle pointed to by 'hdk' is now valid.
- ERROR_INVALID_PARAMETER – the 'hdk' pointer is null, or the specified connector is invalid.
- ERROR_INVALID_DATA – an exception was generated.
- ERROR_NOT_SUPPORTED – this device is not supported by the HDK.
- Other errors are possible.

Sample Code

```
DWORD OpenAndCloseHdk( const wchar_t *ActiveKey )
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Invalid;

        DWORD result = Hdk7545_ExpansionSlotFromActiveRegKey(ActiveKey,&expansionSlot);

        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_FOUND;
        }

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        // ...

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.3 Hdk7545_Close

Syntax

DWORD Hdk7545_Close(HANDLE *hdk);

Parameters

- hdk – [in] pointer to a valid open HDK handle.

Description

This function is used to close an open HDK handle and release all the resources it owns. The handle cannot be used after it is closed.

Returns

- ERROR_SUCCESS – if successful. The handle is now closed.
- ERROR_INVALID_HANDLE – the specified handle is invalid or null.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

See sample code for *"Hdk7545_Open"*.

4.7.4.4 Hdk7545_SetPower

Syntax

DWORD Hdk7545_SetPower(HANDLE hdk, BOOL enable);

Parameters

- hdk – [in] an open HDK handle.
- enable – [in] the new power state of the connector being controlled (see *"Hdk7545_Open"*).

Description

Powers on/off the connector being controlled.

The power state is reference-counted. If this function is called multiple times with the 'enable' parameter set to TRUE, it has to be called the same number of times with the 'enable' parameter set to FALSE in order to power the connector off.

The default power state for connectors is off.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
DWORD SetPower()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        result = Hdk7545_SetPower(hdkHandle, TRUE);
        if( result != ERROR_SUCCESS ) {
                Hdk7545_Close(&hdkHandle);
                return result;
        }

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.5 Hdk7545_GetPower

Syntax

DWORD Hdk7545_GetPower(HANDLE hdk, BOOL *enabled);

Parameters

- hdk – [in] an open HDK handle.
- enabled – [out] pointer to a BOOL containing the current connector power state.

Description

This function is used to determine the current power state of a connector.

The default power state for connectors is off.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – one of the parameters is incorrect or invalid.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
DWORD GetPower()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        BOOL powerEnabled = FALSE;

        result = Hdk7545_GetPower(hdkHandle, &powerEnabled);
        if( result != ERROR_SUCCESS ) {
                Hdk7545_Close(&hdkHandle);
                return ERROR_GEN_FAILURE;
        }

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.6 Hdk7545_SetPowerMode

Syntax

DWORD Hdk7545_SetPowerMode(HANDLE hdk, Hdk7545_PowerMode mode);

Parameters

- hdk – [in] an open HDK handle.
- mode – [in] the new power mode for the device.

Description

This function is used to configure the power mode for the device attached to the connector. There are currently two modes available: Auto and Manual.

If the power mode of the device is Manual, the connector power will not be controlled by the Peripherals Driver. A loaded device driver/application must enable and disable the power.

If the power mode of the device is Auto, the Peripherals driver will enable/disable power to the connectors automatically. Power to the connector is:

1. Applied initially before the device driver for the connected hardware is loaded.
2. Removed when the hand-held is suspended.
3. Reapplied when the hand-held resumes from suspend.

The default power mode is Manual.

This function can only be called by a driver, not by an application. The driver that calls this function must be loaded by the Peripherals driver at startup.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – one of the parameters is incorrect or invalid.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
DWORD SetPowerMode()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        result = Hdk7545_SetPowerMode(hdkHandle, Hdk7545_PowerMode_Manual);
        if( result != ERROR_SUCCESS ) {
                Hdk7545_Close(&hdkHandle);
                return ERROR_GEN_FAILURE;
        }

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.7 Hdk7545_GetPowerMode

Syntax

DWORD Hdk7545_GetPowerMode(HANDLE hdk, Hdk7545_PowerMode *mode);

Parameters

- hdk – [in] an open HDK handle.
- mode – [out] pointer to a Hdk7545_PowerMode value that will contain the current power mode of the connector.

Description

This function is used to retrieve the current power mode of the device attached to the connector. There are currently two modes available: Auto and Manual.

The default power mode is Manual.

This function can only be called by a driver, not by an application. The driver that calls this function must be loaded by the Peripherals driver at startup.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – one of the parameters is incorrect or invalid.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
DWORD GetPowerMode()
{
	Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

	HANDLE hdkHandle = INVALID_HANDLE_VALUE;

	DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
	if( result != ERROR_SUCCESS ) {
		return ERROR_NOT_SUPPORTED;
	}

	Hdk7545_PowerMode mode = Hdk7545_PowerMode_Manual;

	result = Hdk7545_GetPowerMode(hdkHandle, &mode);
	if( result != ERROR_SUCCESS ) {
		Hdk7545_Close(&hdkHandle);
		return ERROR_GEN_FAILURE;
	}

	if( mode != Hdk7545_PowerMode_Manual )
	{
		result = Hdk7545_SetPowerMode(hdkHandle, Hdk7545_PowerMode_Manual);
		if( result != ERROR_SUCCESS ) {
			Hdk7545_Close(&hdkHandle);
			return ERROR_GEN_FAILURE;
		}
	}

	Hdk7545_Close(&hdkHandle);
	return ERROR_SUCCESS;
}
```

4.7.4.8 Hdk7545_ExpansionSetPinDirection

Syntax

DWORD Hdk7545_ExpansionSetPinDirection(HANDLE hdk, DWORD pin, Hdk7545_PinDirection direction);

Parameters

- hdk – [in] an open HDK handle.
- pin – [in] the ID of the expansion connector pin. The pin parameter must be set to one of the HDK7545_GPIO_PIN_EXPANSION_xxx values.
- direction – [in] the new pin direction (input/output).

Description

Sets the specified pin on the expansion connector to be either an input or an output.

Expansion connector pins can be configured for use as a GPIO interface, or they can be reassigned to an alternate function (serial or SPI) using the Hdk7545_ExpansionSetPinFunction function (see page 46). The pin direction does not need to be set if the alternate (non-GPIO) function for the pin is enabled.

Drivers and/or applications must set the pin direction to output before calling the Hdk7545_ExpansionSetPinState function.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – the specified pin is not valid.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
// configure expansion 1's GPIO 0-3 pins (0, 3 as outputs, and 1, 2 as inputs).
DWORD SetDirection()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        Hdk7545_ExpansionSetPinDirection(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD,Hdk7545_PinDirection_Output);
        Hdk7545_ExpansionSetPinDirection(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO1_RXD,Hdk7545_PinDirection_Input);
        Hdk7545_ExpansionSetPinDirection(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO2_CTSN,Hdk7545_PinDirection_Input);
        Hdk7545_ExpansionSetPinDirection(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO3_RTSN,Hdk7545_PinDirection_Output);

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.9 Hdk7545_ExpansionGetPinDirection

Syntax

DWORD Hdk7545_ExpansionGetPinDirection(HANDLE hdk, DWORD pin, Hdk7545_PinDirection *direction);

Parameters

- hdk – [in] an open HDK handle.
- pin – [in] the ID of the expansion connector pin. The pin parameter must be set to one of the HDK7545_GPIO_PIN_EXPANSION_xxx values.
- direction – [out] a pointer to the current pin direction (input/output).

Description

Reports the current direction (input/output) for the specified pin.

Expansion connector pins can be configured for use as a GPIO interface, or they can be reassigned to an alternate function (serial or SPI) using the Hdk7545_ExpansionSetPinFunction function (see page 46). The pin direction does not need to be set if the alternate (non-GPIO) function for the pin is enabled.

Drivers and/or applications must set the pin direction to output before calling the Hdk7545_ExpansionSetPinState function.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – the specified pin is not valid, or the direction pointer is null.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
DWORD GetDirection()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        Hdk7545_PinDirection direction = Hdk7545_PinDirection_Output;
        Hdk7545_ExpansionGetPinDirection(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD, &direction);

        RETAILMSG(1, (L"The pin direction for %u is %u\r\n",
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD, direction));

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.10 Hdk7545_ExpansionSetPinMode

Syntax

DWORD Hdk7545_ExpansionSetPinMode(HANDLE hdk, DWORD pin, DWORD mode);

Parameters

- hdk – [in] an open HDK handle.
- pin – [in] the ID of the expansion connector pin. The pin parameter must be set to one of the HDK7545_GPIO_PIN_EXPANSION_xxx values.
- mode – [in] the new pin mode(s) to set (see description below).

Description

Changes the mode of the specified pin on the expansion connector. The mode can be set to either 'Hdk7545_PinMode_NoInterrupt' or to a combination of one or more of the following values:

Hdk7545_PinMode_InterruptLowHigh	[edge-triggered interrupts]
Hdk7545_PinMode_InterruptHighLow	[edge-triggered interrupts]
Hdk7545_PinMode_InterruptLow	[level-triggered interrupts]
Hdk7545_PinMode_InterruptHigh	[level-triggered interrupts]

The following restrictions apply to the pin mode configuration:

1. Omnii XT10 only allows level-trigger interrupts to be configured, and only one type (low or high, not both).
2. Future Omnii models will allow both edge- and level-triggered interrupts to be configured, but not at the same time. If level-triggered interrupts are configured, only one type can be enabled (low or high, not both). If edge-triggered interrupts are configured, either one type (high-low or low-high) or both types can be configured. If a driver or application attempts to configure both level- and edge-triggered interrupts at the same time, the level-triggered interrupts take precedence.

The pin mode only needs to be configured if a pin is being used as a GPIO line.

A non-zero pin mode should only be set for pins configured as inputs (see "Hdk7545_ExpansionSetPinDirection" on page 41).

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – the specified pin is not valid.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
// set the RX & CTSN pins as inputs, and configure their modes to level-triggered
// interrupts [high and low, respectively].
DWORD SetMode()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        Hdk7545_ExpansionSetPinDirection(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO1_RXD,Hdk7545_PinDirection_Input);
        Hdk7545_ExpansionSetPinDirection(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO2_CTSN,Hdk7545_PinDirection_Input);

        Hdk7545_ExpansionSetPinMode(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO1_RXD,Hdk7545_PinMode_InterruptHigh);
        Hdk7545_ExpansionSetPinMode(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO2_CTSN,Hdk7545_PinMode_InterruptLow);

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.11 Hdk7545_ExpansionGetPinMode

Syntax

DWORD Hdk7545_ExpansionGetPinMode(HANDLE hdk, DWORD pin, DWORD *mode);

Parameters

- hdk – [in] an open HDK handle.

- pin – [in] the ID of the expansion connector pin. The pin parameter must be set to one of the HDK7545_GPIO_PIN_EXPANSION_xxx values.
- mode – [out] pointer to the pin mode(s) set for the specified pin.

Description

Retrieves the current mode configuration for the specified pin.

For more details see the description for *"Hdk7545_ExpansionSetPinMode"*.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – the specified pin is not valid, or the mode pointer is null.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
// retrieve and print out the mode flags for the RX and CTSN pins
DWORD GetMode()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        DWORD mode[2] = { 0, 0 } ;
        DWORD pins[2] = { HDK7545_GPIO_PIN_EXPANSION_GPIO1_RXD,
                HDK7545_GPIO_PIN_EXPANSION_GPIO2_CTSN } ;

        if( Hdk7545_ExpansionGetPinMode(hdkHandle, pins[0],
                &mode[0]) != ERROR_SUCCESS ) {
                Hdk7545_Close(&hdkHandle);
                return ERROR_GEN_FAILURE;
        }

        if( Hdk7545_ExpansionGetPinMode(hdkHandle, pins[1],
                &mode[1]) != ERROR_SUCCESS ) {
                Hdk7545_Close(&hdkHandle);
                return ERROR_GEN_FAILURE;
        }

        RETAILMSG(1, (L"GPIO1: Low:%u High:%u Low/High:%u High/Low:%u\r\n",
                (mode[0] & Hdk7545_PinMode_InterruptLow) != 0,
                (mode[0] & Hdk7545_PinMode_InterruptHigh) != 0,
                (mode[0] & Hdk7545_PinMode_InterruptLowHigh) != 0,
                (mode[0] & Hdk7545_PinMode_InterruptHighLow) != 0));
        RETAILMSG(1, (L"GPIO2: Low:%u High:%u Low/High:%u High/Low:%u\r\n",
                (mode[1] & Hdk7545_PinMode_InterruptLow) != 0,
                (mode[1] & Hdk7545_PinMode_InterruptHigh) != 0,
                (mode[1] & Hdk7545_PinMode_InterruptLowHigh) != 0,
                (mode[1] & Hdk7545_PinMode_InterruptHighLow) != 0));

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.12 Hdk7545_ExpansionSetPinFunction

Syntax

DWORD Hdk7545_ExpansionSetPinFunction(HANDLE hdk, DWORD pin, Hdk7545_PinFunction function);

Parameters

- hdk – [in] an open HDK handle.
- pin – [in] the ID of the expansion connector pin. The pin parameter must be set to one of the HDK7545_GPIO_PIN_EXPANSION_xxx values.
- function – [in] the pin function (GPIO or alternate).

Description

Each of the three expansion ports on Omnii has eight configurable pins. These pins can be configured for use as a GPIO interface, or they can be reassigned to an alternate function (serial or SPI) using the Hdk7545_ExpansionSetPinFunction function. This function allows drivers and applications to switch between the two functional modes. If a driver or application is going to use one or more of the eight pins as GPIOs, it must first call this function for each of the pins used.

If a device driver loaded by the peripherals driver requires the UART functionality of the pin, it should set the appropriate bit in the "LoadFlags" registry value (see Section 4.4.1.1: "Device Information Registry Keys").

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – the specified pin is not valid.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
// set the first four pins to UART functionality
// make the other four pins GPIOs
DWORD SetFunction()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;
        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        Hdk7545_ExpansionSetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO1_RXD,Hdk7545_PinFunction_Alternate);
        Hdk7545_ExpansionSetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD,Hdk7545_PinFunction_Alternate);
        Hdk7545_ExpansionSetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO2_CTSN,Hdk7545_PinFunction_Alternate);
        Hdk7545_ExpansionSetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO3_RTSN,Hdk7545_PinFunction_Alternate);
        Hdk7545_ExpansionSetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO4_MOSI,Hdk7545_PinFunction_Alternate);
        Hdk7545_ExpansionSetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO5_MISO,Hdk7545_PinFunction_Alternate);
        Hdk7545_ExpansionSetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO6_SCLK,Hdk7545_PinFunction_Alternate);
```

```
        Hdk7545_ExpansionSetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO7_CSN,Hdk7545_PinFunction_Alternate);
                Hdk7545_Close(&hdkHandle);

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.13 Hdk7545_ExpansionGetPinFunction

Syntax

DWORD Hdk7545_ExpansionGetPinFunction(HANDLE hdk, DWORD pin, Hdk7545_PinFunction *function);

Parameters

- hdk – [in] an open HDK handle.
- pin – [in] the ID of the expansion connector pin. The pin parameter must be set to one of the HDK7545_GPIO_PIN_EXPANSION_xxx values.
- function – [out] pointer to the current pin function (GPIO or alternate).

Description

Retrieves the currently configured function of the specified pin. See the description for *"Hdk7545_ExpansionSetPinFunction"* above for details.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – the specified pin is not valid, or the pin function pointer is null.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
// get and print the current functions for Expansion 1's GPIO 0 - 3
DWORD GetFunction()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;
        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        Hdk7545_PinFunction functions[4] = {
                Hdk7545_PinFunction_GPIO,Hdk7545_PinFunction_GPIO,
                Hdk7545_PinFunction_GPIO,Hdk7545_PinFunction_GPIO
        } ;

        Hdk7545_ExpansionGetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD, &functions[0]);
        Hdk7545_ExpansionGetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO1_RXD, &functions[1]);
        Hdk7545_ExpansionGetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO2_CTSN, &functions[2]);
        Hdk7545_ExpansionGetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO3_RTSN, &functions[3]);
```

```
        RETAILMSG(1, (L"PIN Functions:\r\nPin 0: %u  Pin 1: %u  Pin 2: %u  Pin 3: %u\r\n",
                functions[0], functions[1], functions[2], functions[3]));

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.14 Hdk7545_ExpansionSetPinState

Syntax

DWORD Hdk7545_ExpansionSetPinState(HANDLE hdk, DWORD pin, Hdk7545_PinState state);

Parameters

- hdk – [in] an open HDK handle.
- pin – [in] the ID of the expansion connector pin. The pin parameter must be set to one of the HDK7545_GPIO_PIN_EXPANSION_xxx values.
- state – [in] the new pin state.

Description

Changes the state of the specified pin to Set or Clear.

This function may fail to change the pin state if the pin direction is not set to output (see *"Hdk7545_ExpansionSetPinDirection"*).

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – the specified pin is not valid.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
DWORD SetState()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        Hdk7545_ExpansionSetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD,Hdk7545_PinFunction_GPIO);

        Hdk7545_ExpansionSetPinDirection(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD,Hdk7545_PinDirection_Output);

        Hdk7545_ExpansionSetPinState(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD, Hdk7545_PinState_Set);

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.15 Hdk7545_ExpansionGetPinState

Syntax

DWORD Hdk7545_ExpansionGetPinState(HANDLE hdk, DWORD pin, Hdk7545_PinState *state);

Parameters

- hdk – [in] an open HDK handle.
- pin – [in] the ID of the expansion connector pin. The pin parameter must be set to one of the HDK7545_GPIO_PIN_EXPANSION_xxx values.
- state – [out] pointer to the current pin state.

Description

Retrieves the state of the specified pin.

This function may fail if the pin direction is not set to output (see "Hdk7545_ExpansionSetPinDirection" on page 41).

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – the specified pin is not valid, or the state pointer is null.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
// query the state of a pin, and change it if the state is not clear.
DWORD GetState()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        Hdk7545_ExpansionSetPinFunction(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD,
                Hdk7545_PinFunction_GPIO);

        Hdk7545_ExpansionSetPinDirection(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD,
                Hdk7545_PinDirection_Output);

        Hdk7545_PinState state = Hdk7545_PinState_Unknown;
        Hdk7545_ExpansionGetPinState(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD, &state);

        if( state != Hdk7545_PinState_Clr ) {
                Hdk7545_ExpansionSetPinState(hdkHandle,
                        HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD,
                        Hdk7545_PinState_Clr);
        }

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.16 Hdk7545_ExpansionSetPullUpDown

Syntax

DWORD Hdk7545_ExpansionSetPullUpDown(HANDLE hdk, DWORD pin, Hdk7545_PullUpDown upDown, BOOL enable);

Parameters

- hdk – [in] an open HDK handle.
- pin – [in] the ID of the expansion connector pin. The pin parameter must be set to one of the HDK7545_GPIO_PIN_EXPANSION_xxx values.
- upDown – [in] set to *Hdk7545_PullUpDown_Pullup* to enable/disable a pull-up, or to *Hdk7545_PullUpDown_PullDown* to enable/disable a pull-down.
- enable – [in] if TRUE, the pull-up/pull-down is enabled. If FALSE it is disabled.

Description

Used to enable/disable a pull-up or pull-down on a pin.

It is not possible to enable both a pull-up and pull-down on the same pin simultaneously. If a pull-up is enabled, an existing pull-down is disabled, and vice versa.

This function is not supported on Omnii XT10**.**

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – the specified pin is not valid.
- ERROR_INVALID_FUNCTION – the computer does not support setting a pull-up/pull-down on this pin.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
// enable a pull-up on the Expansion 1
// HDK7545_GPIO_PIN_EXPANSION_GPIO0_CTSN pin.
DWORD SetPulldown()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        Hdk7545_ExpansionSetPullUpDown(hdkHandle,
                HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD,
                Hdk7545_PullUpDown_Pullup, FALSE);

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.17 Hdk7545_ExpansionGetIrq

Syntax

DWORD Hdk7545_ExpansionGetIrq(HANDLE hdk, DWORD pin, DWORD *irq);

Parameters

- hdk – [in] an open HDK handle.
- pin – [in] the ID of the expansion connector pin. The pin parameter must be set to one of the HDK7545_GPIO_PIN_EXPANSION_xxx values.
- irq – [out] pointer to the IRQ value.

Description

Used to retrieve the IRQ value associated with the specified pin. The value returned can then be used to request a system interrupt.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – the specified pin is not valid, or the irq pointer is null.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
// interrupt handling for an expansion GPIO.
DWORD GetIRQ()
{
	Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

	HANDLE hEvent = CreateEvent(0, FALSE, FALSE, 0);
	if( hEvent == 0 ) {
		return ERROR_OUTOFMEMORY;
	}

	HANDLE hdkHandle = INVALID_HANDLE_VALUE;

	DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
	if( result != ERROR_SUCCESS ) {
		return ERROR_NOT_SUPPORTED;
	}

	// assume HDK7545_GPIO_PIN_EXPANSION_GPIO1_RXD is configured as an
	// input with interrupts enabled.

	DWORD irq = 0;
	result = Hdk7545_ExpansionGetIrq(hdkHandle,
		HDK7545_GPIO_PIN_EXPANSION_GPIO1_RXD, &irq);
	if( result != ERROR_SUCCESS ) {
		Hdk7545_Close(&hdkHandle);
		return result;
	}

	DWORD sysIntr = 0;
	BOOL ok = KernelIoControl(IOCTL_HAL_REQUEST_SYSINTR, &irq, sizeof(irq),
		&sysIntr, sizeof(sysIntr), 0);
	if( !ok || sysIntr == SYSINTR_UNDEFINED || sysIntr == 0 ) {
		Hdk7545_Close(&hdkHandle);
		return ERROR_NO_SYSTEM_RESOURCES;
	}
```

```
        ok = InterruptInitialize(sysIntr, hEvent, 0, 0) ;
        if( !ok ) {
                KernelIoControl(IOCTL_HAL_RELEASE_SYSINTR, &sysIntr,
                        sizeof(sysIntr), 0, 0, 0);
                Hdk7545_Close(&hdkHandle);
                return ERROR_NO_SYSTEM_RESOURCES;
        }

        // ...

        InterruptDisable(sysIntr);
        KernelIoControl(IOCTL_HAL_RELEASE_SYSINTR, &sysIntr,
                sizeof(sysIntr), 0, 0, 0);

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.18 Hdk7545_ReadEepromHeader

Syntax

DWORD Hdk7545_ReadEepromHeader(HANDLE hdk, Hdk7545_Eeprom *eeprom);

Parameters

- hdk – [in] an open HDK handle.
- eeprom – [out] pointer to an instance of Hdk7545_Eeprom.

Description

Reads the contents of an EEPROM into the structure supplied by the caller. This function can only read from the EEPROM on the connector specified in the call to Hdk7545_Open().

The caller must set the correct version [HDK7545_EEPROM_VERSION] and the correct size [sizeof(Hdk7545_Eeprom)] in the supplied Hdk7545_Eeprom structure. If the values supplied are not correct, an error is returned.

The format of the EEPROM is described in Chapter 8: “EEPROM Specifications”. It is the responsibility of anyone developing expansion hardware to correctly program the EEPROMs. If an EEPROM is not programmed properly, the values reported by the Hdk7545_ReadEepromHeader function may be invalid.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – one of the parameters is incorrect/invalid.
- ERROR_INVALID_DATA - an exception was generated.
- ERROR_REVISION_MISMATCH – the EEPROM structure size/format/version supplied by the caller does not match what was expected by the HDK and/or the hand-held device.
- Other errors are possible.

Sample Code

```
DWORD ReadEeprom()
{
	Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

	HANDLE hdkHandle = INVALID_HANDLE_VALUE;

	DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
	if( result != ERROR_SUCCESS ) {
		return ERROR_NOT_SUPPORTED;
	}

	Hdk7545_Eeprom eeprom = { 0 };
	eeprom.m_Size = sizeof(eeprom);
	eeprom.m_Version = HDK7545_EEPROM_VERSION;

	result = Hdk7545_ReadEepromHeader(hdkHandle, &eeprom);
	if( result != ERROR_SUCCESS )
	{
		// found the EEPROM
	}
	else if( result == ERROR_NOT_FOUND )
	{
		// Expansion 1 is empty - no EEPROM found
	}
	else if( result == ERROR_REVISION_MISMATCH )
	{
		// software mismatch. the OS image or HDK expects the
		// Hdk7545_Eeprom size to be different.
	}
	else
	{
		// some other error occurred.
	}

	Hdk7545_Close(&hdkHandle);
	return ERROR_SUCCESS;
}
```

4.7.4.19 Hdk7545_WriteEepromHeader

Syntax

DWORD Hdk7545_WriteEepromHeader(HANDLE hdk, Hdk7545_Eeprom *eeprom);

Parameters

- hdk – [in] an open HDK handle.
- eeprom – [in] pointer to an instance of Hdk7545_Eeprom containing the information to be written to the EEPROM header.

Description

This function is used to modify the contents of the EEPROM header of the EEPROM on the connector specified in the call to Hdk7545_Open().

The caller must set the correct version [HDK7545_EEPROM_VERSION] and the correct size [sizeof(Hdk7545_Eeprom)] in the supplied Hdk7545_Eeprom structure. If the values supplied are not correct, an error is returned.

The caller must also set a valid EEPROM size (m_EepromSize = 128|256|etc.). The EEPROM size must be the actual total size of the EEPROM on the connector in bytes, and a multiple of 128. If a size that is larger than the actual size of the EEPROM is written, the behaviour of further

reads/writes is undefined. Since an application or driver may not know the EEPROM size in advance, it should first call the Hdk7545_ReadEepromHeader function (see page 52) to read the EEPROM contents, modify the EEPROM structure contents as required, and write the modified structure back to the EEPROM.

All of the EEPROM header fields can be changed. However, if the EEPROM header data is not formatted properly, unexpected behaviour may occur. For example, the peripherals driver may not be able to properly detect the device type and as a result will not load the driver for the attached device.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – one of the parameters is incorrect/invalid.
- ERROR_INVALID_DATA - an exception was generated.
- ERROR_REVISION_MISMATCH – the EEPROM structure size/format/version supplied by the caller does not match what was expected by the HDK and/or the hand-held device.
- Other errors are possible.

Sample Code

```
DWORD WriteEeprom()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        Hdk7545_Eeprom eeprom = { 0 };
        eeprom.m_Size = sizeof(eeprom);
        eeprom.m_Version = HDK7545_EEPROM_VERSION;

        result = Hdk7545_ReadEepromHeader(hdkHandle, &eeprom);
        if( result != ERROR_SUCCESS ) {
                Hdk7545_Close(&hdkHandle);
                return result;
        }

        // the manufacturing data region has no defined format. Assume for this
        // example that the manufacturer stores the test date in the first three bytes,
        // and the test result in the fourth byte, with 0 indicating no errors occurred.
        eeprom.m_MfgTestRegion[0] = 0x01; // January
        eeprom.m_MfgTestRegion[1] = 0x1e; // 30
        eeprom.m_MfgTestRegion[2] = 0x0a; // 2010
        eeprom.m_MfgTestRegion[3] = 0x00; // test result (0 = okay)

        result = Hdk7545_WriteEepromHeader(hdkHandle, &eeprom);
        if( result != ERROR_SUCCESS ) {
                Hdk7545_Close(&hdkHandle);
                return result;
        }

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.20 Hdk7545_ReadEepromExtendedData

Syntax

DWORD Hdk7545_ReadEepromExtendedData(HANDLE hdk, DWORD offset, DWORD size, BYTE *data);

Parameters

- hdk – [in] an open HDK handle.
- offset – [in] 0-based offset in bytes from the start of the extended data region.
- size – [in] the number of bytes of data to read from the extended data region.
- data – [out] pointer to a BYTE array to where the EEPROM data will be copied. The array must not be null, and must be at least 'size' bytes in length.

Description

This function reads from the extended data area of an EEPROM. The extended data region starts at the first byte past the end of the EEPROM header (see Chapter 8: "EEPROM Specifications" for more information).

The size of the EEPROM may vary. Drivers should use the extended data region size reported in the Hdk7545_Eeprom structure (m_ExtendedSize) to determine how much space is available for extended data. See *"Hdk7545_ReadEepromHeader"* for details on reading the EEPROM header.

The format and contents of the extended data region are device-specific.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – one of the parameters was null/invalid, or the caller specified an invalid offset or size.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
DWORD ReadExtended()
{
        Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;

        HANDLE hdkHandle = INVALID_HANDLE_VALUE;

        DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
        if( result != ERROR_SUCCESS ) {
                return ERROR_NOT_SUPPORTED;
        }

        Hdk7545_Eeprom eeprom = { 0 };
        eeprom.m_Size = sizeof(eeprom);
        eeprom.m_Version = HDK7545_EEPROM_VERSION;

        result = Hdk7545_ReadEepromHeader(hdkHandle, &eeprom);
        if( result != ERROR_SUCCESS ) {
                Hdk7545_Close(&hdkHandle);
                return result;
        }

        BYTE data[24] = { 0 } ;
        DWORD dataSize = sizeof(data);
        DWORD offset = 0;
```

```
        if( eeprom.m_ExtendedSize < dataSize ) {
                Hdk7545_Close(&hdkHandle);
                return ERROR_INVALID_DATA;
        }

        result = Hdk7545_ReadEepromExtendedData(hdkHandle, offset,
                dataSize, data);
        if( result != ERROR_SUCCESS ) {
                // handle error here...
                Hdk7545_Close(&hdkHandle);
                return result;
        }

        Hdk7545_Close(&hdkHandle);
        return ERROR_SUCCESS;
}
```

4.7.4.21 Hdk7545_WriteEepromExtendedData

Syntax

DWORD Hdk7545_WriteEepromExtendedData(HANDLE hdk, DWORD offset, DWORD size, const BYTE *data);

Parameters

- hdk – [in] an open HDK handle.
- offset – [in] 0-based offset in bytes from the start of the extended data region.
- size – [in] the number of bytes of data to write to the extended data region.
- data – [in] pointer to a BYTE array containing the EEPROM data to write. The array must not be null, and must be at least 'size' bytes in length.

Description

This function writes to the extended data area of an EEPROM. The extended data region starts at the first byte past the end of the EEPROM header (see Chapter 8: “EEPROM Specifications” for more information).

The size of the EEPROM may vary. Drivers should use the extended data region size reported in the Hdk7545_Eeprom structure (m_ExtendedSize) to determine how much space is available for extended data. See *“Hdk7545_ReadEepromHeader”* for details on reading the EEPROM header.

The format and contents of the extended data region are device-specific.

Returns

- ERROR_SUCCESS – if successful.
- ERROR_INVALID_HANDLE – the specified handle is invalid.
- ERROR_INVALID_PARAMETER – one of the parameters was null/invalid, or the caller specified an invalid offset or size.
- ERROR_INVALID_DATA - an exception was generated.
- Other errors are possible.

Sample Code

```
DWORD WriteExtended()
{
	Hdk7545_Connector expansionSlot = Hdk7545_Connector_Expansion1;
	HANDLE hdkHandle = INVALID_HANDLE_VALUE;

	DWORD result = Hdk7545_Open(&hdkHandle, expansionSlot);
	if( result != ERROR_SUCCESS ) {
		return ERROR_NOT_SUPPORTED;
	}

	Hdk7545_Eeprom eeprom = { 0 };
	eeprom.m_Size = sizeof(eeprom);
	eeprom.m_Version = HDK7545_EEPROM_VERSION;

	result = Hdk7545_ReadEepromHeader(hdkHandle, &eeprom);
	if( result != ERROR_SUCCESS ) {
		Hdk7545_Close(&hdkHandle);
		return result;
	}

	const BYTE data[24] =
		{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23};
	DWORD dataSize = sizeof(data);
	DWORD offset = 0;

	if( eeprom.m_ExtendedSize < dataSize ) {
		Hdk7545_Close(&hdkHandle);
		return ERROR_INVALID_DATA;
	}

	result = Hdk7545_WriteEepromExtendedData(hdkHandle, offset, dataSize, Data);
	if( result != ERROR_SUCCESS ) {
		// handle error here...
		Hdk7545_Close(&hdkHandle);
		return result;
	}

	Hdk7545_Close(&hdkHandle);
	return ERROR_SUCCESS;
}
```

4.7.5 API Structures

The following C Structures are declared in the file hdk7545Structs.h:

4.7.5.1 Hdk7545_Eeprom

The Hdk7545_Eeprom structure is defined as follows:

```
struct Hdk7545_Eeprom
{
	DWORD m_Size;
	DWORD m_Version;
	DWORD m_EepromId;
	BYTE m_MfgTestRegion[HDK7545_EEPROM_MFGTESTREGION_SIZE];
	char m_PartNumber[HDK7545_EEPROM_PART_NUMBER_SIZE+1];
	char m_SerialNumber[HDK7545_EEPROM_SERIAL_NUMBER_SIZE+1];
	char m_Manufacturer[HDK7545_EEPROM_MANUFACTURER_SIZE+1];
	char m_Model[HDK7545_EEPROM_MODEL_SIZE+1];
	char m_HardwareRevision[HDK7545_EEPROM_HWREVISION_SIZE+1];
	DWORD m_EepromSize;
	DWORD m_ExtendedSize;
}
```

The fields of the Hdk7545_Eeprom structure are described below:

- **m_Size** – must be set to the size of the Hdk7545_Eeprom structure [sizeof(Hdk7545Eeprom)].
- **m_Version** – must be set to the structure version [HDK7545_EEPROM_VERSION]. The EEPROM version is defined in the *Hdk7545Consts.h* header file.
- **m_EepromID** – this field can be ignored. It is set by the *Hdk7545_ReadEepromHeader* and *Hdk7545_WriteEepromHeader* functions.
- **m_MfgTestRegion** – this field can be ignored. It is only used by the manufacturer for manufacturing test purposes.
- **m_PartNumber** – this field stores the device part number.
- **m_SerialNumber** – this field stores the device serial number.
- **m_Manufacturer** – this field stores the device manufacturer.
- **m_Model** – this field stores the device model.
- **m_HardwareRevision** – this field stores the device hardware revision.

Note: The above five null-terminated ASCII char fields have an extra character in order to allow the use of string functions. the HDK will strip the terminating null character if the character data completely fills the EEPROM field.

- **m_EepromSize** – this field stores the EEPROM size in bytes (default value is 128). The field size (if modified) is rounded down to the nearest multiple of 128 before being stored in the EEPROM. The maximum value allowed is 32640 (255*128) bytes, or the actual size of the EEPROM, whichever is less.
- **m_ExtendedSize** – this field stores the amount of extended data available to developers. This is a read-only field, derived from the total EEPROM size less the EEPROM header size, typically 38 bytes (128 - 90). It can be accessed with the *Hdk7545_ReadEepromExtendedData* and *Hdk7545_WriteEepromExtendedData* functions.

For more information on the use of this structure and the reading and writing of EEPROMs, see Section 8.3: “EEPROM Data Specification”.

4.7.6 API Enumerations

The following enumerations are declared in the file Hdk7545Consts.h:

4.7.6.1 Hdk7545_PowerMode

The Hdk7545_PowerMode enumeration is defined as follows:

```
typedef enum {
       Hdk7545_PowerMode_Auto = 1,
       Hdk7545_PowerMode_Manual = 2,
       Hdk7545_PowerMode_Invalid = 0xffffffff
} Hdk7545_PowerMode;
```

4.7.6.2 Hdk7545_Connector

The Hdk7545_Connector enumeration is defined as follows:

```
typedef enum {
       Hdk7545_Connector_Expansion1 = 0,
       Hdk7545_Connector_Expansion2 = 1,
       Hdk7545_Connector_Expansion3 = 2,
       Hdk7545_Connector_Invalid = 0xffffffff
} Hdk7545_Connector;
```

4.7.6.3 Hdk7545_PinDirection

The Hdk7545_PinDirection enumeration is defined as follows:

```
typedef enum {
       Hdk7545_PinDirection_Output = 0,
       Hdk7545_PinDirection_Input
} Hdk7545_PinDirection;
```

4.7.6.4 Hdk7545_PinFunction

The Hdk7545_PinFunction enumeration is defined as follows:

```
typedef enum {
       Hdk7545_PinFunction_GPIO = 0,
       Hdk7545_PinFunction_Alternate
} Hdk7545_PinFunction;
```

4.7.6.5 Hdk7545_PinState

The Hdk7545_PinState enumeration is defined as follows:

```
typedef enum {
       Hdk7545_PinState_Clr = 0,
       Hdk7545_PinState_Set = 1,
       Hdk7545_PinState_Unknown = 0xffffffff
} Hdk7545_PinState;
```

4.7.6.6 Hdk7545_PullUpDown

The Hdk7545_PullUpDown enumeration is defined as follows:

```
typedef enum {
       Hdk7545_PullUpDown_Pullup = 1,
       Hdk7545_PullUpDown_Pulldown
} Hdk7545_PullUpDown;
```

4.7.6.7 Hdk7545_PinMode

The Hdk7545_PinMode enumeration is defined as follows:

```
typedef enum {
       Hdk7545_PinMode_NoInterrupt = 0x00,
       Hdk7545_PinMode_InterruptLowHigh = 0x02,     // edge-triggered*
       Hdk7545_PinMode_InterruptHighLow  = 0x04,    // edge-triggered*
       Hdk7545_PinMode_InterruptLow      = 0x08,    // level-triggered
       Hdk7545_PinMode_InterruptHigh     = 0x10,    // level-triggered
} Hdk7545_PinMode;
// *Edge-triggered modes are not supported on Omnii XT10.
```

4.7.7 Omnii HDK API Constants

The following constants are defined in the file Hdk7545Consts.h:

```
// transmit
#define HDK7545_GPIO_PIN_EXPANSION_GPIO0_TXD               0
// receive
#define HDK7545_GPIO_PIN_EXPANSION_GPIO1_RXD               1
// clear-to-send (active low)
#define HDK7545_GPIO_PIN_EXPANSION_GPIO2_CTSN              2
// ready-to-send (active low)
#define HDK7545_GPIO_PIN_EXPANSION_GPIO3_RTSN              3

// Master Output, Slave Input (output from master)
#define HDK7545_GPIO_PIN_EXPANSION_GPIO4_MOSI              4
// Master Input, Slave Output (output from slave)
#define HDK7545_GPIO_PIN_EXPANSION_GPIO5_MISO              5
```

```
// Serial Clock (output from master)
#define HDK7545_GPIO_PIN_EXPANSION_GPIO6_SCLK              6
// Chip Select (active low; output from master)
#define HDK7545_GPIO_PIN_EXPANSION_GPIO7_CSN               7

// the EEPROM structure version information
#define HDK7545_EEPROM_VERSION_MAJOR                       0x0001
#define HDK7545_EEPROM_VERSION_MINOR                       0x0002
#define HDK7545_EEPROM_VERSION \
        ((HDK7545_EEPROM_VERSION_MAJOR << 16) |
        (HDK7545_EEPROM_VERSION_MINOR))

#define HDK7545_EEPROM_MFGTESTREGION_SIZE                  10
#define HDK7545_EEPROM_PART_NUMBER_SIZE                    16
#define HDK7545_EEPROM_SERIAL_NUMBER_SIZE                  16
#define HDK7545_EEPROM_MANUFACTURER_SIZE                   20
#define HDK7545_EEPROM_MODEL_SIZE                          20
#define HDK7545_EEPROM_HWREVISION_SIZE                     4
```

MECHANICAL CONSIDERATIONS 5

5.1 Overview

This chapter describes the physical connectors, space and mounting of expansion modules.

5.2 Materials

The recommended material for manufacturing expansion modules that attach to the end-cap, back cover or main housing, is Sabic Lexan EXL9134. The recommended texturing is VDI27 or VDI33.

5.3 HDK Mechanical Files

The Hardware Development Kit provides the following mechanical models and drawings:

5.3.1 3D Files

IGES and STEP files provide 3D models of Omnii components for viewing with CAD software. These models give the exact forms and dimensions of the components so that modules can be designed to fit with precision.

Table 5.1 3D Files

Description	Filename
3D CAD models of the auto-range standard back cover	Back_Cover_Auto_Std_3D.igs
	Back_Cover_Auto_Std_3D.stp
3D CAD models of the expansion back cover, showing end-cap and pod expansion openings with sealing overmoulds, and keep-away areas for camera and speaker options	Back_Cover_Expan_3D.igs
	Back_Cover_Expan_3D.stp
3D CAD models of the large standard back cover	Back_Cover_Large_Std_3D.igs
	Back_Cover_Large_Std_3D.stp
3D CAD models of the end-cap with GPS antenna	Endcap_GPS_3D.igs
	Endcap_GPS_3D.stp
3D CAD model of the standard end-cap	Endcap_Standard_3D.stp
3D CAD model of Omnii chassis with back cover removed showing keep-away areas, with no GPS or WWAN radio installed	Omnii_Connectors_NoRadio_3D.stp
3D CAD model of Omnii chassis with back cover removed showing keep-away areas, with GPS and WWAN radio installed	Omnii_Connectors_Radio_3D.stp
3D CAD models of the standard scanner pod	Scanner_Pod_Std_3D.igs
	Scanner_Pod_Std_3D.stp

5.3.2 2D Files

PDF and DWG files provide 2D drawings of the surfaces and attachment points where expansion modules can be mounted to Omnii. The drawings show the exact locations and relative positions of screw mountings, etc.

Table 5.2 2D Files

Description	Filename
2D line drawings of the auto-range standard back cover with locations of mounting points for the back cover and pistol grip	Back_Cover_Auto_Std_2D.dwg
	Back_Cover_Auto_Std_2D.pdf
2D line drawings of the expansion back cover with locations of mounting points for the back cover, pistol grip, end-cap and pod expansion	Back_Cover_Expan_2D.dwg
	Back_Cover_Expan_2D.pdf
2D line drawings of the large standard back cover with locations of mounting points for the back cover and pistol grip	Back_Cover_Large_Std_2D.dwg
	Back_Cover_Large_Std_2D.pdf
2D line drawings of the end-cap with GPS antenna	Endcap_GPS_2D.dwg
	Endcap_GPS_2D.pdf
2D line drawings of the standard end-cap	Endcap_Standard_2D.dwg
	Endcap_Standard_2D.pdf
Artwork for the 36-key, alpha modified, numeric calculator, 12 Fn keyboard hard caps	KB_HardCaps_36ModNumCal12.pdf
Artwork for the 36-key numeric telephony, 12 Fn keyboard hard caps	KB_HardCaps_36NumTel12.pdf
Artwork for the 59-key, alpha ABC, numeric telephony, 6 Fn keyboard hard caps	KB_HardCaps_59ABCTel6.pdf
Artwork for the 36-key, alpha modified, numeric calculator, 12 Fn keyboard overlay	KB_Overlay_36ModNumCal12.pdf
Artwork for the 36-key numeric telephony, 12 Fn keyboard overlay	KB_Overlay_36NumTel12.pdf
Artwork for the 59-key, alpha ABC, numeric telephony, 6 Fn keyboard overlay	KB_Overlay_59ABCTel6.pdf
2D line drawings of Omnii chassis and MLB showing locations of expansion ports with no GPS radio installed	Omnii_Connectors_NoRadio_2D.dwg
	Omnii_Connectors_NoRadio_2D.pdf
2D line drawings of Omnii chassis and MLB showing locations of expansion ports with GPS and WWAN radio installed	Omnii_Connectors_Radio_2D.dwg
	Omnii_Connectors_Radio_2D.pdf
2D line drawings of the standard scanner pod with locations of mounting points for the pod and for the scanner assembly	Scanner_Pod_Std_2D.dwg
	Scanner_Pod_Std_2D.pdf

5.4 Expansion Module and Device Design and Installation

5.4.1 Physical Space Considerations

Whether installing a non-standard expansion device into an existing Psion Teklogix module (back cover, end-cap or pod expansion), or creating a custom module to contain an expansion device, it is vitally important that your device does not intrude on space already occupied by the internal components of the hand-held computer.

To assist you in this, the following 3D files are included in the HDK:

Table 5.3 Files Showing Occupied Areas

Description	Filename
3D CAD model of Omnii chassis with back cover removed showing keep-away areas, with **no** GPS or WWAN radio installed	Omnii_Connectors_NoRadio_3D.stp
3D CAD model of Omnii chassis with back cover removed showing keep-away areas, with GPS and WWAN radio installed	Omnii_Connectors_Radio_3D.stp

To assist in forward-compatibility, these files show areas that will be occupied by modules (such as the WWAN radio), which are unavailable in the Omnii XT10 but which will appear in future models.

The STEP files also show the exact locations of the expansion ports in three dimensions so that you can design your device and flex cables to align properly. These locations are also shown in the following two-dimensional .dwg files:

Table 5.4 Files Showing Occupied Areas

Description	Filename
2D line drawings of Omnii chassis and MLB showing locations of expansion ports with **no** GPS or WWAN radio installed	Omnii_Connectors_NoRadio_2D.dwg
	Omnii_Connectors_NoRadio_2D.pdf
2D line drawings of Omnii chassis and MLB showing locations of expansion ports with GPS and WWAN radio installed	Omnii_Connectors_Radio_2D.dwg
	Omnii_Connectors_Radio_2D.pdf

The information in the above files, in conjunction with the files indicated in the sections that follow, will allow you to design modules that fit and connect precisely with your Omnii hand-held computer.

5.4.2 End-Cap Modules and Devices

This section describes the specifics of designing and mounting devices for the end-cap area of the Omnii expansion back cover.

Table 5.5 Associated Files for End-Cap

Description	Filename
2D line drawings of the auto-range standard back cover with locations of mounting points for the back cover and pistol grip	Back_Cover_Expan_2D.dwg
	Back_Cover_Expan_2D.pdf
3D CAD models of the expansion back cover, showing end-cap and pod expansion openings with sealing overmoulds, and keep-away areas for camera and speaker options	Back_Cover_Expan_3D.igs
	Back_Cover_Expan_3D.stp
2D line drawings of the end-cap with GPS antenna	Endcap_GPS_2D.dwg
	Endcap_GPS_2D.pdf
3D CAD models of the end-cap with GPS antenna	Endcap_GPS_3D.igs
	Endcap_GPS_3D.stp
2D line drawings of the standard end-cap	Endcap_Standard_2D.dwg
	Endcap_Standard_2D.pdf
3D CAD model of the standard end-cap	Endcap_Standard_3D.stp

Table 5.6 Required Fasteners

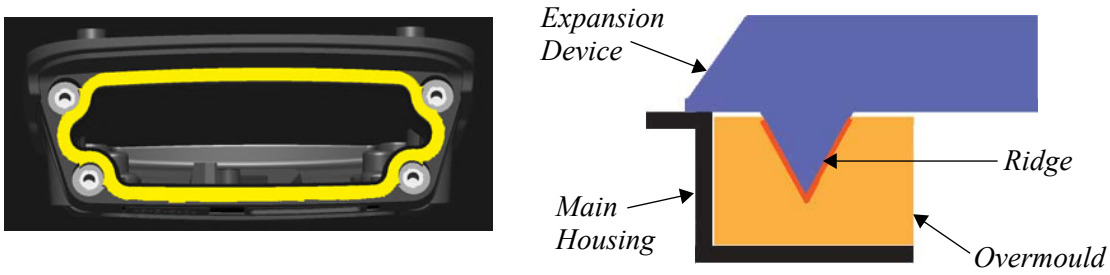
How Used	Quantity	Part Number	Screw Size	Torque (in-lb)	Bit
End-Cap to Main Housing	4	1004948	M3x6	5.0	T10
Back Cover to Main Housing (Back)	8	1004948	M3x6	5.0	T10
Back Cover to Main Housing (Front)	2	1004946	M2x8	2.5	T6

Designing End-Cap Modules

The 3D CAD model of the expansion back cover shows the precise shape and size of the end-cap opening, so that an end-cap module can be constructed to fit with precision.

To maintain proper protection against water and dust ingress, end-cap modules must form a tight seal with the back cover. A soft overmould around the perimeter of the opening is used to create a seal with the attached end-cap. The end-cap module must be designed with a corresponding hard ridge that presses into the overmould to form this seal. The 3D CAD model of the standard end-cap can be used to obtain the required width, depth and shape of this ridge.

Figure 5.1 End-Cap Overmould



The expansion back cover has options to attach a speaker and/or a camera mounted in the back of the unit. End-cap mounted devices should not intrude on the space required for these options, even if they are not installed at the time of development. These “keep-away” areas are indicated in the 3D CAD file for the expansion back cover.

End-cap modules are attached to the expansion back cover with four M3 screws. The end-cap module must be designed with holes for these four screws, to be torqued to 5 in-lb (0.56 N-m) in order to maintain a proper seal.

Figure 5.2 Location of End-Cap Mounting Points

Installing End-Cap Modules

The standard end-cap is attached with four screws. Removing and installing these screws requires a T10 Torx screwdriver.

To remove the end-cap:

1. Remove the four T10 screws securing the end-cap to the main housing.
2. Remove the end-cap.
3. If a GPS radio is installed in the end-cap, remove the back cover and disconnect the antenna from the socket on the GPS board. To remove the back cover:
   a. If a scanner or imager pod module is installed, first remove the four T10 Torx screws holding the pod to the housing, and disconnect the intermediate scanner flex cable from the scanner module (the intermediate scanner flex cable connects the scanner module to the scanner port on the main logic board).
   b. Remove the eight T10 Torx screws holding the back cover to the main housing.
   c. Remove the two T6 Torx screws at the top of the display on the front of the unit.
   d. Separate the back cover from the main housing. You can now disconnect the antenna wire from the GPS module.

To install the end-cap mounted device:

1. If the device requires connection to an internal socket:
   a. If the back cover is still attached, remove the back cover as described in step 3 above.
   b. Route the device cable through the top of the unit and connect it to the appropriate internal socket on the GPS or main logic board.
   c. Install eight M3x6 T10 Torx screws to secure the back cover to the main housing. Torque the screws to 5.0 in-lb (0.56 N-m) in the sequence indicated below to ensure a consistent seal around the perimeter.

   d. Install the two M2x8 T6 Torx screws at the top of the display on the front of the unit. Torque the screws to 2.5 in-lb (0.28 N-m).
   e. If a scanner or imager pod module is used, connect the scanner intermediate flex cable to the scanner through the pod opening, then secure the pod in place using four M3x5 T10 Torx screws. Torque the screws to 5.0 in-lb (0.56 N-m).
2. Fit the end-cap to the top of the unit.
3. Insert and tighten the four M3x6 T10 Torx screws that hold the end-cap to the top of the unit. Torque the screws to 5.0 in-lb (0.56 N-m).

5.4.3 Pod Expansion Modules and Devices

This section describes the specifics of designing and mounting expansion devices in the pod opening on the back of Omnii.

Table 5.7 Associated Files for Pod Expansion

Description	Filename
2D line drawings of the standard scanner pod with locations of mounting points for the pod and for the scanner assembly	Scanner_Pod_Std_2D.dwg
	Scanner_Pod_Std_2D.pdf
3D CAD models of the standard scanner pod	Scanner_Pod_Std_3D.igs
	Scanner_Pod_Std_3D.stp

Table 5.8 Required Fasteners

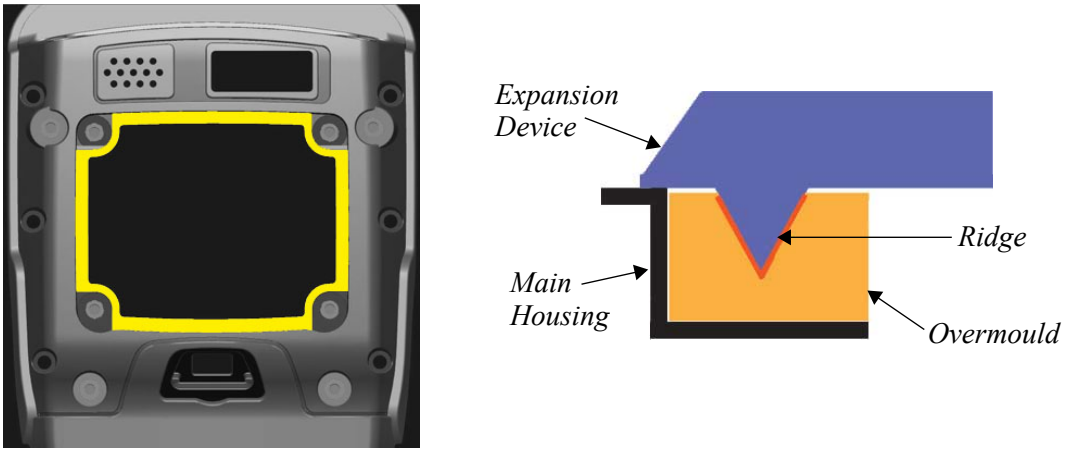
How Used	Quantity	Part Number	Screw Size	Torque (in-lb)	Bit
Pod Expansion/Blanking Plate to Main Housing	4	1005090	M3x5	5.0	T10
Scanner Assembly to Pod Expansion	4	1004945	M2.5x5	2.5	T6

Designing Pod Expansion Modules

The 3D CAD model of the expansion back cover shows the precise shape and size of the pod opening, so that a pod expansion module can be constructed to fit with precision.

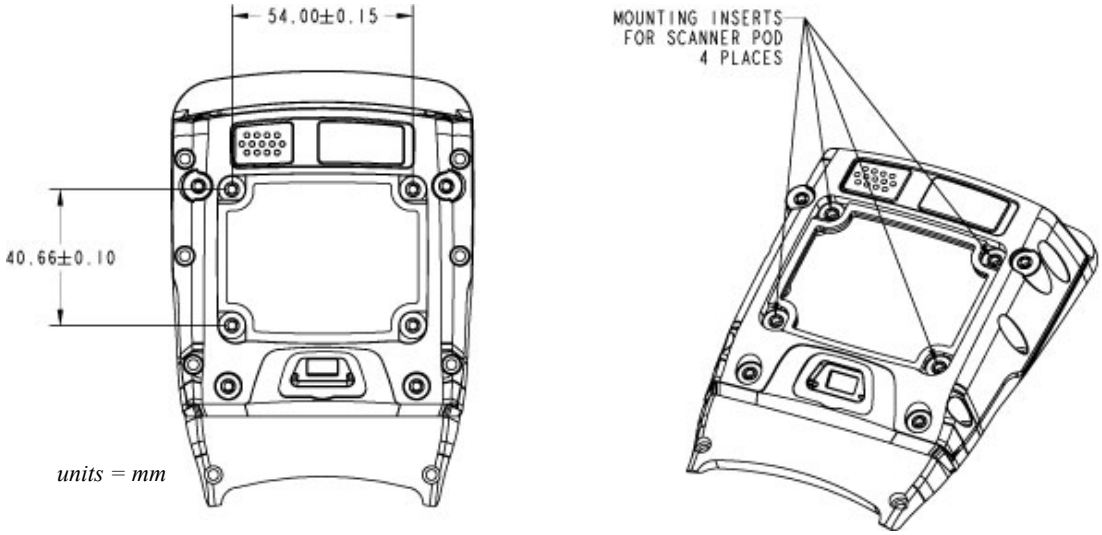
To maintain proper protection against water and dust ingress, pod expansion modules must form a tight seal with the back cover. A soft overmould around the perimeter of the opening is used to create a seal with the attached pod. The pod expansion module must be designed with a corresponding hard ridge that presses into the overmould to form this seal. The 3D CAD model included with the HDK of the standard scanner pod can be used to obtain the required width, depth and shape of this ridge.

Figure 5.3 Pod Expansion Overmould



Pod expansion modules are attached to the expansion back cover with four M3 screws. The pod expansion module must be designed with holes for these four screws, to be torqued to 5 in-lb (0.56 N-m) in order to maintain a proper seal.

Figure 5.4 Location of Pod Expansion Mounting Points



Installing Pod Expansion Modules

The Omnii expansion back cover has an opening where a pod expansion module can be attached. If no pod expansion is in use, a blanking plate is attached in its place to seal the opening. The pod expansion or blanking plate is attached using four screws which can be removed using a T10 Torx screwdriver.

To remove the pod expansion/blanking plate:

1. Remove the four T10 Torx screws securing the pod or plate to the main housing.
2. Separate the pod or plate from the unit.
3. When removing a pod expansion, there is a flex cable that connects the expansion device to the main logic board. Disconnect the flex cable from the pod expansion module.

Removing the pod expansion or blanking plate allows a flex cable to be routed into the unit through the pod opening and attached to one of the internal connectors. It will likely be necessary to separate the whole back cover from the unit in order to properly access the connector and install the cable.

To remove the back cover:

1. Remove the eight T10 Torx screws holding the back cover to the main housing.
2. Remove the two T6 Torx screws at the top of the display on the front of the unit.

To attach the back cover:

1. Make sure all cables are attached to the appropriate sockets on the main unit.
2. Seat the back cover on the main housing, routing the flex cable for the pod expansion module through the pod opening.
3. Install eight M3x6 T10 Torx screws to secure the back cover to the main housing. Torque the screws to 5.0 in-lb (0.56 N-m) in the sequence indicated below to ensure a consistent seal around the perimeter.

4. Install two M2x8 T6 Torx screws in the two mounting points at the top of the display on the front of the unit. Torque the screws to 2.5 in-lb (0.28 N-m).

To attach the new module in the pod opening:

1. Connect the pod expansion flex cable to the pod module.
2. Seat the pod module on the main housing.
3. Install four M3x5 T10 Torx screws to secure the pod module to the main housing. Torque the screws to 5.0 in-lb (0.56 N-m).

5.4.4 Back Cover Modules and Devices

This section describes the specifics of designing and mounting custom-made back covers on Omnii.

Table 5.9 Associated Files for Back Covers

Description	Filename
2D line drawings of the auto-range standard back cover with locations of mounting points for the back cover and pistol grip	Back_Cover_Auto_Std_2D.dwg
	Back_Cover_Auto_Std_2D.pdf
3D CAD models of the auto-range standard back cover	Back_Cover_Auto_Std_3D.igs
	Back_Cover_Auto_Std_3D.stp
2D line drawings of the expansion back cover with locations of mounting points for the back cover, pistol grip, end-cap and pod expansion	Back_Cover_Expan_2D.dwg
	Back_Cover_Expan_2D.pdf
3D CAD models of the expansion back cover, showing end-cap and pod expansion openings with sealing overmoulds, and keep-away areas for camera and speaker options	Back_Cover_Expan_3D.igs
	Back_Cover_Expan_3D.stp
2D line drawings of the large standard back cover with locations of mounting points for the back cover and pistol grip	Back_Cover_Large_Std_2D.dwg
	Back_Cover_Large_Std_2D.pdf
3D CAD models of the large standard back cover	Back_Cover_Large_Std_3D.igs
	Back_Cover_Large_Std_3D.stp

Table 5.10 Required Fasteners

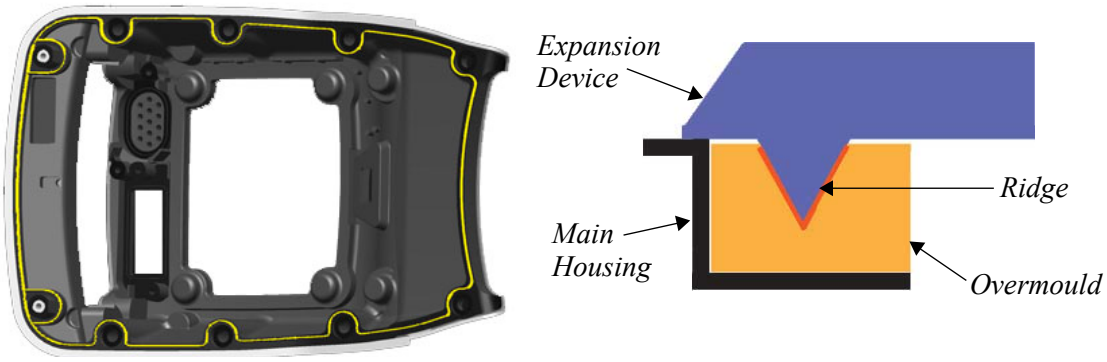
How Used	Quantity	Part Number	Screw Size	Torque (in-lb)	Bit
Pod Expansion to Expansion Back Cover	4	1005090	M3x5	5.0	T10
Back Cover to Main Housing (Back)	8	1004948	M3x6	5.0	T10
Back Cover to Main Housing (Front)	2	1004946	M2x8	2.5	T6
Inserts	2	1005273	M2x5	N/A	N/A

Designing Back Cover Modules

To maintain proper protection against water and dust ingress, back cover modules must form a tight seal with the main housing. A soft overmould around the perimeter of the opening creates a seal with the attached back cover. The back cover module must be designed with a corresponding hard ridge that presses into the overmould to form this seal. The 3D CAD model of any of the supplied back covers can be used to obtain the required width, depth and shape of this ridge.

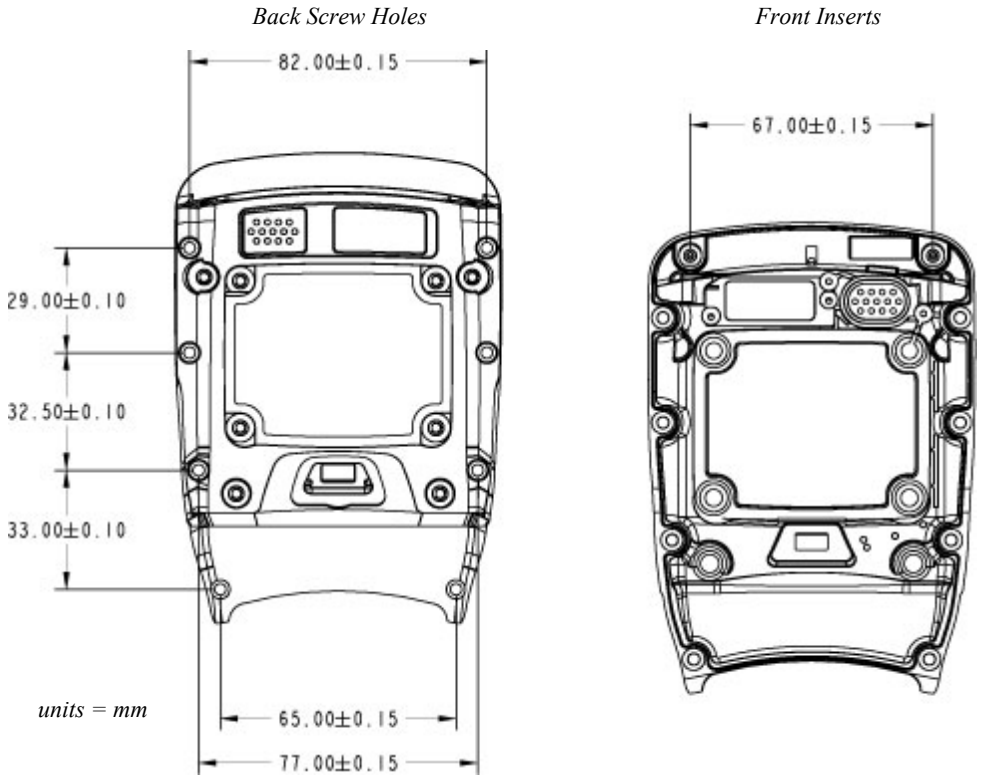
The following illustration indicates the ridge on the expansion back cover. This ridge is identical on all back cover variants.

Figure 5.5 Sealing Ridge on Expansion Back Cover



Back cover modules are attached to the Omnii main housing with eight T10 Torx screws that enter from the back, plus two T6 Torx screws that enter through the front of the unit above the display, and fit into inserts mounted in the inside of the back cover module. Any back cover module you design must have these eight screw holes and two inserts to secure the back cover to the main housing.

Figure 5.6 Location of Back Cover Mounting Points



Installing Back Cover Modules

To remove the existing back cover:

1. If a pod expansion module is installed on the expansion back cover, it should be removed first:
   a. Remove the four T10 Torx screws securing the pod to the main housing.
   b. Lift the pod module from the unit.
   c. Disconnect the flex cable from the pod expansion module.
2. Remove the eight T10 Torx screws holding the back cover to the main housing.
3. Remove the two T6 Torx screws at the top of the display on the front of the unit.
4. Gently lift the back cover from the main housing to allow access to the cables that are still attached.
5. Disconnect any cables attaching the back cover to the main unit (trigger switch, scanner, speaker, camera, GPS antenna are all possibilities).
6. Remove the back cover module.

To attach the new back cover module:

1. Connect any cables to the appropriate sockets on the main logic board and GPS module (if installed).
2. Seat the back cover on the main housing.
3. Install eight M3x6 T10 Torx screws to secure the back cover to the main housing. Torque the screws to 5.0 in-lb (0.56 N-m) in the sequence indicated below to ensure a consistent seal around the perimeter.

4. Install two M2x8 T6 Torx screws in the two mounting points at the top of the display on the front of the unit. Torque the screws to 2.5 in-lb (0.28 N-m).

5.4.5 Pistol Grip Modules

This section describes the specifics of designing and mounting attachments to the pistol grip mounting points on the back cover modules.

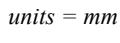
Table 5.11 Associated Files for Pistol Grips (Back Covers)

Description	Filename
2D line drawings of the auto-range standard back cover with locations of mounting points for the back cover and pistol grip	Back_Cover_Auto_Std_2D.dwg
	Back_Cover_Auto_Std_2D.pdf
3D CAD models of the auto-range standard back cover	Back_Cover_Auto_Std_3D.igs
	Back_Cover_Auto_Std_3D.stp
2D line drawings of the expansion back cover with locations of mounting points for the back cover, pistol grip, end-cap and pod expansion	Back_Cover_Expan_2D.dwg
	Back_Cover_Expan_2D.pdf
3D CAD models of the expansion back cover, showing end-cap and pod expansion openings with sealing overmoulds, and keep-away areas for camera and speaker options	Back_Cover_Expan_3D.igs
	Back_Cover_Expan_3D.stp
2D line drawings of the large standard back cover with locations of mounting points for the back cover and pistol grip	Back_Cover_Large_Std_2D.dwg
	Back_Cover_Large_Std_2D.pdf
3D CAD models of the large standard back cover	Back_Cover_Large_Std_3D.igs
	Back_Cover_Large_Std_3D.stp

Table 5.12 Required Fasteners

How Used	Quantity	Part Number	Screw Size	Torque (in-lb)	Bit
Pistol Grip to Main Housing	4	1005118	M3x6	5.0	Phillips

Figure 5.7 Location of Pistol Grip Mounting Points on Expansion Back Cover



5.4.6 Keyboard Modules

Custom keyboard overlays and hard caps may be developed for use on Omnii. The HDK provides PDF files with details of the artwork that is used on the factory-installed components so that the colours may be faithfully reproduced to maintain the contrast, visibility and overall look of the unit.

5.4.6.1 Keyboard Overlays

This section describes the specifics of designing and installing new keyboard overlays on Omnii.

Table 5.13 Associated Files for Keyboard Overlays

Description	Filename
Artwork for the 36-key, alpha modified, numeric calculator, 12 Fn keyboard overlay	KB_Overlay_36ModNumCal12.pdf
Artwork for the 36-key numeric telephony, 12 Fn keyboard overlay	KB_Overlay_36NumTel12.pdf
Artwork for the 59-key, alpha ABC, numeric telephony, 6 Fn keyboard overlay	KB_Overlay_59ABCTel6.pdf

The factory keyboard overlay may be replaced with a custom keyboard overlay. There are two form factors of Omnii keyboards: a 36-key numeric format, and a 59-key full alphanumeric format. Make sure you use the correct files and parts for your keyboard type.

You can install the custom overlay on an existing keyboard bezel, provided you first remove (peel off) the factory-installed overlay. Clean the bezel surface thoroughly with isopropyl alcohol to remove all traces of the old adhesive before installing the new overlay.

If you wish to install the overlay on a new keyboard bezel, or if you need to replace a damaged bezel, blank ones can be ordered using the following part numbers:

Table 5.14 Blank Keyboard Bezel Part Numbers

Part Number	Keyboard Bezel Form Factor
1004988	59-key alphanumeric
1004982	36-key numeric

5.4.6.2 Keyboard Hard Caps

This section describes the specifics of designing and installing new keyboard hard caps on Omnii.

Table 5.15 Associated Files for Keyboard Hard Caps

Description	Filename
Artwork for the 36-key, alpha modified, numeric calculator, 12 Fn keyboard hard caps	KB_HardCaps_36ModNumCal12.pdf
Artwork for the 36-key numeric telephony, 12 Fn keyboard hard caps	KB_HardCaps_36NumTel12.pdf
Artwork for the 59-key, alpha ABC, numeric telephony, 6 Fn keyboard hard caps	KB_HardCaps_59ABCTel6.pdf

Installing new keyboard hard caps requires a new, clean, keyboard elastomer. You cannot remove the hard caps from an existing elastomer to install new ones. Blank elastomers can be ordered using the following part numbers:

Table 5.16 Blank Keyboard Elastomer Part Numbers

Part Number	Keyboard Elastomer Form Factor
1004986	59-key alphanumeric
1004980	36-key numeric

5.5 Installing Devices Inside Existing Modules

Schematics and drawings are provided for the purpose of mounting custom devices inside existing modules, including back covers, pod expansion modules and end-caps.

5.5.1 Standard Scanner Pod

The following files show the dimensions and contours of the standard scanner pod for the expansion back cover:

Table 5.17 Associated Files for Standard Scanner Pod

Description	Filename
2D line drawings of the standard scanner pod with locations of mounting points for the pod and for the scanner assembly	Scanner_Pod_Std_2D.dwg
	Scanner_Pod_Std_2D.pdf
3D CAD models of the standard scanner pod	Scanner_Pod_Std_3D.igs
	Scanner_Pod_Std_3D.stp

The following fasteners are required to secure the standard scanner pod to the expansion back cover:

Table 5.18 Required Fasteners

How Used	Quantity	Part Number	Screw Size	Torque (in-lb)	Bit
Scanner Pod to Expansion Back Cover	4	1005090	M3x5	5.0	T10

To remove the existing scanner pod:

1. Remove the four T10 Torx screws securing the scanner pod to the main housing.
2. Lift the pod from the unit.
3. Disconnect the scanner flex cable from the scanner module.

To attach the new pod module:

1. Attach any cables to the appropriate sockets on the main logic board or GPS module (if installed).
2. Seat the new pod module on the opening in the back cover.
3. Install four M3x5 T10 Torx screws to secure the scanner pod to the back cover. Torque the screws to 5.0 in-lb (0.56 N-m).

5.5.2 Large/Auto-Range Standard Back Covers

The following files show the dimensions and contours of the large and auto-range standard back covers:

Table 5.19 Associated Files for Large Standard Back Cover

Description	Filename
2D line drawings of the auto-range standard back cover with locations of mounting points for the back cover and pistol grip	Back_Cover_Auto_Std_2D.dwg
	Back_Cover_Auto_Std_2D.pdf
3D CAD models of the auto-range standard back cover	Back_Cover_Auto_Std_3D.igs
	Back_Cover_Auto_Std_3D.stp
2D line drawings of the large standard back cover with locations of mounting points for the back cover and pistol grip	Back_Cover_Large_Std_2D.dwg
	Back_Cover_Large_Std_2D.pdf
3D CAD models of the large standard back cover	Back_Cover_Large_Std_3D.igs
	Back_Cover_Large_Std_3D.stp

The following fasteners are required to secure a standard back cover to the main housing:

Table 5.20 Required Fasteners

How Used	Quantity	Part Number	Screw Size	Torque (in-lb)	Bit
Back Cover to Main Housing (Back)	8	1004948	M3x6	5.0	T10
Back Cover to Main Housing (Front)	2	1004946	M2x8	2.5	T6
Scanner Module to Back Cover	4	1004945	M2.5x5	2.5	T6

To remove the standard back cover:

1. Remove the eight T10 Torx screws holding the back cover to the main housing.
2. Remove the two T6 Torx screws at the top of the display on the front of the unit.
3. Gently lift the back cover from the main housing to allow access to the cables that are still attached.
4. Disconnect any cables attaching the back cover to the main unit.
5. Remove the back cover module.
6. If an existing scanner (or other) module is mounted in the back cover, remove the four T6 Torx screws holding the assembly to the back cover.

Insert the new module in the back cover, and secure using four M2.5x5 T6 Torx screws. Torque the screws to 2.5 in-lb (0.28 N-m).

To attach the back cover to the main housing:

1. Attach any cables to the appropriate sockets on the main logic board.
2. Seat the back cover on the main housing.

3. Install eight M3x6 T10 Torx screws to secure the back cover to the main housing. Torque the screws to 5.0 in-lb (0.56 N-m) in the sequence indicated below to ensure a consistent seal around the perimeter.

4. Install two M2x8 T6 Torx screws in the two mounting points at the top of the display on the front of the unit. Torque the screws to 2.5 in-lb (0.28 N-m).

5.5.3 End-Cap

The following files show the dimensions and contours of the standard and GPS end-caps:

Table 5.21 Associated Files for Large Integrated Back Cover

Description	Filename
2D line drawings of the end-cap with GPS antenna	Endcap_GPS_2D.dwg
	Endcap_GPS_2D.pdf
3D CAD models of the end-cap with GPS antenna	Endcap_GPS_3D.igs
	Endcap_GPS_3D.stp
2D line drawings of the standard end-cap	Endcap_Standard_2D.dwg
	Endcap_Standard_2D.pdf
3D CAD model of the standard end-cap	Endcap_Standard_3D.stp

The following fasteners are required to secure an end-cap to the standard back cover:

Table 5.22 Required Fasteners

How Used	Quantity	Part Number	Screw Size	Torque (in-lb)	Bit
End-Cap Module to Main Housing	4	1005090	M3x5	5.0	T10

To remove the existing end-cap:

1. If the existing end-cap contains a module connected to the main unit (such as the end-cap with GPS antenna), you must first remove the back cover of your Omnii:
   a. Remove the eight T10 Torx screws holding the back cover to the main housing.
   b. Remove the two T6 Torx screws at the top of the display on the front of the unit.
   c. Gently lift the back cover from the main housing to allow access to the cables that are still attached.
   d. Disconnect any cables attaching the back cover and end-cap to the main unit.

2. Remove the four Torx screws securing the end-cap to the main housing.
3. Gently pull the end-cap away from the main housing.

To attach the new pod module:

1. Route the cable for the end-cap module through the opening in the top of the unit, and connect to the appropriate socket on the main unit.
2. Fit the new end-cap module into place on the main housing, being careful not to catch the cable in the seal.
3. Install four M3x5 T10 Torx screws to secure the end-cap to the main housing. Torque the screws to 5.0 in-lb (0.56 N-m).
4. Replace the back cover.
   a. Attach any back cover cables to the appropriate sockets on the main logic board.
   b. Seat the back cover in place on the main housing.
   c. Install eight M3x6 T10 Torx screws to secure the back cover to the main housing. Torque the screws to 5.0 in-lb (0.56 N-m) in the sequence indicated below to ensure a consistent seal around the perimeter.

   d. Install two M2x8 Torx screws in the two mounting points at the top of the display on the front of the unit. Torque the screws to 2.5 in-lb (0.28 N-m).

OMNII EXPANSION PORTS AND CONNECTORS 6

6.1 Overview

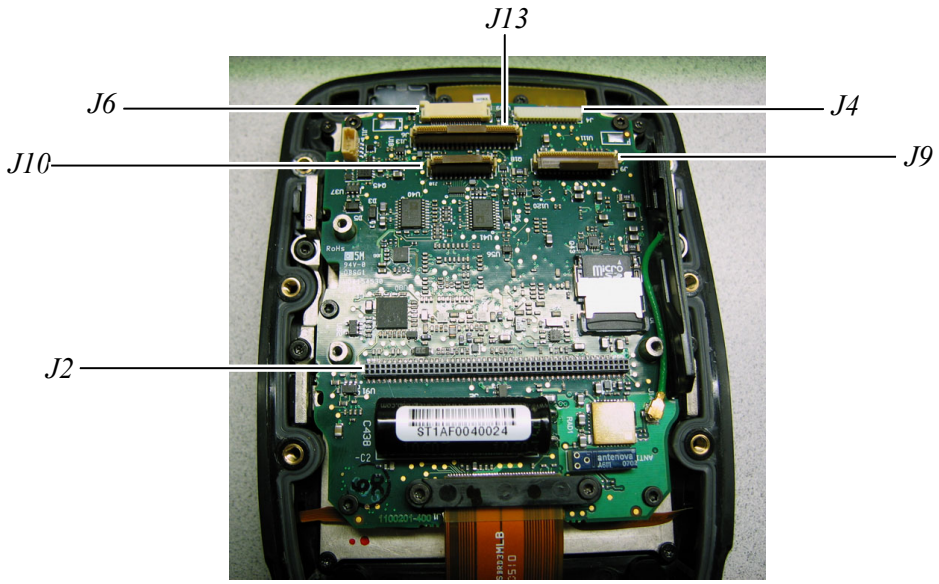
This chapter describes the Omnii connectors used for attaching expansion devices to the hand-held computer.

6.2 Connector Locations

Omnii has the following physical connectors located on the main logic board:

- Audio expansion connector (J4).
- End-cap connector / **Expansion Port 1** (J6).
- Pod expansion connector / **Expansion Port 2** (J10).
- 100-pin multi-function connector, includes **Expansion Port 3** (J2).
- Camera connector (J9) (not for use with Omnii XT10 HDK).
- Scanner/imager connector (J13) (not for use with Omnii XT10 HDK).

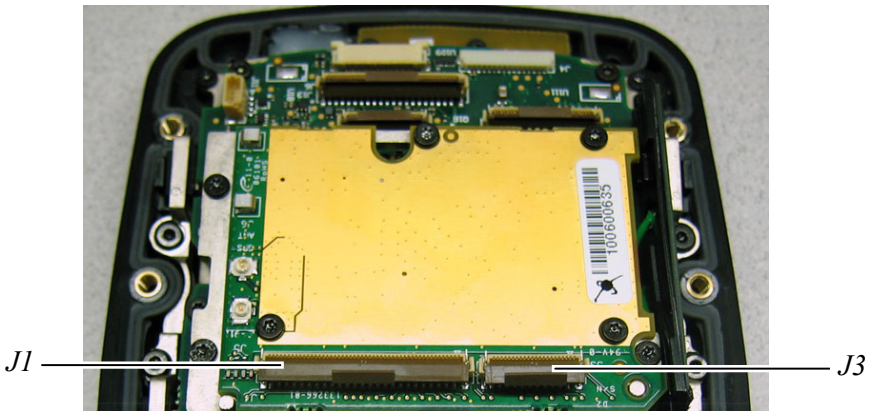
Figure 6.1 Connector Locations on the Omnii Main Logic Board



If a GPS module is installed, it will occupy the 100-pin connector. However, the unused pins are passed through to connectors on the back of the module, so they are still available for other devices. The connectors on the back of the GPS module provide access to:

- Proprietary interface pins reserved for future use (J1).
- **Expansion Port 3** (J3). This connector is identical to the pod expansion connector (expansion port 2) on the main logic board.

Figure 6.2 Connector Locations on the GPS Module



6.3 Audio Connector

Theory of Operation

The Omnii main logic board exposes a 16-pin audio connector, providing a means of connecting audio input and output devices such as headsets, push-to-talk handsets and external speakers.

Included in the expansion interface are the following connections:

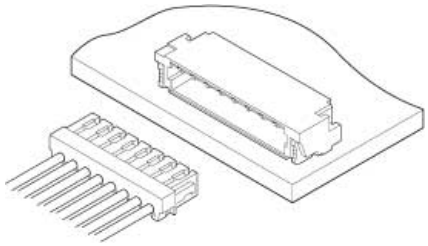
- **HPL** and **HPR**: left and right audio outputs (Headphone Left and Headphone Right)
- **HP_AMP_EN**: enable line for headphone amplifier circuit
- **HP_DETECT_N**: headset detection
- **MIC+** and **MIC-**: differential audio inputs from expansion board into main logic board
- **MIC_AMP_EN**: enable line for microphone amplifier and biasing circuitry
- **PTT_DETECT**: detection of push-to-talk switch
- **SPKR_EXP_R** and **SPKR_EXP_R+**: differential audio outputs for driving a loudspeaker (4 or 8 Ohms)

Audio Connector Details

The audio connector is a side-entry 0.8 mm pitch disconnectable insulation displacement connector.

Manufacturer:	JST
Manufacturer Part Number:	SM16B-SURS-TF(LF)(SN)
Mating Connector:	16SUR-32S
Number of Pins:	16
Current Rating:	0.5 A per pin for 32 AWG wire (do not use 36 AWG wire)

Figure 6.3 Diagram of JST Audio Connector

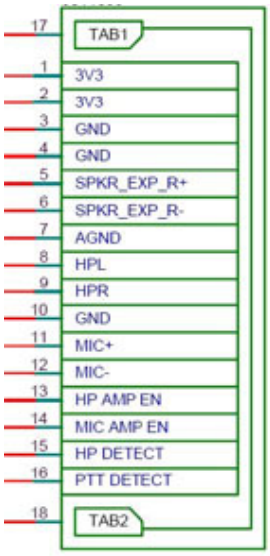


Pinout

Table 6.1 Pinout of 16-pin Audio Connector

Pin	Name	Direction (referenced to MLB)	Type	Function
1, 2	3V3	Output	Power	3.3 V supply input into audio expansion
3, 4, 7, 10	GND	Ground	Ground	Ground signal connected to ground plane on the main logic board
5	SPKR_EXP_R+	Output	Analog	Bridge tied load (BTL) audio signals from class D amplifier on MLB to drive a loudspeaker. Designed to drive a load down to a minimum of 4 Ohms. Capable of driving up to 510 mW into an 8 Ohm speaker with 1% THD.
6	SPKR_EXP_R-			
8	HPL	Output	Analog	Left headphone signal from audio codec on MLB. Capable of driving from -3.3 V to +3.3 V. Biased at 0 V.
9	HPR	Output	Analog	Right headphone signal from audio codec on MLB. Capable of driving from -3.3 V to +3.3 V. Biased at 0 V.
11	MIC+	Input	Analog	Differential microphone signals coming from expansion board. These inputs are fed into the codec on the main logic board. Maximum input level of each of these signals is 0.50 VRMS. The codec applies its own bias voltage on these signals and therefore they need to be AC coupled on the expansion board.
12	MIC-			
13	HP_AMP_EN	Output	Digital	Active-high signal to enable headphone amplifier when headphones are in use. 1.8 V signal coming from the main logic board.
14	MIC_AMP_EN	Output	Digital	Active-high signal to enable microphone preamplifier and/or biasing circuitry when microphone is in use. 1.8 V signal coming from the main logic board.
15	HP_DETECT	Input	Digital	Active-low signal when a headset is plugged into the expansion board. 1.8 V signal. 100 kohm pull-up resistor on the main logic board.
16	PTT_DETECT	Input	Digital	Active-low signal when a push-to-talk microphone is enabled. Enables reading of the MIC lines when active. A 100 kohm pull-up resistor to 1.8 V is placed on the MLB.

Figure 6.4 Pin Diagram of 16-pin Audio Expansion



6.3.1 Audio Reference Designs

Three sample reference designs are included in the files for the Omnii HDK. There are sample designs for a single-ended headset, a push-to-talk handset and an external speaker.

Table 6.2 Audio Reference Design Files

Description	Filename
Reference design for a single-ended headset	Audio_Single-Ended_Headset.pdf
Reference design for a push-to-talk handset	Audio_PTT.pdf
Reference design for an external speaker	Audio_External_Speaker.pdf

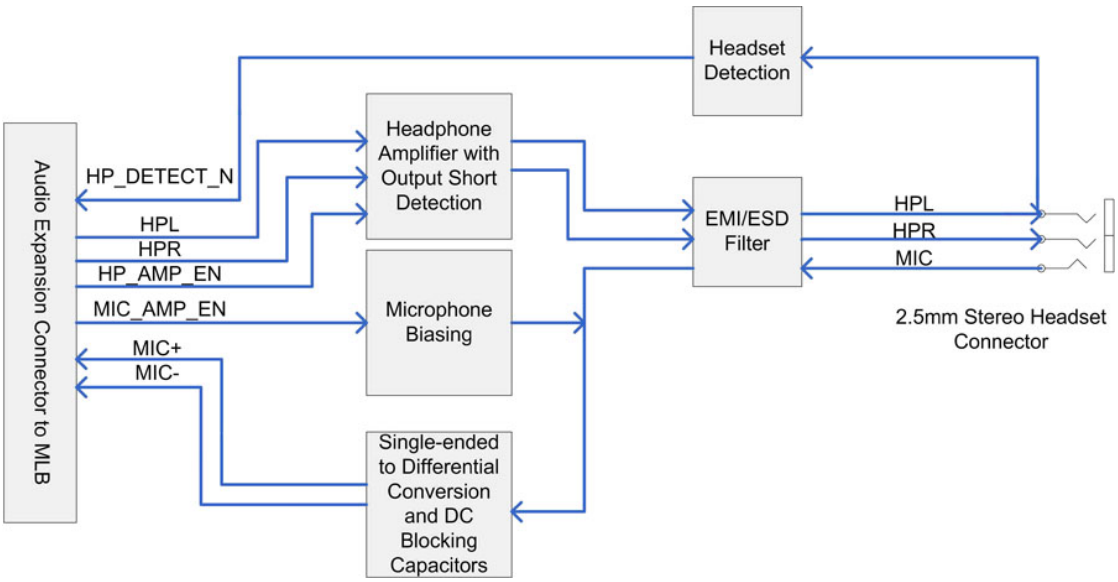
The designs are examined in detail in the following sections to demonstrate how to design modules that connect to the audio expansion interface connector.

6.3.1.1 Single-Ended Headset

The single-ended headset audio expansion board can be used with most commercially available 2.5 mm audio headsets. The MAX9720 headphone amplifier in the reference design will detect whether a mono or stereo headset has been inserted and adjust the output signals accordingly.

To prevent GSM interference in the 850/900/1800/1900 MHz bands from coupling into the audio path and demodulating into the audio band, 33 pF and 10 pF capacitors are placed at various locations throughout the circuit.

Figure 6.5 Block Diagram of the Single-Ended Headset Expansion Board



Headphone Amplifier

The headphone amplifier is used to drive the speakers in the headset. It is only enabled when a headset is plugged into the audio jack and the HP_AMP_EN signal is asserted. Additionally, the MAX9720 headphone amplifier has automatic mono/stereo detection. In the event that a mono headset is plugged in, the amplifier will automatically reroute its audio inputs to a single output channel.

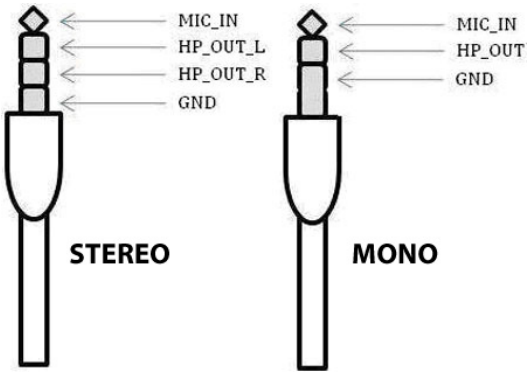
Microphone Biasing

The headset microphone is biased at 3.0 V when in use. It is biased from a low noise, low drop-out linear regulator (LDO). When the microphone is not in use, the LDO is disabled and the biasing is removed.

2.5 mm Audio Jack

The 2.5 mm audio jack can be used with a single-ended mono, or stereo, headset. When the headset connector is inserted into the jack, the HP_DETECT line drops low. Without the HP_DETECT signal, audio will not be routed to/from the audio expansion board. The figure below illustrates the necessary connections on standard commercial stereo and mono audio plugs that will allow them to interface with the audio jack in the reference design.

Figure 6.6 2.5 mm Single-Ended Headset Audio Jack Connection Diagram



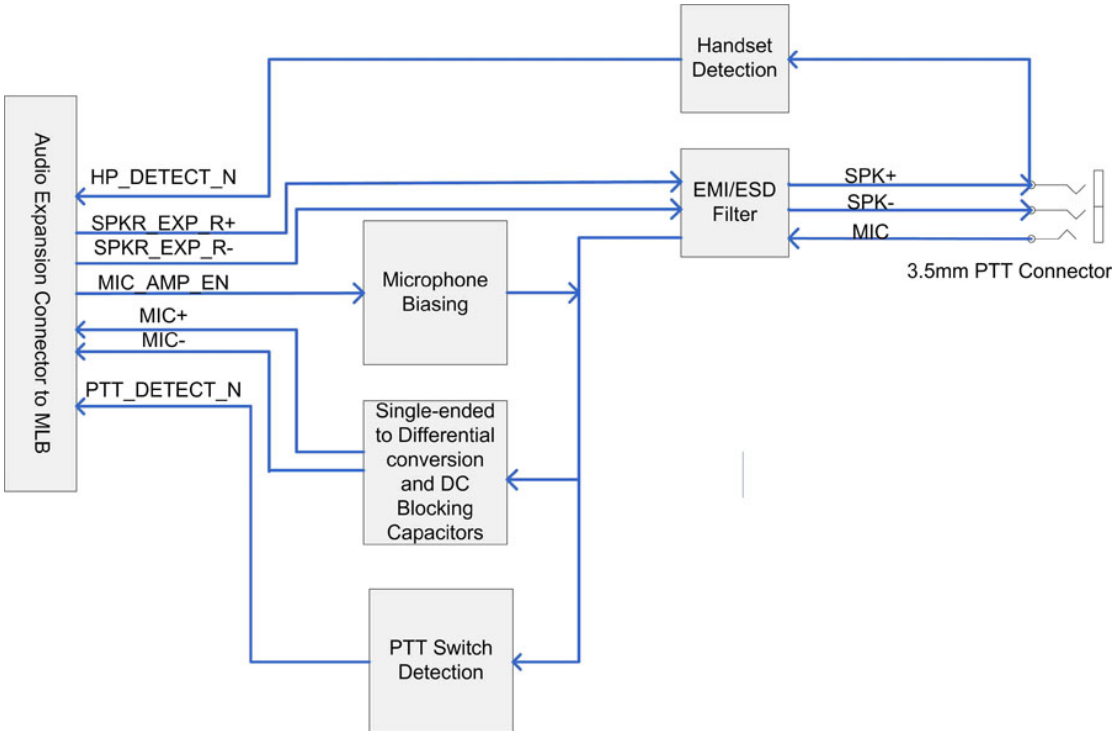
6.3.1.2 Push-to-Talk Handset

The push-to-talk handset reference design uses a 3.5 mm audio jack to accept a variety of push-to-talk handsets with a differentially driven loudspeaker. These handsets have a switch which enables/disables the microphone in the handset allowing audio to be played through the handset while the microphone is disabled. To speak, the user pushes the switch enabling the microphone. Audio output is driven from a class D amplifier on the main logic board which provides sufficient power to drive the loudspeaker.

Similar to the single-ended headset audio expansion board, filtering capacitors are placed at various locations throughout the audio paths to prevent GSM interference from coupling in. As well, the microphone biasing and signal conversion is performed identically to the single-ended headset reference design.

If a push-to-talk radio with a different activation scheme is used, a different detection circuit will need to be designed.

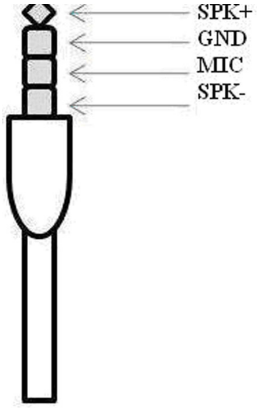
Figure 6.7 Block Diagram of the Push-to-Talk Handset Expansion Board



3.5 mm Audio Jack

The reference design for a push-to-talk audio expansion board has a 3.5 mm audio jack which provides connections to a single-ended microphone and a differentially driven loudspeaker. This is the only connector configuration supported in the reference design.

Figure 6.8 3.5 mm Push-to-Talk Handset Audio Jack Connection Diagram



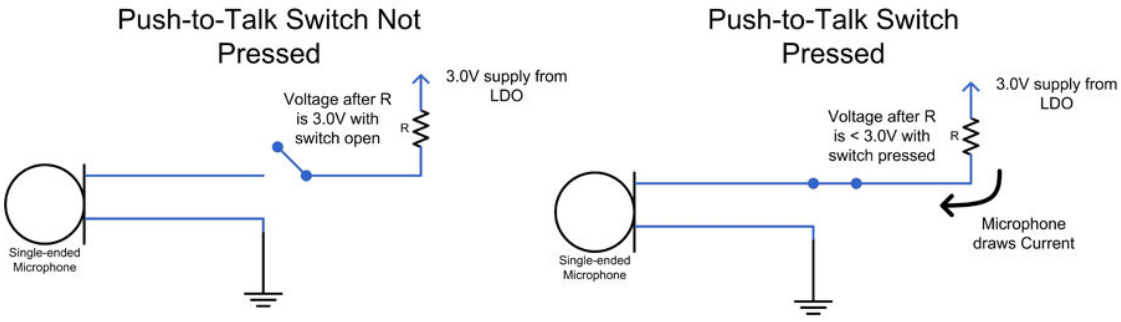
Push-to-Talk Switch Detection

Push-to-talk handsets provide a switch to the user which will enable the microphone when pushed, otherwise leaving the microphone deactivated. The PTT Switch Detection will generate an active low signal when the switch is pressed. In the reference design, this is accomplished with a comparator. When the switch is not pressed, the microphone is not connected and the DC voltage applied at the microphone matches the 3.0 V biasing voltage. When the switch is pressed, the microphone will draw some current, lowering the applied bias voltage below 3.0 V. A comparator with hysteresis is used to toggle the PTT_DETECT signal.

Table 6.3 Comparator Thresholds and Hysteresis for PTT_DETECT

Condition	Comparator Input Thresholds		
	Minimum (V)	Typical (V)	Maximum (V)
PTT_DETECT Falling Edge	2.77	2.81	2.86
PTT_DETECT Rising Edge	2.70	2.75	2.80

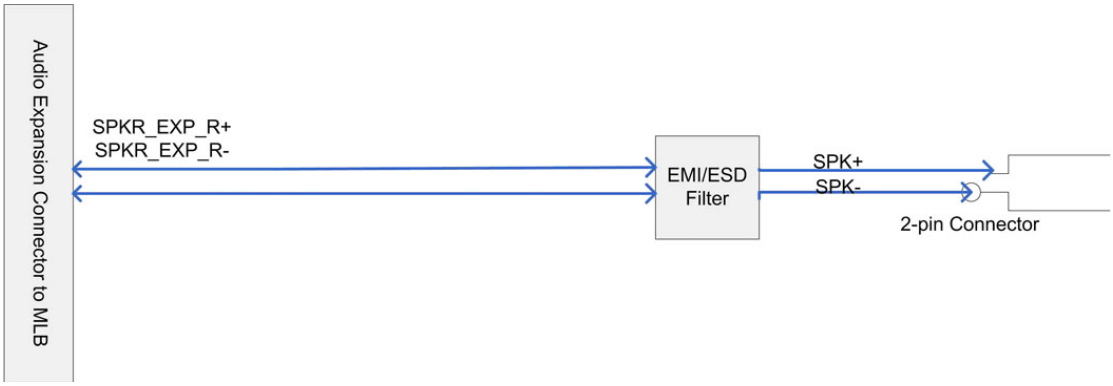
Figure 6.9 Push-to-Talk Handset Switch Operation



6.3.1.3 External Speaker

This reference design provides support for a loudspeaker attached through an audio expansion board. This design is identical to the push-to-talk configuration, without support for microphone input.

Figure 6.10 Block Diagram of the External Speaker Expansion Board



6.4 Expansion Ports

Omnii includes three designated “expansion ports” that expose standard GPIO, serial and USB interfaces for expansion modules. Each expansion port operates independently of the other two, allowing up to three expansion modules to be connected at the same time.

The three expansion ports are located as follows:

- Expansion Port 1: End-Cap Connector (see Section 6.4.2: “Expansion Port 1 (End-Cap Connector)”)
- Expansion Port 2: Pod Expansion Connector (see Section 6.4.3: “Expansion Port 2 (Pod Expansion Connector)”)
- Expansion Port 3: Multi-Function 100-Pin Connector (see Section 6.4.4: “Expansion Port 3 (100-pin Multi-Function Connector)”)

Note: If a GPS/WWAN module is installed in the 100-pin socket, the pins for expansion port 3 are passed through to a 22-pin connector on the back of the module. In this case, the pinout for expansion port 3 is the same as for expansion port 2.

Figure 6.11 Location of Expansion Ports on Omnii Main Logic Board

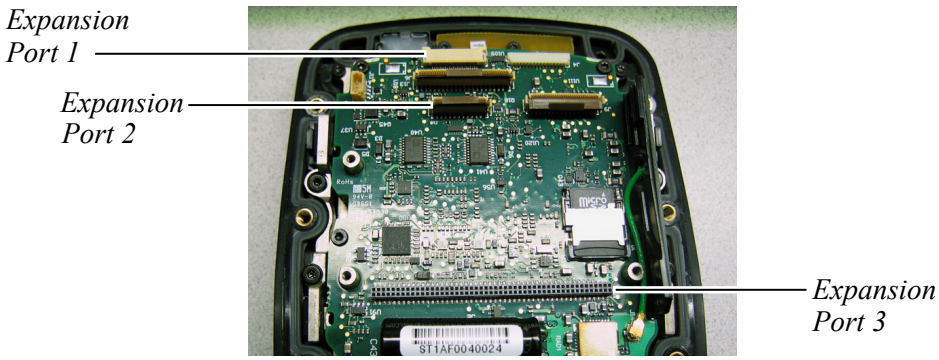
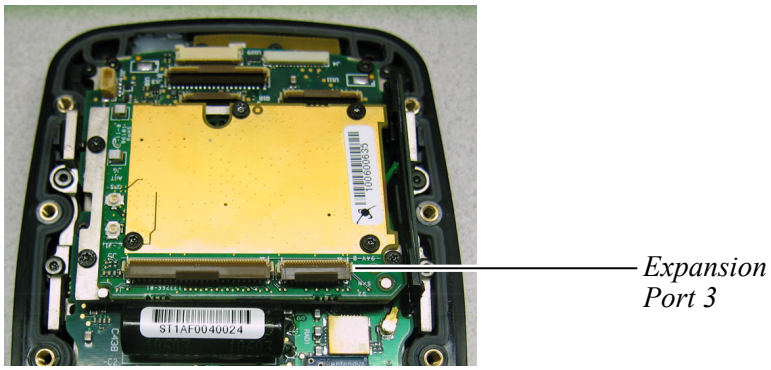


Figure 6.12 Location of Expansion Port 3 on GPS/WWAN Board



6.4.1 Expansion Port Power

This section describes the power supplies used by the expansion ports, and how to calculate the power available to an expansion module.

A 3.3 volt power supply is supplied for low power interface logic. This power supply is limited to 100 mA current.

VSYS is an unregulated power supply derived from the battery or from the regulated DC power supply generated by the AC/DC converter or adaptor powering a docking accessory (referred to as AC). VSYS is the rail from which the other power supplies are derived. The voltage of VSYS can vary between 3 volts and 4.2 volts depending on the state of the battery.

Expansion ports 1 and 2 each have switched VSYS (pins 3, 4 and 5) on the connector which should not be used to draw more than 1000 mA current. Expansion port 3 on the multi-function connector has one switched VSYS pin (pin 3) limited to 1000 mA, and there are also three unswitched pins on the same connector (pins 2, 4 and 6) which provide VSYS power up to a maximum current of 2000 mA. If a GPS module is installed in the multi-function connector, the 22-pin connector on the back of the module that keeps expansion port 3 exposed and provides the same power output as expansion ports 1 and 2 (pins 3, 4 and 5 switched VSYS limited to 1000 mA).

If only one expansion port is being used you can disregard the following current usage calculations and simply use the values specified by the expansion port pinout table as the maximum current available for the expansion module.

Although there is no restriction on the simultaneous use of the three expansion ports, the total amount of power being used by the system must be considered. This includes the Omnii core electronics, on-board peripherals and any standard options. The following table gives the typical current drawn from VSYS for each subsystem under given conditions, when VSYS is at a typical battery voltage of 3.7 V.

Table 6.4 Subsystem Power Usage

Subsystem	Condition	Current (mA)
Display (DISP)	Maximum brightness	100
	Minimum brightness	40
	Default brightness	72

Table 6.4 Subsystem Power Usage

Subsystem	Condition	Current (mA)
Processor & Memory (P+M)	Idle	205
	Maximum (Processor and memory under heavy load.)	550
System Controller (SYSCON)	Average	50
Keyboard (KYB)	Backlight maximum	285
	Backlight minimum (off)	10
	Backlight default	115
Power Microcontroller (PWR)	Average	11.5
System Flash (SYSMEM)	Writing	75
	Reading	75
USB OTG (USBOTG)	Transferring/Receiving at 480 Mbps	121
	Connected but idle	9
Dock USB Host (DUSB)	Transferring/Receiving at 12 Mbps	21
Expansion n USB (EXNUSB)	Transferring/Receiving at 12 Mbps	113
Speaker (SPK)	Maximum volume	84
Receiver (RXV)	Maximum volume	53
Recording (Mic) (REC)		14
Beeper (BEEP)	Resonant frequency, max volume	88
WiFi (WIFI)	Idle, associated to access point	187
	Pinging access point	220
Bluetooth (BT)	Idle, associated to device	39
	Transmitting/Receiving	64
	Idle, not associated	11.5
Vibrator Motor (VIB)	Vibrating	150
Micro SD (USD)	Writing	165
	Reading	121
SE1223 (SE1223)	Idle	0
	Scanning	136
SE1224 (SE1224)	Idle	0
	Scanning	142
SE1524 (SE1524)	Idle	0
	Scanning	201
SE955 *(not on* Omnii XT10*)*	Idle	0
	Scanning	98

Table 6.4 Subsystem Power Usage

Subsystem	Condition	Current (mA)
5080	Idle	41
	Scanning	447
EV15	Idle	0
	Scanning	101
Camera (CAM)	Idle	11
	Taking a picture with flash	223
	Taking a picture without flash	145
	Preview mode	523
GPS	Acquiring satellites	88

The core electronics of the system will always be active while Omnii is powered on, and therefore must always be included in power calculations. The core components include: Processor and Memory, Display, Keyboard, System Controller, Power Microcontroller and System Flash (Reading).

The amount of current which can be drawn from the battery or AC is limited to 3 A. Therefore, the current available to an expansion port can be calculated by subtracting the power usage for the core electronics and each optional subsystem from 3000 mA.

For example, if the processor and memory are both at idle, the display and keyboard backlight are both at default brightness, the WiFi radio is transmitting and the expansion port is using USB, the formula for determining the available current is:

3000 – (P+M(Idle) + DISP(Def) + KYB(Def) + SYSCON + PWR + SYSMEM(Read) + WiFi(Tx) + EXNUSB)

= 3000 – (205 + 72 + 115 + 50 + 11.5 + 75 + 220 + 113)

= 3000 – 861.5

= 2138.5 mA

When Omnii is being powered by the battery, higher current draw results in shorter battery life. The standard battery has a 5000 mAh capacity, so a constant load of 2000 mA will drain the battery in approximately 2.5 hours. Expansion devices with high power requirements should always be placed in the lowest possible power state, or the power supply to the expansion device should be disabled, when the device is not in use.

High current loads should be soft-started to avoid an unexpected system shutdown due to instantaneous appliation of a high current load to the battery. If the expansion device has a current draw profile that includes short duration high current peaks, add capacitance to the expansion board to smooth the peaks and average the current draw over time. The internal resistance of Li-ion batteries increases as the temperature decreases, so be aware that operating the system at low temperatures (below 0 °C) will make the battery more susceptible to voltage drops due to transient loads.

Table 6.5 Expansion Port 1 & 2 Power Pins

Pin Name (*n*=Expansion Port #)	Description	Expansion Port 1 & 2 Pin
EXP*n*_1Wire	Connection for 1-Wire EEPROM	19
EXP*n*_3V3	3.3 V power - 100 mA current limited	1
EXP*n*_VSYS	3.0 V to 4.2 V power - raw unregulated battery voltage 1000 mA current limited	3,4,5
GND	Ground	2,6,9,18,20,21,22

Table 6.6 Expansion Port 3 Power Pins

Pin Name	Description	Expansion Port 3 Pin (100-pin MLB)	Expansion Port 3 Pin (22-pin GPS module)
EXP3_1Wire	Connection for 1-Wire EEPROM	29	19
EXP3_3V3	3.3 V power - 100 mA current limited	1	1
EXP3_VSYS	3.0 V to 4.2 V power - raw unregulated battery voltage 1000 mA current limited	3	3,4,5
GND	Ground	5,8,10,11,12,22,36,49,53, 54,57,61,65,78,94,99,100	2,6,9,18,20,21,22

6.4.2 Expansion Port 1 (End-Cap Connector)

Theory of Operation

The end-cap connector is a standard expansion port that provides a means of connecting a GPIO, USB, or serial device to the system, mounted in the end-cap module of the standard back cover. This connection is identified to the system as Expansion Port 1.

Note: The end-cap expansion port (Expansion Port 1) does not function on Omnii XT10 computers with main logic boards prior to revision "-500".

End-Cap Connector Details

Manufacturer:	AVX/KYOCERA
Manufacturer Part Number:	08-6222-022-101-829+
Mating Connector:	Flex – 0.5 mm pitch
Number of Pins:	22
Current Rating:	0.4 A per pin

The AVX/Kyocera end-cap connector is a low-insertion-force, right-angle mounted, bottom-contact connector.

Pinout

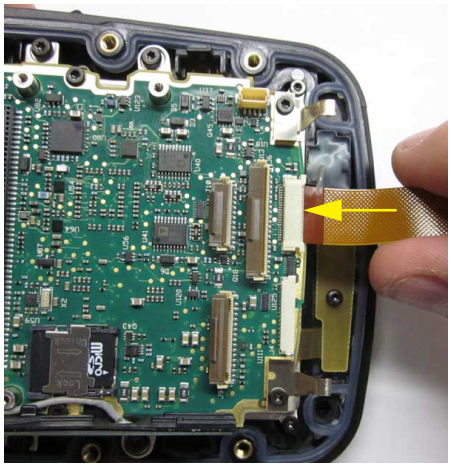
The 22-pin end-cap connector provides access to expansion port 1, with the following pinout:

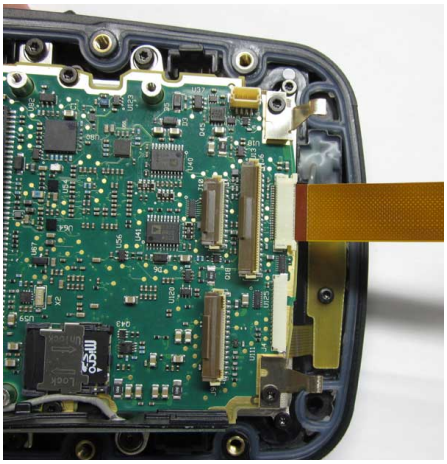
Table 6.7 Pinout of 22-pin End-Cap Connector

Pin	Signal	Function	I/O Type	Voltage (V)
1	EXP1_3V3	Power. 100 mA current limited	Power	3.3
2,6,9,18, 20,21,22	GND	Ground	Ground	
3,4,5	EXP1_VSYS	Power. Raw unregulated battery voltage 1000 mA current limited	Power	3.0-4.2
7	EXP1_USB+	USB Host D+	I/O	
8	EXP1_USB-	USB Host D-	I/O	
10	EXP1_RTS_GPIO3	Serial RTS / GPIO pin 3	I/O	3.3
11	EXP1_CTS_GPIO2	Serial CTS / GPIO pin 2	I/O	3.3
12	EXP1_RXD_GPIO1	Serial RXD / GPIO pin 1	I/O	3.3
13	EXP1_TXD_GPIO0	Serial TXD / GPIO pin 0	I/O	3.3
14	EXP1_CS_N_GPIO7	SPI chip select / GPIO pin 7 (Modem DTR)	I/O	3.3
15	EXP1_SCLK_GPIO6	SPI SCLK / GPIO pin 6 (Modem DSR)	I/O	3.3
16	EXP1_MISO_GPIO5	SPI MISO / GPIO pin 5 (Modem DCD)	I/O	3.3
17	EXP1_MOSI_GPIO4	SPI MOSI / GPIO pin 4 (Modem RI)	I/O	3.3
19	EXP1_1WIRE	Connection for 1-Wire EEPROM	I/O	3.3

Connecting to Expansion Port 1

Expansion port 1 is located on the top edge of the main logic board. There is no latch on this connector; the cable is held by contact force alone. Push the stiffened end of the flex cable, with the exposed contacts facing downwards towards the main logic board, straight into the socket until it is firmly seated.





6.4.3 Expansion Port 2 (Pod Expansion Connector)

Theory of Operation

The pod expansion connector is a standard expansion port that provides a means of connecting a GPIO, USB, or serial device to the system, mounted in a pod expansion module attached to the standard back cover. This connection is identified to the system as Expansion Port 2.

Pod Expansion Connector Details

Manufacturer:	HIROSE
Manufacturer Part Number:	FH12-22S-0.5SV(55)
Mating Connector:	Flex – 0.5 mm pitch
Number of Pins:	22
Current Rating:	0.4 A per pin

The Hirose pod connector is a zero-insertion-force, vertical-mounted connector.

Pinout

The 22-pin pod expansion connector provides access to expansion port 2, with the following pinout:

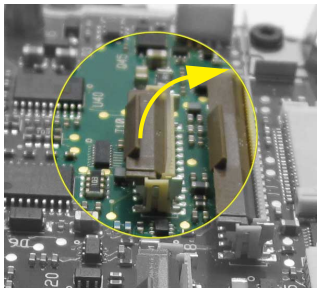
Table 6.8 Pinout of 22-pin Pod Expansion Connector

Pin	Signal	Function	I/O Type	Voltage (V)
1	EXP2_3V3	Power. 100 mA current limited	Power	
2,6,9,18, 20,21,22	GND	Ground	Ground	
3,4,5	EXP2_VSYS	Power. Raw unregulated battery voltage 1000 mA current limited	Power	3.0-4.2
7	EXP2_USB+	USB Host D+	I/O	
8	EXP2_USB-	USB Host D-	I/O	
10	EXP2_RTS_GPIO3	Serial RTS / GPIO pin 3	I/O	3.3
11	EXP2_CTS_GPIO2	Serial CTS / GPIO pin 2	I/O	3.3
12	EXP2_RXD_GPIO1	Serial RXD / GPIO pin 1	I/O	3.3
13	EXP2_TXD_GPIO0	Serial TXD / GPIO pin 0	I/O	3.3
14	EXP2_CS_N_GPIO7	SPI chip select / GPIO pin 7 (Modem DTR)	I/O	3.3
15	EXP2_SCLK_GPIO6	SPI SCLK / GPIO pin 6 (Modem DSR)	I/O	3.3
16	EXP2_MISO_GPIO5	SPI MISO / GPIO pin 5 (Modem DCD)	I/O	3.3
17	EXP2_MOSI_GPIO4	SPI MOSI / GPIO pin 4 (Modem RI)	I/O	3.3
19	EXP2_1WIRE	Connection for 1-Wire EEPROM	I/O	3.3

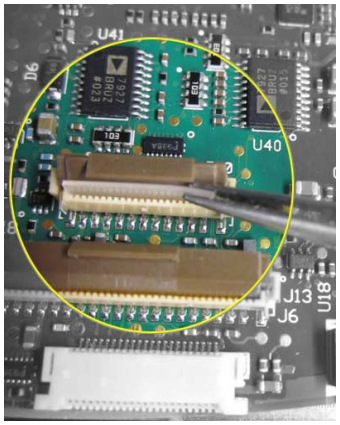
Connecting to Expansion Port 2

Expansion port 2 may have a piece of dummy "stiffener" plastic already installed in the socket. This prevents the latch from moving and breaking under sudden impact, such as may occur if the terminal is dropped.

Open the latch, and if the stiffener plastic is present, remove it from the socket.



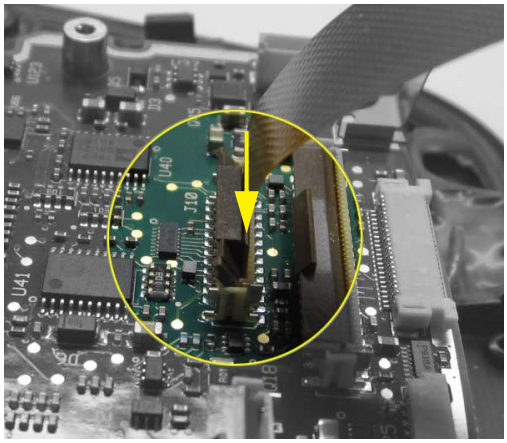
Lift Latch



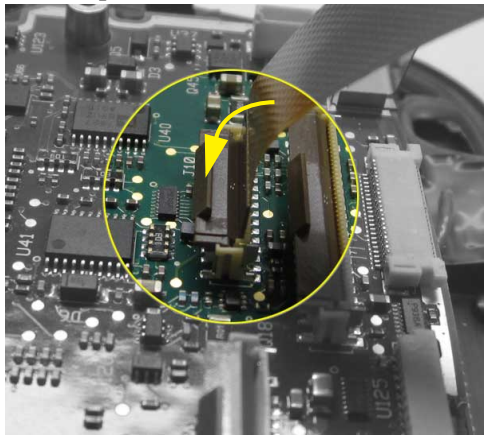
Remove Stiffener

If the terminal is reassembled without a cable in this socket, insert the stiffener plastic back in the socket, and close the latch to secure it in place.

Insert the stiffened end of the flex cable into the socket, with the exposed contacts facing the top of the unit. Close the latch on the socket to secure the cable in place.



Insert Cable



Close Latch

6.4.4 Expansion Port 3 (100-pin Multi-Function Connector)

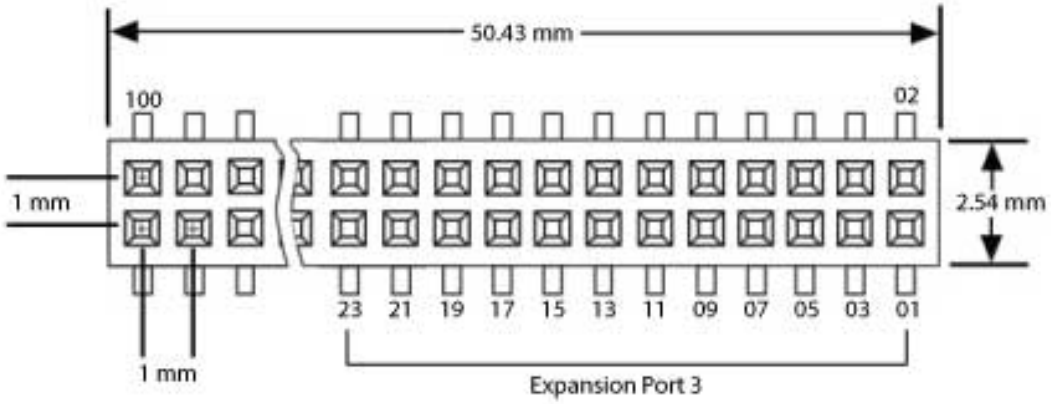
Theory of Operation

The 100-pin multi-function connector on the main logic board includes pins for a standard expansion port, providing GPIO, serial and USB connections. This collection of pins is identified to the system as Expansion Port 3. For details on the other interfaces exposed by this connector, see Section 6.5 on page 102.

Connector Details

Manufacturer:	SAMTEC
Manufacturer Part Number:	CLM-150-020-F-D-K
Mating Connector:	MW-50-03-G-D-150-065
Number of Pins:	100
Current Rating:	1.5A per pin

Figure 6.13 Expansion Port 3 Pins on 100-Pin Connector (Top View)



If a GPS module is installed on the 100-pin connector, the pins for expansion port 3 are passed through to a 22-pin connector on the back of the module (see Section 6.2: “Connector Locations”). This connector is identical in form and pinout to expansion port 2 as described in Section 6.4.3: “Expansion Port 2 (Pod Expansion Connector)” – use those specifications instead when using this connector.

The following pinout table only shows the pins for the expansion port 3 interface of the 100-pin connector. For descriptions of the remaining pins and interfaces, see Section 6.5 on page 102.

Pinout

Table 6.9 Pinout of Expansion Port 3 (100-pin Connector)

Pin	Signal	Function	I/O Type	Voltage (V)
1	EXP3_3V3	Power. 100 mA current limited, switched	Power	3.3
3	EXP3_VSYS	Power. Raw unregulated battery voltage 1000 mA current limited, switched	Power	3.0–4.2
5	GND	Ground	Ground	
7	EXP3_USB+	USB Host D+	I/O	
9	EXP3_USB-	USB Host D–	I/O	
11	GND	Ground	Ground	
13	EXP3_RTS_GPIO3	Serial RTS / GPIO pin 3	I/O	3.3
15	EXP3_CTS_GPIO2	Serial CTS / GPIO pin 2	I/O	3.3
17	EXP3_RXD_GPIO1	Serial RXD / GPIO pin 1	I/O	3.3
19	EXP3_TXD_GPIO0	Serial TXD / GPIO pin 0	I/O	3.3
21	EXP3_CS_N_GPIO7	SPI chip select / GPIO pin 7 (Modem DTR)	I/O	3.3
23	EXP3_SCLK_GPIO6	SPI SCLK / GPIO pin 6 (Modem DSR)	I/O	3.3
25	EXP3_MISO_GPIO5	SPI MISO / GPIO pin 5 (Modem DCD)	I/O	3.3
27	EXP3_MOSI_GPIO4	SPI MOSI / GPIO pin 4 (Modem RI)	I/O	3.3
29	EXP3_1WIRE	Connection for 1-Wire EEPROM	I/O	3.3

Connecting to Expansion Port 3

Note: If a GPS/WWAN module is installed, expansion port 3 is routed to a 22-pin connector on the back of the module. This connector is identical to expansion port 2; follow the instructions for that port instead.

Expansion Port 3 is incorporated into the 100-pin multi-function socket J2 on the main logic board. See Section 6.4.4 on page 98 for more information on the 100-pin connector.

To connect to the 100-pin socket, align the pins of the connector to the socket and press down firmly and evenly across the entire connector.

6.4.5 Expansion Port Standard Interfaces

6.4.5.1 Serial (UART) Interface

Serial expansion modules can be connected as standard UART devices through the following signal pins on the expansion ports:

Table 6.10 Expansion Port Serial Pins

Pin Name (*n*=Expansion Port #)	Description	Expansion Port 1 & 2 Pin	Expansion Port 3 Pin
EXP*n*_TXD_GPIO0	Serial Transmit Data (output)	13	19
EXP*n*_RXD_GPIO1	Serial Receive Data (input)	12	17
EXP*n*_CTS_GPIO2	Serial Clear-to-Send (input)	11	15
EXP*n*_RTS_GPIO3	Serial Request-to-Send (output)	10	13

Note: All pin directions are given with reference to the Omnii connector. e.g., the TXD line is for data transmitted from Omnii to the expansion device.

These pins can be configured for two different functions: GPIO (General Purpose Input/Output), or serial communication. To configure these pins for serial communication, configure the **PinFunctions** registry entry for the expansion device accordingly (see Section 4.4.1: “Registry Settings for Expansion Devices”).

To identify an expansion device as a serial device and load the UART driver, configure its **LoadFlags** registry entry to *0* (see Section 4.4.1: “Registry Settings for Expansion Devices”).

Additionally, the serial driver can be instructed to provide serial modem functions (RI, DCD, DSR and DTR) on pins 4 through 7 of the GPIO interface.

Table 6.11 Expansion Port Modem Pins (GPIO)

Pin Name (*n*=Expansion Port #)	Description	Expansion Port 1 & 2 Pin	Expansion Port 3 Pin
EXP*n*_MOSI_GPIO4	GPIO pin 4 / Modem RI (input)	17	27
EXP*n*_MISO_GPIO5	GPIO pin 5 / Modem DCD (input)	16	25
EXP*n*_SCLK_GPIO6	GPIO pin 6 / Modem DSR (input)	15	23
EXP*n*_CS_N_GPIO7	GPIO pin 7 / Modem DTR (output)	14	21

To enable these signal pins, make sure that they are configured for GPIO operation (see Section 6.4.5.3 on page 101), and set the following registry keys for the serial driver:

```
[HKLM\Drivers\Expansion\iglooUart_Exp1]
     “GpioPinRI”=dword:1          ; Enable/Disable GPIO RI  (1=Enable)
     “GpioPinDCD”=dword:1         ; Enable/Disable GPIO DCD
     “GpioPinDSR”=dword:1         ; Enable/Disable GPIO DSR
     “GpioPinDTR”=dword:1         ; Enable/Disable GPIO DTR
```

Serial communication is supported up to 115200 kbps.

6.4.5.2 USB Interface

The expansion ports provide access to a USB (Universal Serial Bus) host connection. Communication with USB client devices is supported at low speed (1.5 Mbps) and full speed (12 Mbps).

USB devices are identified to the system by setting the **LoadFlags** registry entry for the device to *4* (see Section 4.4.1: "Registry Settings for Expansion Devices"). When a USB device is detected by the system, the USB driver is loaded on Omnii.

The following pins support USB communication on the expansion ports:

Table 6.12 Expansion Port USB Pins

Pin Name (n = Expansion Port #)	Description	Expansion Port 1 & 2 Pin	Expansion Port 3 Pin
EXP*n*_USB+	USB Host D+	7	7
EXP*n*_USB+	USB Host D–	8	9

6.4.5.3 GPIO (General Purpose Input/Output) Interface

Eight data pins on each expansion interface can be used as GPIO (General Purpose Input/Output) communication lines. These same pins are also used for serial and SPI communications; the function of the pins is assigned by setting the **PinFunctions** registry entry for the connected device (see Section 4.4.1: "Registry Settings for Expansion Devices").

Table 6.13 Expansion Port GPIO Pins

Pin Name (*n*=Expansion Port #)	Description	Expansion Port 1 & 2 Pin	Expansion Port 3 Pin
EXP*n*_TXD_GPIO0	GPIO pin 0	13	19
EXP*n*_RXD_GPIO1	GPIO pin 1	12	17
EXP*n*_CTS_GPIO2	GPIO pin 2	11	15
EXP*n*_RTS_GPIO3	GPIO pin 3	10	13
EXPn_MOSI_GPIO4	GPIO pin 4	17	27
EXPn_MISO_GPIO5	GPIO pin 5	16	25
EXPn_SCLK_GPIO6	GPIO pin 6	15	23
EXPn_CS_N_GPIO7	GPIO pin 7	14	21

6.4.5.4 SPI (Serial Peripheral Interface)

The SPI (Serial Peripheral Interface) function of the expansion ports is not available on Omnii XT10, but will be included in later models. Information is included here for future reference.

Table 6.14 Expansion Port SPI Pins

Pin Name (*n*=Expansion Port #)	Description	Expansion Port 1 & 2 Pin	Expansion Port 3 Pin
EXPn_MOSI_GPIO4	Master output/Slave input	17	27
EXPn_MISO_GPIO5	Master input/Slave output	16	25
EXPn_SCLK_GPIO6	Serial clock	15	23
EXPn_CS_N_GPIO7	Chip select	14	21

These pins can be configured for two different functions: GPIO or SPI. To configure these pins for SPI serial communication, configure the **PinFunctions** registry entry for the expansion device accordingly (see Section 4.4.1: “Registry Settings for Expansion Devices”)

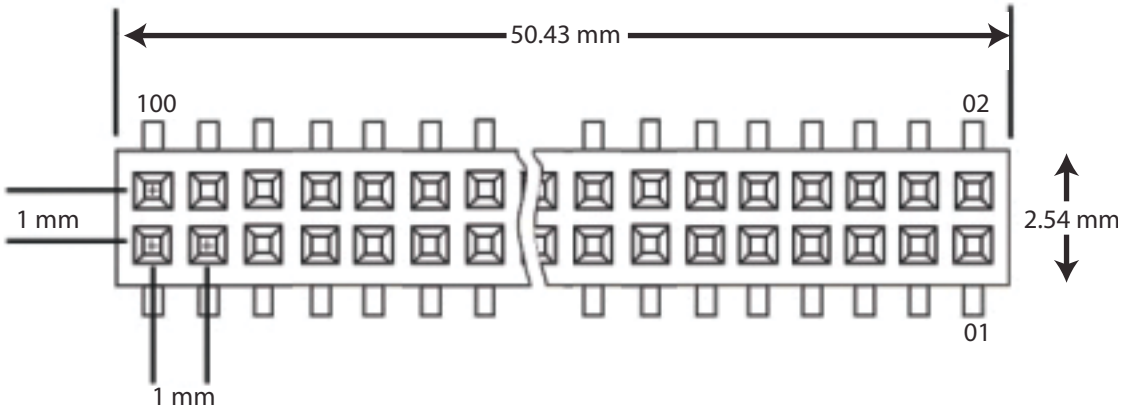
6.5 100-Pin Multi-Function Connector

The 100-pin multi-function connector on the Omnii XT10 main logic board exposes an interface for GPS modules. Future Omnii models will include interfaces for WWAN radios and other expansion devices on this connector as well. In addition, standard expansion port 3 occupies some of the pins on this connector – see Section 6.4.4 on page 98 for details on using this expansion port.

Connector Details

Manufacturer:	SAMTEC
Manufacturer Part Number:	CLM-150-020-F-D-K
Mating Connector:	MW-50-03-G-D-150-065
Number of Pins:	100
Current Rating:	1.5 A per pin

Figure 6.14 Diagram of SAMTEC 100-Pin Connector (Top View)



Pinout

Table 6.15 Pinout of Multi-Function 100-Pin Connector

Pin	Signal	Function	I/O Type	Voltage (V)
1	EXP3_3V3	Power. 100 mA current limited, switched	Power	3.3
2, 4, 6	VSYS	Power. Raw unregulated battery voltage 2000 mA maximum, unswitched	Power	3.0–4.2
3	EXP3_VSYS	Power – raw unregulated battery voltage 1000 mA current limited, switched	Power	3.0–4.2
5, 8, 10, 11, 12, 22, 36, 49, 53, 54, 57, 61, 65, 78, 94, 99, 100	GND	Ground	Ground	
7	EXP3_USB+	USB Host D+	I/O	
9	EXP3_USB-	USB Host D–	I/O	
13	EXP3_RTS_GPIO3	Serial RTS / GPIO pin 3	I/O	3.3
14	BACKUP_ACTIVE	Backup active signal, used to shed all loads when running from backup power	Output	3.3
15	EXP3_CTS_GPIO2	Serial CTS / GPIO pin 2	I/O	3.3
16	5V0	5 V power supply, 100 mA maximum	Power	5.0
17	EXP3_RXD_GPIO1	Serial RXD / GPIO pin 1	I/O	3.3
18	3V3	3.3 V power supply, 100 mA maximum	Power	3.3
19	EXP3_TXD_GPIO0	Serial TXD / GPIO pin 0	I/O	3.3
20	1V8	1.8 V power supply, 100 mA maximum	Power	1.8
21	EXP3_CS_N_GPIO7	SPI chip select / GPIO pin 7 (Modem DTR)	I/O	3.3
23	EXP3_SCLK_GPIO6	SPI SCLK / GPIO pin 6 (Modem DSR)	I/O	3.3
24	GPS_PWR_EN	GPS power enable	Output	1.8
25	EXP3_MISO_GPIO5	SPI MISO / GPIO pin 5 (Modem DCD)	I/O	3.3
26	GPS_RX	GPS serial port receive (input to GPS)	Output	1.8
27	EXP3_MOSI_GPIO4	SPI MOSI / GPIO pin 4 (Modem RI)	I/O	3.3
28	GPS_TX	GPS serial port transmit (input from GPS)	Input	1.8
29	EXP3_1Wire	Connection for 1-Wire EEPROM	I/O	3.3
30	GPS_BOOT	GPS boot mode control	Output	1.8

Table 6.15 Pinout of Multi-Function 100-Pin Connector

Pin	Signal	Function	I/O Type	Voltage (V)
31, 33, 35, 37-48, 50, 51, 52, 55, 56, 58-60, 62-64, 66-77, 79-93, 95-98	RESERVED	Reserved for future use		
32	GPS_CTS	GPS serial port CTS / GPIO	I/O	1.8
34	GPS_RTS	GPS serial port RTS / GPIO	I/O	1.8

DOCKING STATIONS 7

7.1 Overview

This chapter describes the connectors and features of the Omnii docking stations, to which expansion modules may be attached.

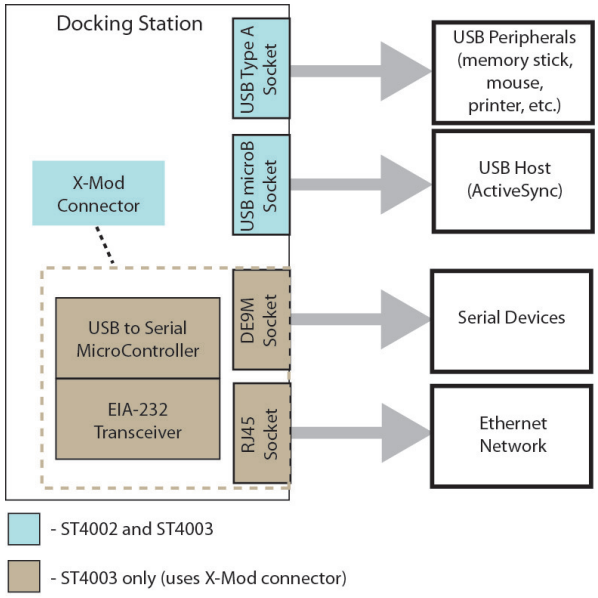
7.2 Desktop Docking Stations

There are two single-unit desktop docking stations available for Omnii: models ST4002 and ST4003.

Model ST4002 provides power and charging functions, as well as standard USB Type-A and microB connectors. An expansion module ("X-Mod") connector provides support for external devices.

Model ST4003 provides the same power, charging and USB capabilities as ST4002, and also comes with a built-in expansion PCB assembly mounted in the X-Mod connector to provide standard RS-232 serial and Ethernet ports. (This serial/Ethernet module is available as a kit, model ST4100).

Figure 7.1 Block Diagram of Desktop Docking Station Connectors



7.2.1 Docking Station USB Connectors

The ST4002 and ST4003 docking stations provide two standard USB interfaces:

- one USB 2.0 microB interface that supports USB client connections up to 480 Mbps.
- one USB 2.0 Type A host interface that supports connections to USB clients at speeds up to 12 Mbps.

USB 2.0 microB Interface

The USB microB connector on the docking station allows Omnii to connect to a USB client device. It supports low speed (1.5 Mbps), full speed (12 Mbps) and high speed (480 Mbps) communications.

Figure 7.2 USB MicroB Connector

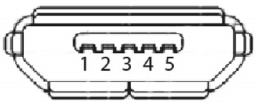


Table 7.1 Pinout Of The USB MicroB Connector

Pin	Name	Description	Direction
1	VBUS	DC current from external host	Input from connected external device.
2	USB_D-	USB Client D-	Bidirectional (half-duplex).
3	USB_D+	USB Client D+	
4	ID	ID connected plug (Only microB plug is supported in the desktop docking stations.)	
5	Ground		

USB 2.0 Type A Interface

The USB Type A connector on the desktop docking station is a standard USB 2.0 compliant interface that allows Omnii to connect to USB client devices. It supports low speed (1.5 Mbps) and full speed (12 Mbps) communications.

In addition, the Type A port provides VBUS power to a maximum of 500 mA. If a power adaptor is connected to the docking station, it provides the VBUS power; otherwise, Omnii provides the power from the main battery. If there is no Omnii docked in the docking station, no power is provided to the VBUS power line, even if a power adaptor is connected.

Figure 7.3 USB Host Type A Connector

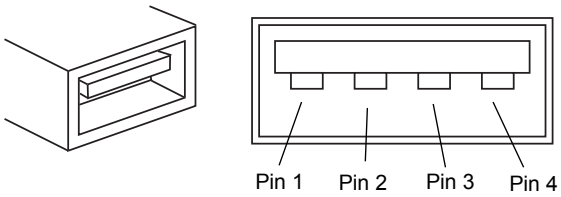


Table 7.2 Pinout Of The USB Host Connector

Pin	Name	Description	Direction
1	VBUS	DC supply from Omnii. Current 500 mA maximum. Voltage minimum 4.35 V. Voltage maximum 5.25 V.	Output to connected external device.
2	USB_H4_D-	USB Host port 4 D-	Bidirectional (half-duplex).
3	USB_H4_D+	USB Host port 4 D+	
4	Ground		

7.2.2 Docking Station RS-232 Connector

The ST4003 desktop docking station provides a male DE9 port for connecting serial devices. The port is capable of communicating at speeds from 2400 kbps up to 115200 kbps.

Figure 7.4 RS-232 DE9M Connector

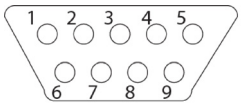


Table 7.3 Pinout Of The RS-232 DE9M Connector

Pin	Name	Description
1	DCD	Data Carrier Detect
2	RXD	Received Data
3	TXD	Transmitted Data
4	DTR	Data Terminal Ready
5	GND	Ground
6	DSR	Data Set Ready
7	RTS	Request To Send
8	CTS	Clear To Send
9	RI	Ring Indicator

7.2.3 Docking Station Ethernet RJ45 Connector

The ST4003 desktop docking station provides a female RJ45 port for connecting the docked device to an Ethernet network. This port supports speeds up to 10 Mbps.

Figure 7.5 RJ45 Connector

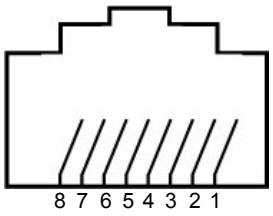


Table 7.4 Pinout Of The RJ45 Connector

Pin	Name	Description
1	TX+	Transmit Data +
2	TX-	Transmit Data -
3	RX+	Receive Data +
4, 5, 7	NC	No Connection
6	RX-	Receive Data -
8	DGND	Ground
Shield	SGND	Shell Ground

7.2.4 Docking Station Expansion Module (X-Mod) Connector

Theory of Operation

The ST4002 and ST4003 desktop docking stations have an expansion module (X-Mod) connector concealed under a plate in the bottom of the unit. In the ST4003 model, this connector is occupied by an expansion module that provides the RS-232 and Ethernet ports. This module can be removed and replaced with a custom expansion module if desired, but typically it is more sensible to simply add the custom module to the ST4002 model.

A custom-built expansion module installed in the X-Mod connector should have an 845R resistor connected to pin 20 on the header. This identifies the expansion module as a developer-created device to the hand-held computer.

X-Mod Connector Details

For details on the X-Mod connector socket, and the specifications for designing a matching header, see the files *114-24004-2011 2x10 connector on Xmod PCA (header).pdf* and *124-21022-201 2x10 connector on Main PCA (socket).pdf*, included with the HDK.

Pinout

The 20-pin X-Mod connector has the following pinout:

Table 7.5 Pinout of 20-pin X-Mod Connector

Pin	Signal	Function	I/O Type	Voltage (V ±5%)
1, 4, 5, 8, 9, 11, 12, 15, 16, 19	Digital Ground	Ground	Ground	
2, 3	VIN_12V	Power. 500 mA maximum	Power	12.0
6, 7	VDD_5V	Power. Turned on when the hand-held computer is connected. 500 mA maximum	Power	5.0
10	VDDChargeMicro	Power. Reference voltage. 10 mA maximum	Power	3.3
13	USB_H1_D-	USB Host port 1 D-	I/O	
14	USB_H1_D+	USB Host port 1 D+	I/O	
17	USB_H2_D-	USB Host port 2 D-	I/O	
18	USB_H2_D+	USB Host port 2 D+	I/O	
20	XModDetect	Connect to ground through 845R resistor on expansion module PCB assembly. Identifies module as custom-built to hand-held computer.		

Figure 7.6 X-Mod Connector Pin Diagram

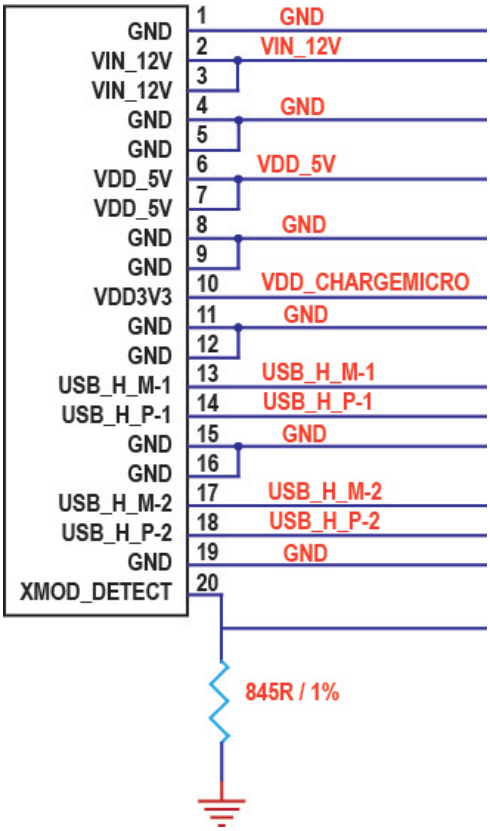
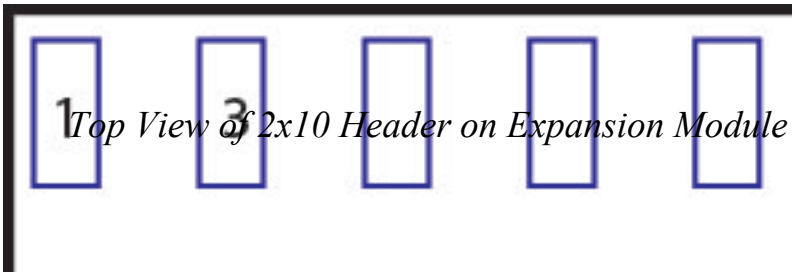


Figure 7.7 X-Mod Connector Pin Sequence



Designing X-Mod Expansion Modules

The 3D CAD model of the ST400x docking stations shows the dimensions and mounting point locations of the X-Mod cavity in the bottom of the unit. Also included in the HDK is the 3D CAD model of the RS-232 / Ethernet module from the ST4003 docking station, showing the size, shape and screw locations of an X-Mod PCB assembly.

Table 7.6 Associated Files for Docking Station X-Mod Connector

Description	Filename
Datasheet for 20-pin X-Mod connector header (on X-Mod PCB)	114-24004-2011 2x10 connector on Xmod PCA (header).dwg
Datasheet for 20-pin X-Mod connector socket (on docking station)	124-21022-201 2x10 connector on Main PCA (socket).dwg

Table 7.6 Associated Files for Docking Station X-Mod Connector

Description	Filename
2D drawing of the RS-232 / Ethernet expansion module for the ST4003 docking station	Xmod Board Outline.pdf
Zip file containing 3D models of the docking station bottom housing, X-Mod covers and the ST4003 RS-232 / Ethernet expansion module.	ProE model of parts.zip
2D drawings of the bottom housing of the ST400x docking stations, showing the X-Mod cavity.	9.0. Bottom_Housing(1).dwg
	9.0. Bottom_Housing(1).pdf
2D drawings of the X-Mod blank cover (with no openings) for the ST4002 docking station.	10.0 X_mod_cover_wIO(01).dwg
	10.0 X_mod_cover_wIO(01).pdf
2D drawings of the X-Mod RS-232 / Ethernet cover for the ST4003 docking station	10.0_2 X_mod_cover_nIO(01).dwg
	10.0_2 X_mod_cover_nIO(01).pdf

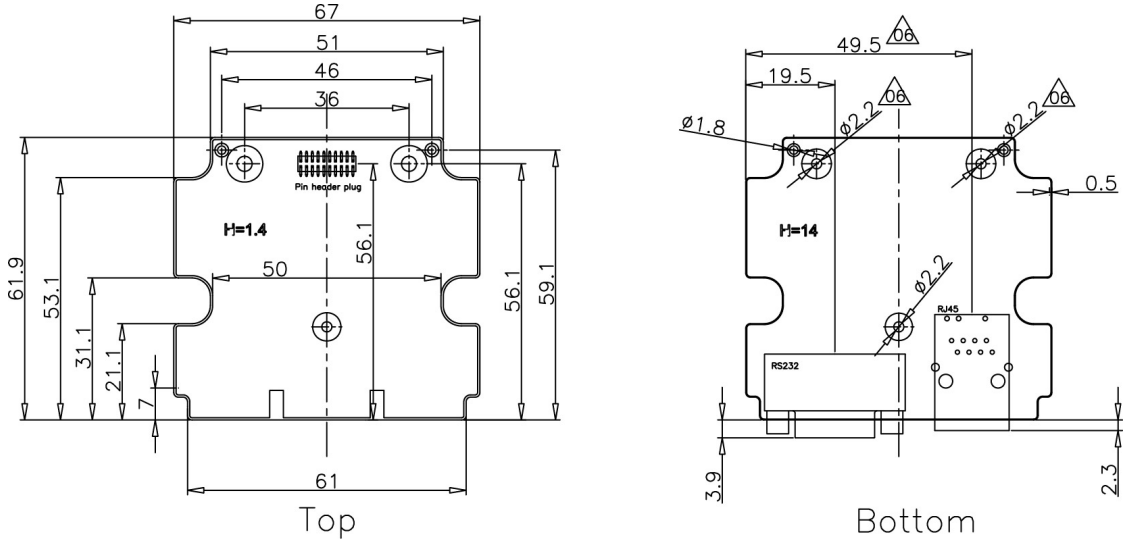
Table 7.7 Required Fasteners

How Used	Quantity	Part Number	Screw Size	Torque (in-lb)	Bit
Expansion Module to Docking Station	3	9009780	M2 x 4	2.5	Phillips
X-Mod Cover to Docking Station	4	n/a	M3 x 8	2.5	Phillips

X-Mod expansion modules are attached to the docking station with three M2 x 4 mm screws. The expansion module must be designed with holes for these three screws.

The cover that fits over the X-Mod cavity and protects the expansion module is attached to the docking station with four M3 x 8 mm screws. The expansion module must be designed with cutouts to fit around the mounting standoffs for these screws.

Figure 7.8 Sample X-Mod PCB Assembly with Connector and Mounting Points



For a custom X-Mod expansion module that has one or more external connectors, a custom cover must also be designed, with openings in the back to accommodate these connectors. The external connectors should be flush with the back cover when installed, and the cover openings should fit as closely around the connector as possible to prevent dust ingress. Reference files are included with the HDK which show 2D and 3D views of both the standard blank X-Mod cover for the ST4002 docking station, and the cover for the RS-232 / Ethernet expansion module of the ST4003 docking station.

Installing X-Mod Expansion Modules

Installing an expansion module in the X-Mod socket is a simple procedure.

1. Remove all power from the docking station.
2. Use a Phillips screwdriver to remove the four M3 screws holding the cover to the bottom of the docking station.
3. If an existing module is installed, use a Phillips screwdriver to remove the three M2 screws securing it to the docking station, then carefully lift the expansion PCB to disconnect it from the X-Mod socket.
4. Align the 20-pin header of the new expansion module with the X-Mod socket on the docking station, and press evenly on the PCB to snap it into place.
5. Install three M2 x 4 mm Phillips screws to secure the expansion module to the docking station. Torque the screws to 2.5 in-lb (0.28 N-m).
6. Fit the cover into place over the X-Mod cavity, making sure any external connectors fit properly through the openings in the back of the cover.
7. Install four M3 x 8 mm Phillips screws to secure the expansion module to the docking station. Torque the screws to 2.5 in-lb (0.28 N-m).

EEPROM SPECIFICATIONS 8

8.1 Overview

This chapter describes and defines the content of the EEPROMs that are used to identify modules attached to Omnii.

8.2 EEPROM Hardware

Theory of Operation

Expansion modules should be equipped with a Maxim DS2431 1-wire EEPROM to identify the module and its properties to the system. This information is displayed to the user in the System Properties Control Panel applet, and is used to load appropriate software drivers.

The required pull-ups for the operation of this EEPROM are included on the main logic board.

Due to the slow transfer speed of the EEPROMs, it is not recommended to use larger EEPROMs, or to store larger amounts of information in them. Also, since the EEPROM data is read from the SysCon (system controller), it is not recommended to use the EEPROM storage for any operational functions beyond initial device setup at boot time. Accessing the EEPROM memory during device operation will consume the SysCon processing time and may slow the system down.

8.3 EEPROM Data Specification

The module EEPROM is used to identify the hardware and load appropriate software drivers. The identity will be displayed to the user in the System Properties panel of the System control panel, and will be accessible to drivers and applications.

This section describes the fields of the EEPROM which are predefined for this purpose. Manufacturers are free to use unused areas of the EEPROM for their own configuration purposes.

Note: The expansion module one-wire EEPROM may not be writable at all times during device operation and therefore should not be relied on for use after boot time.

All modules will share the common EEPROM fields below, and each module type may have specific additional fields.

8.3.1 Common EEPROM Fields

Unused fields/bytes have the value 0xFF. If the specified fields are set to 0xFF, "Unknown" is displayed in the corresponding entry in the System Properties Control Panel.

Data in a text field must start at the first byte of the field. If the data does not fill the text field, the first occurrence of byte 0xFF or 0x00 terminates the text string.

Table 8.1 Common EEPROM Fields

Field Number	Field Name	Address	Size	Contents
1	Mfg. Test Region	0-9	10 bytes	Any. (Contents ignored by online software.)
2	Part Number	10-25	16 bytes	ASCII text.
3	Serial Number	26-41	16 bytes	ASCII text.
4	Manufacturer	42-61	20 bytes	ASCII text. Character set is restricted to alphanumeric plus space and null.
5	Model	62-81	20 bytes	ASCII text. Character set is restricted to alphanumeric plus space and null.
6	Hardware Revision	82-85	4 bytes	ASCII text.
7	EEPROM Size	86	1 byte	Binary data; Size = value of this field * 128 + 128. ie: 0x00 = 128 bytes 0x01 = 256 bytes 0x03 = 512 bytes, etc. (Exception: default value 0xFF returns 128 bytes).
8	Reserved	87-89	3 bytes	All 0xFF until defined.
9+	Module Specific Parameters	90+	(variable)	Module dependent.

- **Mfg. Test Region** – This area is provided to allow manufacturing test of the module EEPROM. The contents are not used for any other purpose. This is reserved for the original board manufacturer.
- **Part Number** – This field will be displayed to the user in the System Properties panel of the System applet in the Windows Control Panel. If the ASCII string is less than 16 bytes, it must be null-terminated.
  For OEM expansion modules, this field may be any format.
  For Psion Teklogix modules, this field will use the 12-digit Psion Teklogix part number format.
- **Serial Number** – This field will be displayed to the user in the System Properties panel of the System applet in the Windows Control Panel. If the ASCII string is less than 16 bytes, it must be null-terminated.
  For OEM expansion modules, this field may be any format.
  For Psion Teklogix modules, this field will use the 14-digit Psion Teklogix serial number format.
- **Manufacturer and Model** – These two fields together uniquely define the module hardware. These fields will be displayed to the user in the System Properties panel of the System applet in the Control Panel. They will also define the registry key of the software driver to be loaded (if applicable).
  The character set is restricted to alphanumeric characters plus the space character, as these strings will be used as part of a WinCE registry key. If the ASCII string for either field is less than 20 bytes, it must be null-terminated.

- **Hardware Revision** – This field will be displayed to the user in the System Properties panel of the System applet in the Control Panel. This field is optional, but if used it will be at this known location and contain printable ASCII text. If the ASCII string is less than 4 bytes, it must be null-terminated.
- **EEPROM Size** – This byte identifies the EEPROM size. This size may be programmed to be smaller than the actual capacity of the part installed. If it is not programmed (ie 0xFF), the size will default to 128 bytes. Software will not be able to access any data beyond this configured size.

Warning: Setting a field size larger than the actual EEPROM size is likely to result in unexpected behaviour.

- **Reserved** – The bytes in this field are reserved for future use and must be defaulted to or programmed as 0xFF.
- **Module Specific Parameters** – Starting at this address parameters specific to the module type can be appended.

8.4 EEPROM Reading/Writing

Note: For more information on the C functions, structures and constants that appear in this section, refer to Section 4.7: "Omnii HDK Application Development Software".

EEPROMs on devices attached to Omnii are read using the **Hdk7545_ReadEepromHeader** function (see Section 4.7.4.18 on page 52), and modified using the **Hdk7545_WriteEepromHeader** function (see Section 4.7.4.19 on page 53). These functions use the **Hdk7545_Eeprom** structure (see page 57) to perform reads and writes.

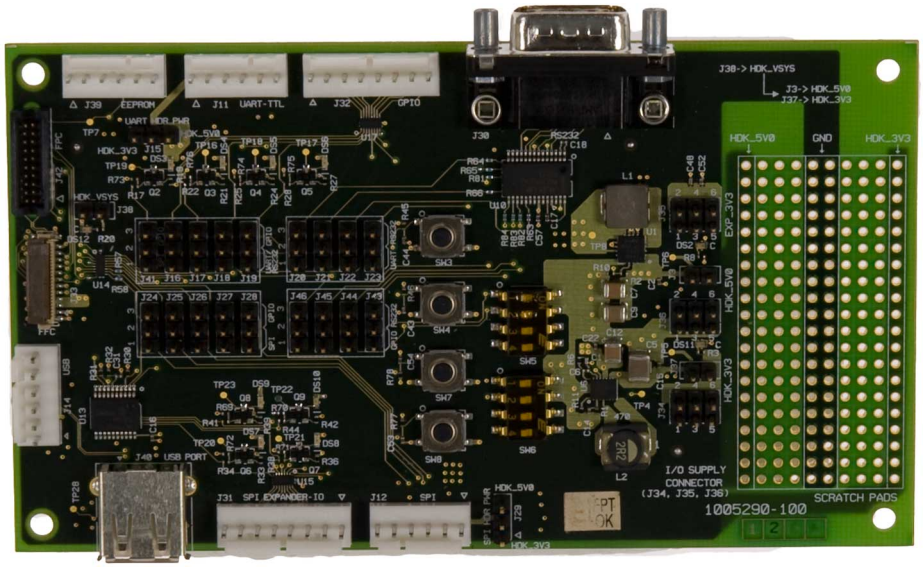
The HDK Demo application program included with the Omnii HDK can be used to more easily read and write these fields. See Chapter 10: "HDK Demo Application" for information on how to install and use this application, and Section 10.6.2.4: "EEPROM" for specific details regarding EEPROM reading and writing.

BREAKOUT BOARD 9

9.1 Overview

This chapter describes the features and functions of the development breakout board for Omnii. The board connects to the Omnii expansion ports, and breaks out the underlying interfaces into distinct connection ports of their own for testing and development. In addition to ease-of-access, the board provides standard connectors that will stand up to the repeated insertion and removal of cables required during the development process, preventing wear-and-tear on the Omnii low- and zero-insertion force connectors.

Figure 9.1 The Breakout Board

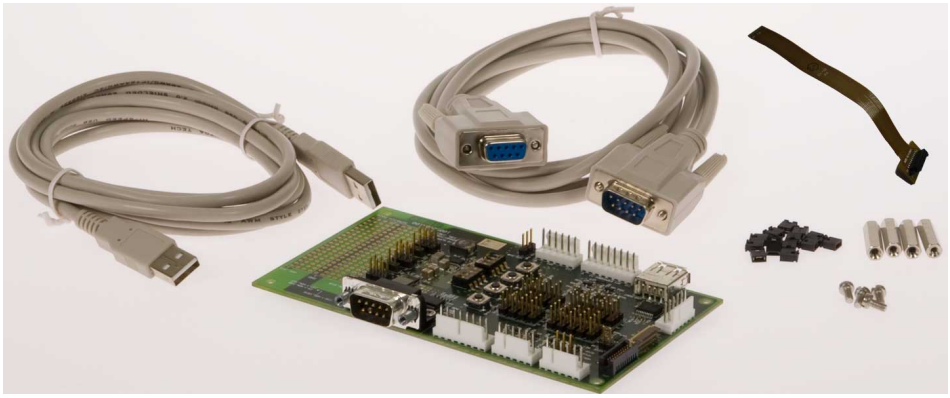


9.2 Contents of the Breakout Board Kit

The breakout board kit for Omnii includes the following items:

- 1 breakout board
- 1 flex cable
- 1 DCE-DTE serial communication cable
- 1 USB patch cable
- 25 jumpers
- 4 standoffs
- 4 screws

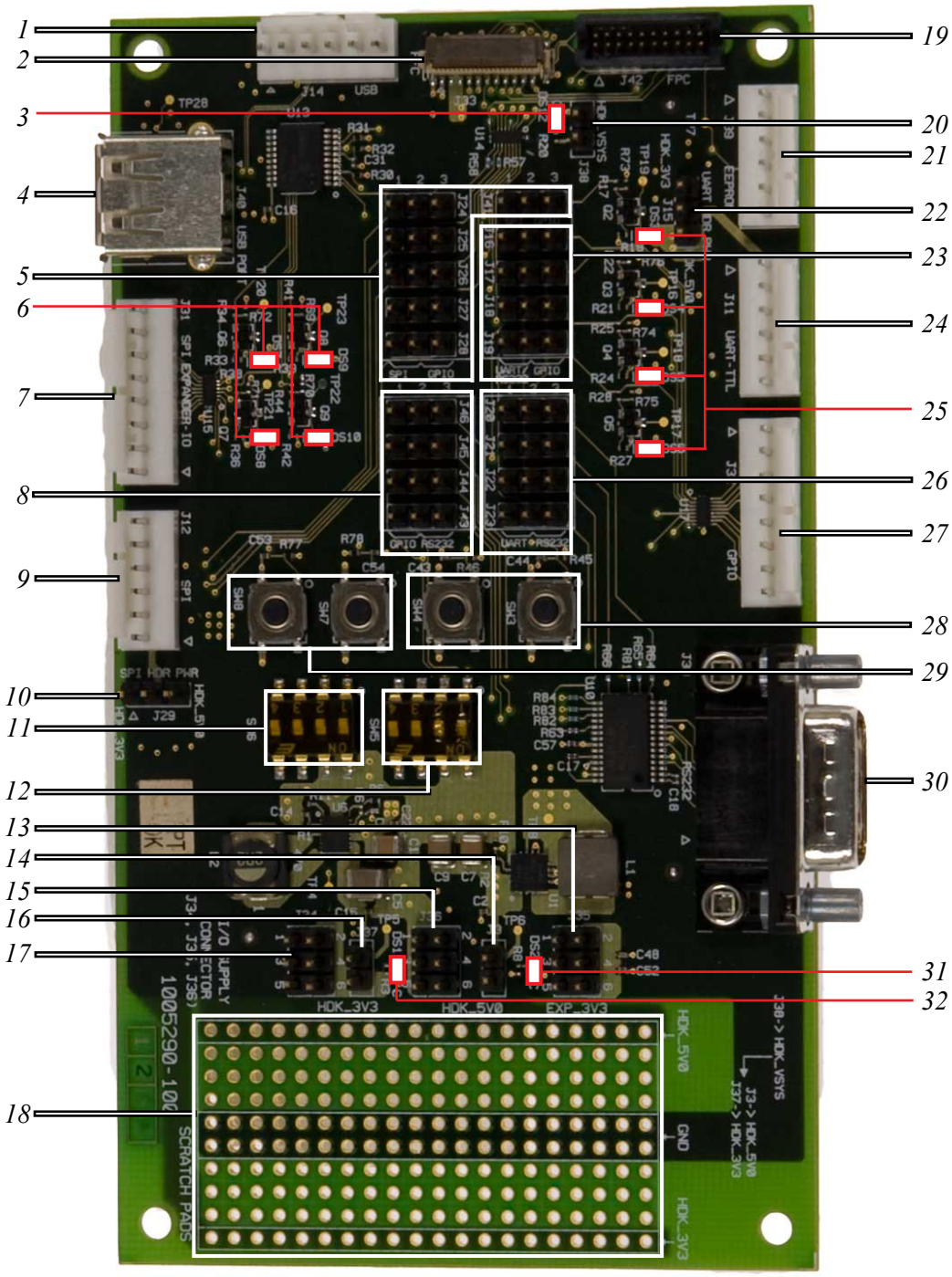
Figure 9.2 Breakout Board Kit Contents



9.3 Board Components

The components of the breakout board are indicated on the following illustration:

Figure 9.3 Breakout Board Components



1. J14: 6-pin header for USB devices
2. J33: 26-pin ZIF connector for connection to Omnii expansion port (recommended to use J42 instead)
3. DS12: LED indicator for VSYS power enabled
4. J40: USB Type A port for USB devices
5. J24 - J28, J41: Jumper headers for SPI / GPIO selection
6. DS7 - DS10: LED indicators for SPI I/O expander (not for use with Omnii XT10)
7. J31: 8-pin header for SPI I/O expander (not for use with Omnii XT10)
8. J43 - J46: Jumper headers for GPIO / RS-232 selection
9. J12: 6-pin header for external SPI devices (not for use with Omnii XT10)
10. J29: Jumper header for external SPI 3.3 V / 5 V selection (not for use with Omnii XT10)
11. SW6: DIP switches for SPI I/O expander (not for use with Omnii XT10)
12. SW5: DIP switches for GPIO_4 and GPIO_5 input
13. J35: Header for accessing 3.3 V power from Omnii (EXP_3V3)
14. J3: Jumper header to enable 5 V VSYS power (HDK_5V0)
15. J36: Header for accessing VSYS 5 V power (HDK_5V0)
16. J37: Jumper header to enable 3.3 V VSYS power (HDK_3V3)
17. J34: Header for accessing VSYS 3.3 V power (HDK_3V3)
18. Scratch pad for HDK_5V0 and HDK_3V3 power connections
19. J42: 20-pin header for connection to Omnii expansion port
20. J38: Jumper header to enable VSYS power
21. J39: 6-pin header for 1-Wire EEPROM programming
22. J15: Jumper header for UART 3.3 V / 5 V selection
23. J16 - J19: Jumper headers for UART / RS-232 / GPIO selection
24. J11: 6-pin header for UART devices
25. DS3 - DS6: LED indicators for GPIO output
26. J20 - J23: Jumper headers for UART (J11) / RS-232 (J30) selection
27. J32: 8-pin header for GPIO devices
28. SW3, SW4: Tactile switches for GPIO_6 and GPIO_7 input
29. SW7, SW8: Tactile switches for SPI I/O expander (not for use with Omnii XT10)
30. J30: DE9 port for RS-232 serial devices (with full modem support)
31. DS2: LED indicator for VSYS 5 V (HDK_5V0) power enabled
32. DS11: LED indicator for VSYS 3.3 V (HDK_3V3) power enabled

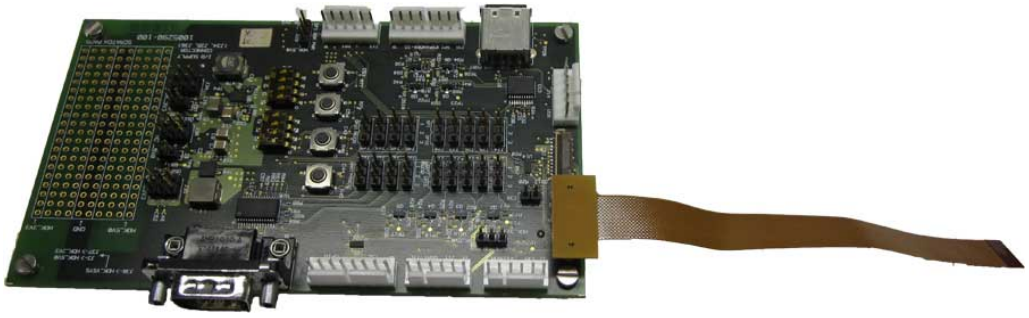
Detailed schematic diagrams for the breakout board, showing how all the components are connected, can be found in the file *BreakoutSch.pdf*, included with the Omnii HDK.

9.4 Connecting to Omnii

Before connecting the breakout board to the hand-held computer, install the standoffs included with the kit to keep the board components safely suspended above the work surface. Align the standoffs under the holes in the four corners of the board, and install a screw in each to secure them in place. **Do not overtighten!**

The breakout board connects to any of the three standard expansion ports on Omnii and isolates the specific functions of those ports for design and testing purposes. A flex cable is included with the breakout board kit which connects to the 20-pin socket (J42) of the breakout board on one end, and to the Omnii 22-pin expansion ports on the other end.

Expansion port 3 is not a 22-pin port but is instead incorporated into the 100-pin multi-function connector; therefore the flex cable cannot be used to connect directly to this port. A cable with a 100-pin connector can be constructed using the pinout and socket information provided in Section 6.4.4: “Expansion Port 3 (100-pin Multi-Function Connector)”. However, if a GPS/WWAN module is installed in the 100-pin connector, the included flex cable **can** be used with the 22-pin expansion port located on the back of that module. In this configuration, the socket may be considered identical to Expansion Port 2; follow the instructions for that port instead.



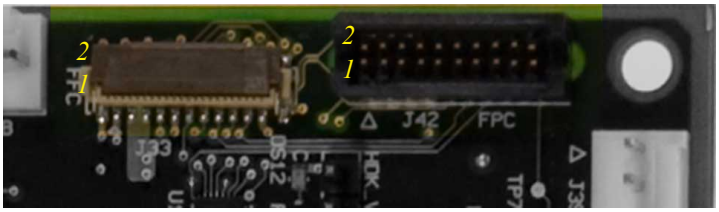
The 26-pin ZIF (zero-insertion force) connector J33 is redundant with J42 and can be used as an alternate means of connecting to the hand-held computer. However, ZIF connectors are not designed to withstand repeated cable insertion/removal cycles, therefore it is not recommended to use the J33 socket unless necessary.

The J33 and J42 connectors are not independent and cannot be used to connect to two expansion ports at one time.

Table 9.1 Omnii Headers J33 and J42 Pinout

Pin	Name	Description
1	EXP_3V3	Power. 100mA current limited
2,6,9,18,20	GND	Ground
3,4,5	VSYS	Power. Raw unregulated battery voltage 1000 mA current limited
7	USB+	USB HOST D+
8	USB-	USB HOST D-
10	GPIO3 / RTS	GPIO pin 3 or serial RTS
11	GPIO2 / CTS	GPIO pin 2 or serial CTS
12	GPIO1 / RXD	GPIO pin 1 or serial RXD
13	GPIO0 / TXD	GPIO pin 0 or serial TXD
14	GPIO7 / DTR / CS_N	GPIO pin 7, Serial DTR or SPI CS_N
15	GPIO6 / DSR / SCLK	GPIO pin 6, Serial DSR or SPI SCLK
16	GPIO5 / DCD / MISO	GPIO pin 5, Serial DCD or SPI MISO
17	GPIO4 / RI / MOSI	GPIO pin 4, Serial RI or SPI MOSI
19	1WIRE	Connection for 1-Wire EEPROM
21-26	GND	Ground (J33 only)

Figure 9.4 Omnii Headers J33 and J42 Pin Order



For further details on the functions of the expansion ports, see Section 6.4: “Expansion Ports”.

To connect to the Omnii expansion ports the back cover of the unit must be removed. A T10 and a T6 Torx screwdriver are required to do this.

Remove the Omnii Back Cover

1. Power Omnii off.
2. If a pod expansion module is installed on the expansion back cover, it should be removed first:
   a. Remove the four T10 Torx screws securing the pod to the back cover.
   b. Lift the pod from the unit.
   c. Disconnect the flex cable from the expansion pod module.
3. Remove the eight T10 Torx screws holding the back cover to the main housing.
4. Remove the two T6 Torx screws at the top of the display on the front of the unit.

5. Gently lift the back cover from the main housing to allow access to the cables that are still attached.
6. Disconnect any cables attaching the back cover to the main unit (trigger switch, scanner/imager, speaker, camera, GPS antenna are all possibilities).
7. Remove the back cover module.
8. Connect one end of the cable to the expansion port you wish to use (see Section 6.4: "Expansion Ports" for details on connecting to these ports).
9. If the expansion back cover is used, run the cable through the end-cap or pod openings and reattach the back cover for stability and security while testing.

Warning: Exercise extreme caution if operating Omnii with the back cover removed. The unit may be damaged if anything makes contact with the exposed circuitry while it is powered on.

10. Connect the other end of the cable to socket J33 or J42 on the breakout board.

Warning: Zero- and low-insertion force connectors such as J33 and those on the Omnii main logic board are not designed to withstand many cable insertion/removal cycles and may be damaged by repeated use. It is highly recommended to use socket J42 to connect to the hand-held computer.
Also, when breaking the connection from the hand-held computer, disconnect the cable from the breakout board rather than from the Omnii whenever possible.

11. Power Omnii on.
12. Load and run the required software for testing the expansion device.

Note: The breakout board will not work with the end-cap expansion port (Expansion Port 1) on Omnii XT10 computers with main logic boards prior to revision "-500".

Attach the Omnii Back Cover

When you are finished with your testing, follow these steps to re-attach the back cover to your Omnii.

1. Connect all cables from the back cover to the main logic board (trigger/speaker, camera, etc.).
2. Fit the back cover in place on the main housing, making sure no cables are caught in the seal. If a pod expansion module is required, fold the flex cable so that it is accessible through the pod opening in the expansion back cover.

3. Install eight M3x6 T10 Torx screws to secure the back cover to the main housing. Torque the screws to 5.0 in-lb (0.56 N-m) in the sequence indicated below to ensure a consistent seal around the perimeter.

4. Install the two T6 Torx screws in the front cover above the display, and torque to 2.5 in-lb (0.28 N-m).
5. If a pod expansion module is required on the expansion back cover, attach it now:
   a. Connect the flex cable from the pod opening to the device mounted in the pod.
   b. Seat the pod in place on the back cover.
   c. Install the four T10 Torx screws to secure the pod to the back cover, and torque to 5 in-lb (0.56 N-m).

9.5 Power Configuration

Power from Omnii can be accessed in several ways on the breakout board. Power convertors on the breakout board convert VSYS power from Omnii to 3.3 V and 5 V power lines. These lines in turn provide power to circuits for testing GPIO, UART and USB devices; to power headers J34 (HDK_3V3) and J36 (HDK_5V0); and to the power scratch pad grid.

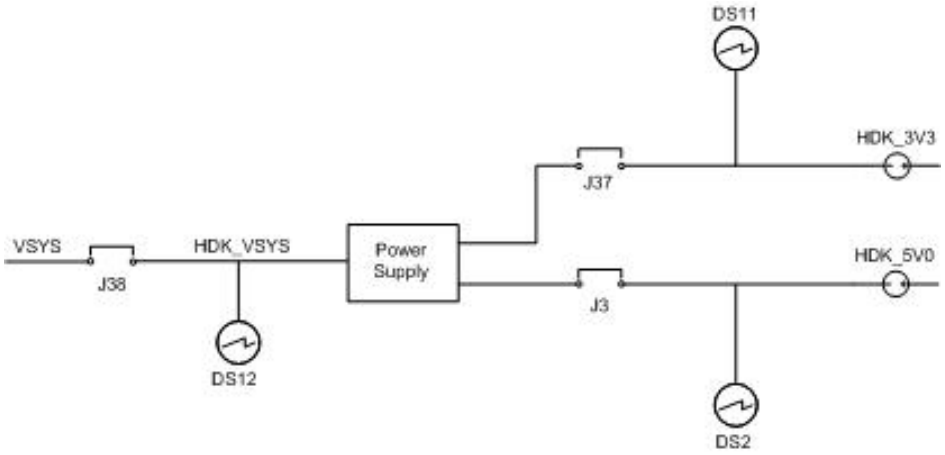
3.3 V power is also derived from the EXP_3V3 pin on Omnii and supplied to the DE9 RS-232 port (J30), to power header J35 (EXP_3V3), and to header J39 for programming the onboard 1-wire EEPROM.

Access VSYS Power

VSYS power is required to test devices on the GPIO, UART and USB interfaces. It must also be enabled to access power from power headers J34 and J36, or from the scratch pad grid.

1. Install a jumper across the pins of jumper header J38 to enable VSYS power from Omnii.
   LED DS12 illuminates to indicate VSYS power is enabled.
2. Install a jumper across the pins of jumper header J3 to enable the 5 V power supply.
   LED DS2 illuminates to indicate the 5 V (HDK_5V0) power supply is enabled.
3. Install a jumper across the pins of jumper header J37 to enable the 3.3 V power supply.
   LED DS11 illuminates to indicate the 3.3 V (HDK_3V3) power supply is enabled.

Figure 9.5 VSYS Power Configuration



With the jumpers installed, power is supplied to the GPIO, UART and USB interfaces, the HDK_3V3 (J34) and HDK_5V0 (J36) headers, and the scratch pad grid.

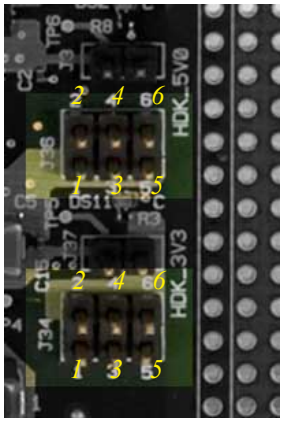
For further information on working with GPIO, UART or USB devices, see the appropriate section (Section 9.6: “GPIO Devices”, Section 9.7: “RS-232 / UART Devices”, Section 9.8: “USB Devices”).

The J34 and J36 headers have similar pin configurations. To access power from these headers, construct wire harnesses with the following pinouts:

Table 9.2 Power Headers J34 and J36 Pinout

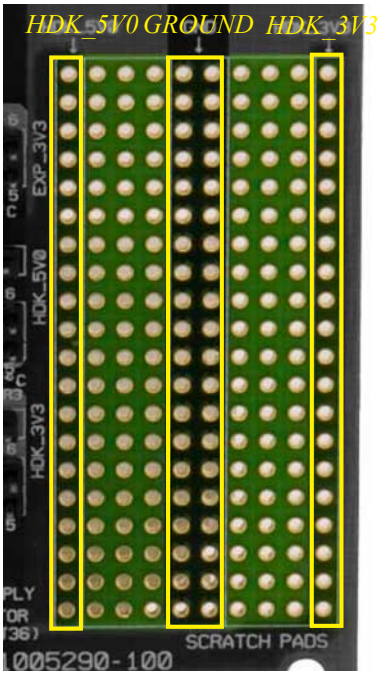
Pin	Name	Description
1,3,5	GND	Ground
2,4,6	HDK_3V3 HDK_5V0	VSYS 3.3 V power (J34) VSYS 5 V power (J36)

Figure 9.6 Power Headers J34 and J36 Pin Order



VSYS 3.3 V and 5.0 V power can be accessed directly from the scratch pad area at the side of the breakout board. The scratch pad provides 20 solder points for HDK_3V3 power, 20 solder points for HDK_5V0 power, and 40 solder points for connecting to ground. These solder points are clearly marked on the board:

Figure 9.7 Power Scratch Pad Contact Points



Access Direct 3.3V Power

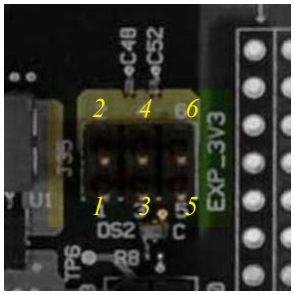
No action is required to configure the breakout board to access the 3.3 V power from the EXP_3V3 line. When the board is connected to Omnii, power is automatically supplied directly to the DE9 RS-232 connector (J30), the EXP_3V3 power header (J35), and the 1-wire EEPROM header (J39).

Power header J35 can be used to access the 3.3 V power from EXP_3V3 directly. To access power from this header, construct a wire harness with the following pinout:

Table 9.3 Power Headers J35 Pinout

Pin	Name	Description
1,3,5	GND	Ground
2,4,6	EXP_3V3	3.3 V power

Figure 9.8 Power Header J35 Pin Order



For further information on working with RS-232 devices, see Section 9.7 on page 134. For more information on programming the 1-wire EEPROM, see Chapter 8: “EEPROM Specifications”.

9.6 GPIO Devices

This section describes how to configure the breakout board to connect to and test GPIO (General Purpose Input/Output) devices.

9.6.1 GPIO Power

Power for GPIO devices is provided by the VSYS power from Omnii. Power converters on the breakout board split the VSYS power into 3.3 V (HDK_3V3) and 5 V (HDK_5V0) lines.

The GPIO features on the breakout board require power from both the 3.3 V and 5 V lines. To enable these power lines, install jumpers across the pins of jumper headers J38, J3 and J37 (see Section 9.5: “Power Configuration” for more details).

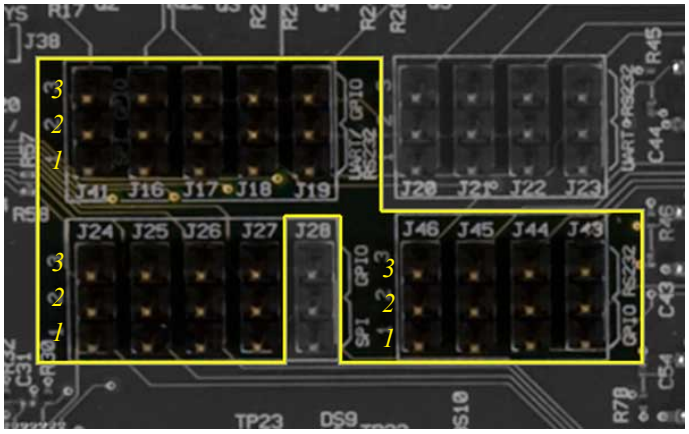
9.6.2 GPIO Data Pins

The expansion ports on Omnii use common data pins for connecting to GPIO, RS-232 / UART and SPI devices. In addition to configuring the pins on Omnii for the specific type of device being attached to the expansion port (see **PinFunctions** registry entry in Section 4.4.1 on page 26), the breakout board must also be configured to direct the data pins to the correct header. To configure the data pins on the breakout board for a GPIO device, set the following jumpers as indicated:

Table 9.4 GPIO Jumper Settings

Jumper	Pins	Function
J16	2-3	UART_TXD & RS-232 TXD / **SPI INT & GPIO_0**
J17	2-3	UART_RXD & RS-232 RXD / **GPIO_1**
J18	2-3	UART_CTS & RS-232_CTS / **GPIO_2**
J19	2-3	UART_RTS & RS-232_RTS / **GPIO_3**
J24	2-3	SPI_MOSI / **RS-232 RI & GPIO_4**
J25	2-3	SPI_MISO / **RS-232_DCD & GPIO_5**
J26	2-3	SPI_SCLK / **RS-232_DSR & GPIO_6**
J27	2-3	SPI_CS_N / **RS-232_DTR & GPIO_7**
J41	2-3	SPI INT / **GPIO_0**
J43	1-2	**GPIO_4** / RS-232_RI
J44	1-2	**GPIO_5** / RS-232_DCD
J45	1-2	**GPIO_6** / RS-232_DSR
J46	1-2	**GPIO_7** / RS-232_DTR

Figure 9.9 GPIO Jumper Headers and Pin Order

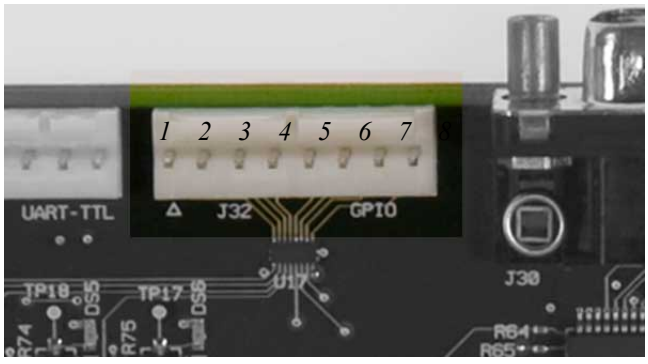


Setting the jumpers this way enables the data pins on header J32 for connecting to GPIO devices. The pin order for header J32 is as follows:

Table 9.5 GPIO Header J32 Pinout

Pin	Name	Description
1	GPIO_0	GPIO pin 0
2	GPIO_1	GPIO pin 1
3	GPIO_2	GPIO pin 2
4	GPIO_3	GPIO pin 3
5	GPIO_4	GPIO pin 4
6	GPIO_5	GPIO pin 5
7	GPIO_6	GPIO pin 6
8	GPIO_7	GPIO pin 7

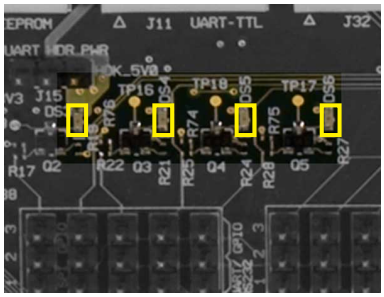
Figure 9.10 GPIO Header J32 Pin Order



9.6.3 GPIO Inputs and Outputs

GPIO lines 0 through 3 are connected to red LEDs DS3, DS4, DS5 and DS6, respectively. Activity on these LEDs indicates activity on their respective data lines. VSYS 5 V(HDK_5V0) power must be enabled to illuminate these LEDs.

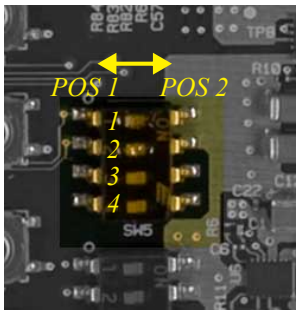
Figure 9.11 GPIO LED Indicators (DS3, DS4, DS5, DS6)



DIP switches 1 and 2 on switch bank SW5 are connected to GPIO lines 4 and 5, respectively. One end of each DIP switch is pulled high with 10 kohm to 3.3 V, and the other end is grounded. With the DIP switch in position 1, the corresponding line reports High status to Omnii. Moving the switch over to position 2 changes the line status to Low. VSYS 3.3 V (HDK_3V3) power must be enabled to power these switches.

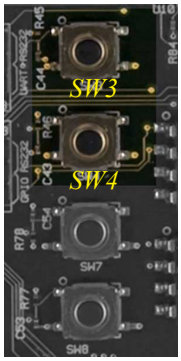
DIP switches 3 and 4 on bank SW5 are not connected.

Figure 9.12 GPIO DIP Switch Bank (SW5)



The tactile switches SW3 and SW4 are connected to GPIO lines 6 and 7 respectively. The default position of each switch is pulled high with 10 kohm to 3.3 V, and the other position is grounded. With the switch unpressed (position 1), the corresponding line reports High status to Omnii. Press the switch down to position 2 to change the line status to Low. VSYS 3.3 V (HDK_3V3) power must be enabled to power these switches.

Figure 9.13 GPIO Tactile Switches (SW3 & SW4)



9.7 RS-232 / UART Devices

This section describes how to configure the breakout board to test serial devices.

There are two headers on the board for serial interfaces: one standard RS-232 DE9 serial connector (J30), and one 6-pin in-line connector for custom UART connections (J11). Both connectors support serial communications at data rates up to 115200 Kbps.

9.7.1 RS-232 / UART Power

Power for the RS-232 interface is provided by the EXP_3V3 power line from Omnii. When the breakout board is connected to Omnii, power is automatically supplied to this port.

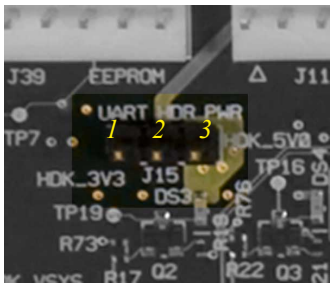
Power for devices connected to the custom UART connector is provided by VSYS power from Omnii. Power converters on the breakout board split the VSYS power into 3.3 V (HDK_3V3) and 5 V (HDK_5V0) lines.

The UART header on the breakout board can draw power from either the 3.3 V or 5 V lines, but not both at the same time. These power lines must first be enabled by installing jumpers across the pins of jumper headers J38, J37, and J3 (see Section 9.5: “Power Configuration” for more details). After the power lines are enabled, select the UART voltage by installing a jumper on jumper header J15 as indicated in the following table:

Table 9.6 UART Power Jumper J15 Settings

Position	Function
1-2	UART 3.3 V Power
2-3	UART 5 V Power

Figure 9.14 UART Power Jumper J15 Pin Order



9.7.2 RS-232 / UART Data Pins

The expansion ports on Omnii use common data pins for connecting to GPIO, RS-232 / UART and SPI devices. In addition to configuring the pins on Omnii for the specific type of device being attached to the expansion port (see **PinFunctions** registry entry in Section 4.4.1 on page 26), the breakout board must also be configured to direct the data pins to the correct header.

Configuring the data pins for a serial connection is done in two stages. First, the pins must be directed between GPIO and serial function, using jumpers J16-J19. Next, having established serial connectivity, the data pins must be routed to either the standard RS-232 port (J30) or the custom inline UART connector (J11) using jumpers J20-J23.

Data lines for full modem support are also provided and are similarly configured in two stages. First, set jumpers J24-J27 to indicate GPIO and RS-232 functions. Next, use jumpers J43-J46 to select the modem signals.

To configure the data pins on the breakout board for a serial device connected to the RS-232 DE9 connector (J30), set the following jumpers as indicated:

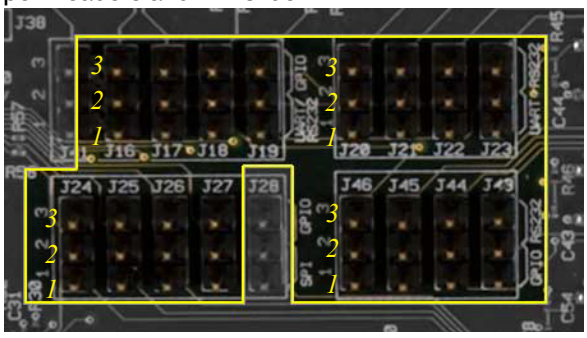
Table 9.7 RS-232 / UART Jumper Settings

Jumper	Pins	Function
J16	1-2	**UART_TXD & RS-232_TXD** / SPI_INT & GPIO_0
J17	1-2	**UART_RXD & RS-232_RXD** / GPIO_1
J18	1-2	**UART_CTS & RS-232_CTS** / GPIO_2
J19	1-2	**UART_RTS & RS-232_RTS** / GPIO_3
J20	2-3	UART_TXD / **RS-232_TXD**
J21	2-3	UART_CTS / **RS-232_CTS**
J22	2-3	UART_RXD / **RS-232_RXD**
J23	2-3	UART_RTS / **RS-232_RTS**

Table 9.8 Modem Signal Jumper Settings

Jumper	Pins	Function
J24	2-3	SPI_MOSI / **RS-232 RI & GPIO_4**
J25	2-3	SPI_MISO / **RS-232_DCD & GPIO_5**
J26	2-3	SPI_SCLK / **RS-232_DSR & GPIO_6**
J27	2-3	SPI_CS_N / **RS-232_DTR & GPIO_7**
J43	2-3	GPIO_4 / **RS-232_RI**
J44	2-3	GPIO_5 / **RS-232_DCD**
J45	2-3	GPIO_6 / **RS-232_DSR**
J46	2-3	GPIO_7 / **RS-232_DTR**

Figure 9.15 RS-232 Jumper Headers and Pin Order



The RS-232 standard male DE9 serial connector (J30) has the following pinout:

Table 9.9 RS-232 Header J30 Pinout

Pin	Name	Description
1	DCD	Data Carrier Detect
2	RXD	Received Data

Table 9.9 RS-232 Header J30 Pinout

Pin	Name	Description
3	TXD	Transmitted Data
4	DTR	Data Terminal Ready
5	GND	Ground
6	DSR	Data Set Ready
7	RTS	Request to Send
8	CTS	Clear to Send
9	RI	Ring Indicator

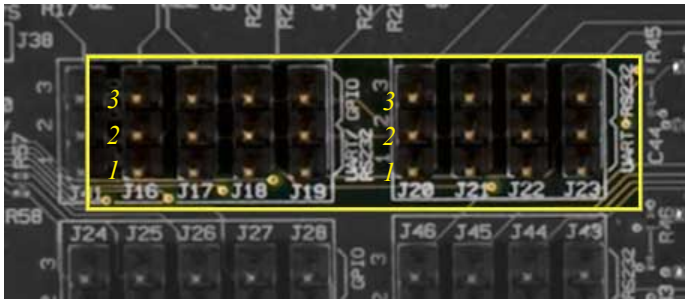
Figure 9.16 RS-232 Header J30 Pin Order

To configure the data pins on the breakout board for a serial device connected to the 6-pin in-line UART connector (J11), set the following jumpers as indicated:

Table 9.10 UART Data Pin Jumper Settings

Jumper	Pins	Function
J16	1-2	**UART_TXD & RS-232 TXD** / SPI_INT & GPIO_0
J17	1-2	**UART_RXD & RS-232 RXD** / GPIO_1
J18	1-2	**UART_CTS & RS-232 CTS** / GPIO_2
J19	1-2	**UART_RTS & RS-232 RTS** / GPIO_3
J20	1-2	**UART_TXD** / RS232_TXD
J21	1-2	**UART_CTS** / RS232_CTS
J22	1-2	**UART_RXD** / RS232_RXD
J23	1-2	**UART_RTS** / RS232_RTS

Figure 9.17 UART Data Pin Jumper Headers



The 6-pin in-line custom UART connector (J30) has the following pinout:

Table 9.11 UART Header J11 Pinout

Pin	Name	Description
1	HDK_VDD	UART 3.3 V or 5 V power
2	UART_RXD	Received data
3	UART_TXD	Transmitted data
4	UART_RTS	Request to send
5	UART_CTS	Clear to send
6	GND	Ground

Figure 9.18 UART Header J11 Pin Order

9.8 USB Devices

This section describes how to use the breakout board to test USB devices.

9.8.1 USB Power

Power for USB devices is provided by the VSYS power from Omnii. Power converters on the breakout board split the VSYS power into 3.3 V (HDK_3V3) and 5 V (HDK_5V0) lines.

The USB headers on the breakout board require power only from the 5 V line. Install jumpers across the pins of jumper headers J38 and J3 to enable this power line (see Section 9.5 on page 129 for more details).

9.8.2 USB Data Pins

The breakout board supplies two headers for connecting to USB devices: J40 is a standard Type A USB port, and J14 is a 6-pin in-line connector. The two headers are not independent and cannot be used to connect to two different USB devices simultaneously.

Table 9.12 USB Headers J14 and J40 Pinout

Pin	Name	Description
1	VBUS	USB 5 V power
2	D-	USB host D-
3	D+	USB host D+
4	GND	Ground

Table 9.12 USB Headers J14 and J40 Pinout

Pin	Name	Description
5	SHL1 (J40) not connected (J14)	Ground (shield)
6	SHL2 (J40) not connected (J14)	Ground (shield)

Figure 9.19 USB Header J14 Pin Order

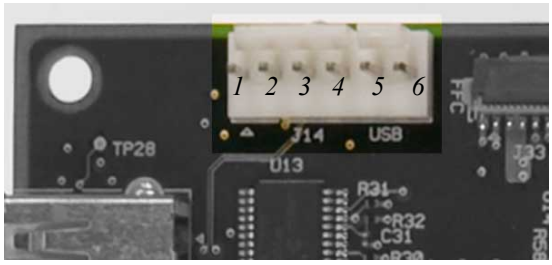
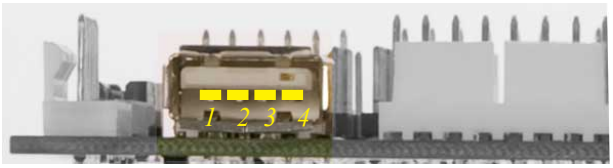


Figure 9.20 USB Header J40 Pin Order



9.9 1-Wire EEPROM Programming

This section describes how to access and program the 1-wire EEPROM on the breakout board to identify itself as a specific device model to Omnii.

9.9.1 1-Wire EEPROM Power

Power for the 1-wire EEPROM can be provided by the EXP_3V3 power line from Omnii, or from an external device.

To take power from the EXP_3V3 power line, install a jumper across pins 1 and 2 of the EEPROM header J39. As long as the breakout board is connected to Omnii, power will be available across these pins.

To power the EEPROM from an external device, construct a cable to connect to EEPROM header J39 with the following pinout:

Table 9.13 1-Wire EEPROM External Device Cable Pinout

Pin	Description
1	No connection
2	No connection
3	External 3.3 V power supply
4	1-Wire EEPROM data
5	Ground
6	No connection

Warning: When constructing a cable with external power, ensure all 6 pins are used, with pins 1 and 2 disconnected. If power is supplied from the cable at the same time as EXP_3V3 power is being supplied across pins 1 and 2, damage to the board may result.

9.9.2 1-Wire EEPROM Details

The breakout board provides a 1-Wire EEPROM similar to those required in expansion devices to identify the make and model of the device to Omnii, so that the appropriate device drivers can be loaded and the device can appear in the Windows Control Panel System applet. By programming this EEPROM the breakout board can identify itself as any device for development and testing purposes.

The EEPROM chip (U5) has 1024 bits of memory organized into four memory pages of 256 bits each. Data is written to an 8-byte scratch pad, verified and copied to the EEPROM memory.

Table 9.14 1-Wire EEPROM Header J39 Pinout

Pin	Name	Description
1	EXP_3V3	3.3 V power from EXP_3V3
2	SUP_EEPROM	EEPROM power supply from Omnii (jumper pin 1 & 2)
3	SUP_EXT	EEPROM power supply from external device (**NO** jumper on pin 1 & 2)
4	1DATA_EEPROM	Data
5	GND_EEPROM	Ground
6	NC	No connection

Figure 9.21 1-Wire EEPROM Header J39 Pin Order

For details on programming the EEPROM, see Chapter 8: “EEPROM Specifications”.

A simple interface for programming the breakout board EEPROM from the Omnii hand-held computer is provided in the HDK Demo application included with the Omnii HDK. See Chapter 10: “HDK Demo Application” for information on how to install and use this application, and Section 10.6.2.4: “EEPROM” for specific details regarding EEPROM reading and writing.

9.10 Troubleshooting

This section provides tips and suggested solutions for common problems encountered when working with the breakout board.

Table 9.15 Power Troubleshooting

Symptom	Check	Solution
All power issues.	Is breakout board connected to Omnii?	Check and reseat connection at both ends.
All power issues.	Is Omnii powered on?	Power on Omnii hand-held computer.
No VSYS power.	Is LED DS12 lit?	Install jumper across pins of J38.
No HDK_3V3 power.	Is LED DS11 lit?	Install jumper across pins of J37.
No HDK_5V0 power.	Is LED DS2 lit?	Install jumper across pins of J3.
No power from header J34 (HDK_3V3).	Are LEDs DS11 and DS12 lit?	Install jumpers across pins of J38 and J37.
No power from header J35 (EXP_3V3).	Is breakout board connected to Omnii?	Check and reseat connection at both ends.
No power from header J36 (HDK_5V0).	Are LEDs DS2 and DS12 lit?	Install jumpers across pins of J38 and J3.
No power from HDK_3V3 side of scratch pad grid.	Are LEDs DS11 and DS12 lit?	Install jumpers across pins of J38 and J37.
No power from HDK_5V0 side of scratch pad grid.	Are LEDs DS2 and DS12 lit?	Install jumpers across pins of J38 and J3.

Table 9.16 GPIO Troubleshooting

Symptom	Check	Solution
Device not working.	Are GPIO data jumpers set correctly?	Install jumpers J16-J19, J24-J27 and J41 to position 2-3, and jumpers J43-J46 to position 1-2.
Device not working.	Is device being recognized as GPIO by Omnii?	Program 1-wire EEPROM and configure registry entries on Omnii to properly identify device as GPIO. (See Chapter 8: “EEPROM Specifications” and Section 4.4.1: “Registry Settings for Expansion Devices”.)
Device not working.	Are data lines on Omnii expansion port set for GPIO device?	Use API commands to verify and set data lines for GPIO function. (See Section 4.7.4: “Omnii HDK API Functions”.)
LEDs DS3-DS6 not illuminating.	Is HDK_5V0 power enabled (are LEDs DS2 and DS12 lit)?	Install jumpers across pins of J38 and J3.
LED DS3 **only** not illuminating.	Is jumper installed correctly on jumper header J41?	Install jumper across pins 2-3 of J41.

Table 9.16 GPIO Troubleshooting

Symptom	Check	Solution
DIP switches 1 & 2 on bank SW5 not working.	Is HDK_3V3 power enabled (are LEDs DS11 and DS12 lit)?	Install jumpers across pins of J38 and J37.
Tactile switches SW3 and SW4 not working.	Is HDK_3V3 power enabled (are LEDs DS11 and DS12 lit)?	Install jumpers across pins of J38 and J37.

Table 9.17 RS-232/UART Troubleshooting

Symptom	Check	Solution
Device not working.	Are serial data jumpers set correctly?	Install jumpers J16-J19 to position 1-2. Install jumpers J20-J23 to position 1-2 for the J11 UART connector, or position 2-3 for the J30 RS-232 connector.
Modem device not working.	Are modem signal jumpers set correctly?	Install jumpers J24-J27 and J43-J46 to position 2-3.
Device not working (J11 connector).	Is UART power jumper set to provide the correct voltage?	Install jumper on header J15 to provide 3.3 V (position 1-2) or 5 V (position 2-3) power as required. Verify VSYS power enabled (see Power Troubleshooting section).
Device not working (J30 connector).	Is EXP_3V3 power being supplied (is breakout board connected to Omnii)?	Check and reseat connection at both ends.
Device not working.	Is device being recognized as serial (RS-232/UART) by Omnii?	Program 1-wire EEPROM and configure registry entries on Omnii to properly identify device as an RS-232/UART device. (See Chapter 8: “EEPROM Specifications” and Section 4.4.1: “Registry Settings for Expansion Devices”.)
Device not working.	Are data lines on Omnii expansion port set for UART device?	Use API commands to verify and set data lines for UART function. (See Section 4.7.4: “Omnii HDK API Functions”.)

Table 9.18 USB Troubleshooting

Symptom	Check	Solution
Device not working.	Is HDK_5V0 power enabled (are LEDs DS2 and DS12 lit)?	Install jumpers across pins of J38 and J3.
Device not working.	Is device being recognized as USB by Omnii?	Program 1-wire EEPROM and configure registry entries on Omnii to properly identify device as USB. (See Chapter 8: “EEPROM Specifications” and Section 4.4.1: “Registry Settings for Expansion Devices”.)

HDK DEMO APPLICATION 10

10.1 About The HDK Demo Application

The HDK Demo application provides some basic tools to verify communication with a USB, UART serial or GPIO expansion device.

The application is intended for use primarily in conjunction with the breakout board accessory described in Chapter 9: "Breakout Board", which converts the TTL output from the expansion port to the appropriate logic level for the type of device being tested (GPIO, UART, etc.). The application may be used with a device connected directly to an Omnii expansion port, provided the device performs this conversion.

10.2 Installing the HDK Demo Application

Before installing the HDK Demo application, you must first download and install the Windows Embedded CE 6.0 development platform, and Omnii HDK development files on your computer. See Section 4.7: "Omnii HDK Application Development Software" for instructions on installing these packages.

1. Use an Omnii snap module or docking station with USB port to connect your Omnii to your development PC. See the Omnii Hand-Held Computer User Manual for details on establishing this connection so that files may be copied between the PC and Omnii.
2. Browse to the folder where you installed the HDK API files (default folder *C:\Program Files\Windows CE Tools\wce600\PsionTeklogixCE600)*.
3. Browse to the *..\HDKDemoApp* subfolder.
4. Copy the file *HDKDemo.exe* to the Omnii *\Flash Disk* folder.
5. Copy the file *TestDriver.dll* to the Omnii *\Windows* folder.
6. Browse back to the original folder, and into the *..\Lib\ARMV4I* subfolder.
7. Copy the file *7545HDK.dll* to the Omnii *\Windows* folder.
8. Warm reset your Omnii by holding the [FN] key and the [ENTER/Power] key simultaneously.

10.3 Modifying and Compiling the HDK Demo Application

You may wish to make your own alterations to the HDK Demo application program and build new executable and library files. The source code to do this is provided in the *..\HDKDemoApp\Src* subfolder of the HDK API installation, within the file *HDKDemoSrc.zip*. Extract these files into a folder on your development PC.

In Visual Studio 2008, open the project file *HDKDemoRelease.vcproj* to modify and compile the *HDKDemo.exe* file. Open the project file *TestDriver.vcproj* to modify and compile the *TestDriver.dll* file.

Note: Although only Visual Studio 2008 is supported for developing applications on the Omnii HDK, the HDK Demo application is also supported on Visual Studio 2005. To modify and compile these files using Visual Studio 2005, substitute the files ***HDKDemoRelease VS2005.vcproj*** *and* ***TestDriverRelease VS2005.vcproj*** *for the project files in the above paragraph.*

10.4 Creating Registry Keys

Certain keys and values must be created in the Omnii registry in order for Omnii to recognize the expansion device and load the appropriate drivers. A driver for the HDK Demo application (TestDriver.dll) is included with the application and should be copied to the Windows folder (see Section 10.2: “Installing the HDK Demo Application”).

Follow the instructions in this section to create the necessary registry keys for the type of device you wish to test using the HDK Demo application. More detailed information on these registry keys can be found in Section 4.4: “Registry Keys”.

1. If the following registry key does not exist, create it:

```
[HKLM\Drivers\BuiltIn\Peripherals\devices\0]
```

2. Within that key, create a subkey for the device to be tested, constructed as a concatenation of the Manufacturer and Model names stored in the 1-wire EEPROM of the expansion device. For more information on these fields, see Section 10.6.2.4: “EEPROM”. In the examples that follow, the Manufacturer is “Psion Teklogix” and the Model is “Exp1_[devicetype]”.

Note: The EEPROM function of the HDK Demo application can be used to access the 1-wire EEPROM of the connected device, even before everything else (including the registry keys) has been configured. This allows you to ensure these values are properly configured and match the registry entries.

3. Under the device subkey, create the registry values for the type of device being tested as indicated in the following subsections:

USB Device Registry Keys

```
; Registry entry for a USB device
;
[HKLM\Drivers\BuiltIn\Peripherals\devices\0\Psion Teklogix Exp1_USB]
   ; EEPROM Manufacturer field = “Psion Teklogix”, EEPROM Model field = “Exp1_USB”
“Name”=sz:”USB Device”
“ConnectorId”=dword:0        ;0 = Expansion Port 1 (End-Cap Expansion)
“PowerMode”=dword:1          ;Power off in suspend, power on on resume
“LoadFlags”=dword:2          ;0x02, load default USB host driver
```

UART Device Registry Keys

```
; Registry entry for a serial device
;
[HKLM\Drivers\BuiltIn\Peripherals\devices\0\Psion Teklogix Exp1_UART]
   ; EEPROM Manufacturer field = “Psion Teklogix”, EEPROM Model field = “Exp1_UART”
“Name”=sz:”UART Serial Device”
“ConnectorId”=dword:0        ;0 = Expansion Port 1 (End-Cap Expansion)
“PinFunctions”=dword:0F      ;0x0F = Set pins 0-3 for serial function
“PowerMode”=dword:1          ;Power off in suspend, power on on resume
“LoadFlags”=dword:1          ;0x01, load default UART driver
```

GPIO Device Registry Keys

As there is no default GPIO driver on Omnii, additional subkeys and values must be created to load the TestDriver.dll driver. This driver can only be loaded for one expansion port at a time.

```
; Registry entry for a GPIO device
;
[HKLM\Drivers\BuiltIn\Peripherals\devices\0\Psion Teklogix Exp1_GPIO]
   ; EEPROM Manufacturer field = “Psion Teklogix”, EEPROM Model field = “Exp1_GPIO”
“Name”=sz:”GPIO Device”
“ConnectorId”=dword:0        ;0 = Expansion Port 1 (End-Cap Expansion)
“PinFunctions”=dword:0       ;0x00 = Set pins 0-7 for GPIO function
“PowerMode”=dword:1          ;Power off in suspend, power on on resume

; Create the following subkeys and values to load the GPIO driver
[HKLM\Drivers\BuiltIn\Peripherals\devices\0\Psion Teklogix\Exp1_GPIO\Driver]
“Prefix”=sz:“TST”
“Dll”=sz:”TestDriver.dll”        ;Driver filename
“Index”=dword:1               ;Index = Expansion port #

[HKLM\Drivers\BuiltIn\Peripherals\devices\0\Psion Teklogix Exp1_GPIO\RegCopy]
“Drivers\\BuiltIn\\Peripherals\\devices\\0\\Psion Teklogix Exp1_GPIO\\Driver”
=sz:“Drivers\\BuiltIn\\Peripherals\\devices\\active\\0\\Psion Teklogix Exp1_GPIO”
```

4. After the registry entries have been completed, reset your Omnii by holding down the [**FN**] and [**Power/ENTER**] keys for at least three seconds.

10.5 Connecting the Hardware

Connect the breakout board to one of the expansion ports on your Omnii, as described in Section 9.4: “Connecting to Omnii”.

Connecting to the Omnii expansion ports requires removing the back cover of the unit. A T10 and a T6 Torx screwdriver are required to do this.

10.5.1 Remove the Omnii Back Cover

1. Power Omnii off.
2. If a pod is installed on the expansion back cover, it should be removed first:
   i. Remove the four T10 Torx screws securing the pod to the back cover.
   ii. Lift the pod from the unit.
   iii. Disconnect the flex cable from the pod expansion module.
3. Remove the eight T10 Torx screws holding the back cover to the main housing.
4. Remove the two T6 Torx screws at the top of the display on the front of the unit.
5. Gently lift the back cover from the main housing to allow access to the cables that are still attached.
6. Disconnect any cables attaching the back cover to the main unit (trigger switch, scanner/imager, speaker, camera, GPS antenna are all possibilities).
7. Remove the back cover module.

Warning: Exercise extreme caution if operating Omnii with the back cover removed. The unit may be damaged if anything makes contact with the exposed circuitry while it is powered on.

10.5.2 Attach the Omnii Back Cover

When you are finished with your testing, follow these steps to re-attach the back cover to your Omnii.

1. Connect all cables from the back cover to the main logic board (trigger/speaker, camera, etc.).
2. Fit the back cover in place on the main housing, making sure no cables are caught in the seal. If the scanner flex cable for a scanner or imager pod is installed, fold it down so that it is accessible through the pod opening in the expansion back cover.
3. Install the eight T10 Torx screws in the back cover, and torque to 5 in-lb (0.56 N-m). Install the screws in the sequence shown here to ensure a consistent seal around the perimeter.

4. Install the two T6 Torx screws in the front cover above the display, and torque to 2.5 in-lb (0.28 N-m).
5. If a pod expansion is required on the expansion back cover, attach it now:
   i. Connect the flex cable to the device in the pod expansion module.
   ii. Seat the pod in place on the back cover.
   iii. Install the four T10 Torx screws to secure the pod to the back cover, and torque to 5 in-lb (0.56 N-m).

10.5.3 Connect the Test Module to an Expansion Port

Once the back cover has been removed, connect the breakout board flex cable to one of the Omnii expansion ports. Cables can be designed using the socket and pin descriptions provided in Section 6.4: “Expansion Ports”. Instructions on how to connect your cable to the expansion ports can also be found in that section.

10.6 Using the HDK Demo Application

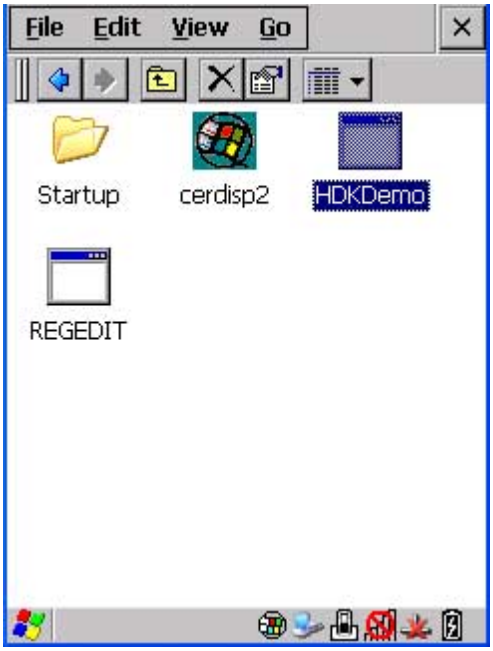
10.6.1 Getting Started

Once the hardware has been attached, insert the battery back in the unit and power it on by pressing the **[Power]** button.

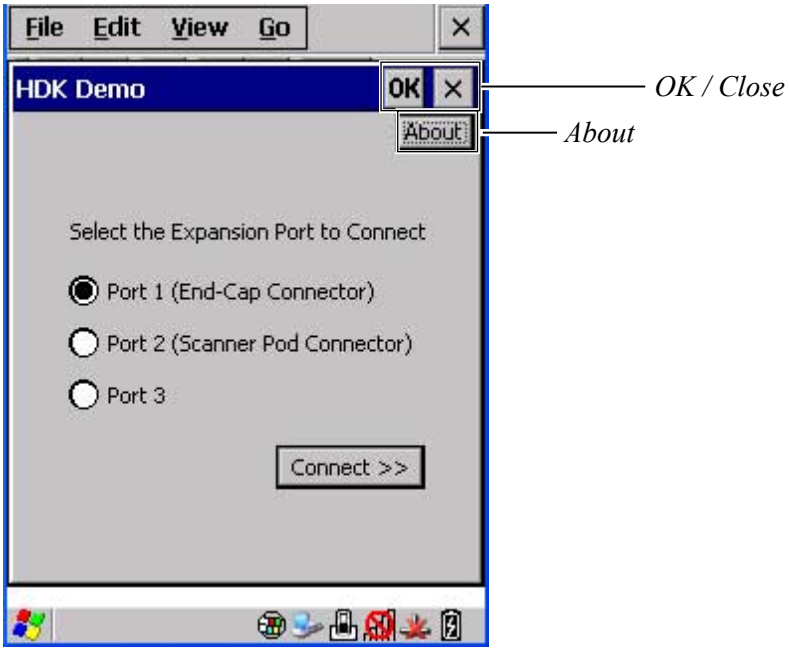
Important: Before you can use the HDK Demo application to test your device, you must first create the required registry keys as described in Section 10.4: "Creating Registry Keys", and program the Manufacturer and Model fields of the 1-wire EEPROM as described in Section 10.6.2.4: "EEPROM".

Before you can use the application program to interact with the expansion device, you must configure the registry entries as described in Section 10.4: "Creating Registry Keys", and programmed the 1-wire EEPROM fields as described in Section 10.6.2.4: "EEPROM".

Browse to the Flash Disk folder and double-tap on the "HDKDemo" icon.



The HDK Demo "Home" screen appears, prompting you to select the expansion port to which your device is connected.



The **OK**, **Close** ([**X**]) and **About** buttons appear at the top-right corner of all application screens.

- **OK** / **Close** – Tapping either of these buttons causes the HDK Demo application program to exit immediately.
- **About** – Tapping on this button displays copyright and version information.

To continue, tap on the radio button next to the expansion port to which your device is connected, then tap the **Connect** button.

A warning message appears reminding you to ensure that the flex cable is connected to the expansion port you selected.

Tap **OK** in the message window to close it and continue.

If the device attached to the selected expansion port is not a GPIO device, or if the registry entries to load the *TestDriver.dll* driver have not been created, the following error message will appear:

Tap **OK** to clear this message and continue. The HDK Demo application program will still function, with the sole exception of the "GPIO Interrupt 7" feature. See Section 10.6.2.3: "GPIO" for more details.

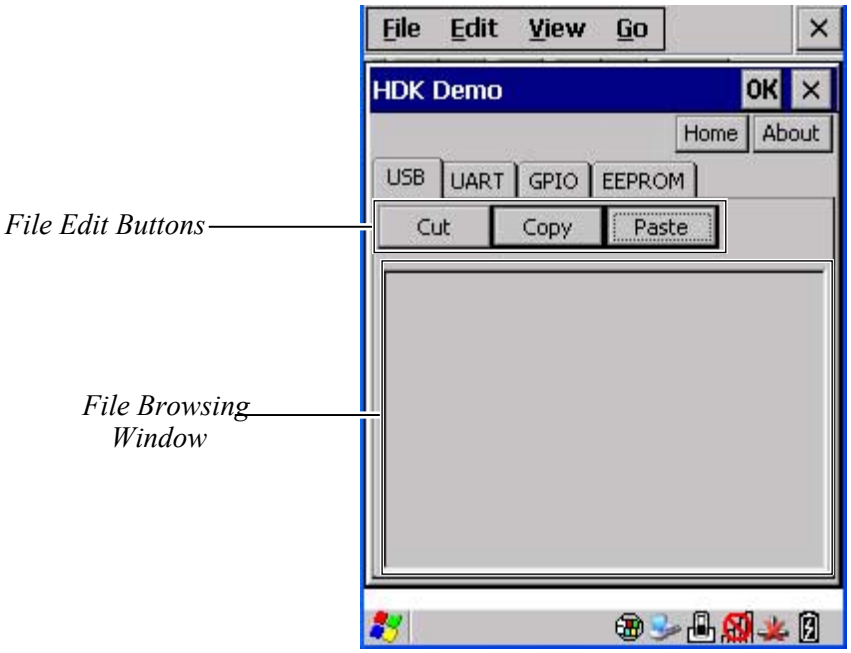
10.6.2 Main Tabs

There are four main tabs in the HDK Demo application: USB, UART, GPIO and EEPROM. Tap on the tab that corresponds to the device attached to the expansion port (**USB**, **UART** or **GPIO**), or tap on the **EEPROM** tab to access the read/write functions of the 1-wire EEPROM of the device.

- **Home** – Tap this button to return to the Home screen and select a different expansion port.
- **Tabs** – The features of each tab and instructions on how to use them are described in the following sections.

10.6.2.1 USB

The USB tab provides an interface to communicate with all USB storage devices connected to Omnii, including those connected via expansion ports.

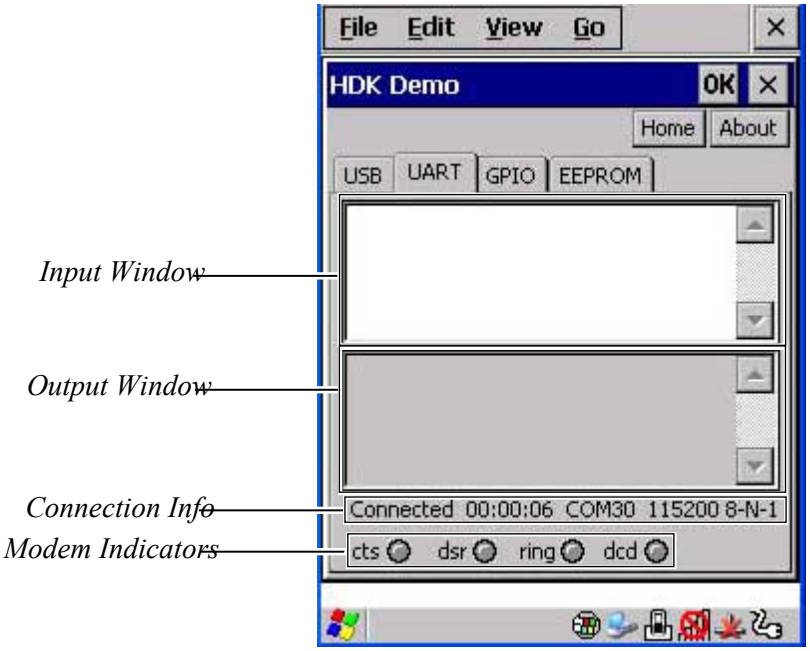


- **File Browsing Window** – The file systems of all USB drives enumerated by Omnii are displayed in the browsing window. The file systems can be navigated as in a standard Windows Explorer window – single-tap an object to highlight it; double-tap an object to open it.
- **File Edit Buttons** – These buttons allow you to move and copy files within the file system of the USB device.
  - **Cut** – Flag the currently highlighted object for relocation to another folder in the file system. The object will not be removed until the *Paste* button is clicked.
  - **Copy** – Flag the currently highlighted object for duplication in another folder in the file system.

- **Paste** – Place a copy the most recent object flagged with either the *Cut* or *Copy* button into the currently selected folder. If *Cut* was selected, the object will be deleted from its original location. If no object was previously flagged the following error message will appear:

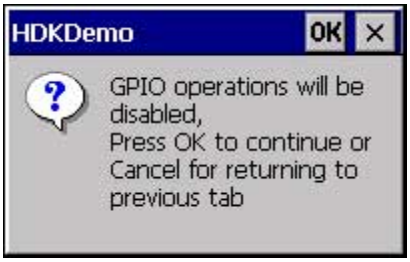
10.6.2.2 UART

The UART tab provides an interface to communicate with a UART serial device (such as a modem) connected to an expansion port.



- **Input Window** – This window provides an interface to send commands or other input to the serial device. Tap in this window to make it active, then type the input string.
- **Output Window** – This window displays command responses and other output sent from the serial device.
- **Connection Info** – This line displays status information about the device connection:
  - Total connection time (hh:mm:ss)
  - COM port being used
  - Baud rate
  - Serial connection parameters (data bits / parity / stop bits)
- **Modem Indicators** – These indicators illuminate to indicate activity on the modem data lines:
  - cts – Clear-to-Send
  - dsr – Data Set Ready
  - ring – Ring Indicator
  - dcd – Data Carrier Detect

UART and GPIO devices share the same data pins on the expansion port, so the current function of the pins must be defined by the application. If the GPIO tab was previously selected, the following warning message appears when selecting the UART tab:



This informs the user that the function of the pins will be changed from GPIO to UART operation, to conform with the tab being selected. Tap on **OK** to continue on to the UART tab, or **Close** (**X**) to remain on the current tab and leave the pins configured for GPIO operation.

After tapping **OK**, the following warning message appears:

Since UART and GPIO devices share the same data communication pins, the breakout board must also be configured so that the pins are routed to the proper circuits on the board. This configuration is done by jumpers on the breakout board, and is described in Chapter 9: “Breakout Board”. Ensure that the jumper settings are correct, and tap **OK** to proceed.

If the UART device driver is not loaded, or the COM port cannot be opened for any reason, the following error message appears:

Tap OK to continue. The features on the UART tab will be unavailable until the issue is corrected. Verify the registry settings for the UART device, and ensure that the subkey name of the device matches the Manufacturer and Model fields of the 1-wire EEPROM, as described in Section 10.4: “Creating Registry Keys”.

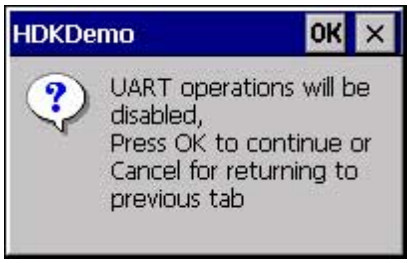
10.6.2.3 GPIO

The GPIO tab provides some screen elements to detect input and output activity on certain GPIO lines of an expansion port.

- **GPIO O/p [0/1/2]** – Tapping these output buttons toggles activity on GPIO line 0, 1 and 2. When a line is active, the corresponding "On/Off" button will be highlighted in green. The corresponding LED on the breakout board (DS3, DS4 or DS5) also illuminates to indicate activity.
- **GPIO I/p [4/5/6]** – These input indicators illuminate when activity is detected on GPIO line 4, 5, or 6. Activating the switch on the breakout board for the corresponding GPIO input (SW5 DIP switch 1: line 4; SW5 DIP switch 2: line 5; SW3: line 6) introduces activity on the line and illuminates the onscreen indicator.
- **GPIO Interrupt 7** – This indicator illuminates to indicate activity on GPIO line 7. When this occurs, activity is toggled on and off on GPIO line 3 for five seconds. Pressing switch SW4 on the breakout board illuminates this indicator, and causes LED DS6 to flash for five seconds.

*Note: If the TestDriver.dll driver has not been loaded, the "GPIO Interrupt 7" indicator still illuminates, but no activity occurs on GPIO line 3 and LED DS6 on the breakout board does **not** flash.*

UART and GPIO devices share the same data pins on the expansion port, so the current function of the pins must be defined by the application. If the UART tab was previously selected, the following warning message will appear when selecting the GPIO tab:



This informs the user that the function of the pins will be changed from UART to GPIO operation, to conform with the tab being selected. Tap on **OK** to continue on to the GPIO tab, or **Close** (**X**) to remain on the current tab and leave the pins configured for UART operation.

After tapping **OK**, the following warning message appears:

Since UART and GPIO devices share the same data communication pins, the breakout board must also be configured so that the pins are routed to the proper circuits on the board. This configuration is done by jumpers on the breakout board, and is described in Chapter 9: “Breakout Board”. Ensure that the jumper settings are correct, and tap **OK** to proceed.

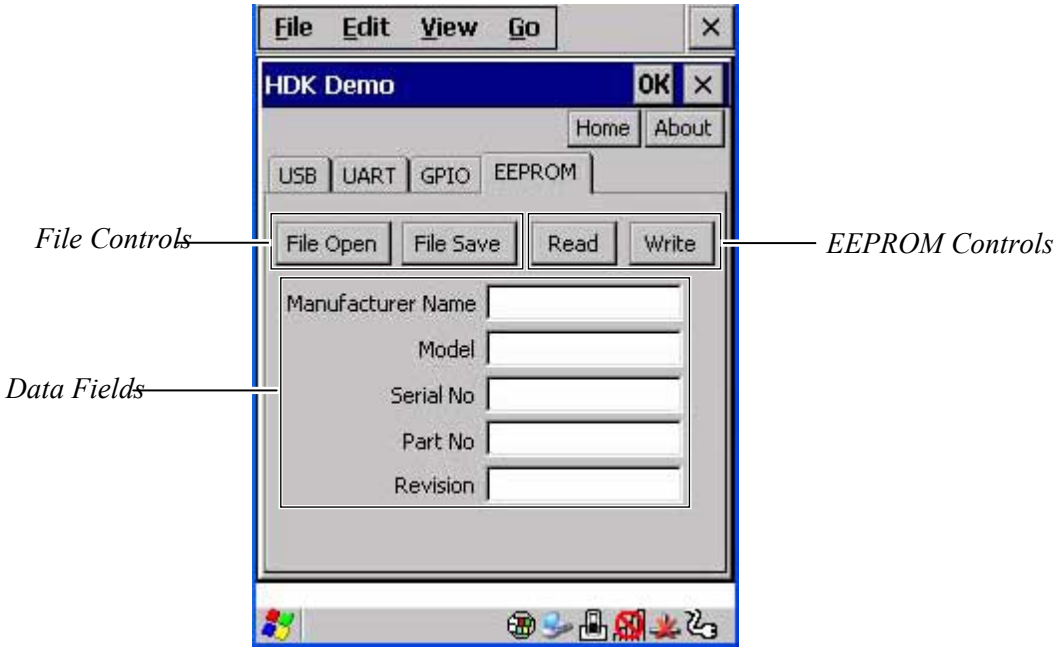
10.6.2.4 EEPROM

The EEPROM tab provides an interface to read and write device information to the 1-wire EEPROM on the expansion device.

This procedure does not require any drivers to be loaded, so you can use the HDK Demo application to program the EEPROM even before configuring anything else (registry keys, etc.). Ignore any error messages regarding drivers in this case.

*Note: When connecting to the breakout board, the 1-wire EEPROM on the breakout board is used, **not** the EEPROM of any device connected to the breakout board.*

Ensure that the information stored in the EEPROM of the breakout board is correct and matches the information in the EEPROM of the device being tested .



- **File Controls** – These buttons allow you to maintain the EEPROM data in a file on the Omnii file system.
  - **File Open** – Tapping this button opens a standard Windows file browser, from which you can select a previously saved EEPROM data file. The information contained in the file will be populated into the data fields on the screen.
  - **File Save** – Tapping this button opens a standard Windows file browser, from which you can specify a folder and filename for an EEPROM data file (standard .txt file format) to store the EEPROM data currently displayed in the data fields on the screen.
- **EEPROM Controls** – These buttons allow you to store and retrieve the data in the 1-wire EEPROM of the connected device.
  - **Read** – Tapping this button reads the information currently stored in the 1-wire EEPROM of the connected device and populates it into the data fields on the screen.
  - **Write** – Tapping this button overwrites the data in the 1-wire EEPROM of the connected device with the information currently displayed in the data fields on the screen.

Important: After writing new values to the EEPROM, you must reset your Omnii before they will take effect!

- **Data Fields** – These fields show the information to be stored in the 1-wire EEPROM of an expansion device. These fields can be filled by reading the current values from the connected device, by loading them from a previously saved file, or by typing them directly into the fields.
  - **Manufacturer Name** – The name of the manufacturer of the expansion device. This field plus the Model field combine to form the subkey name for the registry entries required to make the device function.
  - **Model** – The model name of the expansion device. The Manufacturer Name field plus this field combine to form the subkey name for the registry entries required to make the device function.
  - **Serial No** – The serial number of the expansion device.
  - **Part No** – The part number of the expansion device.
  - **Revision** – The revision number of the expansion device.

APPENDIX A

RESOURCES

Most of the following resources are available on the Psion Teklogix Community website (*http://community.psionteklogix.com*). Website registration is required to log in to the site and obtain the materials.

A.1 Psion Teklogix User Manuals

The following user manuals are available on the Psion Teklogix Community website, under **Knowledge > Knowledge Base > Product Manuals**:

- Psion Teklogix, 2010, *Omnii Hand-Held Computer (Windows Embedded CE 6.0) User Manual* (Part number 8100190)
- Psion Teklogix, 2009, *Mobile Devices SDK Developers Guide* (Part number 8100016)

A.2 Psion Teklogix Downloadable Software

The following software is available on the Psion Teklogix Community website, under **Downloads > Firmware/Software & Demos > Software Demos, Tools & Drivers**:

- Psion Teklogix USB setup utility

The following software is available on the Psion Teklogix Community website, under **Downloads > Developer (SDK/HDK)**:

- Omnii HDK (click on Psion Teklogix HDK)
- Mobile Devices SDK

A.3 Psion Teklogix Accessory And Parts Information

For more information on accessories and parts for Omnii, visit *www.psionteklogix.com/products*.

APPENDIX B

OMNII SPECIFICATIONS

B.1 The Omnii XT10 Hand-Held Computer (Model No. 7545XV)

Note: Performance specifications are nominal and subject to change without notice.

B.1.1 Hardware

Physical Dimensions

- 100 mm width x 44 mm depth x 222 mm length (3.9 in x 1.7 in x 8.74 in).
- Keypad area: 75 mm width x 32 mm depth (2.9 in x 1.3 in).

Weight

- Basic unit with battery: 590 g (1.30 lb).
- Unit with battery, high visibility display, camera, Push-to-Talk speaker, EV15 imager: 636 g (1.4 lb).

User Interface

Colour Touch Display:	9.4 cm (3.7 in.) diagonal. VGA/QVGA, 480 x 640 resolution. High visibility version: 165 cd/m² brightness provides superior sunlight visibility. Extreme Duty version: withstands 1.25 joules impact.
Keyboards:	Backlit, high durability hard-capped keys. Large selection of both alpha and numeric formats. For a list of currently available keyboard configurations, consult your order sheet.
Indicators And Controls:	Four multi-colour LEDs indicate the status of the battery, application, radio, and scanner.
Side Buttons:	Volume, Scan, Enter, Vertical Scroll.
Audio:	High volume beeper: 95 dBA Integrated Microphone/Speaker. Optional PTT Speaker. Walkie-talkie style Push-to-Talk: VoIP over Wi-Fi
Vibration:	You can set Omnii to vibrate as a result of a successful or unsuccessful bar code scan.

Power Management

- **Battery Pack**: lithium-ion 5000 mAh capacity with 8-hour life under normal operating conditions.
- Advanced Smart Battery with gas gauge.
- 3 power source options: Runs off battery, AC power, or automotive power supplies.
- Backup power: >5 minutes.

Communication

Expansion Ports:	MicroSD slot for Flash expansion. Multiple Internal Multi-Function Expansion Interfaces with: - 3v3 TTL serial - USB host - GPIO

Environmental

Standard Operating Temperature:	-20°C to +50°C (-4°F to 122°F)
Storage Temperature:	-40°C to +60°C (-40°F to 140°F) Long exposure to temperatures below -40°C (-40°F) may damage the screen and main battery. Prolonged exposure to temperatures above +60°C (+140°F) will damage the main battery and temperatures above +70°C (+158°F) may damage the unit.
Rain And Dust Resistance:	IEC 60529, classification IP65.
Humidity:	5% - 95% RH non-condensing
Drop Durability:	1.7 m (5.6 ft.), 26 drops to polished concrete (while powered on and with accessories); multiple 2.0 m (6.5 ft.) drops to polished concrete.
ESD:	+/- 8 kV contact, +/- 15 kV air discharge.

B.1.2 Software

Processor and Memory

- Texas Instruments® OMAP3® Processor 600 MHz
- RAM: 256 MB SDRAM standard.
- Flash ROM: 512 MB.

Operating System

- Microsoft® Windows® Embedded CE 6.0

Bundled Applications

- Internet Explorer® 6
- Windows Mobile Device Center
- Wordpad
- ActiveSync

Device Management & Utilities

- PsionVU
- Mobile Control Center (MCC)
- Total Recall / TweakIt / Dr. Debug

Supported Applications

- Open TekTerm Psion Teklogix® Terminal Emulation
- Stay-Linked Terminal Emulation
- Naurtech Browser
- Naurtech Terminal Emulation
- NetMotion Mobility XE VPN

B.1.3 Approvals

Safety:	IEC 60950-1
EMC:	FCC Part 15 Class B, EN 55022, EN 55024, EN 301 489
Laser:	IEC 60825-1 Ed. 2.0, Class 1, Class 2
	FDA 21 CFR 1040.10
	1040.11 Class I, Class II
Bluetooth:	2.0
RF:	*Bluetooth* and 802.11b/g: EN300 328, Part 15.247
RoHS compliant:	EU Directive 2002/95/EC

B.2 Lithium-ion Smart Battery 5000 mAh (ST3000)

For safety instructions, please see "Lithium-ion Battery Safety Precautions" in the *Omnii Hand-Held Computer Regulatory & Warranty Guide (PN 8000191)*.

Parameter	Specification
Model Number	ST3000
Chemistry	lithium-ion (Li-Ion)
Capacity	5000 mAh nominal at 1000 mA discharge 20°C to 3.0 V (min)
Voltage	3.7 V nominal (3.0 V min. to 4.2 V max.)
Cell Configuration	2 P1S (2 parallel connected cells)
Max. Charge Voltage	4.2 V +/- 1%
Recommended Charge Termination Timeout	5.0 hr - charging must stop.
Charge Temperature	0°C to +40°C (32°F to +104°F)
Discharge Temperature	-20°C to +50°C (-4°F to +122°F)
Storage Temperature	-20°C to +50°C (-4°F to +122 ºF). Storage at elevated temperatures not recommended. 25°C (77 ºF) — recommended storage temperature.
Cycle Life	300 cycles minimum with no degradation below 80% of nominal capacity based on 1C charge / 1C discharge rates (to 3.0 V) @ 25°C (77 ºF).

B.3 Wireless Radios

802.11b/g Radio

Parameter	Specification
Form Factor	Embedded surface mount module, 8.2 x 8.4 mm
Antenna Port	U.FL jack
Transmit Power	802.11b/g: 50 mW typical (+17 dBm)
Frequency Range	2.400 - 2.4835 GHz
Channels	1-11 FCC, 1-13 ETSI
RX Sensitivity	-86 dBm typ @ 11 Mbps -82 dBm @ 6 Mbps, -69 dBm @ 54 Mbps
Data Rates	802.11g: 6, 9, 12, 18, 24, 36, 48, 54 Mbps 802.11b: 1, 2, 5.5, 11 Mbps
EVM	802.11b: -28 dB typ (16%) 802.11g: -29 dB typ (13%)
Bluetooth Coexistence	Collaborative with *Bluetooth* radio.

Bluetooth Radio

Parameter	Specification
Form Factor	Embedded (920 kbps serial interface)
Bluetooth Version	Ver 2.0+EDR compliant - Adaptive Frequency Hopping (AFH) for better co-existence with 802.11 radio and Enhanced Data Rate (EDR) (up to 3 Mbps).
Antenna Type	Ceramic chip PIFA
Antenna Gain	1 dBi peak
Transmit Power	-3 dBm (0.5mW) minimum, +4 dBm (2.5 mW) max
Frequency Range	2.400–2.4835 GHz
RX Sensitivity (BER<0.1%)	-80 dBm max
Data Rate	V1.2 = 732.2 kbps and 57.6 kbps asymmetric, 433.9 kbps symmetric V2.0 = 2 & 3 Mbps
802.11 Coexistence	Collaborative with 802.11 radio and adaptive frequency hopping.

B.4 Internal Scanners and Imagers

This section lists specifications for the following internal scanners:

- SE1223LR - Long Range (decoded) Scanner
- SE1224HP - High Performance Scanner
- SE1524ER - Extended Range Scanner
- EV15 1D Standard Range Imager
- 5080 Imager/Decoder

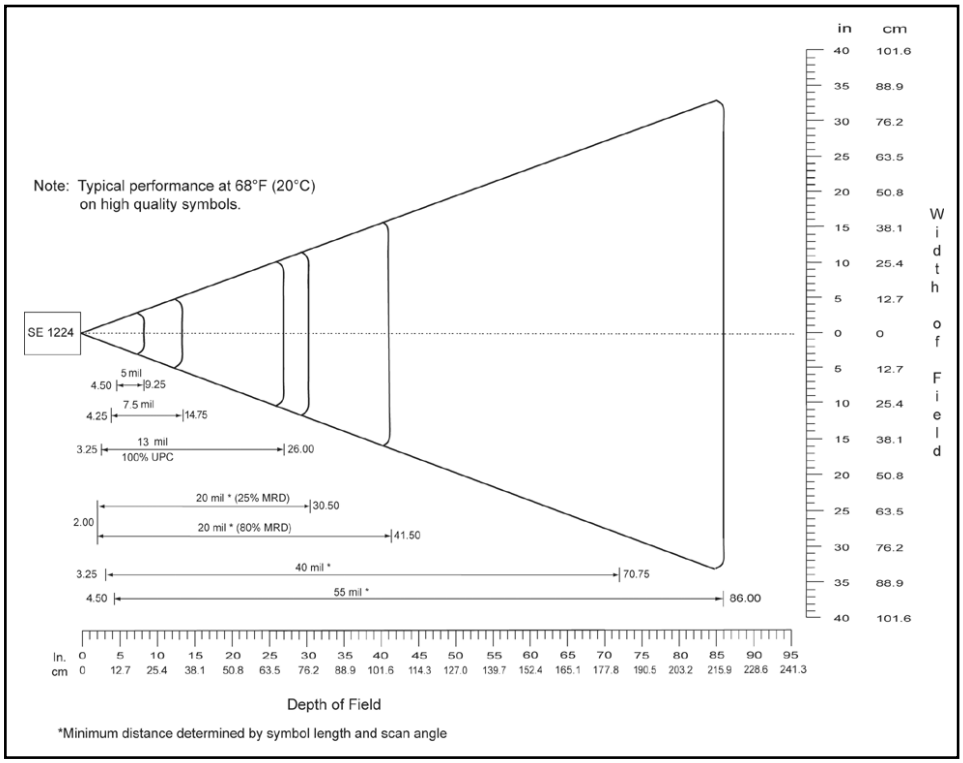
B.4.1 SE1223LR - Long Range (Decoded) Scanner

Parameter	SE1223LR
Light Source	Visible Laser Diode 650 nm
Scan Rate	35 (± 5) scans/sec (bi-directional)
Scan Angle	23º ± 2º
Scan Patterns	Linear
Minimum Print Contrast	Minimum 40% absolute dark/light reflectance measured at 650 nm.
Symbologies Supported	UPC/EAN, Code 128, Code 39, Code 93, I 2 of 5, Discrete 2 of 5, Codabar, MSI, UCC/EAN 128, TriOptic Code 39.
Programmable Parameters	Laser On Time, Aim Duration, Power Mode, Trigger Mode, Bi-directional Redundancy, Symbology types/lengths, Data formatting, Serial Parameters, Beeper Tone.
Ambient Light	Artificial: 450 ft. candles (4,844 Lux). Sunlight: 8,000 ft. candles (86,112 Lux).
Power	Input Voltage: 5.0 VDC ± 10% Input Current: 115 mA typical Standby Current: 70 µA max.
Laser Classification	Intended for use in CDRH Class II and IEC Class 2 devices
Electrical Safety	UL, VDE, and CUL recognized component laser
Environmental	RoHS-compliant

B.4.2 SE1224HP - High Performance Scanner

Parameter	SE1224HP
Type	Laser Class 2
Light Source	Visible Laser Diode 650 nm
Scan Rate	35 (± 5) scans/sec (bi-directional)
Scan Angle/Field of View	42º (typical), 30º (narrow)
Scan Patterns	Linear
Minimum Print Contrast	Minimum 25% absolute dark/light reflectance measured at 650 nm.
Symbologies	UPC/EAN, Code 128, UCC/EAN 128, RSS, Code 39, Code 93, I 2 of 5, Discrete 2 of 5, Codabar, MSI.
Programmable Parameters	Laser On Time, Aim Duration, Power Mode, Trigger Mode, Bi-directional Redundancy, Symbology types/lengths, Data formatting.
Ambient Light	Artificial: 450 ft. candles (4844 Lux). Sunlight: 8000 ft. candles (86112 Lux).
Laser Output Power (peak)	1.35 mW

B.4.2.1 SE1224HP Decode Zones

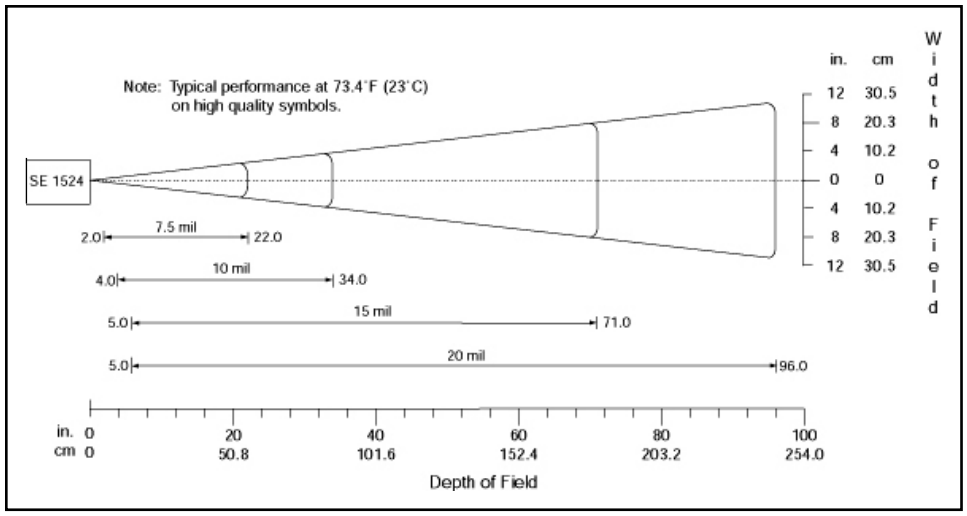


B.4.3 SE1524ER – Extended Range Scanner

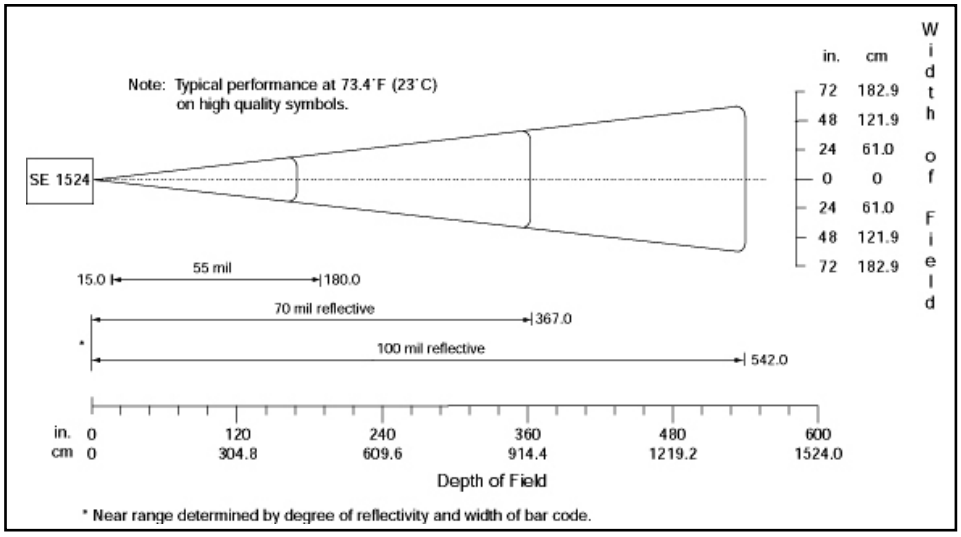
Parameter	Specification
Type	Laser Class 2
Light Source	Visible Laser Diode 650 nm
Scan Rate	35 (±5) scans/sec (bi-directional).
Scan Angle/Field of View	13.5° ±0.7°
Scan Patterns	Linear
Minimum Print Contrast	Minimum 25% absolute dark/light reflectance measured at 650 nm.
Symbologies	UPC/EAN, Code 128, UCC.EAN128, RSS, Code 39, Code 93, I 2 of 5, Discrete 2 of 5, Codabar, MSI.
Programmable Parameters	Laser On Time, Aim Duration, Power Mode, Trigger Mode, Bi-directional Redundancy, Symbology types/lengths, Data formatting.
Ambient Light	Artificial: 450 ft. candles (4,844 Lux) Sunlight: 4,000 ft. candles (86,112 Lux).
Laser Output Power (peak)	1.26 mW

B.4.3.1 SE1524ER Decode Zones

SE1524ER Decode Zone A (Short Range, Small Codes)



SE1524ER Decode Zone B (Long Range, Large Codes)



B.4.4 EV15 Imager

Parameter	Specification
Light Source	617 nm highly visible LED
Scan Angle	40º
Minimum Print Contrast	Minimum 25%
Minimum X. Dimension	0.1 mm (4 mils)
Reading Distance	Up to 90 cm (35 in)
Symbologies	UPC (E&A), EAN, RSS, Code 39, Code 128, UCC/EAN 128, ISBN, ISBT, Interleaved, Matrix, Industrial and Standard 2 of 5, Codabar, Code 93/93i, Code 11, MSI, Plessey, Telepen, PDF417, Micro PDF417
Ambient Light	Works in any lighting conditions, from 0 to 100,000 lux

B.4.4.1 EV15 Imager Decode Zone

0 Lux to 100,000 Lux		
	Minimum Range	Maximum Range
Mil Size	Inches	Inches
5	2.5	7
10	3	14
UPC	2	14.5
20	2.5	22
40	3	35.5
High quality symbols in normal room light.		

B.4.5 5080 Imager/Decoder

Parameter	Specification
Focal Point - SR	7 inches (17.8 cm) from lens plate
Focal Point - SF	4.5 inches (11.4 cm) from lens plate
Image Sensor	752 x 480 CMOS sensor
Motion Tolerance	4 inches per second
Rotational Sensitivity	360°
Viewing Angle	±40°
Ambient Light	Total darkness to 100,000 lux (full sunlight)
Illumination LEDs	626 nm ± 30 nm
Aiming	LEDs: 526 nm ± 30 nm Laser: 650 nm ± 10 nm
Input Voltage - Imager	3.3 VDC ± 5% (23°C)
Input Voltage - 5080	3.0 VDC to 5.5 VDC (23°C)
Current Draw - Imager	Max. Operating Current: 100 mA; Standby Current: 100 µA
Current Draw - 5080	Average Current (Interlaced Mode): 510 mA; Standby Current: 120 µA ; Peak: 600 mA
Symbologies: 2 Dimensional	PDF417, MicroPDF417, MaxiCode, Data Matrix, QR Code, Aztec, Aztec Mesa, Code 49, UCC Composite
Linear	Code 39, Code 128, Codabar, UPC, EAN, Interleaved 2 of 5, Reduced Space Symbology, Code 93, Codablock
Postal	Postnet (US), Planet Code, BPO 4 State, Canadian Post, Japanese Post, KIX (Netherlands) Post
OCR Fonts	OCR-A and OCR-B

B.4.5.1 5080 Working Range

Data is characterized at 23°C (73.4°F) and 0 lux ambient light.

Symbology	Size (mil)	Near	Far
SR			
Linear	8.3 (.020cm)	3.5 in. (8.9cm)	7.6 in. (19.3cm)
PDF417	10 (.025cm)	3.1 in. (7.9cm)	9 in. (22.9cm)
UPC	13 (.033cm)	2.1 in. (5.3cm)	13.2 in. (33.5cm)
Data Matrix	15 (.038cm)	2.3 in. (5.8cm)	10.2 in. (25.9cm)
QR	15 (.038cm)	3.1 in. (7.9cm)	8.8 in. (22.4cm)
MaxiCode	35 (.089cm)	2.0 in. (5.1cm)	13.0 in. (33cm)
SF			
PDF417	6.6 (.017cm)	2.8 in. (7.1cm)	6 in. (15.2cm)
Linear	7.5 (.019cm)	2.5 in. (6.4cm)	6.5 in. (16.5cm)
Data Matrix	12.5 (.021cm)	3.4 in. (8.6cm)	5.7 in. (14.5cm)
QR	8.3 (.021cm)	3.4 in. (8.6cm)	5.4 in. (13.7cm)
Linear	10 Linear	2.2 in. (5.6cm)	7.6 in. (19.3cm)
UPC	13 (.033cm)	2.0 in. (5.1cm)	8.9 in. (22.6cm)

B.5 Accessories

For details about the accessories available with Omnii, please refer to the Omnii Hand-Held Computer User Manual (P/N 8000190).

AC Wall Adaptor (Model No. ST1050)

12 VDC 2.5 A DC power supply.

Carrying Accessories

- Pistol grip.
- Carrying cases, either functional or non-functional.
- Hard and soft shell holsters.
- Hand and wrist straps.
- Forklift holster.

Communications with Power Supply

- **Desktop Docking Stations**
  - **ST4002 and ST4003:**
  - Fast charging of both internal battery and spare battery pack.
  - Host USB port.
  - Client USB port.
  - **ST4003:**
  - A DE9M serial port (unpowered) and an RJ45 10Base-T Ethernet interface. Both compatible with the Omnii USB to Ethernet/serial drivers.

- **Snap Modules**
  - **Model No. ST3101** (Charger only variant): powers and charges the hand-held.
  - **Model No. ST4001** (USB Host/Client variant): powers and charges the hand-held. It provides communications via USB 1.1/2.0 Host and USB 2.0 Client connectors and provides a DC IN port.
  - **Model No. ST4500** (USB DE9M variant): through a powered DE9M serial connector it powers and charges the hand-held, and provides communications to tethered devices. The DE9M connector is capable of speeds up to 115,200 kbp.

B.6 Camera (Optional)

- Colour, 3 Megapixel, autofocus, manual 4X digital zoom, dual LED flash, video capable (optional with or without PTT speaker).

B.7 GPS (Optional)

- High performance antenna.
- SIRF III receiver.

APPENDIX C

HDK LICENSE AGREEMENT

C.1 HARDWARE DEVELOPER KIT LICENSE AGREEMENT

IMPORTANT - READ CAREFULLY:

This Hardware Developer Kit License Agreement ("Agreement") is a legal agreement between you and Psion Teklogix ("we"), the licensor of Psion Teklogix Hardware Developer Kit ("HDK") which is downloaded from the Psion Teklogix website, for developers of hardware expansion modules intended to be used with the Psion Teklogix hand-held mobile devices.

By clicking on the "Accept" or other appropriate assent button and/or installing the HDK, you agree to be and are hereby bound by the terms and conditions of this Agreement. If you do not agree with this Agreement, we do not grant you a license to the HDK, and you may not install or use the HDK or any accompanying documentation.

The HDK is the property of Psion Teklogix Inc. or its licensors and is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The HDK is licensed, not sold. Psion Teklogix Inc. provides the HDK and licenses its use worldwide. You assume responsibility for the selection of the HDK to achieve your intended results, and for the use and results obtained from it.

C.2 GRANT OF LICENSE

We hereby grant you and you hereby accept a non-exclusive, non-transferable, royalty-free license to use the HDK to develop hardware expansion modules to be used with the Psion Teklogix hand-held mobile devices subject to the terms and restrictions set forth in this Agreement. Except as explicitly set forth below, (i) you are not permitted to sell, lease or rent, distribute or sublicense the HDK or to use the HDK in a time-sharing arrangement or in any other unauthorized manner; (ii) no license is granted to you in the human readable code of the HDK (source code); and (iii) this Agreement does not grant you any rights to patents, copyrights, trade secrets, trademarks, intellectual property or any other ownership rights with respect to the HDK.

The HDK is licensed to be used on any personal computer and/or Psion Teklogix hand-held mobile devices, provided that the HDK is used only in connection with your development of hardware expansion modules for use and compatible with the Psion Teklogix hand-held mobile devices (the "Expansion(s)"). The HDK contains certain documentation, drawings, programs, files, specifications, datasheets and APIs. You may distribute the HDK in object code format solely as part of your Expansion. The HDK shall be distributed to your customers under the terms of your standard end user license agreement, provided it includes terms that are substantially similar to those described herein. You are required to include Psion Teklogix' copyright notices on your Expansion that includes the HDK.

C.3 REQUIREMENTS, RESTRICTIONS, RIGHTS AND LIMITATIONS

a. Distribution. Except as provided for in this Agreement, you may not distribute the HDK, in whole or in part, to any other third party.

b. Virus Program. You may not develop or knowingly incorporate any virus program that may be harmful to a computer or a network in conjunction with the HDK, or use the HDK for any other purpose as which may be harmful to a third party.

c. Assignment. You may not assign or transfer the HDK to a third party or allow a third party to use the same.

d. Reverse Engineering. Modification, reverse engineering, reverse compiling, or disassembly of the HDK is expressly prohibited.

e. Export Restrictions. You agree that you will not export or re-export the HDK, or any part or copies thereof, or any products utilizing the HDK in violation of applicable laws or regulations of the United States or the country in which you obtained them.

f. Approvals. You agree that it is your responsibility to obtain any required regulatory approvals required for the sale of products utilizing the HDK anywhere such products are offered for sale.

C.4 HIGH RISK ACTIVITIES.

The HDK is not fault-tolerant and is not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the HDK could lead to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). We specifically disclaim any express or implied warranty of fitness for High Risk Activities.

C.5 DISCLAIMER OF WARRANTY

We do not warrant uninterrupted or error free operation of the HDK nor do we warrant that the HDK will meet your requirements. THE HDK AND DOCUMENTATION ARE PROVIDED "AS-IS" WITHOUT ANY WARRANTY WHATSOEVER AND WITHOUT ANY TECHNICAL SUPPORT OF ANY KIND. WE DISCLAIM ANY AND ALL REPRESENTATIONS, WARRANTIES AND CONDITIONS, WHETHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABLE QUALITY OR FITNESS FOR A PARTICULAR PURPOSE. WE DO NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE HDK IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, USE WITH FUTURE PSION TEKLOGIX DEVICES INTRODUCED, OR OTHERWISE. YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT USE AND MODIFICATION OF THE HDK IS AT YOUR SOLE RISK AND YOU ARE RESPONSIBLE FOR INSTALLATION AND MODIFICATION OF THE HDK ON YOUR COMPUTER.

C.6 LIMITATION OF LIABILITY

Under no circumstances are we or our third party suppliers liable for damages of third parties claimed against you, or for harm to your records or data, or special, incidental, indirect, or consequential damages, including but not limited to lost profits, lost business revenue or failure to realize expected savings, loss of data, loss of use of the HDK or any associated equipment, downtime and user's time, even if you informed us of their possibility, or for breach of any express or implied warranty, breach of contract, negligence, strict liability or any other legal theory related to the HDK. This limitation applies whether you are entitled to claim damages from us or our third party suppliers as a matter of contract or tort.

C.7 COPYRIGHTS, OWNERSHIP AND PROPRIETARY RIGHTS

All title and copyrights in and to the HDK, and any copies thereof, are owned by Psion Teklogix Inc. or its suppliers. The HDK also contains copyrighted material licensed from our suppliers and all rights to such copyrighted material rests with such suppliers. We retain title to the HDK and any copies made from it. Any copies of the HDK you made are subject to the restrictions of this Agreement.

WE DISCLAIM ALL WARRANTIES AND INDEMNITIES, EXPRESS, IMPLIED OR STATUTORY, FOR PATENT OR COPYRIGHT INFRINGEMENT.

C.8 CONFIDENTIALITY

You agree not to use or disclose any proprietary information provided by us, except for the purposes of this Agreement. You agree not to reproduce any of the copyrighted materials unless expressly permitted by this Agreement.

C.9 ENDING THIS AGREEMENT

We may terminate this Agreement and your license immediately without notice if (a) you fail to comply with any term of this Agreement, or (b) your rights are assigned by you, by operation of law or otherwise. In such event, you must return or destroy all copies and component parts of the HDK and documentation, as well as any other Psion Teklogix proprietary information in your possession, within fourteen (14) days of the date of termination. Any rights and obligations under this Agreement that by their nature continue after it ends, will remain in effect until they are completed.

C.10 GENERAL

The laws of the Province of Ontario and the federal laws applicable therein, excluding the conflict of laws provisions, govern this Agreement. If any provision of this Agreement is deemed invalid or unenforceable by any country, that particular provision will be deemed modified to the extent necessary to make the provision valid and enforceable, and the remaining provisions will remain in full force and effect. Failure by us to insist on strict performance or to exercise a right when entitled, does not prevent us from doing so at a later time, either in relation to that default or any subsequent one.

No modifications of this Agreement shall be effective unless in writing and approved by us.

You acknowledge that you have read this Agreement, understand it, and that it is the complete agreement between you and Psion Teklogix with respect to the subject matter hereof and supersedes all prior agreements, oral or written.

INDEX

Symbols

Numbers

A

B

C

D

E

F

G

H

I

K

L

T

U

V

W

X