CN8478/CN8474A/ CN8472A/CN8471A

Evaluation Module Users Guide

Preliminary Review 8/16/99 1600

N8478UG1A
August 16, 1999

CONEXANT™

Reader Response: To improve the quality of our publications, we welcome your feedback. Please send comments or suggestions via e-mail to Conexant Reader Response@conexant.com. Sorry, we can't answer your technical questions at this address. Please contact your local Conexant sales office or local field applications engineer if you have technical questions.

Table of Contents

List of Figures

List of Tables

1.0 Product Overview

1.1 Introduction

This document describes the requirements and plan to build an Evaluation Module (EVM) for CN8478/CN8474A/CN8472A/CN8471A (MUSYCC). This document captures EVM requirements from marketing and engineering for the specification, design, software development, integration, and production phases.

As a marketing collateral, the EVM is sold or otherwise made available to customers for the purposes of demonstrating features and accelerating customers' time to market.

The CN8478 EVM is a demonstration and test platform that allows system developers and product designers to test and evaluate the functionality and performance of the CN8478 chip.

The CN8478 EVM is an eight-channel synchronous communication module with T1 and E1 interfaces. Each physical line is supported by the Bt8370 T1/E1 transceiver, which provides support for data link maintenance, and one High-Level Data Link Control (HDLC) controller supported by the CN8478, which formats up to 256 HDLC channels.

The CN8478 EVM software is implemented under the Windows NT 4.0 operating system as a Network Driver Interface Specification (NDIS) MiniPort driver. The WinNT driver code is available with the CN8478 EVM product.

The CN8478 EVM Toolbox is divided into three levels: physical level, logical level, and application level. It is used to perform several functions:

- Loopback testing
- Reading channel statistics
- Reading PCI configurations
- Configuring dynamic hyperchanneling
- Reading from and writing to device registers
- Reading from and writing to shared memory
- Viewing register map
- Performance monitoring

1.2 Reference Documents

These related documents are available from Conexant, Network Access Products Division:

- CN8478/CN8474A/CN8472A/CN8471A—*Multichannel Synchronous Communications Controller (MUSYCC)*
- *Bt8370—Fully Integrated T1/E1 Framer and Line Interface*

Other documents:

- *Microsoft DDK/NDIS Drivers*

1.3 Online Documentation

Technical product documentation for the CN8478 and Bt8370 is available from Conexant Information website.

1. Go to http://www.conexant.com/techinfo.
2. After registering, select Network Access.
3. On the Network Access page, select the device number.

Alternative component selection and specification for the bill of materials are also provided in Portable Document Format (PDF) in the same directory.

2.0 Installation

2.1 Contents and Requirements

The following items are included in your packing list:

- CN8478 Evaluation Module (EVM) board
- CN8478 EVM CD software
- CN8478/CN8474A/CN8472A/CN8471A datasheet
- Bt8370 datasheet
- *CN8478 Evaluation Module User's Guide*

To use the CN8478 EVM, you need the following system:

- IBM PC or compatible with 32 MB RAM memory and a controller card (PCI 32 bit, 2.0 or higher)
- Windows NT server 4.0 operating system
- A protocol analyzer (optional)
- Your telecom equipment to be tested

To use the online help, you need the following software:

- Acrobat Reader 3.0

2.2 Board Installation

To install the board, follow these steps:

1. Power down the computer and follow the manufacturer's directions to remove the cover from your computer.
2. Locate an empty PCI bus slot and plug in the CN8478 EVM board.
3. Plug the protocol analyzer into a T1/E1 port; otherwise, leave the T1/E1 ports disconnected.
4. Follow the manufacturer's directions to replace the cover on your computer and power up.

2.3 Software Installation

Copy the software for the driver and the Toolbox from the supplied CD onto the hard drive.

2.3.1 Installing the CN8478 EVM Toolbox

Run `Setup.exe` from the CD path: `projects/EVM847x\setup.exe`. The installation batch file adds the CN8478 EVM Toolbox to the Program menu.

2.3.2 Uninstalling the CN8478 EVM Toolbox

To uninstall CN8478 EVM Toolbox, run Add/Remove Programs in the Control Panel window and select CN8478 EVM Toolbox. This removes all files and information from the registry entries.

2.3.3 Driver Setup (OEMSETNT.INF, ROKWAN.SYS)

Each NDIS driver must provide a setup file, called an OEMSETNT.INF script, containing information about how the driver is loaded and its relationships to other network drivers. During system setup, the network is configured on the machine, and the setup file for each network component writes to the registry. This describes how these components fit within the set of installed network drivers that are bound into the network stack. The new network driver is the Conexant WAN Adapter, `ROCKWAN.SYS`. To install the driver, perform the following steps:

1. From the Start menu, choose Settings and Control Panel.
2. From the Control Panel, double click the Network icon.
3. From the Network window, choose Adapters.
4. In the Adapters screen, click Add. This opens a window with all Network Adapters in the system. Click HaveDisk.
5. Type the path: `projects/Windows Driver/wmp-nt40`, and click OK. This path contains the driver file (i.e. OEMSETNT.INF, ROKWAN.SYS). If the path is incorrectly typed, this error message is received:

   ```
   "Setup Message: Setup cannot find OEMSETUP.INF or
   OSMSETNT.INF. Please type a new path to the OEMSETUP.INF
   file. "
   ```

6. The next window is Select OEM Option window. Choose Conexant WAN Adapter. Click OK.
7. The next window is the Conexant WAN Miniport Driver Bus Location screen. Select the PCI Bus Type. Select the Bus Number where the PCI bus is located. Click OK. The following message is displayed:

   ```
   "Rockwell WAN setup is completed.
   ```

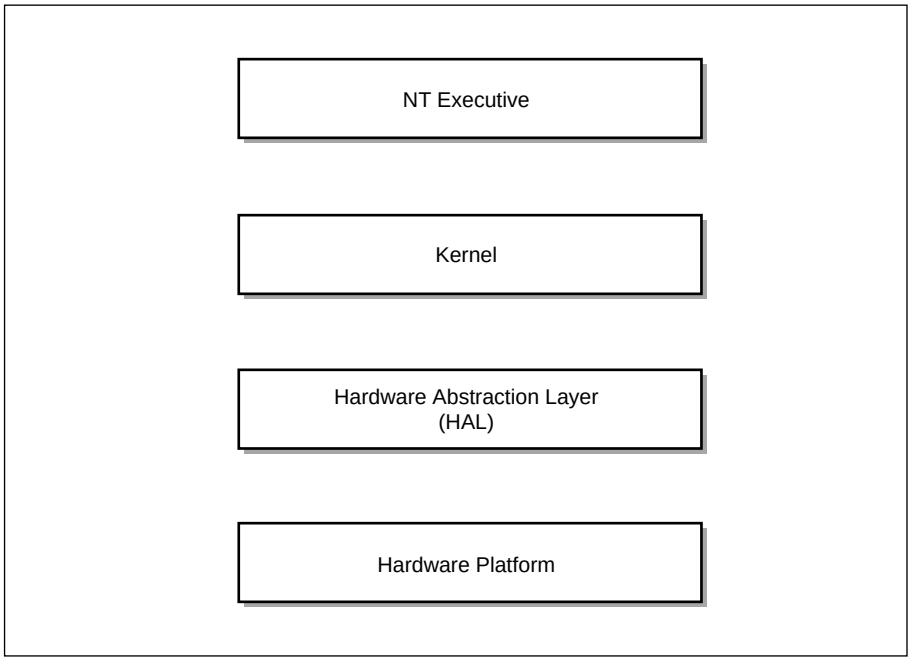
8. Click OK.
9. To set up the bindings of the configuration, click the Close button from Network windows.
10. To complete installation, shut down and restart your computer.

2.4 Windows NT Architectural Model

The generic architecture of the Windows NT CN8478 EVM is defined as a modular system.

Figure 2-1 illustrates the Windows NT architecture as a modular system.

Figure 2-1. Windows NT Architecture

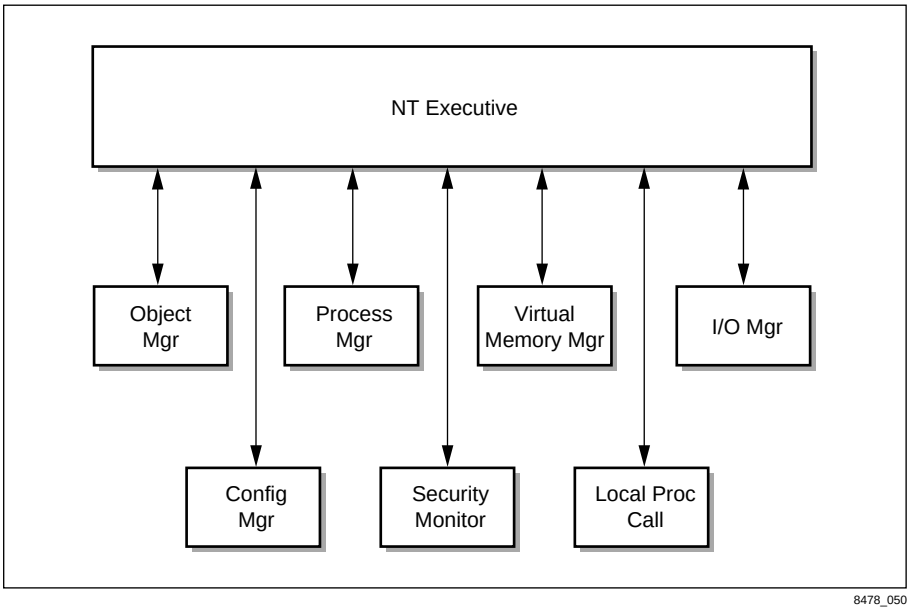


Hardware platform = PCI card and MIC/WAH adaptor, which contains the CN8478 part.

Where:

- Hardware Abstraction Layer (HAL) virtualizes hardware interfaces.
- Kernel manages the basic operation of Windows NT. The activities the kernel schedules are called threads, the most basic entity in the system that can be scheduled. Threads are defined in the context of the process. A process is defined as an address space, a set of objects visible to the process, and a set of threads that run in the context of the process.
- NT Executive system services interface consists of several distinct software components that offer their services both to user-mode process and to one another. These Executive components are independent and communicate through well-defined interfaces as illustrated in Figure 2-2.

Figure 2-2. NT Executive Components



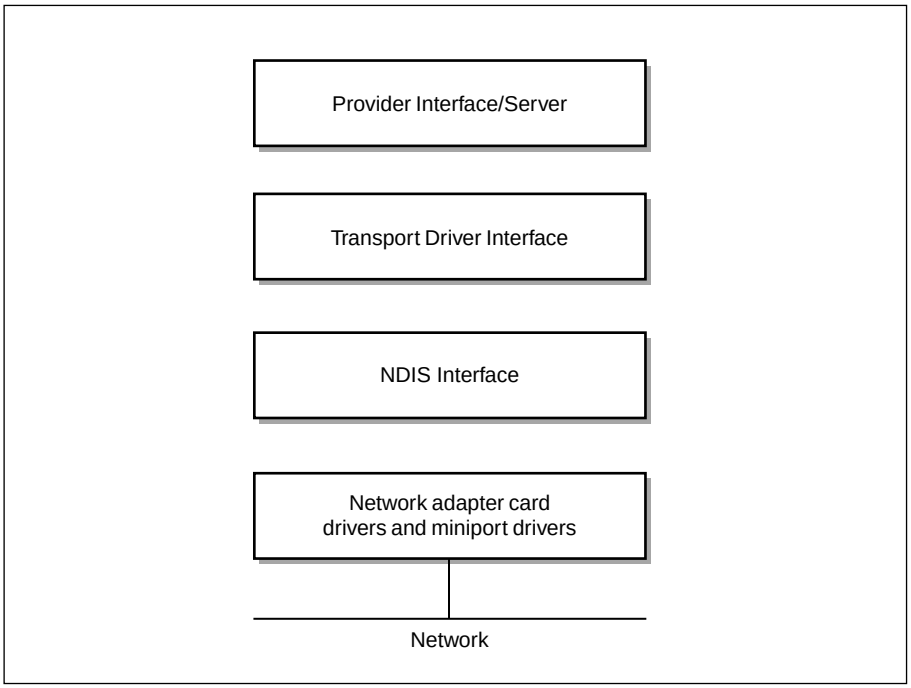
The Backdoor mechanism (messages) communicates to the I/O manager in order to query and get data.

The I/O manager converts the I/O request from user and kernel-mode threads into properly sequenced calls to the NIC driver routine. Through the backdoor mechanism, an interface is defined so all device drivers can communicate with the application (GUI) in the same way. Therefore, the application need not know any specific device drivers' APIs that will be called in the device-specific part of NIC driver.

2.4.1 Network Driver

The CN8500 EVM NIC driver is a network device driver, implemented as a component in the I/O architecture. Windows NT includes integrated networking capabilities and support for distributed applications. The networking level is supported by a series of network drivers, illustrated in Figure 2-3.

Figure 2-3. Networking Level



2.4.2 Windows NT Registry Entry

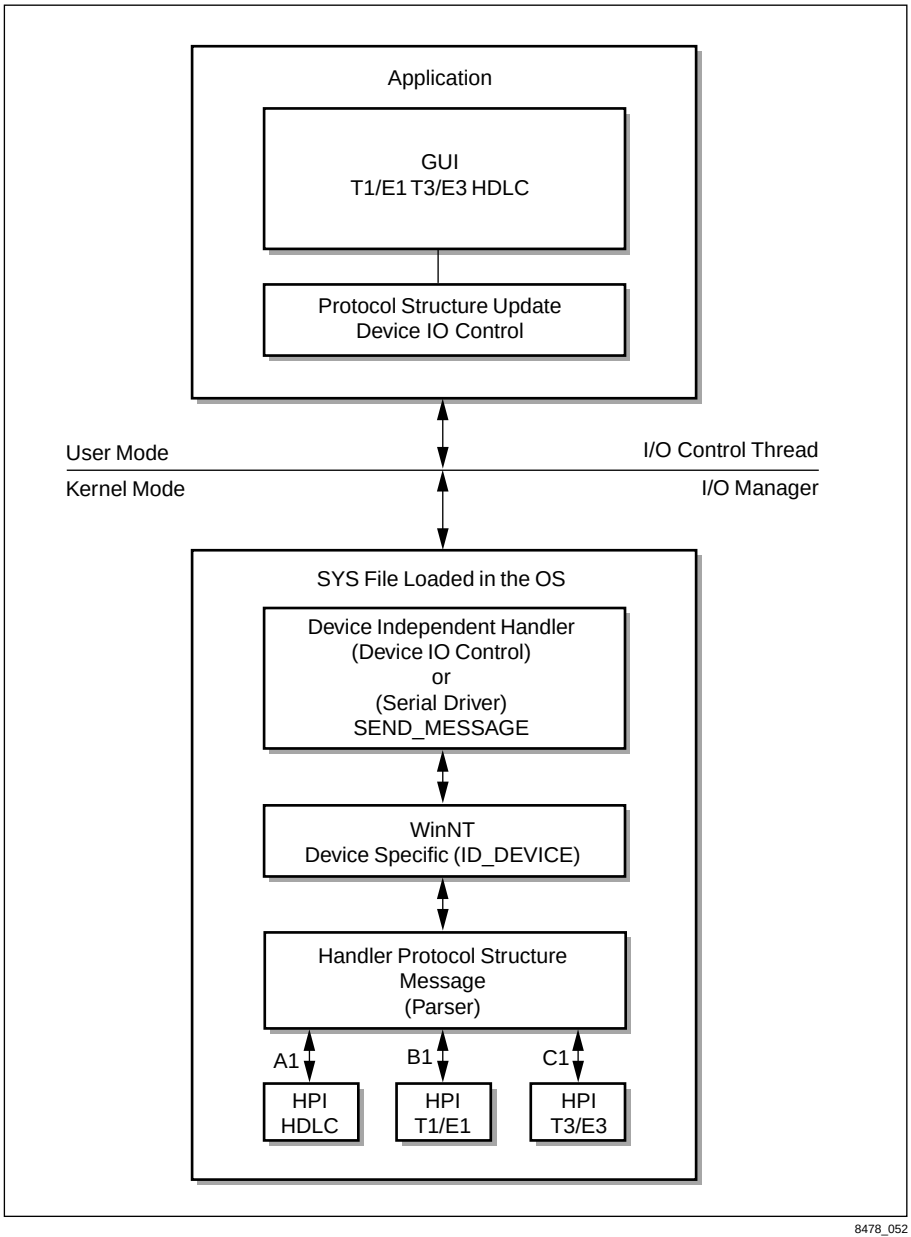
- Setup—loads new configuration data to the Registry.
- Recognizer—places hardware configuration in your Registry when you start a computer running Windows NT.
- Windows NT Kernel—extracts information from the Registry (such as which device drivers to load and their load order) during the system startup.
- Device drivers—send and receive load parameters and configuration data from the Registry.
- A device driver must report system resources it uses, such as hardware interrupts and DMA channels, so the system can add this information to the Registry. Application and device drivers can read this Registry information to provide users with smart installation and configuration programs.
- Administrative tools—can be used to modify configuration data.
- Agenda—
- Process—an address space, set of objects (resources) visible to the process, and a set of threads that run in the context of the process. A thread is most the basic schedulable entity in the system. The process has its own set of registers, its own kernel stack, a thread environment block, and user stack in the address space of its process.
- Kernel—schedules threads to be executed. The threads are defined in the context of the process which defines an address space, a set of objects visible to the process, and a set of threads that runs in the context of the process. The kernel manages two types of objects:

1. Dispatcher objects have a signal state, which includes the following: events, mutants, semaphores, threads and timers.
2. Control objects control the operation of the kernel.

2.4.3 Software Block Diagram

The diagram in Figure 2-4 indicates the basic software blocks .

Figure 2-4. Software Block Diagram



2.4.4 Protocol Structure

2.4.4.1 Backdoor Structure

The Backdoor structure is a message protocol communication between the driver and the GUI.

For WinNT, this structure is passed to the DeviceIOControl which passes the control to the Service Access Point (SAP) which is the WinNT Device-specific function.

The implementation of the Backdoor structure is as follows:

```
typedef struct tBackDoor {
   DWORD KeyCode; / * per device*
   DWORD ReqCode;  / *request Code*/
   DWORD ReqSubCode; /* subrequest Code*/
   DWORD TotalSize; /* backdoor total structure *./
   DWORD NeededSize;
   DWORD ResultCode;

   DWORD Param1;
   DWORD Param2;
   DWORD Param3;
   DWORD Param4;
   DWORD Param5;
   DWORD Param6;

   DWORD vParam1Size;        //v for variable
   DWORD vParam1Offset;
   DWORD vParam2Size;
   DWORD vParam2Offset;
   DWORD vParam3Size;
   DWORD vParam3Offset;
   DWORD vParam4Size;
   DWORD vParam4Offset;

   BYTE data[4];
}BACK_DOOR, *PBACK_DOOR;
```

2.4.5 Generic Queries

READ_SHARED_MEMORY

READ_SHARED_MEMORY_LOCATION

2.4.5.1 T1/E1 Device Driver

The Backdoor structure in order to provide protocol message communication between the Application Level and the Device Driver.

2.4.5.2 T1/E1 HPI Primitive Functions

System and backdoor initialization
Pbd->parm1 = port_number (group number)
Pbd->Reqcode = BACKDOOR _READ
Pbd->ReqSubCode = BACKDOR_READ_XXXX_REGITER_MAP
Pbd->NeedSize = sizeof(BACKDOOR) + sizeof(GRP_REG_MAP)
Pdb->TotolSize = sizeof(BACKDOOR) + sizeof(GRP_REG_MAP)
Pdb->vParm1Offset = sizeof(BACKDOOR)
Pdb->vParma1Size = sizeof(GRP_REG_MAP)

Note:
If the image will be kept in local memory, no call to the HPI is performed; otherwise,

```
NdisMoveMemory(Pdb->parm1Offset,
        &Reg_local_image(device_id), pdb-vParm1Size);
```

Aa call the HPI .

2.5 Api_t1e1ReadRegisterMap()

Description This function calls the HPI T1/E1 to provide the T1/E1 register map.

Synopsis
```
STATUS  Api_t1e1ReadRegisterMap
    (
    DEVADDR   deviceAddr
    Void      *
    )
```

Input Parameters deviceAddr—specifies the device's base address as seen by the MPU.

Output Parameters deviceId— identifier used in subsequent device HPI primitives.

Body Call the adequate HPI.

Returns
- SUCCESS
- ERR_code

Pbd->parm1 = port_number (group number)
Pbd->Reqcode = BACKDOOR _READ
Pbd->ReqSubCode= BACKDOR_READ_XXXX_REGITER_MAP
Pbd->NeedSize=sizeof(BACKDOOR) + sizeof(GRP_REG_MAP)
Pdb-TotolSize = sizeof(BACKDOOR) + sizeof(GRP_REG_MAP)
Pdb->vParm1Offset = sizeof(BACKDOOR)
Pdb->vParma1Size = sizeof(Bt83xx_device)

2.5.1 Api_t1e1Configuration()

Description This function calls the specific device drive API in order to pro-

vide the device driver configuration..

Synopsis STATUS Api_t1e1Configuration

```
(
DEVADDRdeviceAddr
Void *
)
```

Input Parameters deviceAddr—This value specifies the device's base address as seen by the MPU.

Output Parameters **deviceId**—The device identifier used in subsequent device HPI primitives.

Body: Call the adequate HPI

Returns

- SUCCESS

Example:
Call T1/E1 HPI to provide the configuration.

2.5.2 Api_ReadDeviceStatistics

2.5.3 Api_read_device_register()

Description This function calls the specific device drive API in order to provide the device driver configuration..

Synopsis STATUS Api_t1e1Configuration

```
(
DEVADDRdeviceAddr
Void *
)
```

Input Parameters deviceAddr—specifies the device's base address as seen by the MPU.

Output Parameters deviceId – The device identifier used in subsequent device HPI primitives.

Body: Call the adequate HPI

Returns

- SUCCESS

Example:
Call T1/E1 HPI to provide the configuration.

```
Init
BACKDOOR_ADAPTER_INIT
```

```
Init_device_specific_mode(E1,T1)
Init_device_diagnostics

Action_start
Start_hdlc_loopback
Reset_global
Reset_local
start_TX_pulse_template
start_loopback_option
start_TX_isolated_pulse
start_TX_all_ones

Action_stop
stop_hdlc_loopback
stop_TX_pulse_template
stop_loopback_option
stop_TX_isolated_pulse
stop_TX_all_ones

Action_configure
Configure_device
Configure_hdlc_loopback

Action_write
Write_device_register
Write_shared_memory_location

Genereal
Api_readPciConfiguration
Api_readSharedMemory
Api_readRegistryConfiguration

t1e1InitDev()
```

Description — This function initializes a device's registers to an initial, default state as specified by the t1e1 register configuration structure. All registers configured by other HPI primitives will be overwritten. This function is typically called only once, immediately after t1e1RegisterDev().

Synopsis

```
STATUS  t1e1InitDev
  (
```

```
DEVIDdeviceId
tRegisterConfigStruct*pRegisterConfig
)
```

Input Parameters — deviceId – The device identifier.
*pRegisterConfig – Pointer to a device configuration structure. This structure contains values for all of the device's configuration and control registers. Registers not included are those which are updated by the device.

Output Parameters — None

Returns

- SUCCESS
- ERROR

HDLC

HAL

The HAL will inculde services such as read and write for the EBUS.
Read_EBUS ();
Write_EBUS();

Interrupt Handler

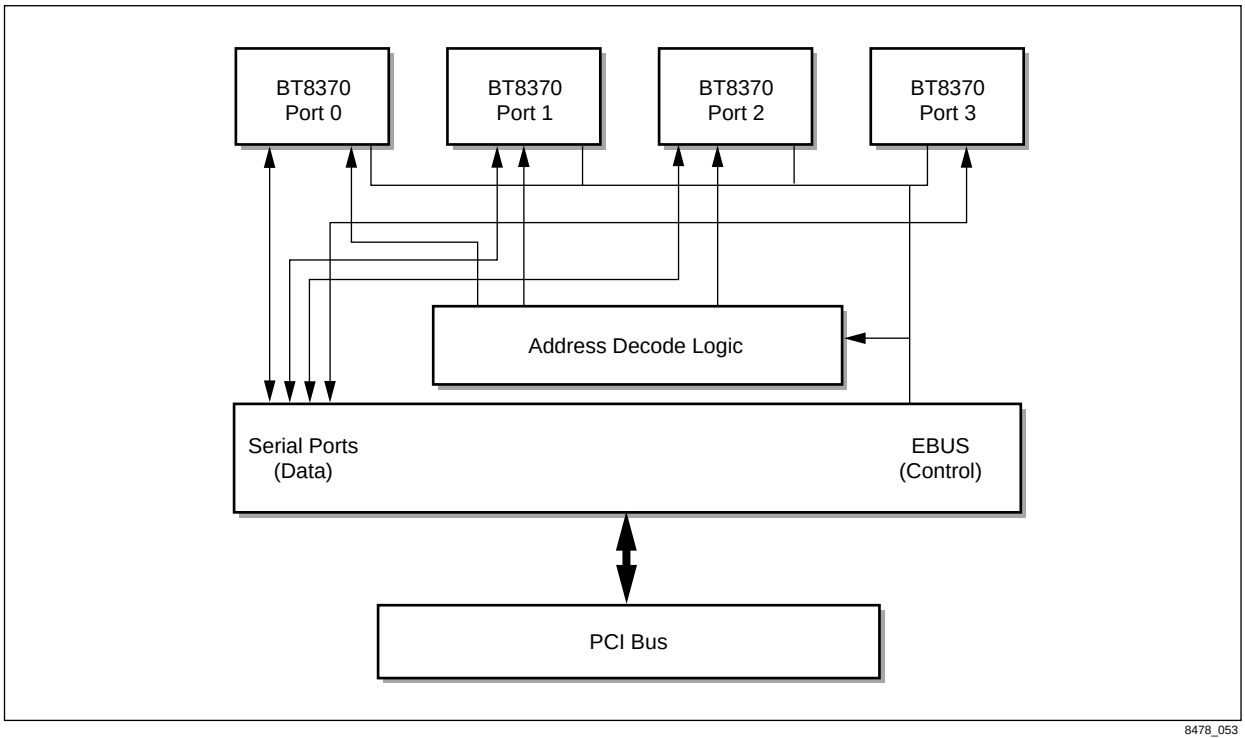
Interrupt Handler per device

3.0 Hardware Description

3.1 Block Diagram

Figure 3-1 illustrates a block diagram of the evaluation board. It includes four Bt8370 circuits, receive and transmit connected (T1 or E1) lines interface, a serial bus used to transfer data between these circuits, and the Bt8474 (MUSYCC). The data is transferred between the Bt8474 and the host through the Peripheral Component Interconnect (PCI) bus. The Bt8370 circuits are configured and controlled using the Expansion Bus (EBUS).

Figure 3-1. Bt8474 Block Diagram

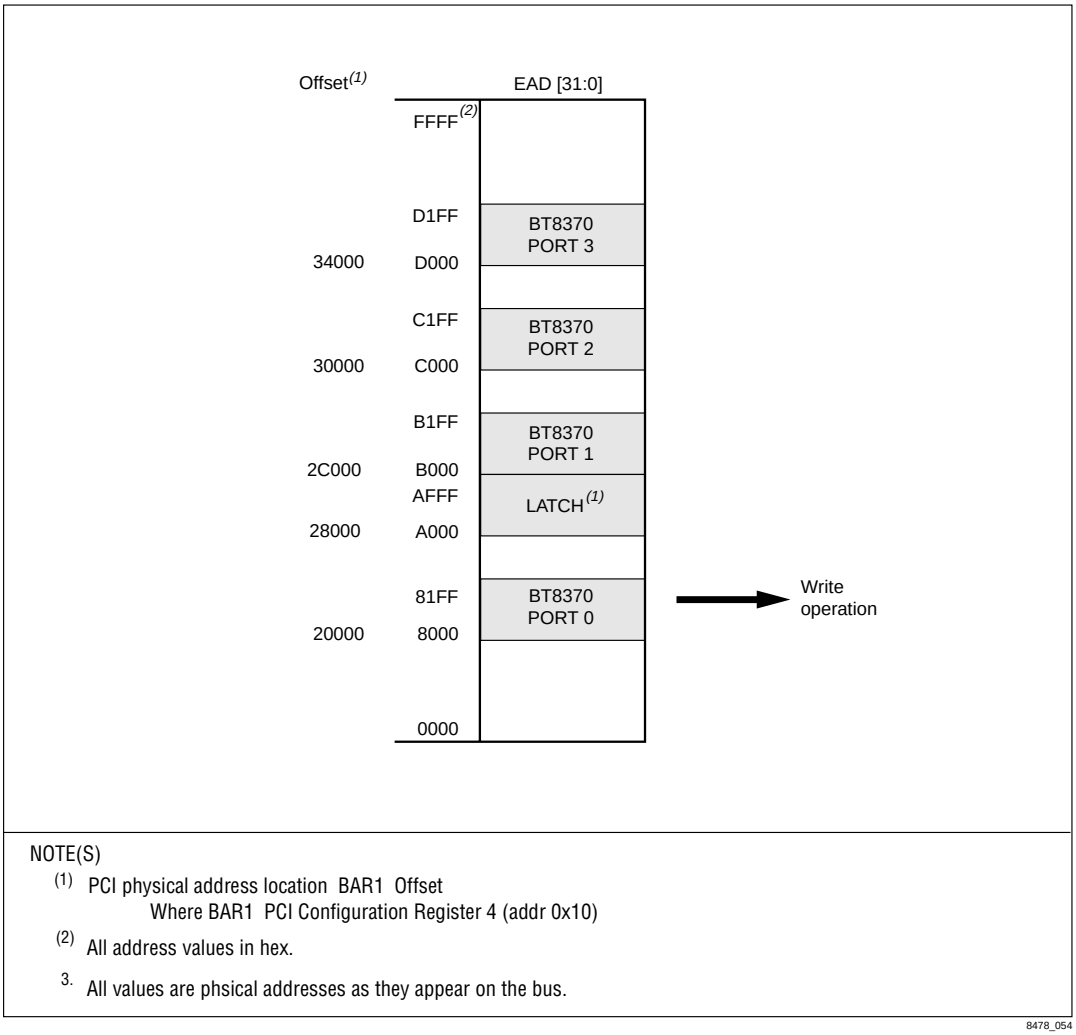


3.2 Expansion Bus

The expansion bus, mastered by the Bt8474, controls and configures the four Bt8370 chips. It is configured as an Intel-style bus (refer to the Bt8474 datasheet for details), and uses only the sixteen lower address/data lines of the 32 bits available on the Bt8474. However, even though 16-bit data transfer can be achieved in this configuration, all EBUS data access will be 8 bits wide because the connected devices (the four Bt8370 chips and the latch register) support only this format.

The address decoding logic is centered on U2 and U3, which generate all necessary chip select signals for the different devices connected to the bus. Figure 3-2 illustrates the resulting memory map of the E-bus.

Figure 3-2. Expansion Bus Memory Map



The latch (U4) controls some hardware signals on the board. Although the latch address ranges from A000 to AFFF (hex), only one physical location can be accessed with any of these. This latch cannot be read, and only the following bits are used (numbering from 0–7):

Bit 1—When reset, disables interrupt A on the Bt8474 (INTA# line disconnected from PCI bus).

Bit 2—When reset, disables interrupt B on the Bt8474.

Bit 4—When reset, activates the reset signal on all four Bt8370 devices. This allows the software driver to perform a hardware reset whenever necessary.

Upon power–up, all these signals are reset (all active); therefore, all Bt8370 circuits will remain in the reset state until the software sets Bit 4.

3.3 PCI Bus

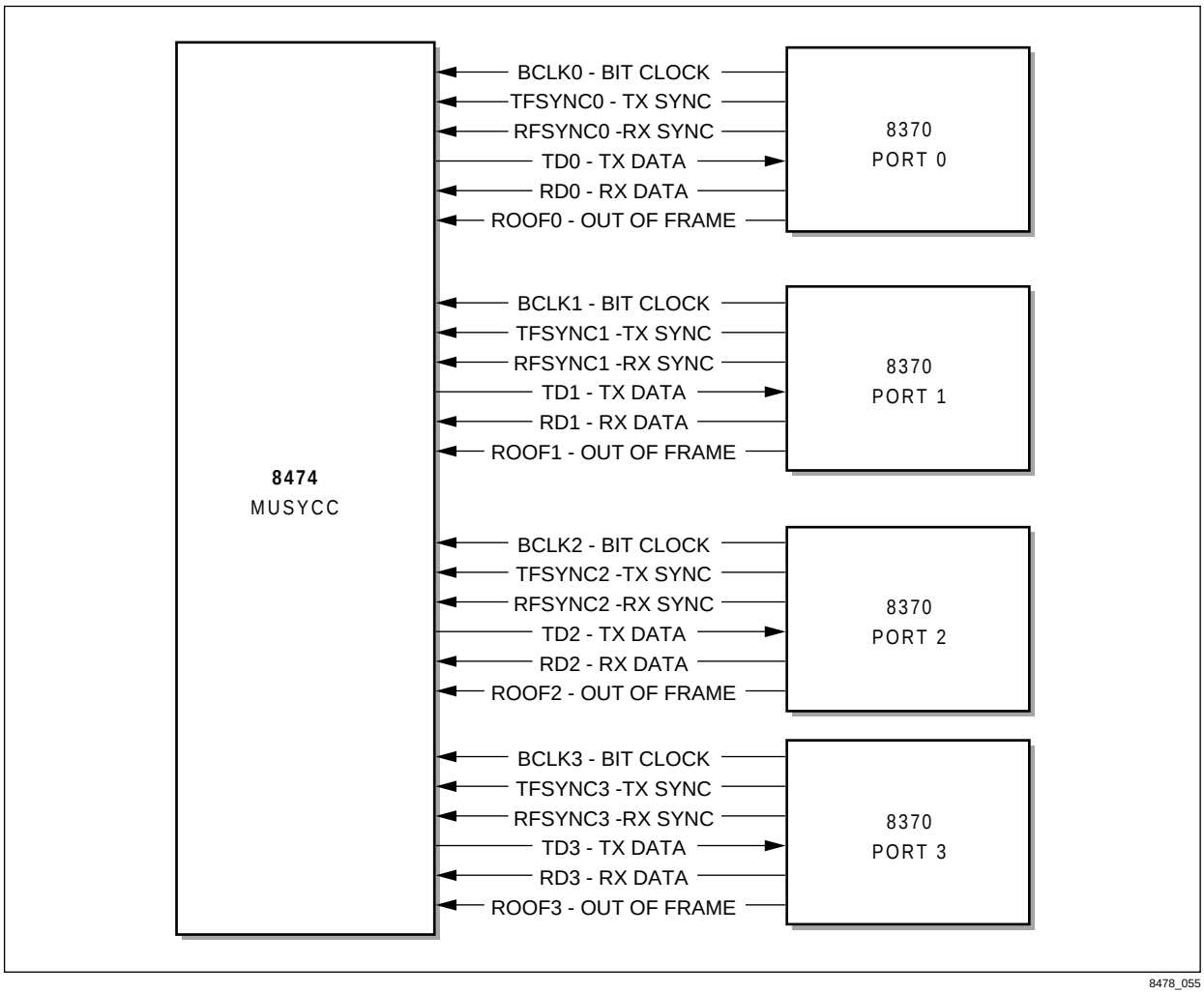
The PCI interface on the Bt8474 is connected to a 5 V, 32-bit PCI bus. The two interrupt lines, INTA* and INTB*, can be disabled by software, if necessary, by setting to zero the appropriate bits in the previously described latch.

The JTAG port is also wired to the connector for testing. The optional J2 header allows connection to the GNT# signal if a PCI bus analyzer is used.

3.4 Serial Buses

The Bt8474 transfers data to and from the four Bt8370 chips, using four independent serial buses. Figure 3-3 illustrates all signals used and their direction.

Figure 3-3. Serial Bus Between MUSYCC and T1 Transceivers



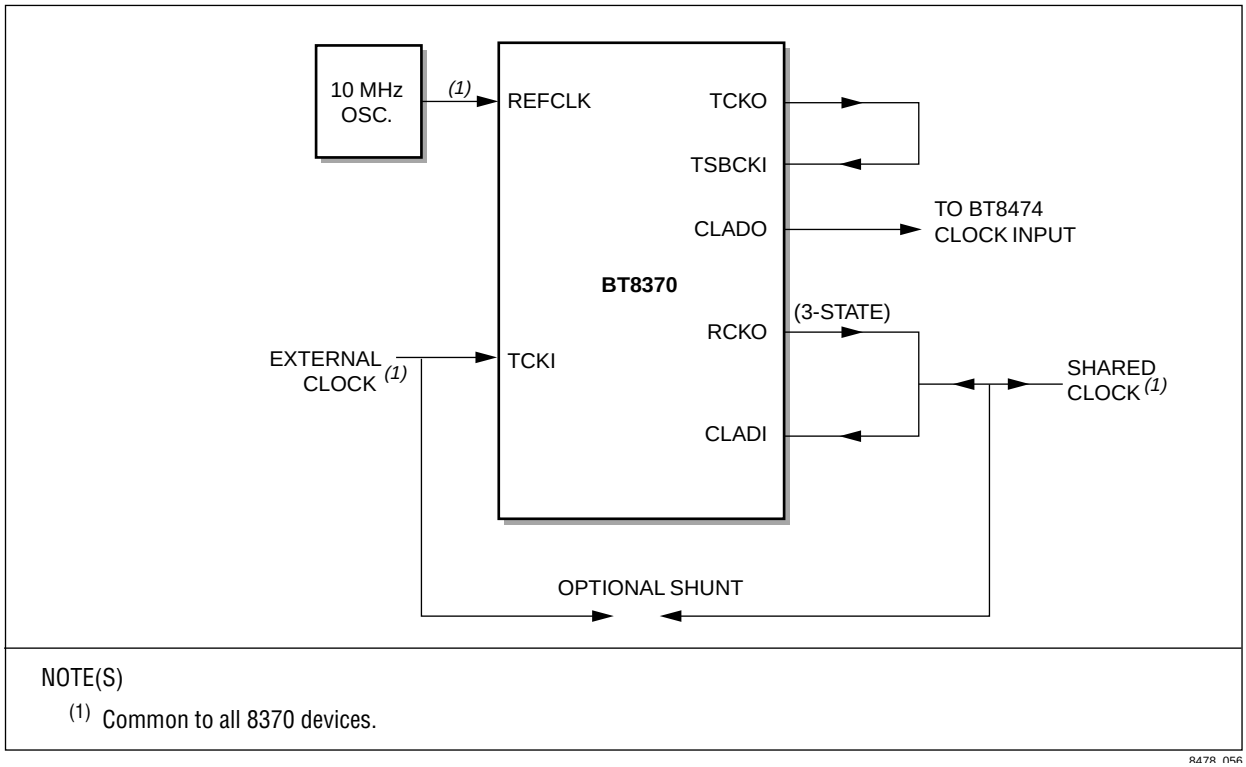
3.5 T1 / E1 Line Interface

The line interface connects the Bt8370 to the T1 or E1 line. The transformer has a 3 kV insulation to allow the board to pass tests in European countries. The line filter reduces the electromagnetic interference. The Polyswitch and P1553AB components (FSx and Zx) protect the transformer from power surges. The optional E1 jumper allows the user to change the receive termination impedance to match the T1 or E1 requirements. By default, the board does not have this header and is configured for T1 line (100 Ω). If the jumper is set, the board is configured for E1 (75 Ω).

3.6 Bt8370 Clocking Options

The Bt8370 offers several synchronization options for its serial port, allowing a flexible clocking configuration. Figure 3-4 illustrates the clock inputs and outputs wiring. See the Bt8370 data sheet for a description of the different pins and how to use them.

Figure 3-4. Bt8370 Serial Clock Generation

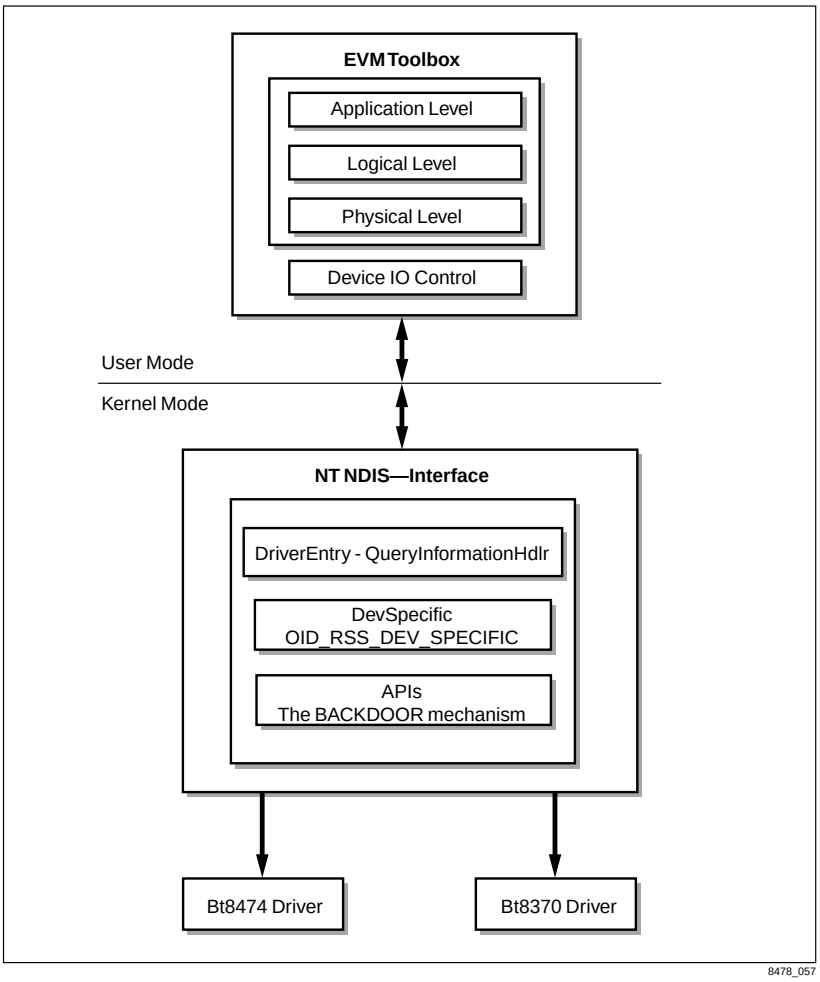


4.0 Software Architecture

4.1 Software Block Diagram

Figure 4-1 illustrates the basic software blocks in the CN8478 EVM.

Figure 4-1. Software Block Diagram



Communication between the user and kernel modes uses the DeviceIOControl function that passes the defined I/O control code to the NDIS driver, requesting the Device Specific driver.

4.1.1 NT NDIS Interface

The NT NDIS interface implements the functions required to comply with the NDIS 4.0 specification. These functions respond to query and set operations from the NDIS wrapper, and maintain proper state information for the WAN links.

In the NT kernel mode driver, the starting point is the DriverEntry function, which initializes driver data structures and prepares the environment for all other components.

The Query InformationHdlr function returns information about the capabilities and status of the driver, in this case, for a specific NIC driver handled by the CN8478 EVM Toolbox. This function calls the DevSpecific function, which is the envelope for all APIs used to configure and perform the application requests.

4.1.2 CN8478 Driver

The CN8478 driver is the set of functions used to reset, configure, and maintain the device. The CN8478 driver supports a dynamically grouped configuration:

- Channel assignments
- Protocols (Transparent, SS7-HDLC-FCS16, HDLC-FCS16, HDLC-FCS32)
- Receive and transmit message loopbacks
- Time slot assignments

Communication between the CN8478 and the host is done through the service request mechanism. The following service requests are supported by the CN8478 driver:

- Soft Chip Reset
- Global Configuration
- Channel Group Configuration
- Read Channel Configuration
- Channel Activation and Deactivation

The device must acknowledge each request to a specific group before any other service request may be issued. All buffer descriptors and packet blocks are statically allocated; however, a large amount of RAM is required. The transmit and receive mechanism is implemented by a linked circular list. The host creates a circular list of buffers for each channel, and the last buffer in the list points to the first one. All buffers in the list are initially host-owned. The host-fill process merely checks the OWNER status of the next buffer before filling. If the next buffer is host-owned, the host fills it and flips the OWNER bit to grant ownership to MUSYCC. If the next buffer is MUSYCC-owned, the host waits for MUSYCC to empty the buffer and become host-owned. At this point, the buffer can be filled.

4.1.3 Bt8370 Driver

The Bt8370 driver is the set of functions used to reset, configure, and maintain the device. Background threads monitor the current state of the device to correct any deviation from the expected performance level.

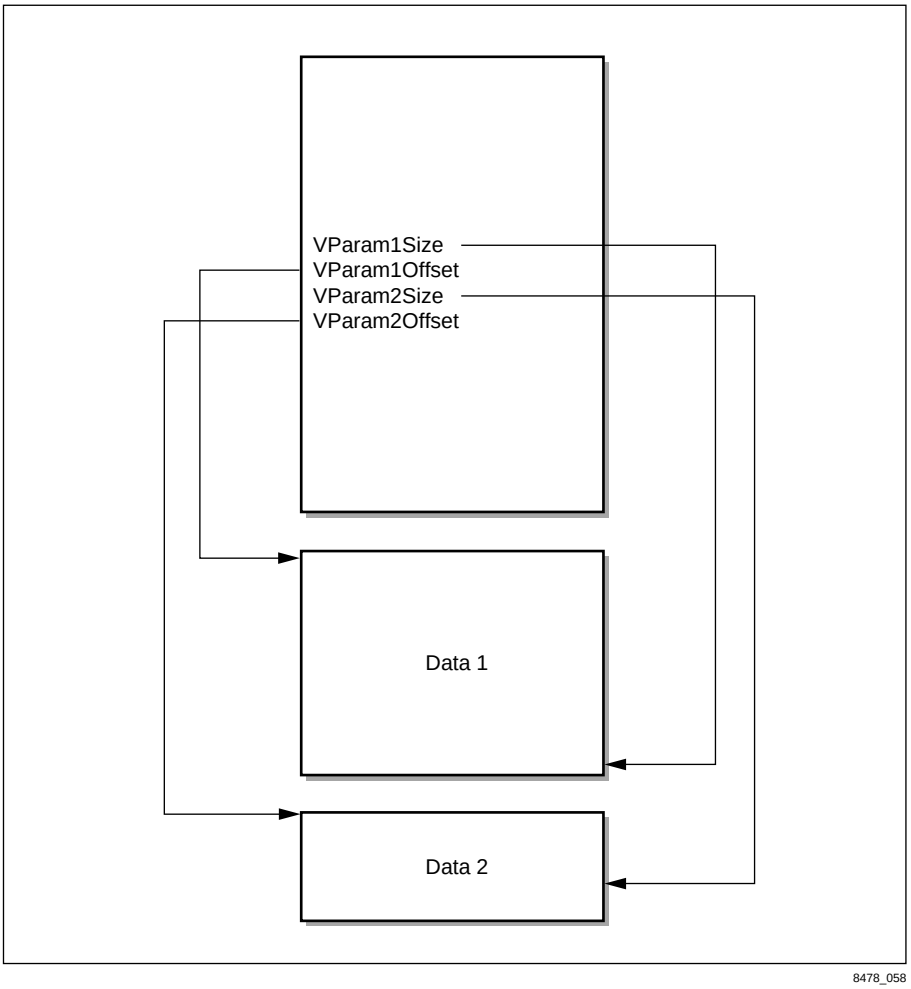
4.2 Development Tools

The development tools are Microsoft Visual C++, Professional Edition; Microsoft Visual Basic, Professional Edition; and Microsoft Development Network–DDK.

4.3 APIs—Backdoor Mechanism

You can interface with the software driver by using the Backdoor structure, submitting requests and receiving responses. This interface is called APIs—Backdoor mechanism. Figure 4-2 illustrates the Backdoor mechanism.

Figure 4-2. Backdoor Mechanism



Tables 4-1 through 4-21 list the Backdoor parameters, instances, and hex values.

4.3.1 Backdoor Read

Table 4-1. Read Shared Memory Map

Backdoor Parameters	Instances
KeyCode	ROK
ReqCode	BACKDOOR_READ
ReqSubCode	BACKDOOR_READ_SHARED_MEM
Param1	Group number
VParam1Size	Shared Memory Map Size
VParam1Offset	Size of (BACKDOOR structure)
NOTE(S): All CN8478 shared memory map values are returned in the Data 1 area.	

Table 4-2. Read Shared Memory Map

Backdoor Parameters	Instances
KeyCode	ROK
ReqCode	BACKDOOR_READ
ReqSubCode	BACKDOOR_READ_SHARED_MEM_LOCATION
Param1	Group number
Param2	RegOffset + Current Channel Number (if required)
NOTE(S): The output of the register content in shared memory is returned in Param 3.	

Table 4-3. Read Shared Memory Map

Backdoor Parameters	Instances
KeyCode	ROK
ReqCode	BACKDOOR_READ
ReqSubCode	BACKDOOR_READ_BT847x_REGISTER_MAP
Param1	Group number
VParam1Size	Shared Memory Map Size
VParam1Offset	Sizeof(BACKDOOR structure)
NOTE(S): All register map values are returned in the Data1 area.	

Table 4-4. Read Bt847x Register

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_READ	0
ReqSubCode	BACKDOOR_READ_BT847x_REGISTER	3
Param1	Group number	0...3
Param2	RegOffset + CurrentChannelNumber (if required)	0...800
NOTE(S): The WAN Miniport driver returns register value in Param3.		

Table 4-5. Read Bt847x Register

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_READ	0
ReqSubCode	BACKDOOR_READ_BT847x_REGISTER	3
Param1	Group number	0...3
Param2	RegOffset + CurrentChannelNumber (if required)	0...800
KeyCode	ROK	524F4B00
NOTE(S): All Bt8370 Registers values are returned in the Data 1 area.		

Table 4-6. Read Bt8370 Register

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_READ	0
ReqSubCode	BACKDOOR_READ_BT8370_REGISTER	7
Param1	Group number	0...3
Param2	RegOffset + CurrentChannelNumber (if required)	0...1FF
NOTE(S): The WAN Miniport driver returns register value in Param3.		

Table 4-7. Read Bt8370 Register

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_READ	0
ReqSubCode	BACKDOOR_READ_BT8370_REGISTER	7
Param1	Group number	0...3
Param2	RegOffset + CurrentChannelNumber (if required)	0...1FF
KeyCode	ROK	524F4B00
NOTE(S): The Wan Miniport driver returns register value in Param3.		

Table 4-8. Read Message Maps

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_READ	0
ReqSubCode	BACKDOOR_READ_MESSAGE_MAPS	9
Param1	Group number	0...3
VParam1Size	MSG_DESC × TX_MESSAGES_PER_CH × NUM_CHANNELS_PER_GROUP	3 × 2 × 20 = C0 (192 dec)
VParam1Offset	Size of (BACKDOOR structure)	54
VParam2Size	MSG_DESC × RX_MESSAGES_PER_CH × NUM_CHANNELS_PER_GROUP	3 × 8 × 20 = 300h (768 dec)
VParam2Offset	Size of (BACKDOOR structure) + Vparam1Size	

NOTE(S):
1. TX_MESSAGES_PER_CH is returned in Param2.
2. TX_MESSAGE_LENGTH is returned in Param3.
3. RX_MESSAGES_PER_CH is returned in Param4.
4. RX_MESSAGE_LENGTH is returned in Param5.
5. All message descriptors (buffer descriptor, data pointer, and pointer to next message descriptor) for each channel for transmitted messages are returned in the Data1 area.
6. All message descriptors (buffer descriptor, data pointer, and pointer to next message descriptor) for each channel for received messages are returned in the Data2 area.

Figure 4-3 illustrates the structure of data in shared memory. There are two transmit messages per channel, and the transmit message length is equal to 2 kB.

Figure 4-3. Structure of Data in Shared Memory

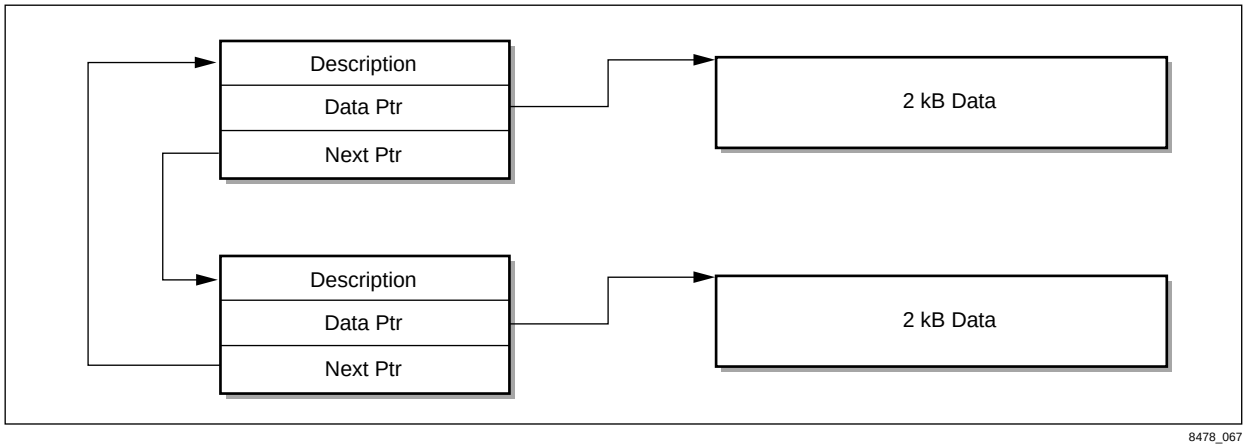


Table 4-9. Read Message Data

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_READ	0
ReqSubCode	BACKDOOR_READ_MESSAGE_DATA	10
Param1	Group number	0...3
Param2	Channel Number	0...1F
Param3	Transmit message number	0 or 1
Param4	Receive message number	0..7
Vparam1Size	TX_MESSAGES_LENGTH	800 (2048 dec)
Vparam1Offset	Size of (BACKDOOR structure)	54
Vparam2Size	RX_MESSAGES_LENGTH	800 (2048 dec)
Vparam2Offset	Size of (BACKDOOR structure) + Vparam1Size	854

NOTE(S):
1. Tx message data is returned in the Data1 area.
2. Rx message data is returned in the Data2 area.

Table 4-10. Read PCI Configuration

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_READ	0
ReqSubCode	BACKDOOR_READ_PCI_CONFIG	11
VParam1Size	—	0C
VParam1Offset	Size of (BACKDOOR structure)	54
VParam2Size	—	0C
VParam2Offset	Size of (BACKDOOR structure) + Vparam1Size	60
NOTE(S): Function 0 PCI Configuration is returned in the Data1 area.		

Table 0-1. Read Registry Configuration

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_READ	0
ReqSubCode	BACKDOOR_READ_REGISTERY_CONFIG	12
VParam1Size	COMMON_PARAMETERS + (GROUP_PARAMETERS × NUM_OF_GROUP)	8 + (10 × 4) = 40h (64 dec)
VParam1Offset	Size of (BACKDOOR structure)	54h
VParam2Size	MAX_GROUPS × NUM_CHANNELS_PER_GROUP× MAX_CALLED_NUMBER_LENGTH	4 × 20 × 28 = 1400h (5120 dec)
VParam2Offset	Size of (BACKDOOR structure) + Vparam1Size	94h
NOTE(S): The registry configuration parameters are returned in the Data1 area.		

4.3.2 Backdoor Write

Table 4-11. Write Bt8370 Register

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_WRITE	5
ReqSubCode	BACKDOOR_WRITE_BT8370_REGISTER	0
Param1	Group number	0...3
Param2	RegOffset + Current Channel Number (if required)	0...1FF

Table 4-12. Write Bt847x Register

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_WRITE	5
ReqSubCode	BACKDOOR_WRITE_BT847x_REGISTER	1
Param1	Group number	0...3
Param2	RegOffset + Current Channel Number (if required)	0...800

Table 4-13. Write Bt847x Register Map

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_WRITE	5
ReqSubCode	BACKDOOR_WRITE_SHARED_MEM_LOCATION	2
Param1	Group number	0...3
Param2	RegOffset + Current Channel Number (if required)	0...800

4.3.3 Backdoor Initialization

Table 4-14. Card Initialization

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_INIT	1
ReqSubCode	BACKDOOR_INIT_ADAPTER	0

Table 4-15. Bt847x Initialization

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_INIT	1
ReqSubCode	BACKDOOR_INIT_BT847x	1

4.3.4 Backdoor Action Start

Table 4-16. Bt847x Loopback Without Self Test

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_ACTION_START	2
ReqSubCode	BACKDOOR_ACTION_START_BT847X_ LOOPBACK	0
Param1	Group number	0...3
Param2	Channel number	0...1F

Table 0-2. Bt847x Loopback With Self Test

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_ACTION_START	2
ReqSubCode	BACKDOOR_ACTION_START_BT847X_ LOOPBACKTEST	1
Param1	Group number	0...3
Param2	Channel number	0...1F

4.3.5 Backdoor Action Stop

Table 4-17. Stop Bt847x Loopback Test

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_ACTION_STOP	3
ReqSubCode	BACKDOOR_ACTION_STOP_BT847X_ LOOPBACK	0
Param1	Group number	0...3
Param2	Channel number or FFFFFFFFh (for all channels)	0...1F

4.3.6 Backdoor Action Configuration

Table 4-18. Configuration of Bt847x Time Slot Assignments and Channel Protocols

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_ACTION_CONFIGURE	4
ReqSubCode	BACKDOOR_ACTION_CONFIGURE_BT847x	4
Param1	Group number	0...3
VParam1Size	Max number of time slots × Number of groups	80h × 4 = 200h (512)
VParam1Offset	Size of (BACKDOOR structure)	54h
VParam2Size	(Not used)	100h × 4 = 400h (1024)
VParam2Offset	Size of (BACKDOOR structure) + vParam1Size	54h +200h = 254h
VParam3Size	Number of channels per group × Number of groups	20h × 4 = 80h (128)
VParam3Offset	Size of (BACKDOOR structure) + vParam1Size + vParam2Size	54h +200h + 400h = 654h
Data 1	Data1[0] = Channel assigned to time slot 0	Data1[1] = Channel assigned to time slot 1
—	Data1[127] = Channel assigned to time slot 127	0...1F
0..1F	...	0...1F
Data2	(not used)	—
Data3	Data3[0] = Protocol assigned to Channel 0	Data3[1] = Protocol assigned to Channel 1
—	Data3[31] = Protocol assigned to Channel 31	0...3
0...3	..	0...3

Table 4-19. Configure Loopback Message

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_ACTION_CONFIGURE	4
ReqSubCode	BACKDOOR_ACTION_CONFIGURE_LOOPBACKMESSAGE	9
Param1	Loopback message	0…Quick brown fox
1…0x00 pattern	2…0x55 pattern	3…0xAA pattern
4…0xFF pattern	5…User define pattern	0...5
Param2 (optional)	User define pattern value	0...FF

4.3.7 Backdoor Loop

Table 4-20. Run Bt8370 Loopback

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_LOOP	6
ReqSubCode	BACKDOOR_LOOP_RUN	2
Param1	Group number	0...3
Param2	0 or channel number for local or remote loop per channel	0 or 0...1F
Param3	Loopback type: 2…FRAMER_LOOPBACK 3…PERCHANNEL_LOCAL_LOOPBACK 4…PERCHANNEL_REMOTE_LOOPBACK	2...4

Table 4-21. Stop Bt8370 Loopback

Backdoor Parameters	Instances	Hex Value
KeyCode	ROK	524F4B00
ReqCode	BACKDOOR_LOOP	6
ReqSubCode	BACKDOOR_LOOP_STOP	2
Param1	Group number	0...3
Param2	0 or channel number for local or remote loop per channel	0 or 0...1F
Param3	Loopback type: 2…FRAMER_LOOPBACK 3…PERCHANNEL_LOCAL_LOOPBACK 4…PERCHANNEL_REMOTE_LOOPBACK	2...4

5.0 Using the CN8478 EVM Toolbox

This chapter describes how to use the CN8478 EVM Toolbox for Windows NT.

5.1 Starting the CN8478 EVM Toolbox

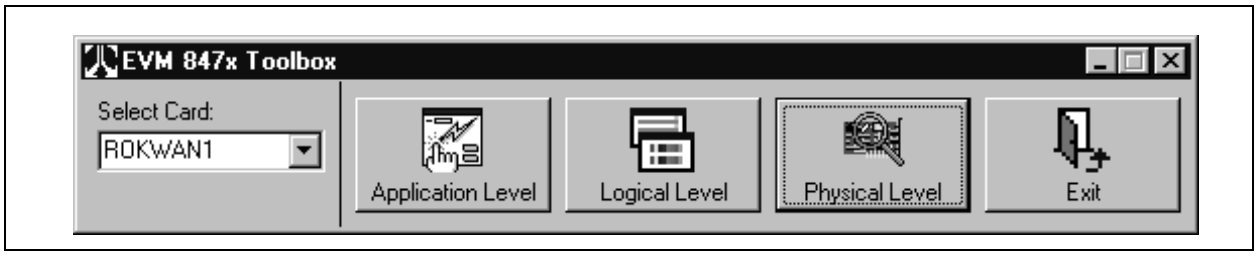
The software installation process adds the CN8478 EVM Toolbox to the Programs menu.

To start the CN8478 EVM Toolbox, perform these steps:

1. From the Start menu, choose Programs.
2. From the Program menu, click on CN8478 EVM Toolbox.

The main menu appears, as illustrated in Figure 5-1. The main menu is always displayed so you can move from one level to another. All windows can be open at the same time.

Figure 5-1. Main Menu of the CN8478 EVM Toolbox



5.2 Main Menu

The CN8478 EVM can be configured at three levels:

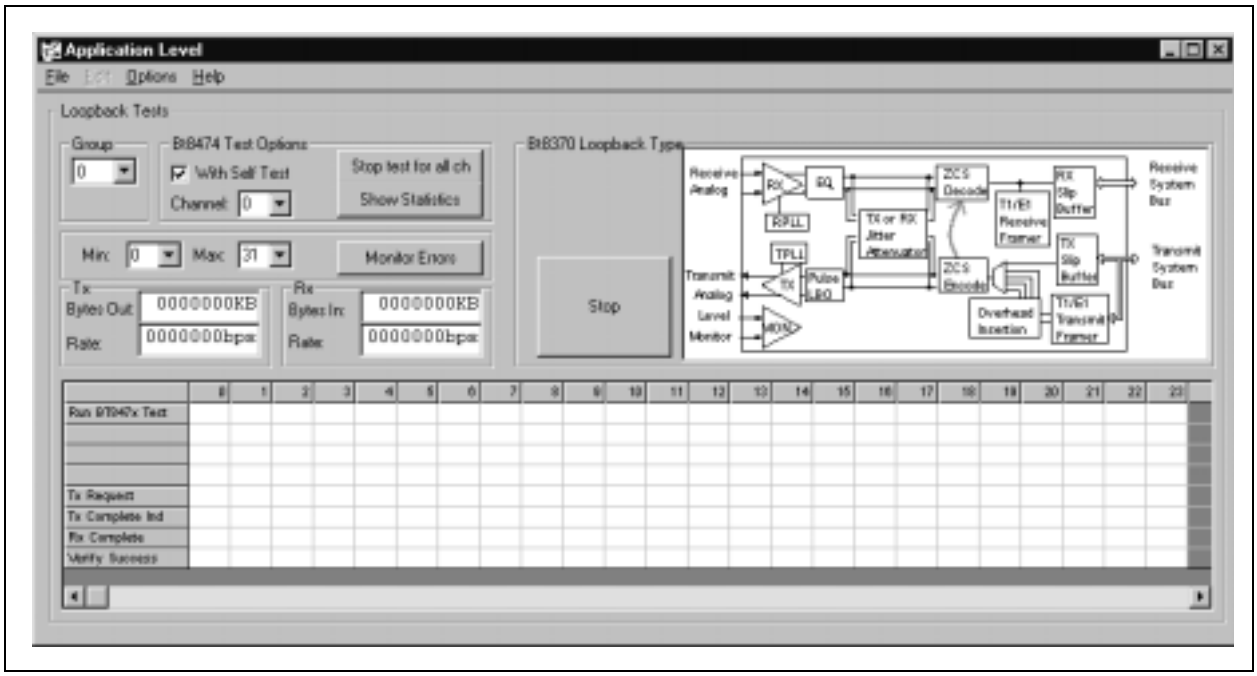
1. Application level—performs testing and maintenance.
2. Logical level—allows channel assignments and time slot monitoring.
3. Physical level—is a component register editor.

5.3 Application Level

Use the application level to perform tests and maintenance. The application level window illustrated in Figure 5-2 is used to run loopback tests, analyze statistics, view received and transmitted messages, examine the PCI bus configuration, and update registers.

The CN8478 is initialized each time the application level is opened. Therefore, open it first and leave it open while working in other levels.

Figure 5-2. Application Level Window



5.3.1 Application Level Menus

The enabled menus are File, Options, and Help. Edit is not enabled.

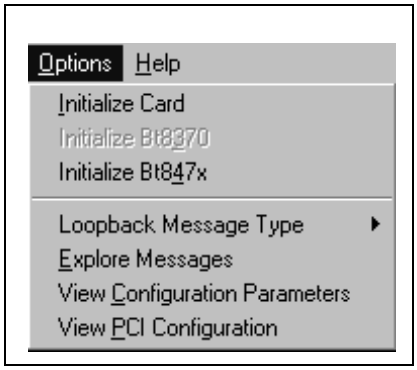
5.3.1.1 File Menu

You can dump either CN8478 registers or Bt8370 registers by saving to a file and folder you select. The address and contents of the device's registers are saved in the file. The file can be retrieved through Notepad, Word, or another Windows NT program.

5.3.1.2 Options Menu

Use the Options menu (illustrated in Figure 5-3) to initialize the card and the CN8478, select loopback message patterns, explore messages, view Card configuration, and view the PCI configuration.

Figure 5-3. Options Menu



Initialize Card and Initialize CN8478

Initialize Card and Initialize CN8478 are like reset buttons. Initialize Card performs a full reset (CN8478 and Bt8370). Initialize CN8478 resets only the CN8478.

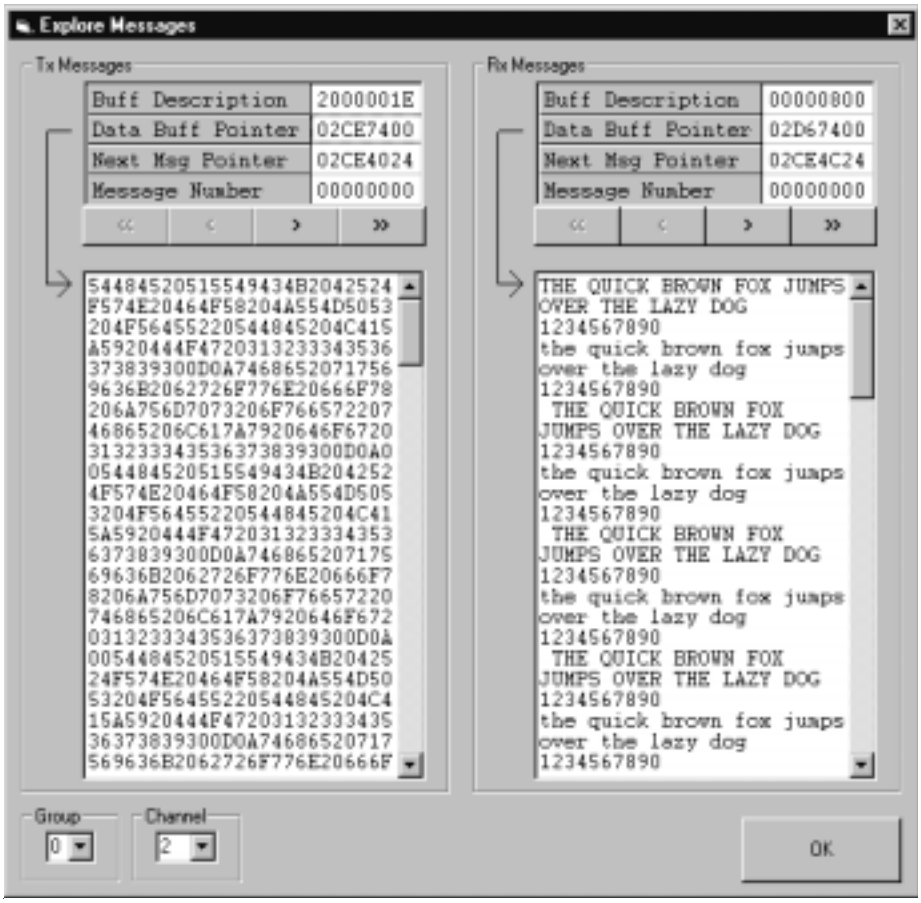
Loopback Messages Type

The Loopback Message Type option allows you to select the pattern of the message type. Six patterns are available (Quick Brown Fox, 0x00 pattern, 0x55 pattern, 0xAA pattern, 0xFF pattern, and User Defined).

Explore Messages

If you select the Explore Messages option during a loopback test, the message descriptor and message data are displayed for the active channel, as illustrated in Figure 5-4.

Figure 5-4. Explore Messages Window

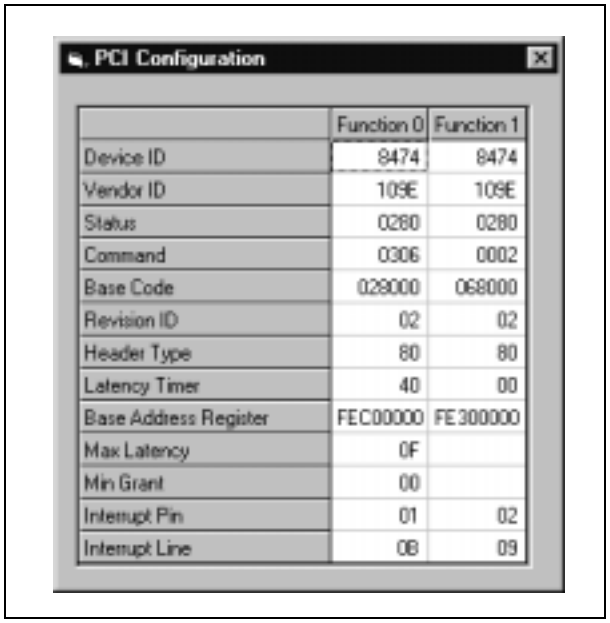


NOTE(S): Message data can be displayed in either hexadecimal or ASCII format, as illustrated above. Double-click the message data to switch between formats.

View PCI Configuration

The PCI Configuration window displays the PCI configuration parameters, as illustrated in Figure 5-5.

Figure 5-5. PCI Configuration Window



5.3.1.3 Help Menu

Online help is accessible from the Help button in the Application Level window. Help includes information about the CN8478 EVM Toolbox and the CN8478 and Bt8370 chips.

Choices include:

- A brief version of this chapter
- CN8478/CN8474A/CN8472A/CN8471A data sheet
- Access to Conexant's Home Page

5.3.2 Loopback Tests

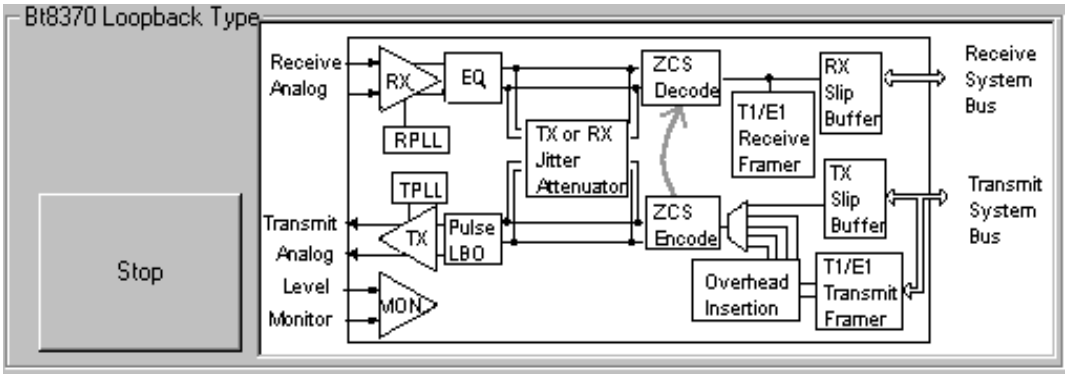
The Application Level window provides a framer loopback test for diagnostics, maintenance, and troubleshooting. There are two loopback interfaces, one for each device—the Bt8370 and the CN8478.

5.3.2.1 Bt8370 Loopback

Select the applicable Bt8370 device and port by selecting the group from the Group drop-down list box.

The default configuration for Bt8370 loopback is run (the arrow is green). Clicking the Stop button stops the loopback (the arrow turns red). The stop condition takes the Bt8370 out of loopback mode but does not stop the transmit message if a CN8478 loopback test is still running. The Bt8370 loopback interface is illustrated in Figure 5-6.

Figure 5-6. Bt8370 Loopback Test Area



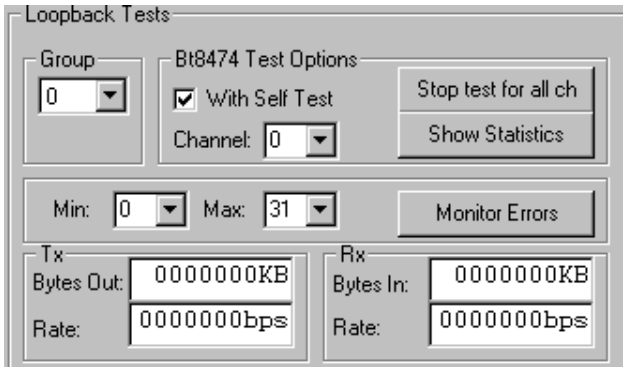
5.3.2.2 CN8478 Loopback

While the Bt8370 is running, you can run a CN8478 loopback test. The CN8478 default configuration is one time slot per channel, protocol HDLC-FCS16. (Use the logical level to change the configurations.)

To run a loopback test, follow these steps:

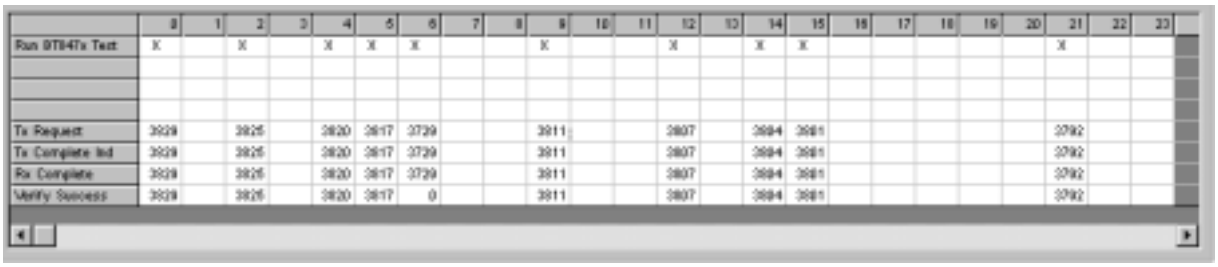
1. Select the group (port) number from the Group drop-down list box in the Loopback Tests area, illustrated in Figure 5-7.

Figure 5-7. CN8478 Loopback Tests Area



2. If you want to disable Self Test for a particular channel, it must be done before you begin the test. Select the channel from the Channel drop-down list box, and click With Self Test to remove the check. When Self Test is disabled, matching between transmit and receive is not performed for that channel during the loopback test.
3. In the grid at the bottom of the application level window, select the channels you want to test by clicking at the junction of the channel to test and the Run Bt847x Test row. This begins the test. An example of the Run CN8478 Test running 10 of the 23 channels is illustrated in Figure 5-8.
4. With Self Test enabled, the Verify Success box indicates whether or not the contents of the transmitted frame are identical with the contents of the received frame; the matching is done byte by byte. In Figure 5-8, all channels have Self Test enabled except channel 6.
5. To stop the loopback test in one channel, click the X under the channel number.

Figure 5-8. Example of Test



NOTE(S): A gray column means that the corresponding channel is disabled. In T1 interface mode, the channels 24 through 31 are disabled.

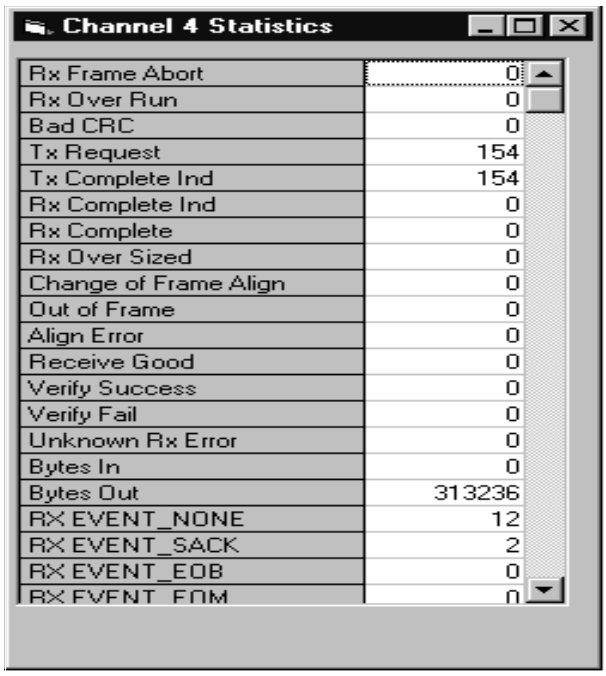
During the loopback test you have four options (in addition to those on the Options menu):

- Stop tests for all channels
- Show statistics
- Monitor errors
- View transmit and receive information rate

Stop Test When you click Stop Test for All Channels, all loopback tests stop. Click again in the grid to restart the tests individually.

Show Statistics When you click Show Statistics, the Channel Statistics window appears, which displays statistics of transmit and receive messages, as well as important events and errors for the selected channel. You can also double-click anywhere in the grid during the loopback test to show statistics. Figure 5-9 illustrates the statistics for Channel 4.

Figure 5-9. Channel Statistics Window

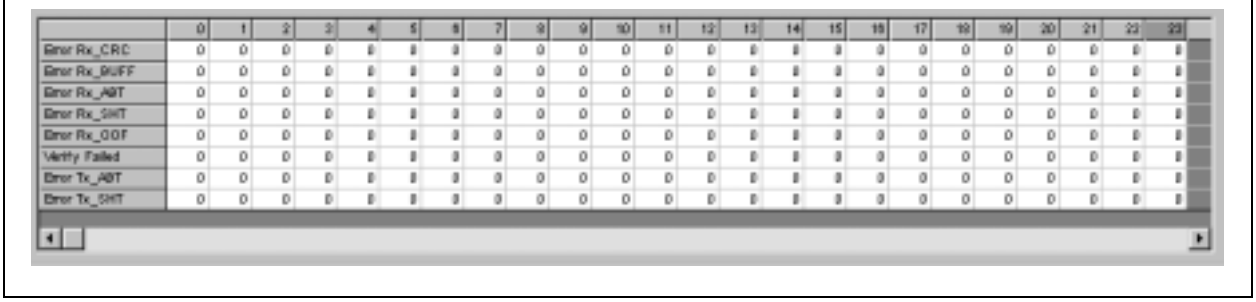


Monitor Errors

When you click Monitor Errors, the Toolbox retrieves real-time information from the driver related to receive and transmit errors in all running channels. The display is updated every five seconds. Receive errors are checksum, buffers, abort, short, and out-of-frame. Transmit errors are abort and short. Another error is Verify Failed. When you click Stop Monitoring, the grid returns to the loopback, which has continued running in the background.

Figure 5-10 illustrates the monitor errors display.

Figure 5-10. Monitor Errors Display



Transmit and Receive Information

The Transmit and Receive Information area, illustrated in Figure 5-11, gives information about the number of bytes received and transmitted and the data rate in bits-per-second for the active channels. When a test is running, transmit and receive information is updated each second.

Figure 5-11. Transmit and Receive Information Area

5.3.3 Summary of Application Level

User interactions in the Application window are listed in Table 5-1.

Table 5-1. Summary of Application Level Window

Name	Description
File	Dumps Bt8487 registers or Bt8370 registers into a file.
Options	Initialize Card—Initialize the Adapter. Initialize CN8478—Initialize the CN8478. Loopback Message Type—Loopback messages type for testing CN8478. Explore Messages—Zoom the messages (ASCII and hex) of the specific channel. View Configuration Parameters—View the registry configuration parameters. View PCI Configuration—View PCI configuration.
Help	Help file online.
Group	Specifies the number of the Bt8370 devices used in the CN8478 EVM board.
Bt8370 Loopback	Stops or runs the Bt8370 loopback.
Self Test	Compares the content of the Tx and Rx messages during loopback tests.
Run CN8478 Test	Starts the CN8478 loopback test for each channel selected.
Stop test for all ch	Stops the CN8478 loopback test for all the channels.
Show statistics	Shows statistics for the group and channel selected.
Min and Max	Displays minimum and maximum number of channels that can be tested.
Tx Bytes Out/Rx Bytes In	Displays Tx and Rx information about transmission speed.
Monitor Errors	Checks for Rx and Tx errors.

5.4 Logical Level

In Logical level, you can configure the protocol and time slots' assignment for each logical channel.

NOTE: The configuration should not be changed while the channels are active. This means that a "Stop test for all ch" is recommended before changing channel configurations.

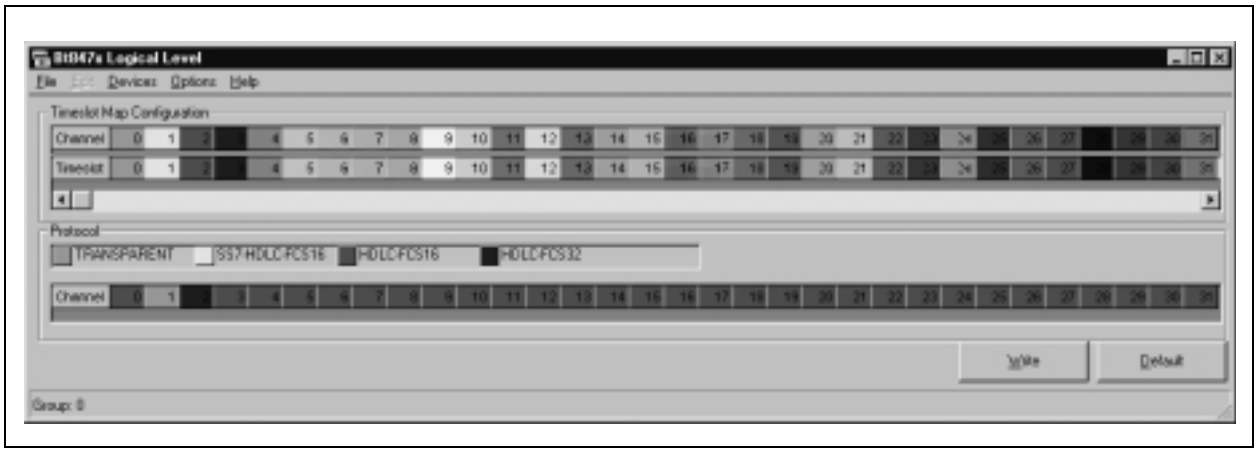
The CN8478 is initialized when the Application level is opened. Therefore, open it and leave it open while working in the Logical level.

There are two modes in logical level: CN8478 and Bt8370.

5.4.1 CN8478 Logical Level

The Timeslot Map Configuration area allows hyperchannel capability tests. You can map several time slots on one logical channel and then run a loopback test in the application level window. Figure 5-12 illustrates the Logical level window. The speed of appropriate channels is shown in the Transmit and Receive information area.

Figure 5-12. Bt847x Logical Level Window



Follow these steps to assign a new time slot to a channel:

1. Select the group in the Options menu.
2. In the Bt847x Logical level window, click the specific channel, then click the time slot. The color of the time slot cell changes to match the channel cell color.
3. When you finish selecting time slots for the channel, click the Write button to save the configuration. (Clicking Default causes the configuration to return to one time slot per channel.)

Follow these steps to assign protocols to a channel:

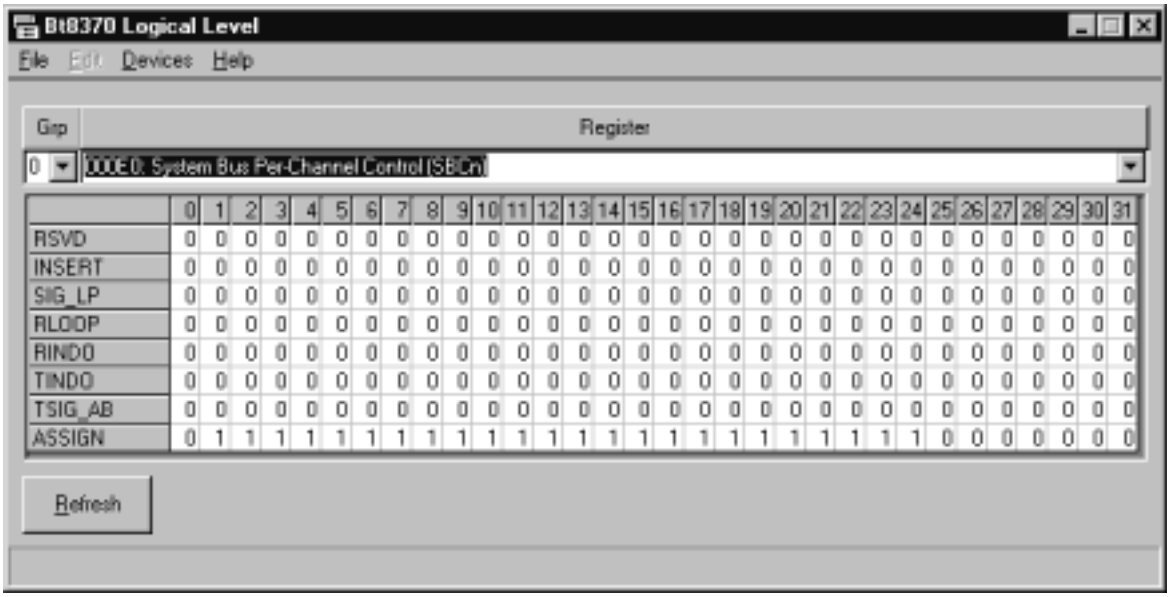
1. Click one of the four protocols in the Protocol area, and then click the channel.

2. Click the Write button to save the desired configuration in the 8474 register map.

5.4.2 Bt8370 Logical Level

The Bt8370 Logical level window, illustrated in Figure 5-13, displays information about all Bt8370 registers on each channel. In this window, the Refresh button updates the register value.

Figure 5-13. Bt8370 Logical Level Window



5.4.3 Summaries of Logical Level Windows

Table 5-2 and Table 5-3 lists user interactions in the CN8478 and Bt8370 Logical level window respectively.

Table 5-2. Summary of CN8478 Logical Level Window

Name	Description
File	Exit
Devices	Bt8370 and CN8478
Options	Refresh—refreshes the screen. Group—specifies the 8370 device to be used Loopback Message Type—displays types of loopback messages View Configuration Parameter—displays the registry configuration parameters View PCI Configuration—displays the PCI configuration
Help	Online help
Time Map Configuration	Time slots assignments for each logical channel
Protocol Configuration	Protocol configuration per channel
Default	Returns to default configuration—one time slot per channel, protocol HDLC-FCS16
Write	Write user's configuration

Table 5-3. Summary of Bt8370 Logical Level Window

Name	Description
File	Exit
Devices	Bt8370 / CN8478
Help	Online help
Group	Specifies the number of the Bt8370 devices used in the CN8478 EVM board
Register	Drop-down menu of registers
Refresh	Provides latest register value

5.5 Physical Level

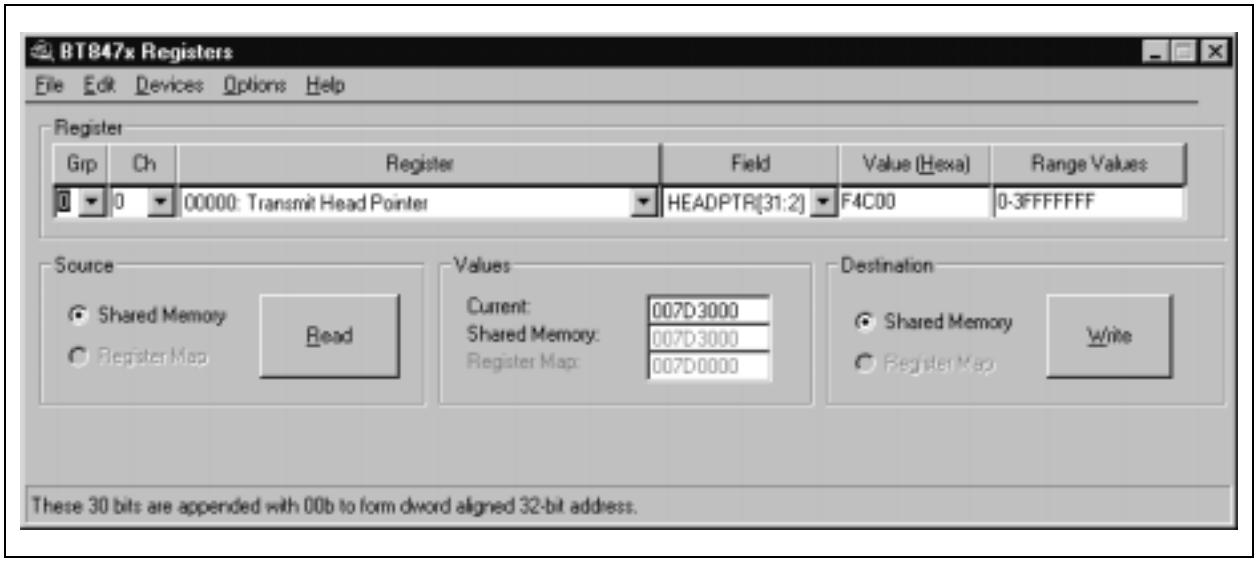
The Physical level is a component register editor where you configure your settings. The default is the CN8478 Registers window, where you can edit the CN8478 and Bt8472 devices. You can select the Bt8370 Registers window from the Devices menu.

The CN8478 is initialized when the Application level is opened. Therefore, open it and leave it open while working in the Physical level.

5.5.1 Register Area

Each register consists of an address, a name, and list of fields. Each field consists of a name, an option list or a range of values, and description. Select the group and channel before selecting the register. Figure 5-14 illustrates the CN8478 registers window.

Figure 5-14. CN8478 Registers Window



5.5.2 Source, Values, and Destination Area

Select where the value is to be read or written in the Source and Destination areas. On the CN8478, some registers are present both in shared memory and in the register map.

5.5.2.1 Read

All registers are read at the opening of the window or during a device selection. To update the register value, click the Read button.

5.5.2.2 Write

When you configure something differently from the default configuration, you must click the Write button. The data can be written into either shared memory or the register map.

5.5.2.3 Values

In the Values area, the contents of a specific register are displayed in three different formats: current, shared memory, and register map.

Current In the Current box, you can enter a specific value for a specific register that can be written in either shared memory or the register map. An automatic check is done that allows you to write to specific registers in the register map area.

Shared Memory This field displays the variable's content from the shared memory.

Register Map This field displays the variable's content from the register map.

5.5.2.4 Description Bar

The description bar at the bottom of the screen describes the selected bit field.

5.5.3 Summary of Physical Level

Table 5-4 lists the user interactions with the Registers windows.

Table 5-4. Summary of Registers Window Interactions

Name	Description
File	Dump—Dumps the registers into a file Exit
Edit	Cut, copy, paste, delete
Devices	CN8478 or Bt8370
Options	Refresh
Help	Online Help
Grp (Group)	Specifies the group (port) from 0–3
Ch (Channel)	Specifies the channel from 0–31
Register	Specifies registers mapped in either shared memory or register map
Field	Specifies the bit field map related to the register map
Options/Value(Hex)/Ranges	Specifies values and ranges for the register map
Source and Destination	Read or write to shared memory or register map in these fields
Values	Displays a register's contents
Current/Shared Memory/Register Map	Displays component contents of register of the selected bit field mask in shared memory or register map
Description bar	Displays the description of the current bit field

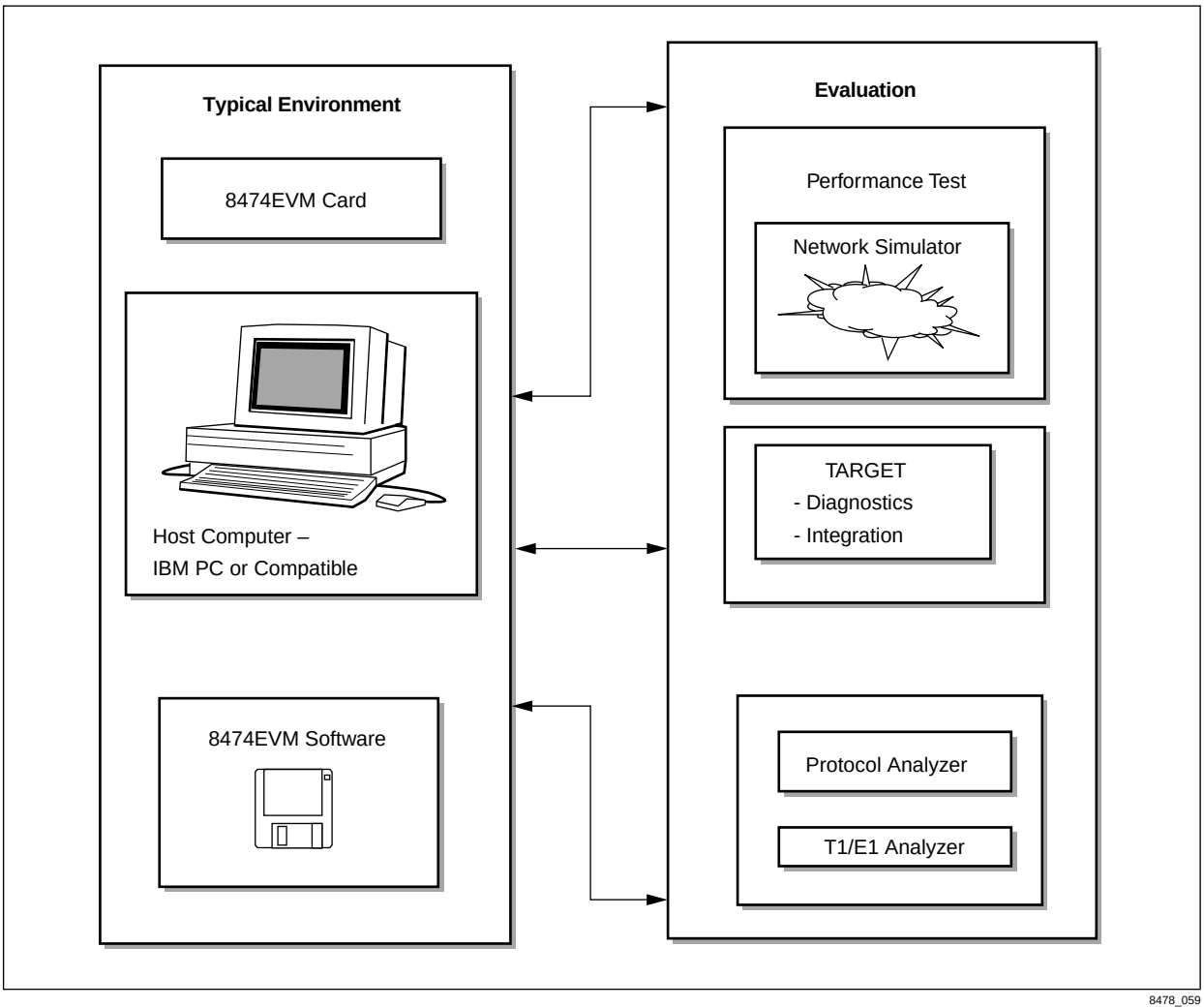
6.0 List of Acronyms and Abbreviations

API	Application Program Interface
EBUS	Expansion Bus
HDLC	High-level Data Link Control
INTA	Integer Array
JTAG	Test access port and boundary scan architecture
NDIS	Network Driver Interface Specification
NIC	Network Interface Controller
PCI	Peripheral Component Interconnect
PDF	Portable Document Format

Appendix A: Sample Connection

The typical development environment illustrated in Figure A-1 consists of a host computer, a PCI bus, version 2.0 or higher, and CN8478 EVM Software. The evaluation environment consists of three types of system components: performance test (which contains the network simulator), target test, and protocol analyzer.

Figure A-1. Typical Development Environment



Appendix B: Bill of Materials

Table B-1 displays the bill of materials.

Figure B-1. Bill of Materials *(1 of 3)*

Item	Qty	Reference	Value	Type	Manufacturer Body	RSS Part #
1	51	C1, C2, C3, C4, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C22, C23, C24, C25, C26, C27, C28, C32, C33, C34, C35, C36, C37, C38, C42, C43, C44, C45, C46, C47, C48, C52, C53, C54, C55, C56, C57, C58, C62, C63, C64, C65	0.1	20% 50 V	0805	5404R20-031
2	5	C5, C21, C31, C41, C51	10	20% 6 V	AVX SMT 12.6x6.3	5402R02-032
3	4	C29, C39, C49, C59	100 p	5% 50 V	0805	5404R19-025
4	4	C30, C40, C50, C60	15 p	5% 100 V	0805	5404R19-015
5	1	C61 (no load)	0.1	20% 50 V	0805	5404R20-031:
6	16	D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15, D16		6.2 V	MOTOROLA MMSZ5234BT1	
7	4	E1, E2, E3, E4			JUMPER-2 CA-S36-25B-45	
8	16	FS1, FS2, FS3, FS4, FS5, FS6, FS7, FS8, FS9, FS10, FS11, FS12, FS13, FS14, FS15, FS16			POLYSWITCH RAYCHEM TR600-160	
9	1	JP1			PCI EDGE CONNECTOR	
10	1	J1			CON2 CA-S36-25B-45	
11	1	J2			CON1 CA-S36-25B-45	
12	4	J3, J4, J5, J6			RJ48C 613R51-003	613R51-003

Figure B-1. Bill of Materials *(2 of 3)* ***(Continued)***

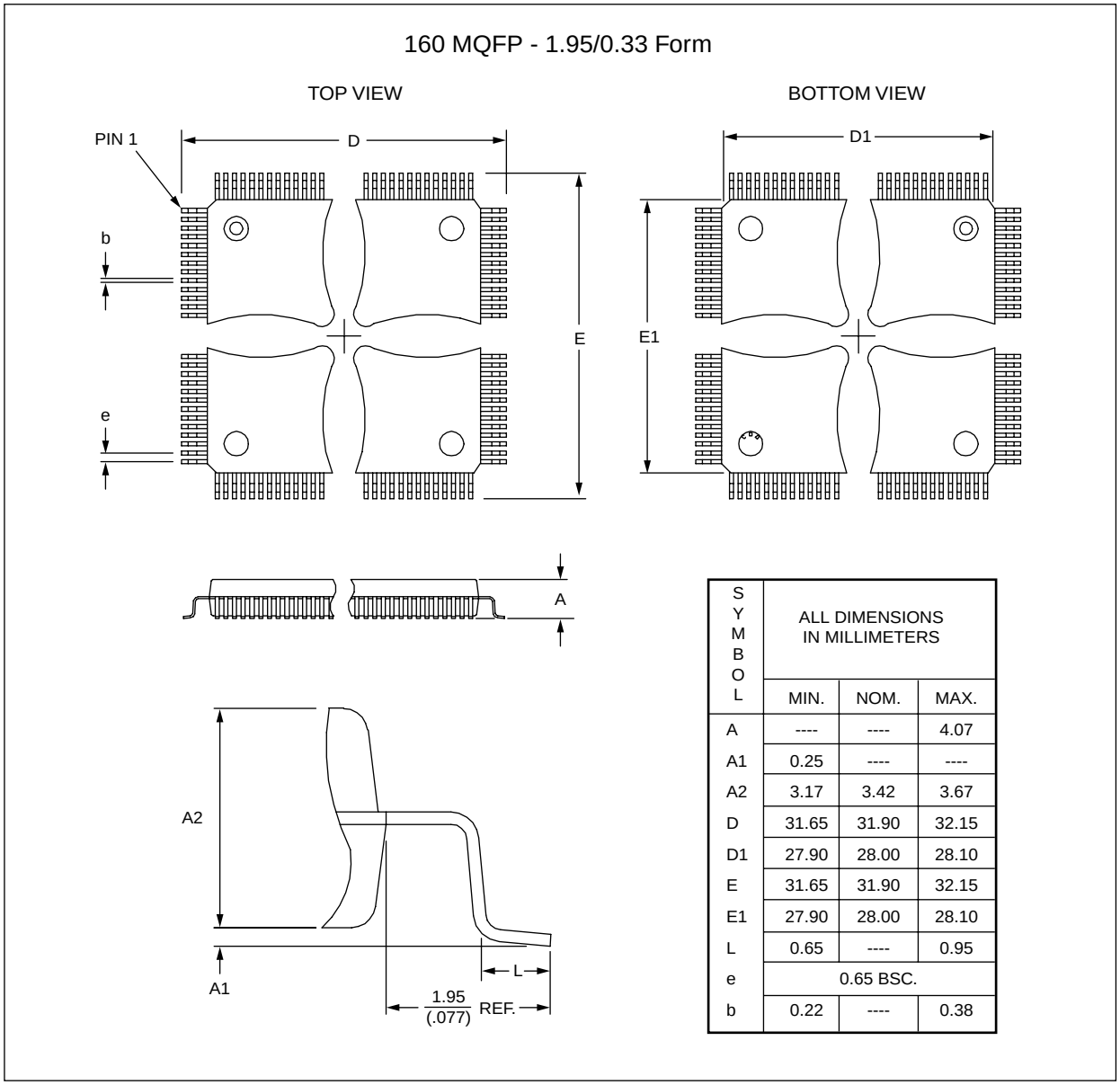
Item	Qty	Reference	Value	Type	Manufacturer Body	RSS Part #
13	4	L1, L2, L3, L4			FILTER PULSE_ENGINEERING PE65854	
14	3	R1, R5, R8	0	5% 0.1W	0805	5424R08-146
15	7	R2, R6, R7, R18, R19, R20, R21	100k	5% 0.1W	0805	5424R08-097
16	2	R3, R9	10k	5% 0.1W	0805	5424R08-073
17	1	R4	100	5% 0.1 W	0805	5424R08-025
18	8	R10, R11, R12, R13, R14, R15, R16, R17	47k	5% 0.1W	0805	5424R08-089
19	2	R22, R23	10k	5% 0.1W	0805	5424R08-073
20	4	R24, R30, R36, R42	51.1	1% 0.25W	1206	5424R01-069
21	4	R25, R31, R37, R43	12.4k	1% 0.1W	0805	5424R07-297
22	4	R26, R32, R38, R44	1k	1% 0.25W	1206	5424R01-193
23	4	R27, R33, R39, R45	240	1% 0.25W	1206	5424R01-134
24	8	R46, R47, R48, R49, R50, R51, R52, R53	100	1% 0.25W	1206	5424R01-097
25	4	TR1, TR2, TR3, TR4			T1044 PULSE_ENG T1044	
26	1	U1			BT8474 BROOKTREE PQFP-160PIN BT8474	
27	1	U2			74F373 TI SOIC-20PIN-DW 74F373-DW	
28	1	U3			74F138 TI SOIC-16PIN-D 74F138-D	

Figure B-1. Bill of Materials *(3 of 3)* ***(Continued)***

Item	Qty	Reference	Value	Type	Manufacturer Body	RSS Part #
29	1	U4			74HCT174 SOIC-16PIN-D 74HCT174-D	
30	1	U5			74F00 SOIC-14PIN-D 74F00-D	
31	1	U6			74HCT03 SOIC-14PIN-D 74HCT03-D	
32	4	U7, U8, U9, U10			BT8370 REV D - 80MQFP	
33	1	Y1	10 MHz OSC.		FOX F3356-10.000 MHZ	
34	1	ZU1			PCI_BRACKET GOMPF 9334-0006	
35	1	ZU2			PCI_RETAINER GOMPF 9100-0000B	
36	2	ZU3, ZU4			SCREW_BRACKET	
37	2	ZU5, ZU6			SCREW_RETAINER	
		Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8			P1553AB TECCOR MOD-TO220 P1553AB	

Appendix C: Mechanical Specifications

Figure C-1. 160-pin PQFP Package Mechanical Drawing



SYMBOL	ALL DIMENSIONS IN MILLIMETERS		
	MIN.	NOM.	MAX.
A	----	----	4.07
A1	0.25	----	----
A2	3.17	3.42	3.67
D	31.65	31.90	32.15
D1	27.90	28.00	28.10
E	31.65	31.90	32.15
E1	27.90	28.00	28.10
L	0.65	----	0.95
e	0.65 BSC.		
b	0.22	----	0.38

8478_060

Appendix D: Circuit Schematics

Figures D-1 through D-5 illustrate the following:

1. PCI Interface
2. T1/E1 Interface—Port 1
3. T1/E1 Interface—Port 2
4. T1/E1 Interface—Port 3
5. T1/E1 Interface—Port 4

Figure D-1. PCI Interface

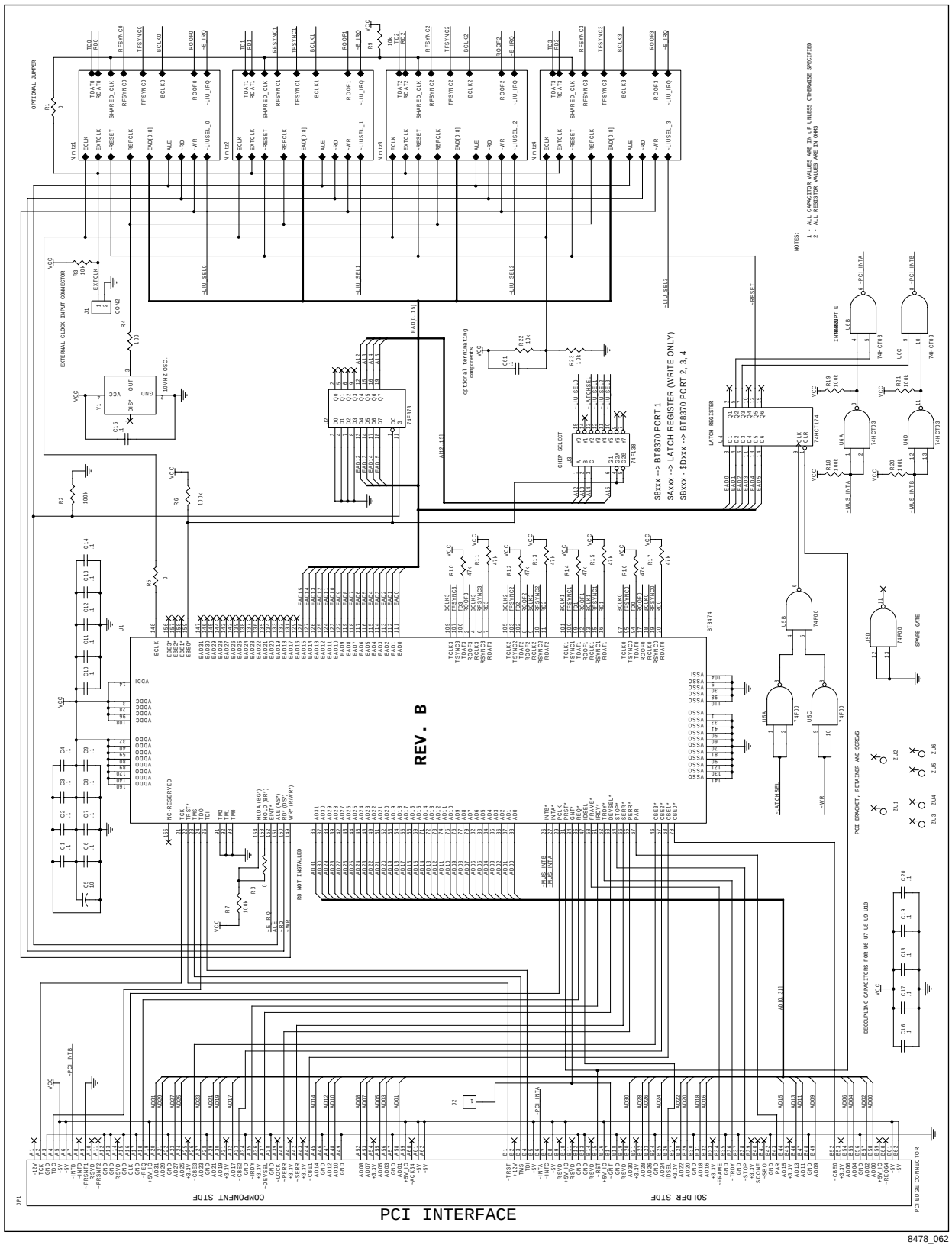


Figure D-2. T1/E1 Interface - Port 1

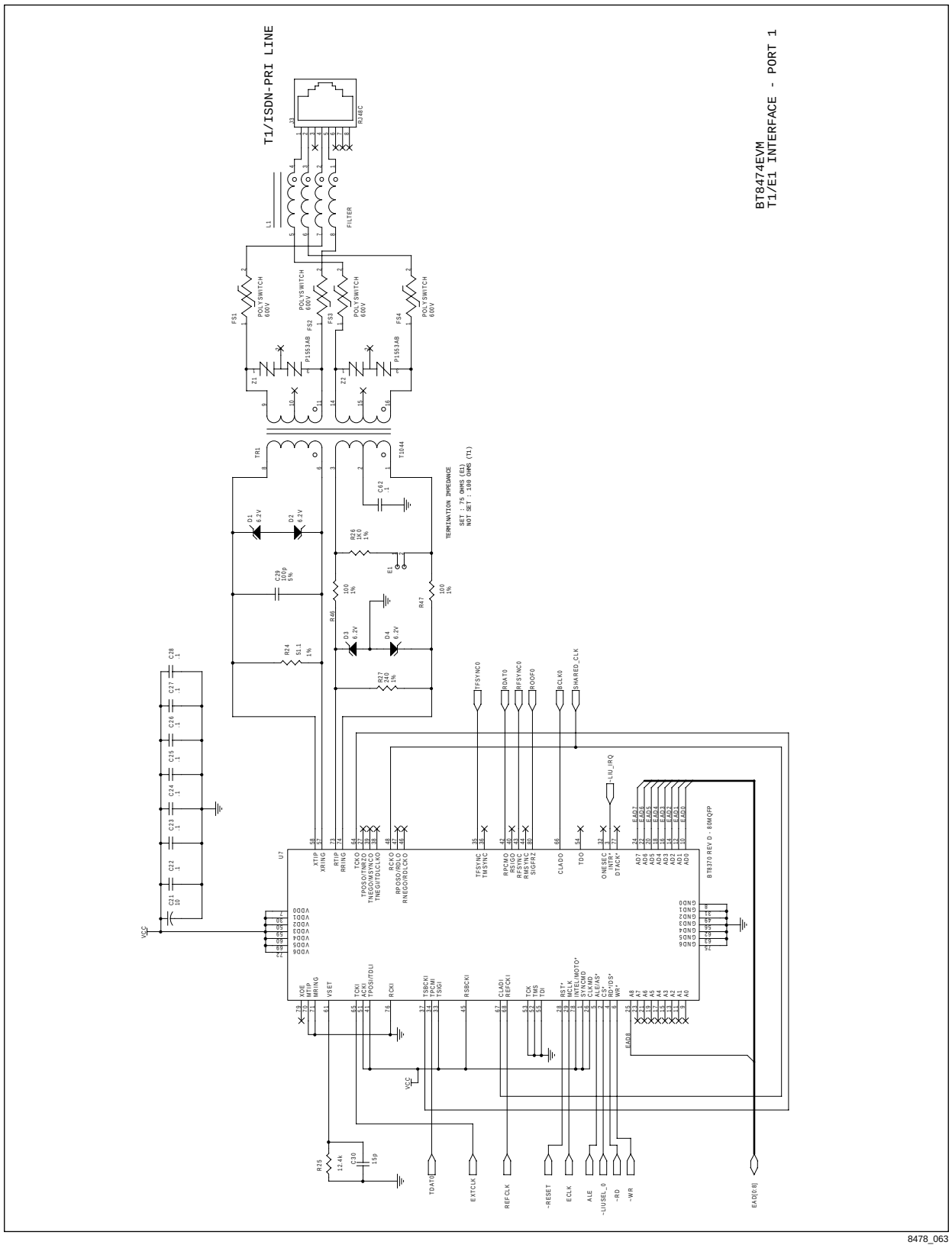


Figure D-3. T1/E1 Interface - Port 2

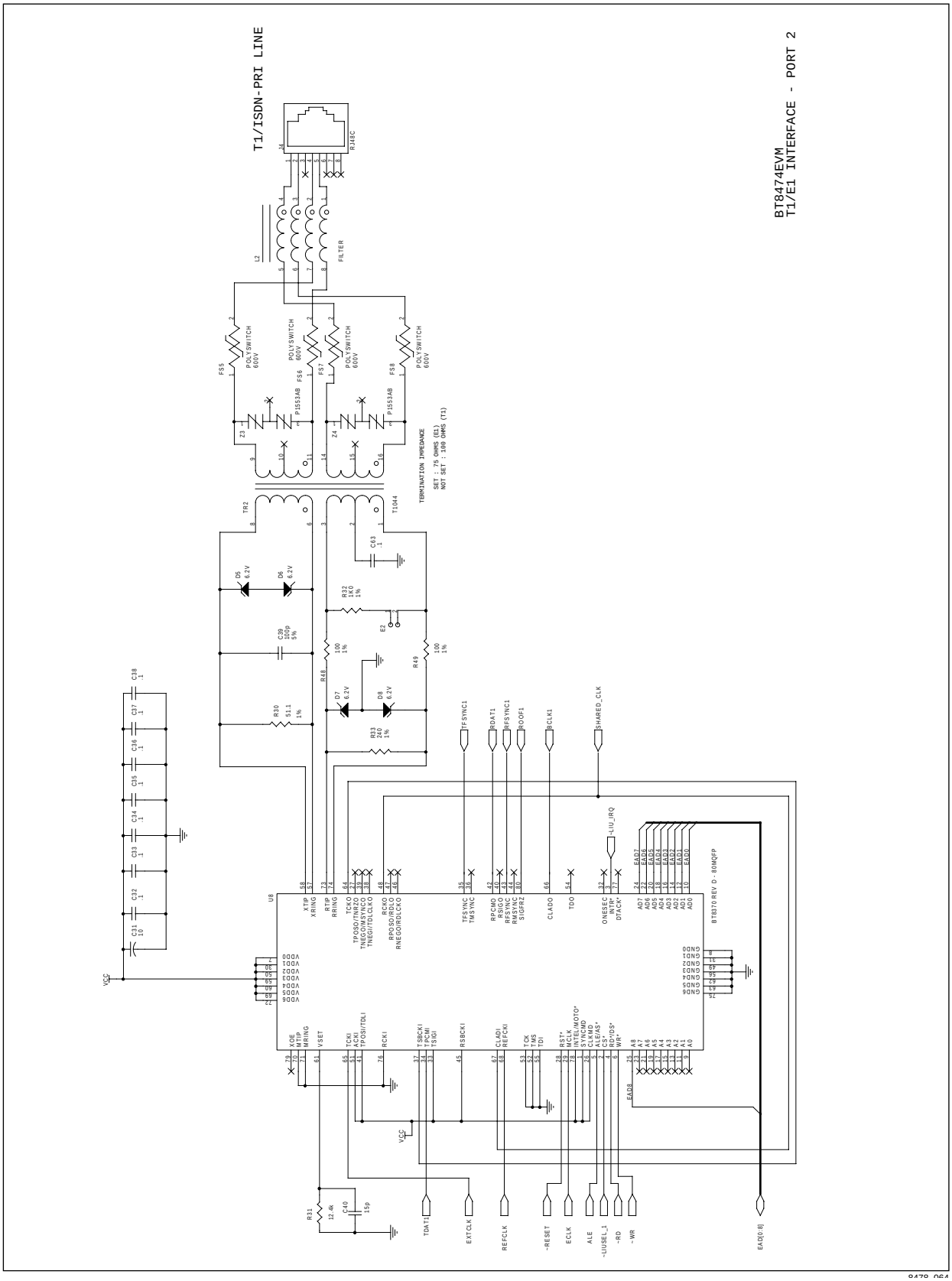


Figure D-4. T1/E1 Interface - Port 3

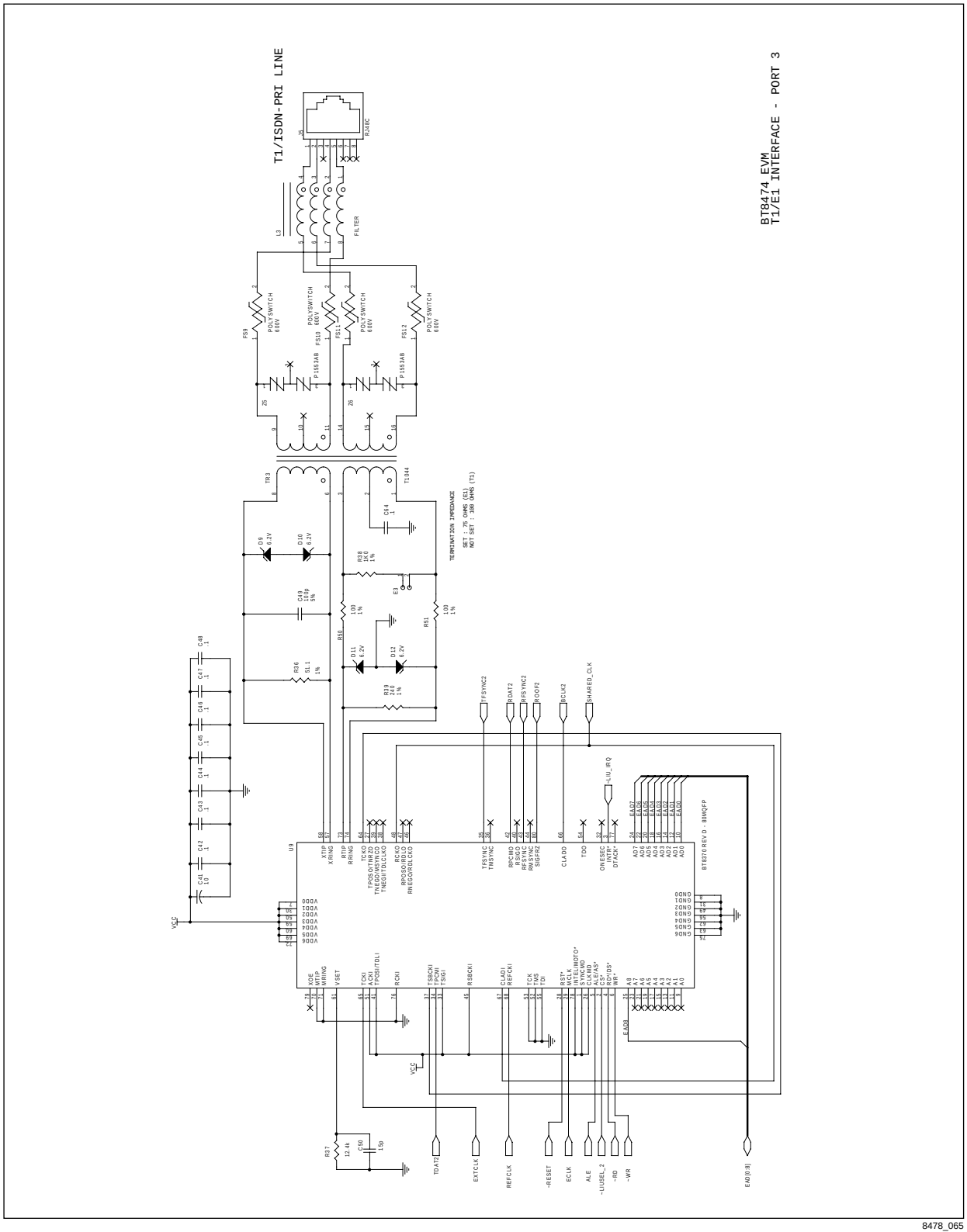
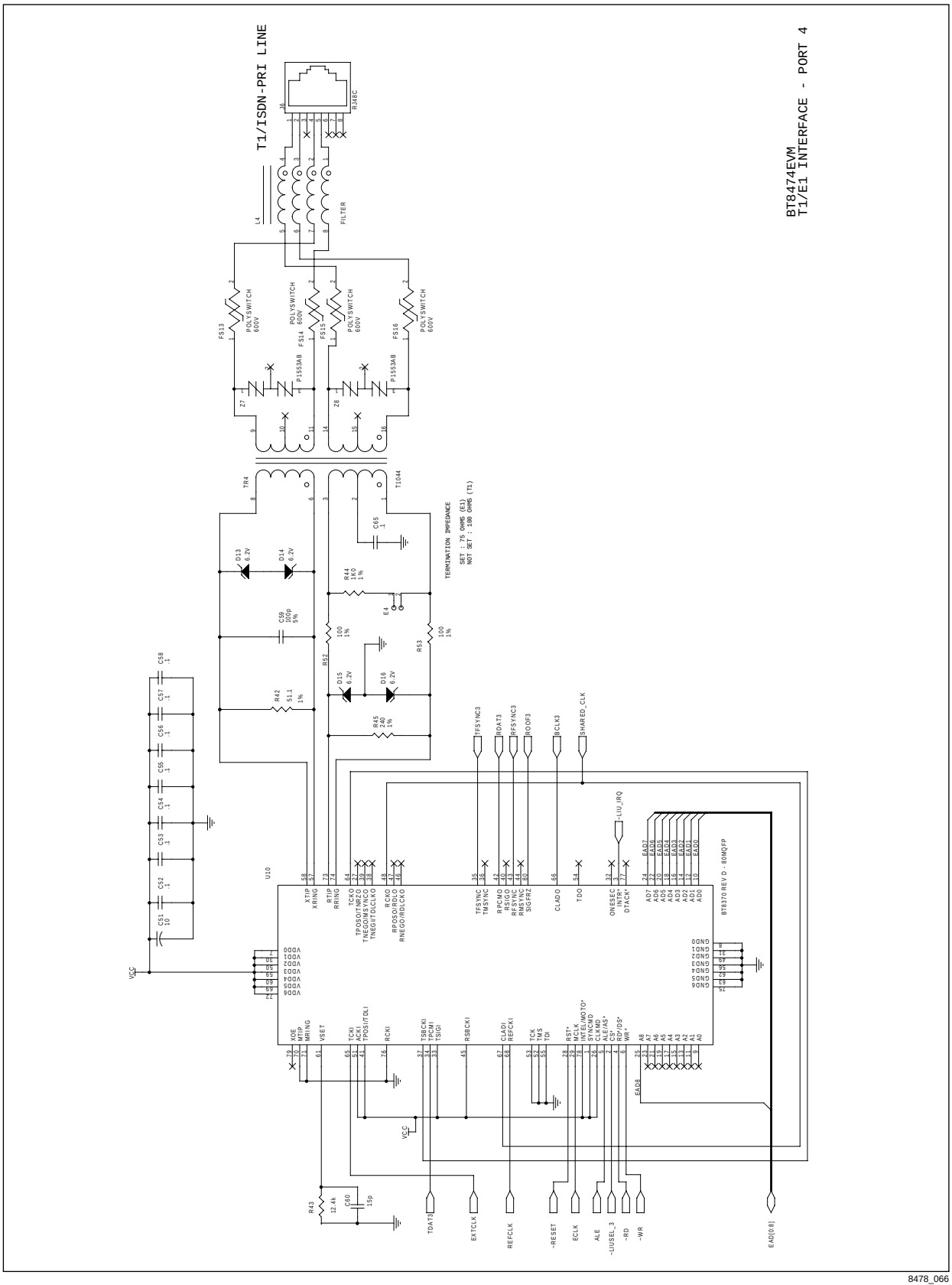


Figure D-5. T1/E1 Interface - Port 4



8478_066

Appendix E: Circuit Board Drawings

Figures E-1 through E-5 illustrate the levelss of the circuit board. Figure E-5 is the silkscreen for the top level, which has all the reference designators and component outlines.

Further Information
literature@conexant.com
1-800-854-8099 (North America)
33-14-906-3980 (International)

Web Site
www.conexant.com

World Headquarters
Conexant Systems, Inc.
4311 Jamboree Road
P. O. Box C
Newport Beach, CA
92658-8902
Phone: (949) 483-4600
Fax: (949) 483-6375

U.S. Florida/South America
Phone: (727) 799-8406
Fax: (727) 799-8306

U.S. Los Angeles
Phone: (805) 376-0559
Fax: (805) 376-8180

U.S. Mid-Atlantic
Phone: (215) 244-6784
Fax: (215) 244-9292

U.S. North Central
Phone: (630) 773-3454
Fax: (630) 773-3907

U.S. Northeast
Phone: (978) 692-7660
Fax: (978) 692-8185

U.S. Northwest/Pacific West
Phone: (408) 249-9696
Fax: (408) 249-7113

U.S. South Central
Phone: (972) 733-0723
Fax: (972) 407-0639

U.S. Southeast
Phone: (919) 858-9110
Fax: (919) 858-8669

U.S. Southwest
Phone: (949) 483-9119
Fax: (949) 483-9090

APAC Headquarters
Conexant Systems Singapore, Pte. Ltd.
1 Kim Seng Promenade
Great World City
#09-01 East Tower
SINGAPORE 237994
Phone: (65) 737 7355
Fax: (65) 737 9077

Australia
Phone: (61 2) 9869 4088
Fax: (61 2) 9869 4077

China
Phone: (86 2) 6361 2515
Fax: (86 2) 6361 2516

Hong Kong
Phone: (852) 2827 0181
Fax: (852) 2827 6488

India
Phone: (91 11) 692 4780
Fax: (91 11) 692 4712

Korea
Phone: (82 2) 565 2880
Fax: (82 2) 565 1440

Phone: (82 53) 745 2880
Fax: (82 53) 745 1440

Europe Headquarters
Conexant Systems France
Les Taissounieres B1
1681 Route des Dolines
BP 283
06905 Sophia Antipolis Cedex
FRANCE
Phone: (33 1) 41 44 36 50
Fax: (33 4) 93 00 33 03

Europe Central
Phone: (49 89) 829 1320
Fax: (49 89) 834 2734

Europe Mediterranean
Phone: (39 02) 9317 9911
Fax: (39 02) 9317 9913

Europe North
Phone: (44 1344) 486 444
Fax: (44 1344) 486 555

Europe South
Phone: (33 1) 41 44 36 50
Fax: (33 1) 41 44 36 90

Middle East Headquarters
Conexant Systems
Commercial (Israel) Ltd.
P. O. Box 12660
Herzlia 46733, ISRAEL
Phone: (972 9) 952 4064
Fax: (972 9) 951 3924

Japan Headquarters
Conexant Systems Japan Co., Ltd.
Shimomoto Building
1-46-3 Hatsudai,
Shibuya-ku, Tokyo
151-0061 JAPAN
Phone: (81 3) 5371-1567
Fax: (81 3) 5371-1501

Taiwan Headquarters
Conexant Systems, Taiwan Co., Ltd.
Room 2808
International Trade Building
333 Keelung Road, Section 1
Taipei 110, TAIWAN, ROC
Phone: (886 2) 2720 0282
Fax: (886 2) 2757 6760