TRACELINK

Advanced Trace Capture Interface For Freescale Devices

USER MANUAL

TRACELINKUM

freescale™
Alliance Member

Purchase Agreement

1   INTRODUCTION

The Tracelink is a powerful development interface capable of capturing external trace data on Freescale microcontrollers.

The Tracelink is designed to help developers find and diagnose software bugs quickly and efficiently. While a microcontroller is running, the Tracelink is constantly recording trace information into its internal memory buffer, which can later be used by supporting software to provide program flow charts and profiling statistics.

Figure 1-1: P&E's Tracelink

In addition to trace capture, the Tracelink also supports all standard run control operations, such as single-stepping, setting breakpoints, register/memory access, and flash programming. The following features make the Tracelink an extremely valuable debug tool:

- **Multiple Freescale Architecture Support**
  - ColdFire V2-4
  - Kinetis
  - S12Z (coming soon)

- **External Trace Capture**
  - Supports trace port speeds up to 250 MHz
  - 128MB of trace storage

- **Multiple Voltage Operation**
  - Automatically detects and caters to target voltages ranging from 1.8V to 5V

- **Target Power Generation**
  - Can generate 2V, 3V, or 5V to directly power target microcontroller

- **Multiple Communication Interfaces**
  - Ethernet 10/100 baseT
  - USB 1.1

Additional Tracelink resources can be found at **pemicro.com/tracelink**.

2   QUICK START GUIDE

Step 1. Install Software

The accompanying software includes all necessary drivers and configuration utilities needed for the Tracelink. After installation is complete, make sure the PC is rebooted before moving on to the configuration step.

Step 2. Power On Tracelink

Power up the Tracelink by connecting the included 9V power supply. A blue LED will be illuminated on the Tracelink to indicate it is receiving power.

Step 3. Ethernet Configuration

Connect the Tracelink to the PC using a USB connection. Run the ConfigureIP software utility to set up the network parameters (IP address, subnet mask, etc.). The Ethernet port will not function properly until this one-time configuration is complete. You may wish to refer to **Section 5 - ETHERNET CONFIGURATION**. If the Tracelink Ethernet connection is not required, this step may be skipped.

Step 4. Hardware Setup

Connect the appropriate ribbon cable between the Tracelink and the target board. The Tracelink case flips open for easy access to the ribbon cable connectors. Power up the target board. A yellow LED on the Tracelink will be illuminated to indicate that it is detecting target power.

Finally, connect the USB or Ethernet cable from the Tracelink to the host PC.

Step 5. Launch PC Software

The Tracelink is now ready to be used with PC software to debug and analyze trace. Please refer to the documentation from the software vendor (e.g., P&E software, Freescale™ CodeWarrior) for more details on any additional configuration that may be required.

3   TRACELINK HARDWARE

This chapter provides more in-depth technical detail about the features and interfaces of the Tracelink unit.

3.1   Tracelink Power Supply

The Tracelink requires a regulated 9V DC center positive power supply with a 2.5/5.5mm female plug. The power supply plugs into the power jack located on the top of the unit.

Figure 3-1: Power Jack
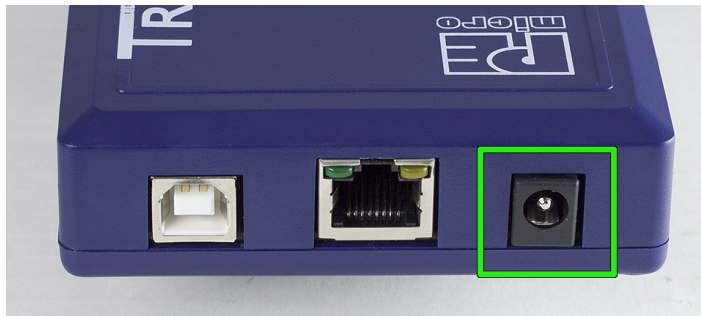
3.2   Ethernet Connector

The Tracelink provides a standard RJ45 socket to communicate with a host PC through the Ethernet Port (10/100 BaseT).

Figure 3-2: Ethernet Connector

3.3   USB Connector

The Tracelink provides a USB connector for Universal Serial Bus communications with the host PC. The Tracelink is a USB 1.1 compliant device.

Figure 3-3: USB Connector
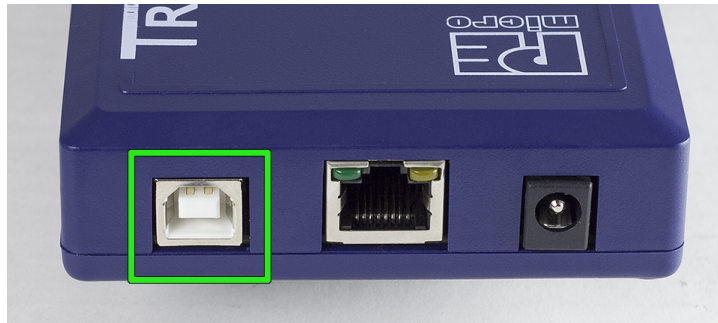
3.4   Target Debug Connectors

The different families of Freescale microcontrollers are supported via the multiple debug headers located on the Tracelink. These headers are accessed by flipping open the plastic case. Pin 1 of each header is marked with the number "1" and also has a rounded corner.

Note: To avoid improper connections, the red stripe of the ribbon cable should always be oriented towards Pin 1.

Warning: Do not attempt to use multiple ports at once, as this may damage both the target processors and the Tracelink.

Figure 3-4: Tracelink Headers, Pin 1 Highlighted
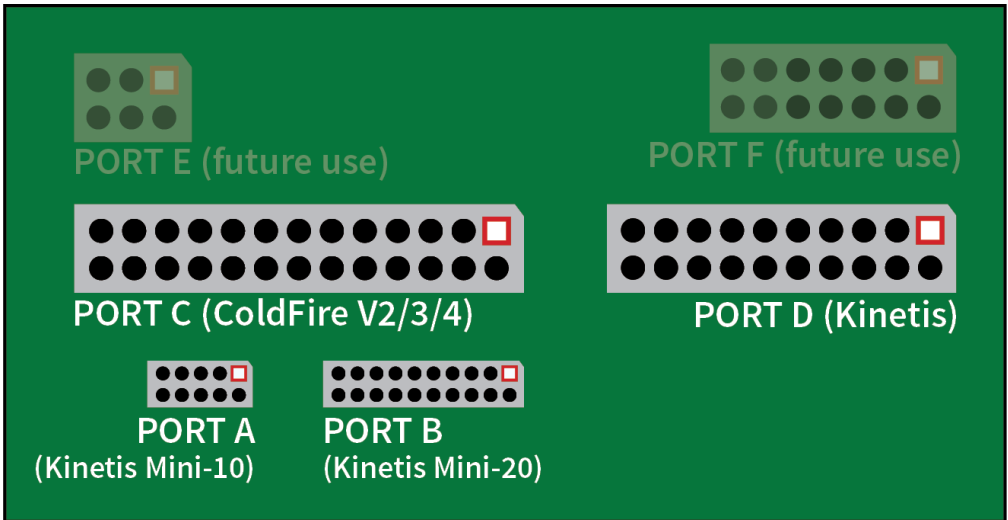
Note: For signals requiring pull-up or pull-down resistors, please note that most microprocessors implement internal resistors to meet these requirements. Otherwise, an external resistor must be used.

3.4.1   PORT A: Kinetis (Mini-10)

Figure 3-5: Port A Pinout

TVCC **1**	• •	**2** TMS/SWD_DIO
GND **3**	• •	**4** TCK/SWD_CLK
GND **5**	• •	**6** TDO
NC **7**	• •	**8** TDI
GND **9**	• •	**10** $\overline{\text{RESET}}$
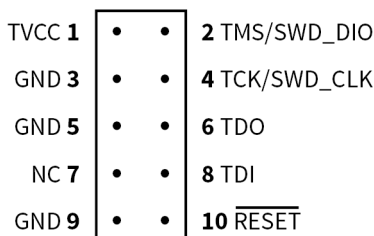
Table C-1. Kinetis Mini-10 Signals (PORT A)

Signal	Direction	Description
TVCC	Input	Target reference voltage. Needs to be connected to the microprocessor's operating voltage (1.8V to 5V) and is used by the Tracelink to determine the correct voltage level to use on output signals. **Required.**
TMS/ SWD_DIO	I/O	Debug communication signal. Pull-up resistor required. **Required.**
GND	Input	Ground signal. Connect to the digital ground signal of the microprocessor. **Required.**
TCK/ SWD_CLK	Output	Debug communication signal. Pull-down resistor required. **Required.**
TDO	Input	Debug communication signal. Pull-up resistor recommended. **Recommended.**
TDI	Output	Debug communication signal. Pull-up resistor required. **Recommended.**
RESET	I/O	Microprocessor reset signal. This signal is driven low during initial debug mode entry. Pull-up resistor required. **Required.**

3.4.2   PORT B: Kinetis (Mini-20)

Figure 3-6: Port B Pinout

```
          TVCC 1  | •   • |  2  TMS/SWD_DIO
           GND 3  | •   • |  4  TCK/SWD_CLK
           GND 5  | •   • |  6  TDO
            NC 7  | •   • |  8  TDI
           GND 9  | •   • | 10  RESET
           NC 11  | •   • | 12  TRACE_CLKOUT
           NC 13  | •   • | 14  TRACE_D0
          GND 15  | •   • | 16  TRACE_D1
          GND 17  | •   • | 18  TRACE_D2
          GND 19  | •   • | 20  TRACE_D3
```

Table C-1. Kinetis Mini-20 Signals (PORT B)

Signal	Direction	Description
TVCC	Input	Target reference voltage. Needs to be connected to the microprocessor's operating voltage (1.8V to 5V) and is used by the Tracelink to determine the correct voltage level to use on output signals. **Required.**
TMS/SWD_DIO	I/O	Debug communication signal. Pull-up resistor required. **Required.**
GND	Input	Ground signal. Connect to the digital ground signal of the microprocessor. **Required.**
TCK/SWD_CLK	Output	Debug communication signal. Pull-down resistor required. **Required.**

Table C-1. Kinetis Mini-20 Signals (PORT B)

Signal	Direction	Description
TDO	Input	Debug communication signal. Pull-up resistor recommended. **Recommended.**
TDI	Output	Debug communication signal. Pull-up resistor required. **Recommended.**
RESET	I/O	Microprocessor reset signal. This signal is driven low during initial debug mode entry. Pull-up resistor required. **Required.**
TRACE_CLKOUT	Input	Trace clock input. **Required for external trace capture.**
TRACE_D[3:0]	Input	Trace data input. **Required for external trace capture.**

3.4.3 PORT C: ColdFire V2/3/4

Figure 3-7: Port C Pinout

NC **1**	•	•	**2** $\overline{\text{BKPT}}$
GND **3**	•	•	**4** DSCLK
GND **5**	•	•	**6** TCK
$\overline{\text{RESET}}$ **7**	•	•	**8** DSI
TVCC **9**	•	•	**10** DSO
GND **11**	•	•	**12** PST3/PSTDDATA7
PST2/PSTDDATA6 **13**	•	•	**14** PST1/PSTDDATA5
PST0/PSTDDATA4 **15**	•	•	**16** DDATA1/PSTDDATA3
DDATA2/PSTDDATA2 **17**	•	•	**18** DDATA1/PSTDDATA1
DDATA0/PSTDDATA0 **19**	•	•	**20** GND
NC **21**	•	•	**22** NC
GND **23**	•	•	**24** PSTCLK
TVCC **25**	•	•	**26** $\overline{\text{TA}}$

Table C-1. ColdFire V2/3/4 Signals (PORT C)

Signal	Direction	Description
BKPT	Output	Debug communication signal. Pull-up resistor required. **Required.**
GND	Input	Ground signal. Connect to the digital ground signal of the microprocessor. **Required.**
DSCLK	Output	Debug communication signal. Pull-up resistor required. **Required.**

Table C-1. ColdFire V2/3/4 Signals (PORT C)

Signal	Direction	Description
TCK	Output	Debug communication signal. Pull-up resistor required. This signal is currently only used by P&E when unsecuring ColdFire processors with internal flash. **Recommended.**
RESET	I/O	Microprocessor reset signal. This signal is driven low during initial debug mode entry. Pull-up resistor is required. **Required.**
DSI	Output	Debug communication signal. Pull-up resistor recommended. **Required.**
TVCC	Input	Target reference voltage. Needs to be connected to the microprocessor's operating voltage (1.8V to 5V) and is used by the Tracelink to determine the correct voltage level to use on output signals. **Required.**
DSO	Input	Debug communication signal. Pull-up resistor recommended. **Required.**
PST[3:0]/ PSTDDATA[7:4]	Input	Trace data input. **Required for external trace capture.**
DDATA[3:0]/ PSTDDATA[3:0]	Input	Trace data input. **Required for external trace capture.**
PSTCLK	Input	Trace clock input. **Required for external trace capture.**
TA	Output	Transfer acknowledge signal. Pull-up resistor is required. Some ColdFire processors do not implement this signal and this pin can be left as a no connect (NC) in these cases. **Highly recommended for applications which use external memory.**

3.4.4 PORT D: Kinetis

Figure 3-8: Port D Pinout

```
          TVCC 1 | •   • | 2 NC
          TRST 3 | •   • | 4 GND
           TDI 5 | •   • | 6 GND
  TMS/SWD_DIO 7 | •   • | 8 GND
  TCK/SWD_CLK 9 | •   • | 10 GND
           NC 11 | •   • | 12 GND
          TDO 13 | •   • | 14 GND
        RESET 15 | •   • | 16 GND
           NC 17 | •   • | 18 GND
           NC 19 | •   • | 20 GND
```

Table C-1. Kinetis Signals (Port D)

| Signals	Direction	Description
TVCC	Input	Target reference voltage. Needs to be connected to the microprocessor's operating voltage (1.8V to 5V) and is used by the Tracelink to determine the correct voltage level to use on output signals. **Required.**
TRST	Output	JTAG signal to reset debug logic. Pull-up resistor required. **Optional.**
GND	Input	Ground signal. Connect to the digital ground signal of the microprocessor. **Required.**
TDI	Output	Debug communication signal. Pull-up resistor required. **Recommended.**

Table C-1. Kinetis Signals (Port D)

Signals	Direction	Description
TMS/ SWD_DIO	I/O	Debug communication signal. Pull-up resistor required. **Required.**
TCK/ SWD_CLK	Output	Debug communication signal. Pull-down resistor required. **Required.**
TDO	Input	Debug communication signal. Pull-up resistor recommended. **Recommended.**
RESET	I/O	Microprocessor reset signal. This signal is driven low during initial debug mode entry. Pull-up resistor required. **Required.**

3.5   Target Power

The Tracelink is capable of generating 2V, 3V, or 5V directly to the TVCC pin of the debug header. This can be used to power target microprocessors requiring up to 500mA of current without the need for a separate power supply.

To enable this feature, a jumper needs to be installed onto option J2, which can be accessed by flipping open the plastic case. This jumper is NOT installed by default.

Figure 3-9: J2 Jumper
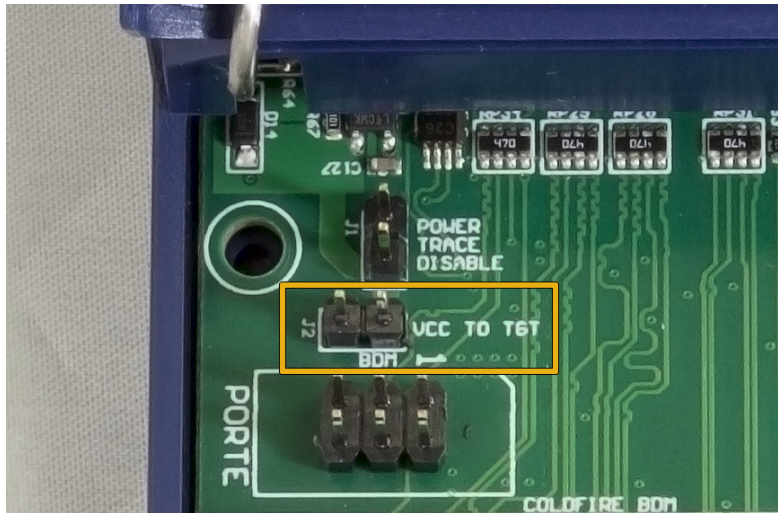
There is additional configuration needed on the PC software to turn on the target voltage and to specify target voltage value (2V, 3V, or 5V). Refer to the software documentation for more details.
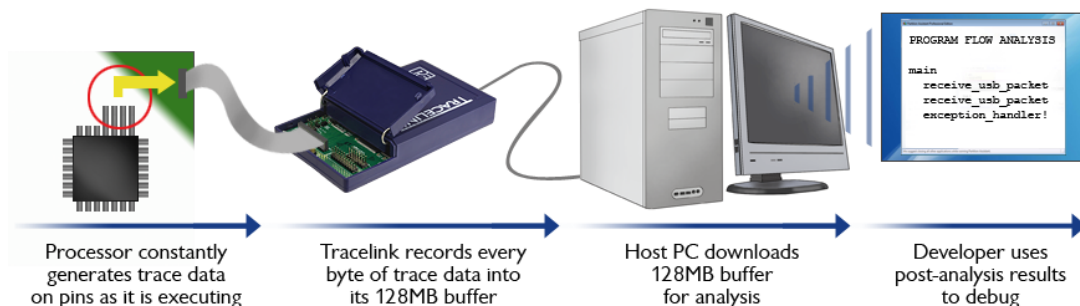
4 EXTERNAL TRACE OVERVIEW

This section provides an introduction to the vocabulary and methodologies of debugging with external trace. Also discussed are general board and layout guidelines to help improve signal integrity for the high-speed trace signals.

4.1 How External Trace Works

Below is a simple diagram of the trace process.

Figure 4-10: "Trace Flow" Diagram

| Processor constantly generates trace data on pins as it is executing | Tracelink records every byte of trace data into its 128MB buffer | Host PC downloads 128MB buffer for analysis | Developer uses post-analysis results to debug |

When a microprocessor is properly configured for external trace and is executing code, it will continuously generate trace data on designated output pins. The width of the trace data port varies depending on the Freescale family and can range from a single bit to 32-bits and higher. The trace data is always synchronized to the rising and/or falling edges of the trace clock signal.

Note: On many microprocessors, the trace pins are often multiplexed with other functions and will default as a general purpose input/output. These pins need to be configured for trace functionality in the application's initialization code.

The Tracelink monitors the trace clock signal and records the value of the trace data pins when the appropriate clock edges occur. The data is saved into the Tracelink's internal 128MB buffer, which is later downloaded onto the host PC for analysis. The software running on the host PC is responsible for decoding the trace data and displaying it in a useful format to the developer.

4.2 Trace Types

Generally there are two categories of trace data being generated by the processor: instruction trace and data trace.

Instruction trace provides "change of flow" information and is mostly concerned with branch and jump type instructions. Instruction trace packets will tell the developer whether a branch was taken and also the destination address of the branch. Instruction trace is the most common type of trace and allows the application's code execution path to be fully reconstructed.

Data trace provides "data access" information and is mostly concerned with load and store type instructions. Data trace packets will tell the developer the address of the memory access, the data value of the memory access, or both. Due to the potentially high bandwidth requirements of data trace, many microprocessors do not directly implement this type of trace.

4.3 Trace Filtering

Trace filtering is a technique used to reduce the amount of trace data generated by the target microprocessor. This is typically done to make better use of the available trace buffer if the user is able to narrow down the scope of the application code where the bug is occurring. For example, a large loop which is used to perform delays may take up a large portion of the 128MB trace buffer but does not contribute useful program flow information to the developer.

Trace filtering focuses primarily on determining when to start and stop trace generation. For example, the microprocessor can be configured to begin generating trace when the program counter matches a specific value and to stop generating trace at a different value.

Each microprocessor family has different trace filtering capabilities, but the goal is the same: try to reduce the amount of trace data generated to only include the most relevant code paths or memory accesses. This allows the developer to make optimal use of the available trace buffer, which may only be large enough to hold a few seconds worth of trace information on a high-speed processor.

4.4 Board Design Considerations

High-speed digital design rules must be observed in order to avoid signal integrity issues that can cause the Tracelink to capture incorrect data. Even with slower trace clock ports, the slew rates of the trace signals can be extremely fast. At a minimum, the following guidelines should be applied to all of the microprocessor trace signals, with emphasis particularly on the trace clock.

4.4.1 Avoid impedance discontinuities

Stubs are printed circuit board tracks that branch off from the main track and are usually formed when placing test points or connecting multiple components to the same net. Stubs should be avoided at all costs as even short stubs can cause serious signal integrity issues.

Vias are a source of impedance change and the number of vias used should be minimized.

Keep track widths constant throughout the signal path.

4.4.2 Minimize skew

Try to match the length of all printed circuit board tracks used to carry the trace signals.

4.4.3 Ground plane

A solid ground plane will ensure a good return path and minimize noise. Avoid having breaks in the ground plane if possible. Ideally the ground plane should be situated in the layer next to the signal layer carrying the trace signals.

4.4.4 Minimize crosstalk

If the design has multiple signals running parallel to each other for long distances, increase the spacing between them to avoid crosstalk issues. Note that crosstalk can also occur across layers if two signal layers are adjacent to each other.

4.4.5 Signal termination

To eliminate reflections on the signal path, three impedances must match: the source, the printed circuit board track, and the load. There are two common termination schemes to minimize reflection:

- Series termination: A resistor is placed in series with the signal as close as possible to the source. The resistor value plus the output impedance of the source should equal the printed circuit board track impedance.

- Parallel termination: A resistor is connected between the signal and ground. This resistor is placed as close as possible to the load (eg. the debug header). The value of the resistor is equal to the printed circuit board track impedance.

5 ETHERNET CONFIGURATION

This section describes the mechanism used by the Tracelink device to transact data over an Ethernet network. It primarily focuses on the User Datagram Protocol (UDP), which is a popular method for sending data over a network when the speed of a data transaction is of more concern than the guarantee of its delivery. The Tracelink takes advantage of the UDP protocol's penchant for speed, and adds an extra layer of logic to guarantee the delivery of UDP packets in order to offer a best-of-both-worlds solution.

5.1 Network Architectures

Before delving into the innards of Ethernet message passing, it is prudent to briefly describe the different network architectures in use today, and how they pertain to the operation of the Tracelink. Computers are, of course, connected to one another through intermediary devices in order to form networks. There are several classes of these intermediary devices, but they generally fall into one of the following three groups:

Hubs

At the most basic level, computers are connected to one another through a Hub. A Hub is a device with several ports that are used to connect multiple computers together. It is a repeater device – a Hub simply copies the data incoming on one port as data outgoing on the other ports. In this manner, if there are four computers connected through a Hub, and if the first computer is sending data to the second computer, then the third and the fourth computers will also receive an identical copy of that data. Hubs are usually used to set up a small Local Area Network (LAN), which may have on the order of 10 to 20 computers.

Switches

The aforementioned type of process, where the data is simply replicated onto every available port, quickly becomes inefficient for larger sized networks. For this reason, a larger sized LAN employs the usage of Switches instead of Hubs. A Switch is essentially a smart Hub, in that it limits the input and output of data to the two transacting computers.

Routers

Larger networks, such as Wide Area Networks (WANs), or the Internet for that matter, use progressively more sophisticated devices to transact data. At the core of these devices is the Router, which functions as a switch between networks.

The Tracelink performs irrespective of the connection mechanism, with one very important caveat: it needs to be set up with the appropriate network parameters for the underlying network architecture.

5.2   Network Parameters

A typical network becomes operational not after the physical connections have been established, but after network parameters in the form of IP (Internet Protocol) numbers have been assigned to the individual computers. An IP number is a unique string that consists of four numbers ranging between 0 and 255, separated by dots, e.g., 192.168.1.2. Every computer that is on a network needs to have a unique IP number. The computer uses this IP number to identify itself on the network, and also to address the recipient of its data.

Assignation of this IP number is sufficient information to transact data on a simple network connected by a hub. On a more complex network, however, routing information becomes important. The routing information consists of two more IP numbers. The first of these is called the Subnet Mask, and is used to determine whether or not the destination address resides on the same subnet (i.e., doesn't need to be forwarded to another network). The other IP number is the Gateway Address, which is the address of the computer that handles forwarding and receiving of packets to and from other networks.

Before first use, the Tracelink needs to be programmed with a unique IP number, the Subnet Mask IP number, and also the default Gateway's IP number. This can be done via the USB port and is described in greater detail in the "Configuring the Tracelink" section of this manual.

5.3 Internet Protocol

Once the network has been established, and the IP numbers have been assigned, data can be transacted over a network with one of several protocols. By far the most prevalent protocol is the Transmission Control Protocol (TCP), which runs on top of the Internet Protocol in what is collectively known as the TCP/IP protocol. The TCP/IP protocol was developed by the Department of Defense to connect different computers from different vendors by a "network of networks," which has become what is known as the Internet today.

The primary purpose of the TCP/IP protocol was to prevent a complete network outage in the case of a nuclear attack, by automatically rerouting data traffic through the functioning part of the network. As such, the TCP/IP mechanism guaranteed delivery of data packets by introducing a system of acknowledgements and sequence numbers for the data packets. This mechanism, while good for transacting large amounts of data (such as email or file transfers), is unsuitable in the real-time type environment in which the Tracelink operates. Because the Tracelink needs to transact data as quickly as possible to the target, it takes advantage of TCP/IP's alternative, the UDP/IP protocol.

Unlike TCP/IP, the UDP/IP protocol is a connectionless, single-packet protocol that sends short data packets at the expense of not guaranteeing their delivery. This makes the UDP/IP protocol efficient in real-time applications such as broadcasting video over the Internet, where the occasional loss of a frame of data is not going to hamper the overall viewing experience. Left unmodified, the UDP/IP, with its lack of guarantees for packet delivery, would be unusable in an environment where the delivery of a single byte of data needs to be guaranteed. The Tracelink firmware adds mechanisms to the UDP/IP protocol, without affecting its underlying efficiency, to guarantee delivery of data packets.

5.4 Connecting The Tracelink Device

There are two methods for establishing a connection between a Tracelink and a PC with an Ethernet cable. The most basic method is to connect the Tracelink directly to a PC, via a cross-over Ethernet cable. However, the more common method is to place the Tracelink and the PC on the same network through a Hub.

5.4.1 Connecting the Tracelink to the PC over a network:

The Tracelink was intended for use on a network of multiple computers (and other Tracelinks). There are many possible network configurations, and to describe them all is beyond the scope of this document. However, most configurations are a

modification of a basic theme, which is that of connecting one or more PCs through a Hub to one or more Tracelinks.

In order to connect these devices to the Hub, you will need to use the provided straight-through Ethernet cable. The straight-through cable, which is the "standard" Ethernet cable, is used to connect devices of different types together, such as a PC to a Hub, or a Hub to a Tracelink.

At this point it once again becomes necessary to program the Tracelink with valid IP numbers, the process for which is described in greater detail in the following section. However, it is important for the Tracelink and the PCs to have matching Subnet and Gateway IP numbers, and for each to have a unique IP number on the network. An example of a setting for above is as follows:

	IP Number	**Gateway IP**	**Subnet Mask**
PC1	192.168.100.1	192.168.100.3	255.255.255.0
PC2	192.168.100.2	192.168.100.3	255.255.255.0
Tracelink	192.168.100.4	192.168.100.3	255.255.255.0
Gateway	192.168.100.3	192.168.100.3	255.255.255.0

It is important to briefly touch upon the underlying network architecture, which can be a 10Mb (Megabit), 100Mb, 10/100Mb, half-duplex, or a full-duplex connection. The details of the underlying network architecture are beyond the scope of this document, but it is sufficient to note that most modern network cards, as well as the Tracelink device, have the capability to configure themselves for the underlying network through the Auto-negotiation mechanism. Auto-negotiation is performed as soon as a network cable is connected to the device, and it sets the operating parameters of the device to match those of the network.

5.4.2 Connecting Tracelink-to-PC via an Ethernet cable

In order to connect the Tracelink to a PC directly via an Ethernet cable, you need to use what is known as a cross-over cable. A cross-over cable, which is not provided by P&E, is normally used to connect two similar devices such as a PC to a PC, or a Hub to a Hub. It is a cable that has its receive and transmit wires crossed over so that the similar devices can effectively communicate with one another.

With this configuration, it is still important to assign IP numbers to both the PC and the Tracelink device. Although at first glance it may not seem necessary to assign a Gateway address in this configuration, the Tracelink was designed to operate on a network of more than two computers, and therefore it needs to be programmed with a Gateway address.

Assuming the desktop's IP number to be 192.168.100.1, this is an example of the three IP numbers that would need to be programmed into the Tracelink:

|	IP Number	Gateway IP	Subnet Mask
PC	192.168.100.1	none	255.255.255.0
Tracelink	192.168.100.2	192.168.100.1	255.255.255.0

For more information on programming these IP numbers into the Tracelink device, please see the following section.

5.5 Tracelink IP Setup Utility User Interface (ConfigureIP)

Before the Tracelink device transacts data on an Ethernet network, it will need to be configured with the relevant network parameters. The application that provides this capability is the Tracelink IP Setup Utility (ConfigureIP), which can be found as part of the distribution software.

This utility is used to configure the Tracelink with network parameters, and also to update the firmware of the Tracelink.

Figure 5-1: IPSetup.exe Default Screen
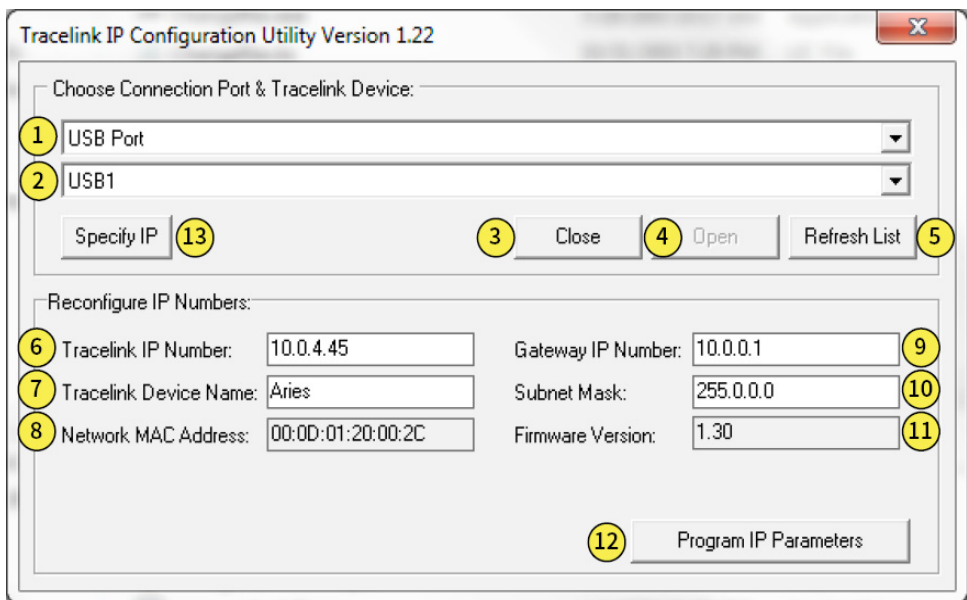
(**1) Drop-down Box 1**

There are two options available in this drop-down box: "Ethernet Port" and "USB Port". Changing to any one of these Ports will list the devices which are found over that specific Port.

(2) Drop-down Box 2

Once one of the communication interfaces (USB or Ethernet) has been selected in the first drop-down box, a list of all available Tracelink devices over that interface will be displayed for selection.

(3) Close Button

The "Close" button is active only when a device has been opened for access. Once a device has been opened for access, it needs to be closed before another device can be opened for access.

(4) Open Button

The "Open" button opens a device for access. This is a required step before changing the parameters on the selected device. Once a device has been selected through the second drop-down box and is opened for access, its information will be displayed at the bottom of the dialog box.

(5) Refresh List

Will refresh the dialog boxes by searching for devices which are currently connected via the USB or Ethernet interfaces.

(6) Tracelink IP Number

This is the IP number which will be associated with the Tracelink. It needs to be a unique IP number which can be accessible on the network.

(7) Tracelink Device Name

This is a label which can be used to identify the Tracelink by name, e.g., "John's Tracelink."

(8) MAC Address

This is the Media Access Control address, the unique number of an Ethernet device on the network. This is programmed by P&E and cannot be modified.

(9) Gateway IP Number

The IP number of a gateway on the network.

(10) Subnet Mask

The subnet mask of the network.

(11) Firmware Version

A read-only field which returns information pertaining to the build date and firmware version of the Tracelink device.

(12) Program Tracelink Parameters

This button saves the information as it appears in the "Reconfigure IP Numbers" area onto the Tracelink device.

(13) Specify IP Button

If the Tracelink cannot be automatically detected over the network, this button allows the user to manually specify the Tracelink IP address.

5.6 Using ConfigureIP.exe To Configure The Tracelink

Before the Tracelink is ready to communicate over an Ethernet network, it will need to be configured with the relevant network parameters. The application that provides this capability is the Tracelink Configuration Utility (IPSetup.exe), and is provided as part of the standard Tracelink software distribution.

In order to update the network parameters, perform the following steps:

1. Connect a Tracelink to the PC via a USB cable, and make sure that it is powered before launching the Tracelink Configuration Utility. The Tracelink Configuration Utility starts up with the following screen:

Figure 5-2: Tracelink IP Setup Utility - Initial Screen
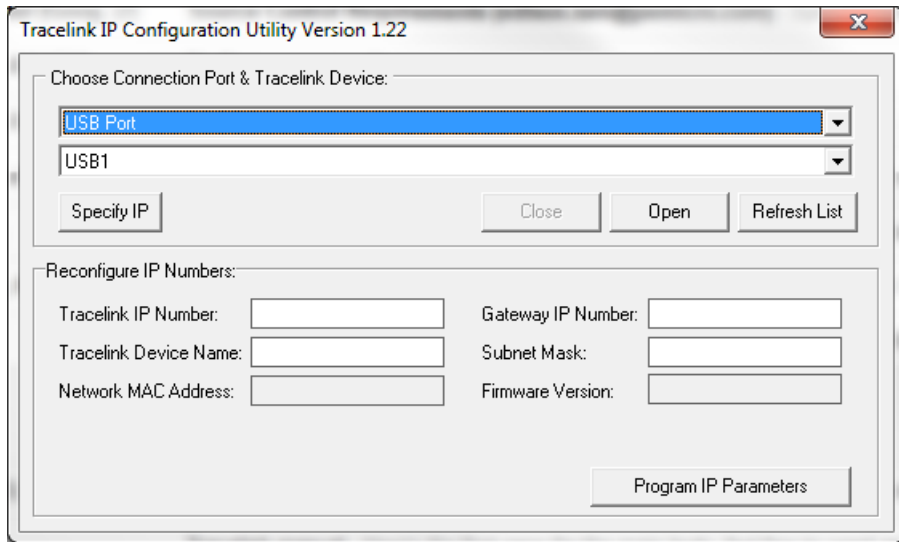
2. Click "Open" to get a dialog box similar to the following:

Figure 5-3: Tracelink IP Setup Utility - Continue Setup
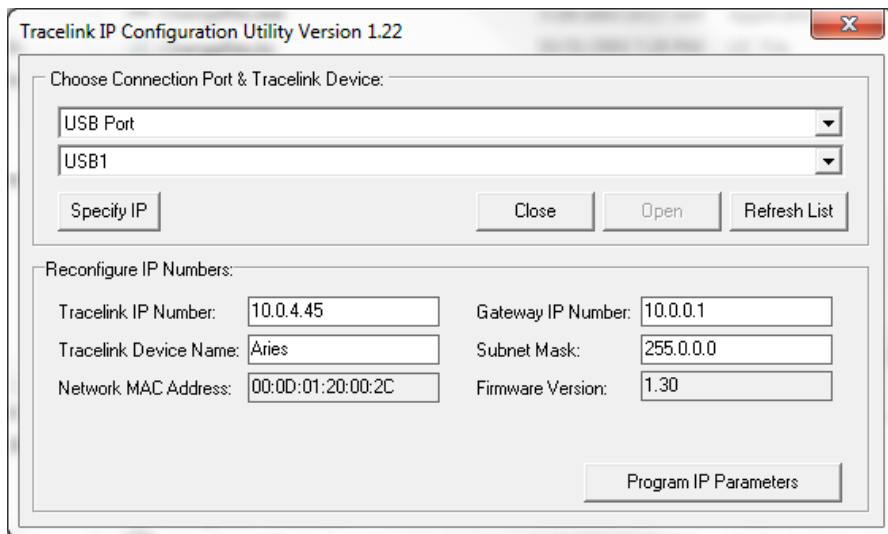
3. The Tracelink now needs to be programmed with IP numbers for the network on which it will operate. The Tracelink IP Number field must contain a unique IP number.

6   USB CONFIGURATION

Standard USB cables may be used for Tracelink USB port configuration. The user may use network hubs as necessary.

7 CODEWARRIOR 10.x CONFIGURATION

This section provides a walkthrough of the configuration steps required to begin using the Tracelink in Freescale's CodeWarrior 10.x development suite. CodeWarrior 10.2 or higher (with all available updates installed) is required. Before following these steps, the Tracelink should already be connected to the target board and both should be powered up.

Step 1. Project Creation

To create a new project, click on File -> New -> Bareboard Project.

After providing a project name and selecting the appropriate Freescale microprocessor, you will be prompted for a run control device:

Figure 7-1: CodeWarrior: Select Run Control Device
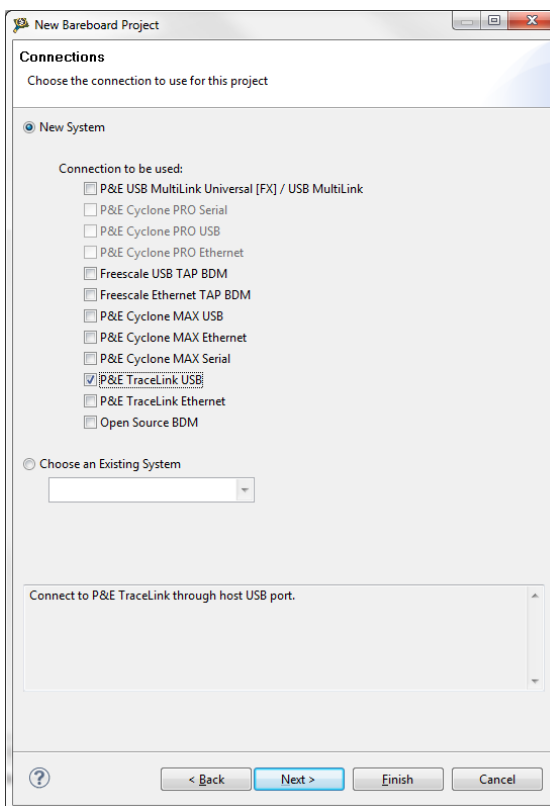
Select the "P&E TraceLink USB" and/or "P&E TraceLink Ethernet" option. Continue with the remaining steps in the wizard to finalize the project.

Step 2. Add Initialization Code

If necessary, add any device specific initialization code to configure the microprocessor for trace generation. For example, the trace pins may be configured as general purpose input/output by default and you may need to add a few lines of code to initialize them for trace output. Once the changes are made, make sure to compile/build the project.

Step 3. Configure Trace Settings

Click on Run -> Debug Configurations. Select the appropriate configuration (Flash, RAM, etc.) from the left panel. Within the "Main" tab, click on the "Edit" button inside the "Connection" groupbox.

Figure 7-2: CodeWarrior: Select Configuration (Edit Button Highlighted)
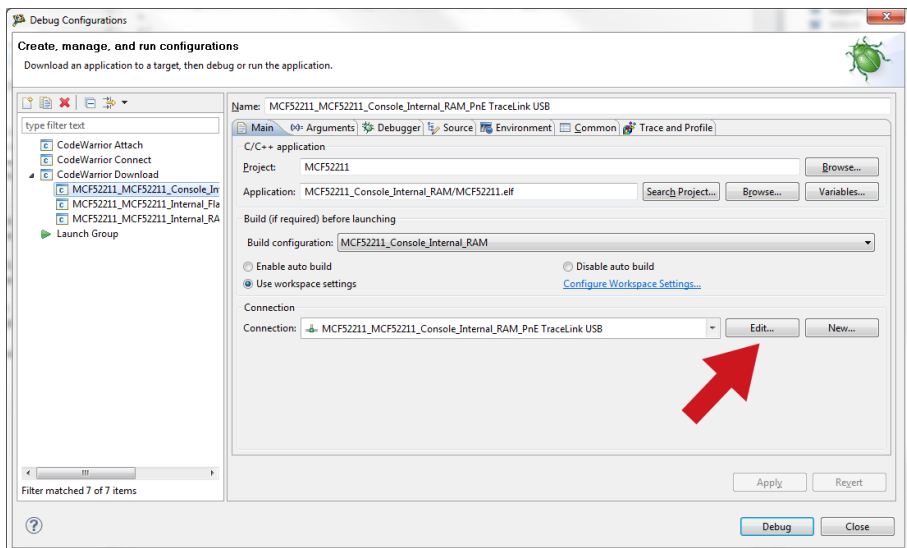
Inside the P&E connection dialog that appears, look for the "Trace Max Buffer Size"

option. This controls the size of the Tracelink's internal buffer. By default, this is set to the lowest value (128KB). As this capacity is increased, the Tracelink can store more trace data, but PC download and processing times are also increased. P&E recommends using one of the center values such as 2MB and increasing it as necessary. Click "OK" to apply the settings.

Figure 7-3: CodeWarrior: Set Trace Max Buffer Size
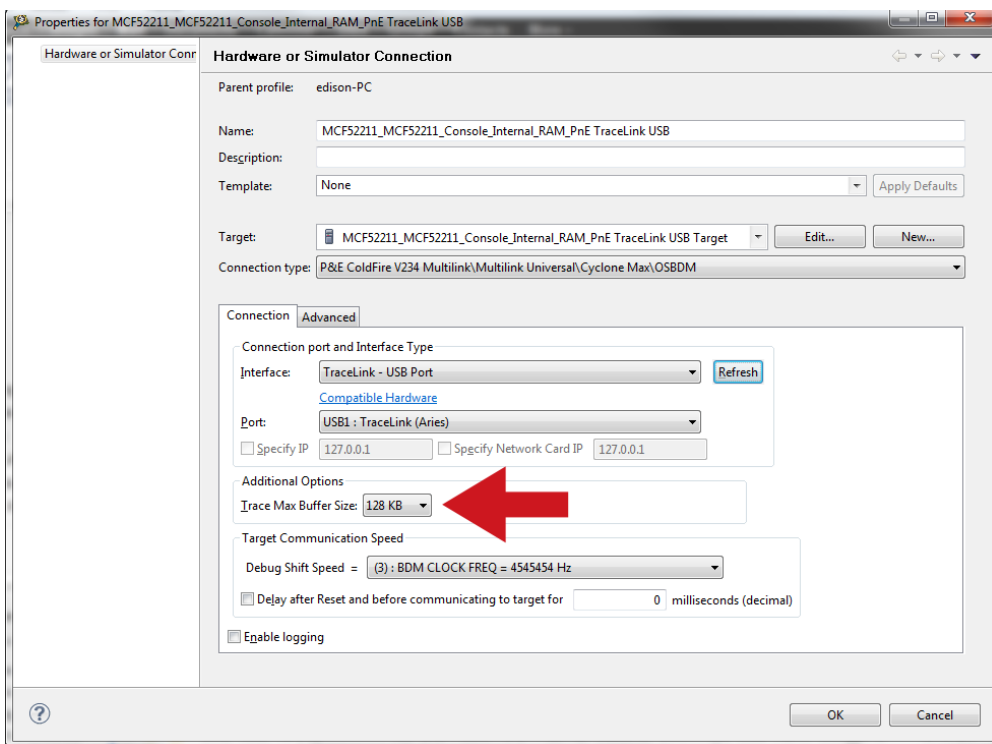
Navigate to the "Trace and Profile" tab and check the "Enable Trace and Profile" checkbox. There are additional device specific settings in this panel that the user should review. These settings control what type of trace filtering will be applied by the CodeWarrior debugger. Click "Apply" and then "Debug" to launch the debug session.
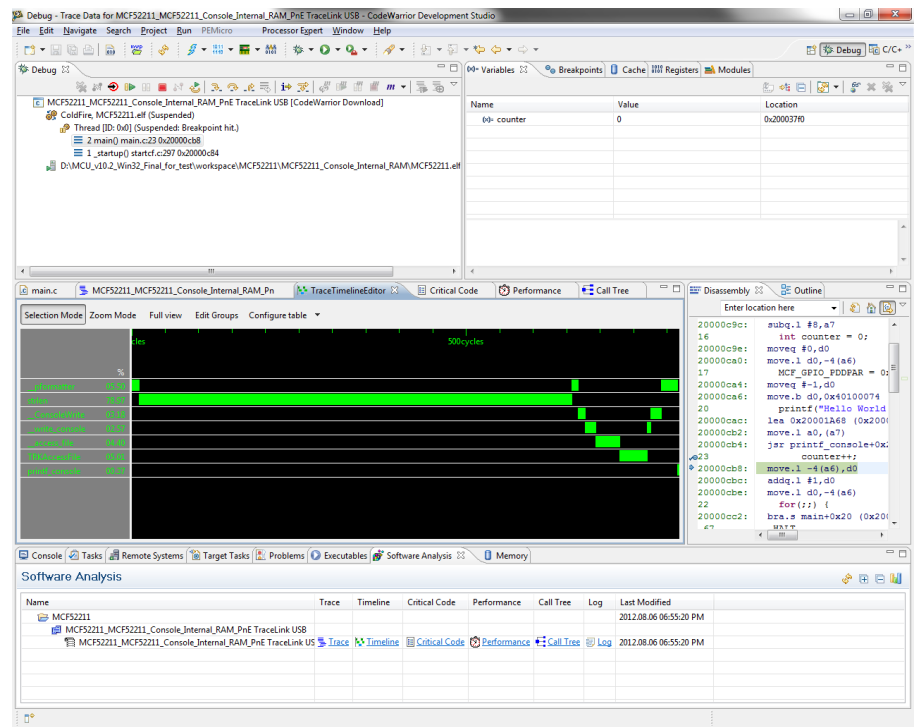
Figure 7-4: CodeWarrior: Enable Trace and Profile
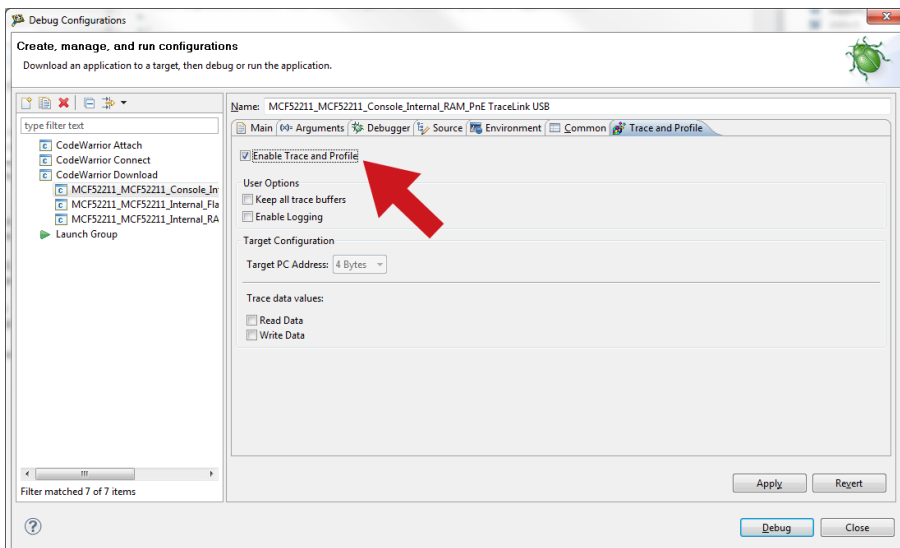
Step 4. Run the project

When the microprocessor halts (eg. after a single-step, run/breakpoint, or run/user halt), CodeWarrior will query the Tracelink for trace data. This data is automatically downloaded and parsed by the CodeWarrior software analysis engine. The results are displayed in the "Software Analysis" tab. Clicking on the individual links (Trace, Timeline, Critical Code, Performance, Call Tree) will display the corresponding view of the trace data.

Figure 7-5: CodeWarrior: Display Trace Data

8   PRODUCTION PROGRAMMING

Once you have accomplished the development stage of your project, you may need a hardware tool to allow you to easily execute low- or high-volume production programming. P&E's Cyclone MAX and Cyclone PRO are versatile and robust programming tools with advanced features and production capabilities. Each supports a different set of Freescale architectures. They each feature multiple communications interfaces (USB, Ethernet, and Serial), stand-alone programming functionality, and many other advanced capabilities. They work with Freescale's CodeWarrior as well as P&E's software to help provide a seamless transition to programming.

For more information about the Cyclones, please visit us at **pemicro.com/cyclonemax** or **pemicro.com/cyclonepro**.

Figure 8-1: Cyclone PRO & Cyclone MAX Stand-Alone Programmers

8.1   Cyclone Key Features

- Advanced programming and debugging capabilities, including:
- PC-Controlled and User-Controlled Stand-Alone Operation
- Interactive Programming via Host PC

- In-Circuit Debugging, Programming, and Testing
- Compatible with many Freescale microcontroller families:

   Cyclone MAX
   - **ColdFire® V2/V3/V4**
   - **Power® MPC5xx/8xx**
   - **Qorivva® MPC55xx/56xx**
   - **Kinetis® ARM**
   - **Power® Architecture PX Series**
   - **DSC (MC56F80xx, MC56F82xx, MC56F83xx, and MC56F84xx)**
   - **ARM (MAC7xxx)**

   Cyclone PRO
   - **HC08**
   - **HCS08**
   - **S12Z (coming soon)**
   - **HC(S)12(X)**
   - **RS08**
   - **ColdFire +V1/ColdFire V1**

- Communication via USB, Serial, and Ethernet Ports
- Multiple image storage
- LCD screen menu interface
- Supported by P&E software and Freescale's CodeWarrior

8.2   Cyclone In-Depth

P&E's Cyclones are extremely flexible tools designed for debugging, testing, and in-circuit flash programming of Freescale microcontrollers. The Cyclone connects your target to the PC via USB, Ethernet, or Serial Port and enables you to debug your code, program, and test it on your board. After development is complete the Cyclone can be used as a production tool on your manufacturing floor.

For production, the Cyclone may be operated interactively via Windows-based programming applications as well as under batch or .dll commands from a PC. Once loaded with data by a PC it can be disconnected and operated manually in a stand-alone mode via the LCD menu and control buttons. The Cyclone has over 7 Mbytes

(MAX) or 3Mbytes (PRO) of non-volatile memory, which allows for the onboard storage of multiple programming images. When connected to a PC for programming or loading it can communicate via the ethernet, USB, or serial interfaces.

8.2.1 Software

The Cyclone comes with intuitive configuration software and interactive programming software, as well as easy to use automated control software. The Cyclone also functions as a full-featured debug interface, and is supported by Freescale's CodeWarrior as well as development software from P&E.

P&E's Cyclones are also available bundled with additional software as part of various Development Packages. In addition to the Cyclone programming hardware, these Development Packages include in-circuit debugging software, flash programming software, a Windows IDE, and register file editor.

8.2.2 Enhancements

P&E also offers add-on features that can increase the Cyclone's versatility.

- The CompactFlash port can be activated, which enables a very useful memory expansion, making it easier to manage larger and/or multiple orgramming images.

- The Cyclone Automated Control Package allows you to control and fully automate multplie Cyclones for efficient gang programming capability.

- The Cyclone Power Pack gives you the ability to power the Cyclone out in the field where a standard power source may be hard to come by. This allows you to ge the most out of the Cyclone's stand-alone programming ability.

For more information about P&E's Cyclones, please visit us at **pemicro.com/cyclonemax** or **pemicro.com/cyclonepro**.

P&E Mircomputer Systems, Inc.
98 Galen St.
Watertown, MA 02472

www.pemicro.com