CodeWarrior™
USB TAP Users Guide

Revised: 11 November 2005

freescale semiconductor

How to Contact Us

Corporate Headquarters	Freescale Semiconductor, Inc. 7700 West Parmer Lane Austin, TX 78729 U.S.A.
World Wide Web	http://www.freescale.com/codewarrior
Technical Support	http://www.freescale.com/support

Table of Contents

1

Introducing the CodeWarrior™ USB TAP Emulator

The CodeWarrior USB TAP emulator is a cost-effective tool that helps you develop and debug a number of processors and microcontrollers.

This chapter introduces you to the USB TAP emulator.

The sections are:

What is the USB TAP Emulator?

The USB TAP emulator uses advanced emulation technology to provide control of and visibility into your target system. Combined with the CodeWarrior IDE, the USB TAP emulator (Figure 1.1) speeds the debugging process by letting you interactively control and examine the state of your target system.

Figure 1.1 USB TAP for ARM®JTAG with USB 2.0 Cable

Product Highlights

The USB TAP emulator has these features:

- Supports the following systems: PowerPC™ processors, ARM® processors, ColdFire™ processors, StarCore™ processors, 56800 Hybrid Controllers (processors and microcontrollers), and RCF base-band controllers.
- Supports Windows® NT/2000/XP and the following Linux operating systems: Red Hat® Linux® version 8.0 or 9.0, Fedora™ Core 3 or Core 4, and Mandrake®Linux® 10.0.
- Supports all CPU core speeds.
- Lets you control and debug software running in-target, with minimal intrusion into target operation.
- Lets you debug code in cache, ROM, RAM, and flash memory.
- Provides high performance:
  - Split-second single-step execution.
  - Capable of download speeds greater than 12 MB per minute from host to target.

NOTE The actual download speed depends on the target processor, the debug port's clock frequency, the network speed, and the debugger.

- Supports one USB 2.0 connection.
- Supports both big and little endian byte-order.

- Automatically supports target signal levels from 1.8V to 3.3V.
- Software debug capabilities including:
  - Controlling instruction execution.
  - Display and modify target memory.
  - Examine and modify any processor registers.
  - Run to breakpoints in ROM, RAM, or flash memory.
  - Single-step through source and assembly language code views.
  - Single-step into, over, or out of functions.

The Debugging Environment

The USB TAP emulator works with the CodeWarrior™ debugger. The debugger has been integrated with the USB TAP emulator to give you control over the emulation functions and your target.

USB TAP Emulator Benefits

The USB TAP emulator provides these key benefits:

- *Visibility*: The USB TAP emulator makes it possible for you to observe registers and the current state of target memory. You can halt program execution at predefined states and examine the data for a particular program state.
- *Control*: You can conveniently control the state of the target system by downloading code, manually modifying processor registers and memory, single-stepping through the code, or setting breakpoints.

Target Connections

The USB TAP connects to your target through the standard debug port for the processor family. Each USB TAP supports a single target connection. The USB TAPs are available in the following versions:

- JTAG/COP, for PowerPC™ targets
- DPI, for PowerPC™ 8xx targets
- BDM, for ColdFire™ targets
- OnCE, for StarCore™, 56800 Hybrid Controllers, RCF base-band, and PowerPC™ 5500 targets
- ARM® JTAG, for i.MX and MAC7100 targets

For more details see “Connecting to the Target and your Host Computer” on page 13.

Operating Requirements

Before setting up your system, you should make sure that the operating environment is prepared.

Standard Electrostatic Precautions

This instrument contains static-sensitive components that are subject to damage from electrostatic discharge. Use standard ESD precautions when transporting, handling, or using the instrument and the target, when connecting/disconnecting the instrument and the target, and when removing the cover of the instrument.

We recommend that you use the following precautions:

- Use wrist straps or heel bands with a 1 MΩ resistor connected to ground.
- On the work surface and floor, use static conductive mats with a 1 MΩ resistor connected to ground.
- Keep high static-producing items, such as non-ESD-approved plastics, tape and packaging foam, away from the instrument and the target.

The above precautions should be considered as minimum requirements for a static-controlled environment.

Electrical Requirements

The USB TAP emulator is powered through the USB cable and does not use an external power supply. The USB TAP emulator is designed to be plugged directly into a host computer, but also can work with self-powered hubs. Bus powered hubs may be unable to provide sufficient power for the USB TAP, which requires 200 mA. If insufficient power is available, your host operating system will indicate this failure and the USB TAP emulator will go into a low power suspend mode. If your hub is not able to provide sufficient power, connect the USB TAP directly to your host PC, or purchase a self-powered USB hub.

Operating Temperature

The USB TAP emulator can operate in a temperature range of 0 to 40 °C (32 to 104 °F).

Target Requirements

The USB TAP emulator automatically supports target signal levels from 1.8V to 3.3V.

NOTE In the case of the PowerPC, for the USB TAP emulator to properly stop and restart a JTAG/COP target processor, the $\overline{\text{QACK}}$ signal must be pulled low. The USB TAP emulator pulls this signal low through the JTAG/COP connector.

Related Documentation

This manual describes the procedures for unpacking the USB TAP emulator, setting up USB communications, and connecting the USB TAP emulator to your target system.

The CodeWarrior documentation explains how to install and configure the CodeWarrior IDE and debugger.

2

Connecting to the Target and your Host Computer

To test your software using the USB TAP emulator, you must have working target hardware.

This chapter explains how to connect the USB TAP emulator to such hardware and to your host computer.

The sections are:

- Debug Port Connector Information
- Setting the Debug Port Clock Frequency
- Connecting to your Host Computer
- What To Do Next

CAUTION The USB TAP emulator contains components that are subject to damage from electrostatic discharge. Whenever you are using, handling, or transporting the USB TAP emulator, or connecting to or disconnecting from a target system, always use proper anti-static protection measures, including using static-free bench pads and grounded wrist straps.

Debug Port Connector Information

The USB TAP emulator is a powerful development tool for use with a wide variety of processors that use either JTAG/COP, DPI, BDM, OnCE or ARM® JTAG debug interfaces

The appendices describe the debug port connector informations for:

- “JTAG/COP Connector Information” on page 21.
- “DPI Connector Information” on page 27.
- “BDM Connector Information” on page 31.
- “OnCE Connector Information” on page 37.
- “ARM® JTAG Connector Information” on page 41.

Target Connection

CAUTION	Failure to properly connect the USB TAP emulator to the target may damage the USB TAP emulator or target. Verify all connections before applying power.

If your target system has a debug port header, you can directly connect the USB TAP emulator to the target debug port header. Make sure that pin 1 of the USB TAP's gray ribbon cable connector lines up with pin 1 on the target's debug port header.

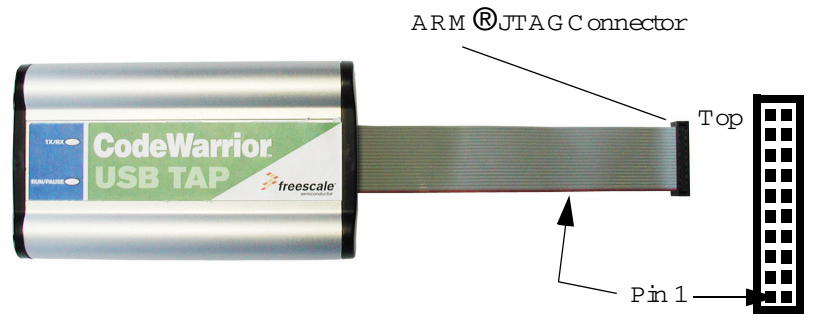
NOTE	Pin 1 is clearly marked on the USB TAP's gray ribbon cable by a red line down one side of the cable and a small triangle in the plastic socket.

To connect the USB TAP connector to the target debug port header:

1. Remove power from your target.
2. Unplug the USB cable from the USB TAP emulator.
3. Make sure that pin 1 of the USB TAP's gray ribbon cable connector lines up with pin 1 on the target's debug port header.

   As an example, Figure 2.1 shows the USB TAP for ARM® JTAG connector.

Figure 2.1 USB TAP for ARM® JTAG



4. Gently (but firmly) press the USB TAP connector onto the target debug port header.
5. Reconnect the USB cable to the USB TAP emulator.
6. Apply power to your target.

Setting the Debug Port Clock Frequency

The debug port on your target is a synchronous interface clocked by the TCK or DSCK signal. The base frequency is set by the debugger and can be adjusted with preferences in the debugger.

NOTE For directions on how to set the debug port clock frequency, please see the CodeWarrior documentation.

Some slow targets might not be able to operate at the initial default rate. Therefore, you may have to adjust the debug port clock rate.

NOTE Because of variations in the design of target systems, it is not possible to guarantee that all systems can be operated at the maximum debug port clock rates. These variations include circuit impedances, trace lengths, and signal terminations. You may need to select a lower clock rate to get reliable operation.

Connecting to your Host Computer

Connect one end of your USB 2.0 Cable to a USB port on your host computer. Connect the other end of the cable to your CodeWarrior USB TAP emulator.

What To Do Next

You can now install your software. See the *IDE Users Guide* or your targeting manual about how to configure the debugger and run a confidence test.

For additional information about using the USB TAP with your target system, refer to "Using the USB TAP" on page 15.

3

Using the USB TAP

This chapter provides system startup procedures, explains how the USB TAP works and provides important information about using the system.

The sections are:

- USB TAP System Startup
- Notes on Using the USB TAP

USB TAP System Startup

This section explains how to start using the USB TAP and the debugger.

Before starting up the USB TAP, make sure you have:

- Connected the USB TAP to your Host Computer with your USB 2.0 Cable
- Connected the USB TAP to the target system (see "Connecting to the Target and your Host Computer" on page 11)
- Installed the USB TAP drivers
- Installed the debugger software and properly configured it to communicate with the USB TAP.

To start up the USB TAP:

1. Apply power to your target system.
2. Start the debugger.

You are now ready to begin your debug session. For information on using the debugger, refer to the *IDE User Guide* and the targeting manual.

Notes on Using the USB TAP

This section provides information specific to the USB TAP. This section is useful for users who are not familiar with USB TAP operation. Also refer to the debugger documentation for becoming familiar with the system operation.

These topics provide information specific to USB TAP operation:

- Run/Pause/Mixed Mode States
- Breakpoints in Exception/Interrupt Handlers

Run/Pause/Mixed Mode States

When the host debugger is connected to the target via the USB TAP, the USB TAP is always in one of these states (modes): *run, pause* or *mixed mode*.

- Run mode — in this mode, all target system processor cores execute the target code
- Pause mode — in this mode, all target system processor cores have stopped executing the target code
- Mixed mode — in this mode, some target system processor cores are in run mode and others are in pause mode

The Run/Pause light on the USB TAP will be green in run mode, red in pause mode, and orange in mixed mode.

Breakpoints in Exception/Interrupt Handlers

Care must be taken when setting breakpoints in exception handler code. A typical exception consists of a preamble that saves processor context, the actual exception handler, and then a postamble that restores processor context. The rule is you can use software breakpoints in the actual exception handler code, but not in the code where processor context is being saved.

For PPC processors, placing the CPU into debug mode is just another interrupt. For example: your code is in an interrupt epilogue and has just placed the return address into SRR0 when a breakpoint occurs. The breakpoint causes the IP for the address of the breakpoint to be written to SRR0, destroying your original return address. Stepping through code which accesses SRR0 and SRR1 exhibits the same problem.

To avoid this problem, always set your breakpoints before or after code which accesses SRR0 and SRR1, and never step through such code. For example, you can set your breakpoint anywhere after the interrupt prologue, but before the epilogue.

Instructions that involve the SRR0 and SRR1 registers are “MTSPR SRR0/1,Rx” “MFSPR Rx,SRR0/1,” and “RFI.”

4

Hardware Specifications

This chapter provides hardware specifications for the USB TAP emulator.

The sections are:

- Connectors and LEDs
- USB TAP Specifications

Connectors and LEDs

Figure 4.1 and Figure 4.2 show the various LEDs and connectors of the USB TAP device.

Figure 4.1 USB TAP Device—Top View

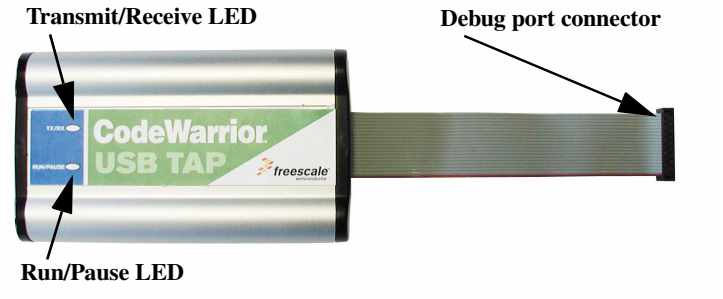
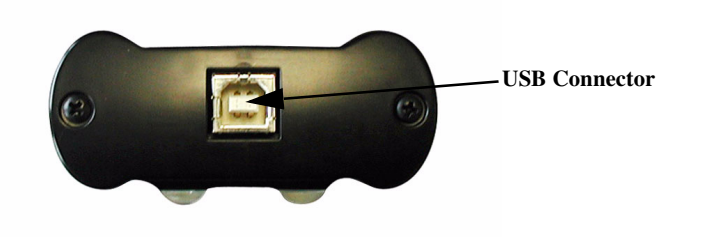


Figure 4.2 USB TAP Device Connector—End View



These topics explain the various LEDs and connectors on the USB TAP device:

Run/Pause Indicator

The USB TAP hardware has a status LED labeled Run/Pause. It indicates the state of execution on the target device as follows:

- The LED is green when the target is running.
- The LED is red when the target is paused.
- The LED is orange when the target is in mixed mode.
- The LED is initially unlit and remains so until the debugger is connected to the USB TAP.

Transmit/Receive Indicator

The transmit/receive (labeled TX/RX) LED indicates the state of the USB interface as follows:

- The LED flashes red when the USB TAP is powered but has not been configured.
- The LED flashes green when the USB TAP is properly configured.
- The LED flashes orange when data is being transferred.
- The LED is off if the USB TAP is disabled or disconnected.

NOTE When the USB TAP is powered on, it performs a brief self-test that displays a test pattern on both LEDs. After the self-test completes, it reverts to the behavior described above.

USB Connector

The USB interface consists of an USB type B connector that connects directly to the provided USB cable.

Debug Port Connector

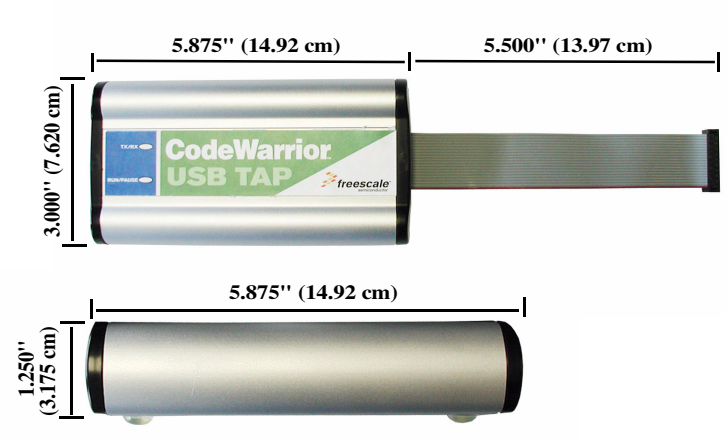
The debug connector consists of a 5.5 inch ribbon cable with the appropriate debug connector attached. The ribbon cable has a red stripe down one side to indicate the location of pin 1.

NOTE The OnCE connector is equipped with a removable plug in pin 8. This follows the keying convention for the OnCE header (pin 8 should be removed). This plug is removable, in case the target being connected to it does not come with pin 8 removed.

USB TAP Specifications

Figure 4.3 shows the dimensions of the USB TAP device.

Figure 4.3 USB TAP Dimensions



Electrical Characteristics

The USB TAP was designed to affect the target processor and target electrical characteristics as little as possible. Care, however, should be taken in designing the target to accommodate the small signal delays associated with any in-circuit emulator or other test equipment.

The USB TAP automatically supports target signal levels from 1.8V to 3.3V.

Physical Considerations

The USB TAP was carefully designed to be as small as possible. Even so, it may not physically fit in all target systems.

Contact Freescale if you have any special considerations that need review.

Table 4.1 shows the physical characteristics of the USB TAP.

Table 4.1 USB TAP—Physical Characteristics

Current consumption	
• USB TAP current consumption from USB cable	< 200 mA
• USB TAP current consumption from target	< 50 µA
Physical dimensions	
• USB TAP dimensions	5.875" x 3.000" x 1.250" (14.92 cm x 7.620 cm x 3.175 cm)
• Target connector ribbon length	5.50” (13.97 cm)
• Target connector dimensions Height Depth JTAG/COP width DPI width ARM® JTAG width OnCE width BDM width	 0.38" (0.97 cm) 0.20" (0.51 cm 0.98" (2.49 cm) 0.68" (1.72 cm) 1.18" (3.00 cm) 0.88" (2.24 cm) 1.48” (3.76 cm)

A

JTAG/COP Connector Information

The CodeWarrior USB TAP for JTAG/COP has a 16-pin connector. The CodeWarrior USB TAP for JTAG/COP automatically supports target signal levels from 1.8V to 3.3V.

Figure A.1 shows the pin assignments of the CodeWarrior USB TAP for JTAG/COP connector.

Table A.1 lists JTAG/COP signal names, direction, pin numbers, descriptions, and drive capabilities for the CodeWarrior USB TAP for JTAG/COP Connector.

Table A.2 provides a general description of each JTAG/COP signal and the CodeWarrior USB TAP for JTAG/COP operational requirements.

NOTE All JTAG/COP signals must meet accepted standards for JTAG/COP signal design. To ensure proper and stable operation between the CodeWarrior USB TAP for JTAG/COP and the target, the JTAG/COP signals must meet the requirements listed in Table A.2.

Figure A.1 CodeWarrior USB TAP for JTAG/COP Connector Pin Assignments

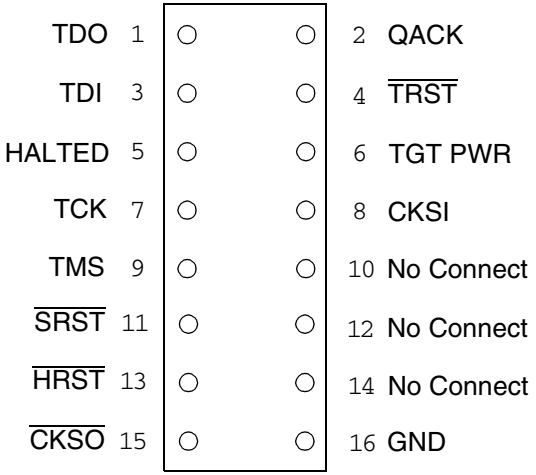


Table A.1 CodeWarrior USB TAP for JTAG/COP Signal Directions

JTAG/ COP Pin	Signal Mnemonic	Signal Direction	Description
1	TDO	From target	30pF load
2	QACK	From USB TAP connector	100Ohm pull-down
3	TDI	From USB TAP connector	50mA driver
4	$\overline{\text{TRST}}$	From USB TAP connector	50mA driver[1]
5	HALTED	Bi-directional	Open-drain, 100Ohm to ground when asserted by USB TAP emulator, 35pF load when not asserted[2]
6	TGT PWR	From target	2MOhm pull-down, plus 0.01uF load
7	TCK	From USB TAP connector	50mA driver
8	CKSI	From USB TAP connector	50mA driver
9	TMS	From USB TAP connector	50mA driver
10	No Connect	- n/a -	
11	$\overline{\text{SRST}}$	Bi-directional	Open-drain. 100Ohm to ground when asserted by USB TAP, 35pF load when not asserted[2]
12	No Connect	- n/a -	

Table A.1 CodeWarrior USB TAP for JTAG/COP Signal Directions (*continued*)

JTAG/ COP Pin	Signal Mnemonic	Signal Direction	Description
13	$\overline{\text{HRST}}$	Bi-directional	Open-drain. 100Ohm to ground when asserted by USB TAP, 35pF load when not asserted[2]
14	No Connect	- n/a -	
15	$\overline{\text{CKSO}}$	From target	30pF load[2]
16	GND	- n/a -	

[1]100KOhm pull-up to buffered TGT PWR.

[2]4.7KOhm pull-up to buffered TGT PWR.

Table A.2 CodeWarrior USB TAP for JTAG/COP Signal Recommendations/Requirements

JTAG/ COP Pin	Signal Mnemonic	Requirement
1	TDO	Must be wired to the target processor. TDO is an output from the target processor and an input to the USB TAP emulator. The TDO trace run should be kept short and maintain a "two-signal-width" spacing from any other parallel dynamic signal trace. TDO should have a series termination resistor located near the target processor.
2	QACK	May be wired to the target processor. $\overline{\text{QACK}}$ is an input to most PowerPC processors and must remain low while the USB TAP emulator is connected to the target. The USB TAP emulator connects this signal internally to the JTAG/COP GND pin (16) through a 100Ohm resistor.
3	TDI	Must be wired to the target processor. The USB TAP emulator drives the TDI output with up to 50mA. The TDI trace should be kept short and maintain a "two-signal-width" spacing from any other parallel dynamic signal trace. TDI should have an RC termination option at the processor.

Table A.2 CodeWarrior USB TAP for JTAG/COP Signal Recommendations/Requirements

JTAG/ COP Pin	Signal Mnemonic	Requirement
4	TRST	Must be wired to the target processor. The USB TAP emulator drives the $\overline{\text{TRST}}$ output with up to 50mA. To gain control of the processor, the USB TAP emulator negates $\overline{\text{TRST}}$ approximately 250 milliseconds before negation of $\overline{\text{HRST}}$. This allows the USB TAP emulator to issue COP commands through the JTAG/COP interface and gain control of the processor upon negation of $\overline{\text{HRST}}$. The $\overline{\text{TRST}}$ trace run should be kept short and maintain a "two-signal-width" spacing from any other parallel dynamic signal trace.
5	HALTED	Need not be wired to the target. The USB TAP emulator does not currently use this signal.
6	TGT PWR	Must be wired to the target. The USB TAP emulator uses this signal to determine if power is applied to the target. This signal is also used as a voltage reference for the signals driven by the USB TAP emulator ($\overline{\text{CKSI}}$, $\overline{\text{TRST}}$, TCK, TMS, TDI). TGT PWR (pin 6) should be connected to target Vcc through a 1KOhm pull-up.
7	TCK	Must be wired to the target processor. The USB TAP emulator drives the TCK output with up to 50mA. The TCK trace run should be kept as short as possible and maintain a "two-signal-width" spacing from any other parallel dynamic signal trace.
8	CKSI	Need not be wired to the target. The USB TAP does not currently use this signal.
9	TMS	Must be wired to the target processor. The USB TAP drives the TMS output with up to 50mA. TMS should be kept as short as possible and maintain a "two signal width" spacing from any other parallel dynamic signal trace. TMS should have a termination option at the processor.
10	No Connect	Not required for emulation
11	SRST	May be wired to the target processor. During reset, the USB TAP drives $\overline{\text{SRST}}$ to ground through a 100Ohm resistor.
12	No Connect	Not required for emulation

Table A.2 CodeWarrior USB TAP for JTAG/COP Signal Recommendations/Requirements

JTAG/ COP Pin	Signal Mnemonic	Requirement
13	HRST	Must be wired to the target processor. During reset, the USB TAP drives $\overline{\text{HRST}}$ to ground through a 100Ohm resistor.
14	No Connect	Not required for emulation
15	CKSO	Should be wired to the target processor. The USB TAP senses $\overline{\text{CKSO}}$ to determine if the processor halted execution in a checkstop state.
16	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP.

Notes Specific to MPC8240, MPC8241, and MPC8245

$\overline{\text{CKSO}}$ is not available on the MPC8240, MPC8241, or MPC8245 processors.

$\overline{\text{SRST}}$ can be deselected in favor of the SDMA12 signal used for Extended Addressing (except on the MPC8240 processor).

$\overline{\text{QACK}}$ is an output signal only, so it need not be connected to the COP header.

The USB TAP emulator fully supports the MPC8240, MPC8241, and MPC8245 processors despite the absence of the signals mentioned above.

Signal width example

Signal line 'A' is 0.005 mil. An adjacent dynamic signal line 'B' should maintain a "two-signal-widths" distance from signal line 'A'. So that from the center of line 'A' to the center of line 'B', there should be 0.0025 + 0.005 + 0.005 + 0.0025 = 0.015mil.

Target bias should maintain "one signal width" spacing from the signal.

B

DPI Connector Information

The USB TAP for DPI has a 10-pin connector. The CodeWarrior USB TAP for DPI automatically supports target signal levels from 1.8V to 3.3V.

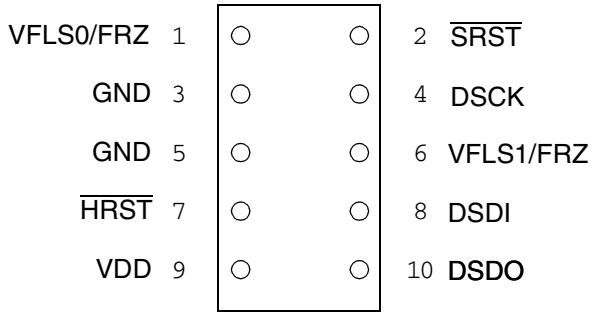
Figure B.1 shows the pin assignments of the CodeWarrior USB TAP for DPI connector.

Table B.1 lists DPI signal names, direction, pin numbers, descriptions, and drive capabilities for the USB TAP DPI Connector.

Table B.2 provides a general description of each DPI signal and the USB TAP for DPI operational requirements.

NOTE All DPI signals must meet accepted standards for DPI signal design. To ensure proper and stable operation between the USB TAP for DPI emulator and the target, the DPI signals must meet the requirements listed in Table B.2.

Figure B.1 CodeWarrior USB TAP for DPI Connector Pin Assignments



To connect the CodeWarrior USB TAP for DPI connector to a DPI header:

1. Turn off power to the target system.
2. Make sure all 10 pins on the USB TAP for DPI connector socket are properly aligned with the DPI header on the target.
3. Gently but firmly press the USB TAP for DPI connector onto the header on the target.

CAUTION Failure to properly connect the USB TAP for DPI emulator to the target may damage the USB TAP for DPI emulator or the target system. Verify all connections before applying power.

Table B.1 CodeWarrior USB TAP Emulator for DPI Signal Directions

DPI Pin	Signal Mnemonic	Signal Direction	Description
1	VLSO/FRZ	From target	30pF load[1]
2	SRST	Bi-directional	Open-drain. 100Ohm to ground when asserted by USB TAP, 35pF load when not asserted[1]
3	GND	- n/a -	
4	DSCK	From USB TAP connector	50mA driver
5	GND	- n/a -	
6	VFLS1/FRZ	From target	35pF load[1]
7	$\overline{\text{HRST}}$	Bi-directional	Open-drain. 100Ohm to ground when asserted by USB TAP, 35pF load when not asserted[1]
8	DSDI	From USB TAP connector	50mA driver
9	VDD	From target	2MOhm pull-down, plus 0.01uF load
10	DSDO	From target	30pF load

[1] 4.7KOhm pull-up to buffered TGT PWR.

Table B.2 CodeWarrior USB TAP for DPI Signal Recommendations and Requirements

DPI Pin	Signal Mnemonic	Requirement
1	VLSO/FRZ	VFLS0/FRZ is not needed for emulation.
2	SRST	Must be wired to the target. During reset, the USB TAP drives SRST to ground through a 100Ohm resistor.
3	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP.

Table B.2 CodeWarrior USB TAP for DPI Signal Recommendations and Requirements

DPI Pin	Signal Mnemonic	Requirement
4	DSCK	Must be wired to the target processor. It is driven by the USB TAP as an output with up to 50mA. This signal is the clock for the DPI interface. It is good design practice to keep the trace length short and isolate the trace from other signals. If the trace must be long, then termination may be needed.
5	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP.
6	VFLS1/FRZ	VFLS1/FRZ is not needed for emulation.
7	$\overline{\text{HRST}}$	Must be wired to the target. During reset, the USB TAP drives HRST to ground through a 100Ohm resistor.
8	DSDI	Must be wired to the target processor. The USB TAP emulator drives the TDI output with up to 50 mA.
9	VDD	Must be wired to the target. The USB TAP emulator uses this signal to determine if power is applied to the target. This signal is also used as a voltage reference for the signals driven by the USB TAP emulator (SRST, SDCK, HRST, DSDI).
10	DSDO	Must be wired to the target processor. DSDO is an output from the target processor and an input to the USB TAP emulator. It is good design practice to keep the trace length short and isolate the trace from the other signals.

C

BDM Connector Information

The USB TAP for BDM has a 26-pin connector. The CodeWarrior USB TAP for BDM automatically supports target signal levels from 1.8V to 3.3V.

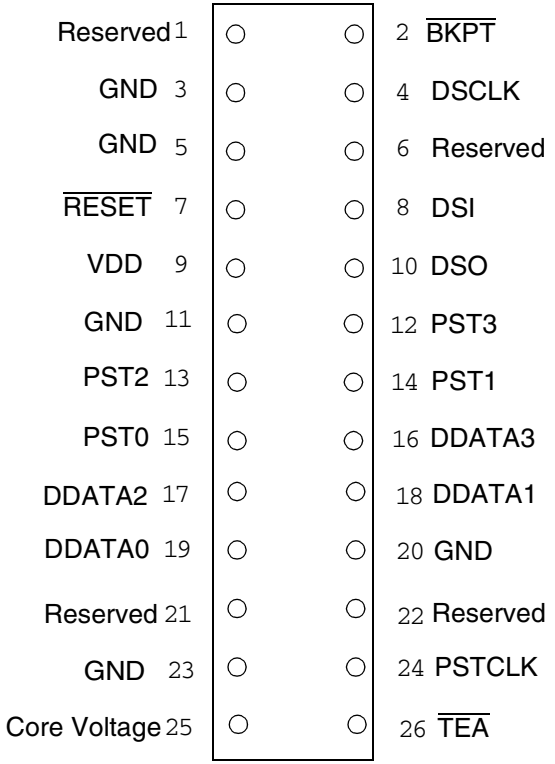
Figure C.1 shows the pin assignments of the CodeWarrior USB TAP for BDM connector.

Table C.1 lists BDM signal names, direction, pin numbers, descriptions, and drive capabilities for the USB TAP BDM Connector.

Table C.2 provides a general description of each BDM signal and the USB TAP for BDM operational requirements.

NOTE All BDM signals must meet accepted standards for BDM signal design. To ensure proper and stable operation between the USB TAP for BDM emulator and the target, the BDM signals must meet the requirements listed in Table C.2.

Figure C.1 CodeWarrior USB TAP for BDM Connector Pin Assignments



To connect the CodeWarrior USB TAP for BDM connector to a BDM header:

1. Turn off power to the target system.
2. Make sure all 26 pins on the USB TAP for BDM connector socket are properly aligned with the BDM header on the target.
3. Gently but firmly press the USB TAP for BDM connector onto the header on the target.

CAUTION Failure to properly connect the USB TAP for BDM emulator to the target may damage the USB TAP for BDM emulator or the target system. Verify all connections before applying power.

Table C.1 CodeWarrior USB TAP Emulator for BDM Signal Directions

BDM Pin	Signal Mnemonic	Signal Direction	Description
1	Reserved	From target	30pF load[1]
2	BKPT	Bi-directional	Open-drain. 100Ohm to ground when asserted by USB TAP, 35pF load when not asserted[1]
3	GND	- n/a -	
4	DSCLK	From USB TAP connector	50mA driver
5	GND	- n/a -	
6	Reserved	Bi-directional	Open-drain, 100Ohm to ground when asserted by USB TAP emulator, 35pF load when not asserted[1]
7	RESET	Bi-directional	Open-drain. 100Ohm to ground when asserted by USB TAP, 35pF load when not asserted[1]
8	DSI	From USB TAP connector	50mA driver
9	VDD	From target	2MOhm pull-down, plus 0.01uF load
10	DSO	From target	30pF load
11	GND	- n/a -	
12	PST3	- n/a -	
13	PST2	- n/a -	
14	PST1	- n/a -	
15	PST0	- n/a -	
16	DDATA3	- n/a -	
17	DDATA2	- n/a -	
18	DDATA1	- n/a -	
19	DDATA0	- n/a -	

Table C.1 CodeWarrior USB TAP Emulator for BDM Signal Directions (*continued*)

BDM Pin	Signal Mnemonic	Signal Direction	Description
20	GND	- n/a -	
21	Reserved	- n/a -	
22	Reserved	- n/a -	
23	GND	- n/a -	
24	PSTCLK	From target	30pF load
25	Core Voltage	- n/a -	
26	TEA	From USB TAP connector	50mA driver[2]

[1]4.7Ohm pull-up to buffered TGT PWR.

[2]100KOhm pull-up to buffered TGT PWR.

Table C.2 CodeWarrior USB TAP for BDM Signal Recommendations and Requirements

BDM Pin	Signal Mnemonic	Requirement
1	Reserved	Need not be wired to the target. The USB TAP does not currently use this signal.
2	BKPT	Must be wired to the target.
3	GND	Must be wired to the target. GND is connected directly the USB TAP ground inside the USB TAP.
4	DSCLK	DSCLK must be connected to the target's processor. It is driven by the USB TAP as an output with up to 50mA. This signal is the clock for the BDM interface. It is good design practice to keep the trace length short and isolate the trace from other signals. If the trace must be long, then termination may be needed.
5	GND	Must be wired to the target. GND is connected directly the USB TAP ground inside the USB TAP.
6	Reserved	Need not be wired to the target. The USB TAP does not currently use this signal.

Table C.2 CodeWarrior USB TAP for BDM Signal Recommendations and Requirements

BDM Pin	Signal Mnemonic	Requirement
7	RESET	Must be wired to the target. During reset, the USB TAP drives $\overline{\text{RESET}}$ to ground through a 100Ohm resistor.
8	DSI	Must be wired to the target processor. The USB TAP emulator drives the TDI output with up to 50mA.
9	VDD	Must be wired to the target. The USB TAP emulator uses this signal to determine if power is applied to the target. The signal is also used as a voltage reference for the signals driven by the USB TAP emulator ($\overline{\text{BKPT}}$, DSCLK, $\overline{\text{RESET}}$, DSI, and $\overline{\text{TEA}}$).
10	DSO	Must be wired to the target processor. It is an input to the USB TAP.
11	GND	Must be wired to the target. GND is connected directly the USB TAP ground inside the USB TAP.
12	PST3	Need not be wired to the target. The USB TAP does not currently use this signal.
13	PST2	Need not be wired to the target. The USB TAP does not currently use this signal.
14	PST1	Need not be wired to the target. The USB TAP does not currently use this signal.
15	PST0	Need not be wired to the target. The USB TAP does not currently use this signal.
16	DDATA3	Need not be wired to the target. The USB TAP does not currently use this signal.
17	DDATA2	Need not be wired to the target. The USB TAP does not currently use this signal.
18	DDATA1	Need not be wired to the target. The USB TAP does not currently use this signal.
19	DDATA0	Need not be wired to the target. The USB TAP does not currently use this signal.
20	GND	Must be wired to the target. GND is connected directly the USB TAP ground inside the USB TAP.

Table C.2 CodeWarrior USB TAP for BDM Signal Recommendations and Requirements

BDM Pin	Signal Mnemonic	Requirement
21	Reserved	Need not be wired to the target. The USB TAP does not currently use this signal.
22	Reserved	Need not be wired to the target. The USB TAP does not currently use this signal.
23	GND	Must be wired to the target. GND is connected directly the USB TAP ground inside the USB TAP.
24	PSTCLK	Need not be wired to the target. The USB TAP emulator does not currently use this signal.
25	Core Voltage	Need not be wired to the target. The USB TAP does not currently use this signal.
26	TEA	Need not be wired to the target. The USB TAP emulator does not currently use this signal.

D

OnCE Connector Information

The CodeWarrior USB TAP for OnCE has a 14-pin connector. The CodeWarrior USB TAP for OnCE automatically supports target signal levels from 1.8V to 3.3V.

Figure D.1 shows the pin assignments of the CodeWarrior USB TAP for OnCE connector.

Table D.1 lists OnCE signal names, direction, pin numbers, descriptions, and drive capabilities for the CodeWarrior USB TAP for OnCE Connector.

Table D.2 provides a general description of each OnCE signal and the CodeWarrior USB TAP for OnCE operational requirements.

NOTE All OnCE signals must meet accepted standards for OnCE signal design. To ensure proper and stable operation between the CodeWarrior USB TAP for OnCE and the target, the OnCE signals must meet the requirements listed in Table D.2.

Figure D.1 CodeWarrior USB TAP for Once Connector Pin Assignments

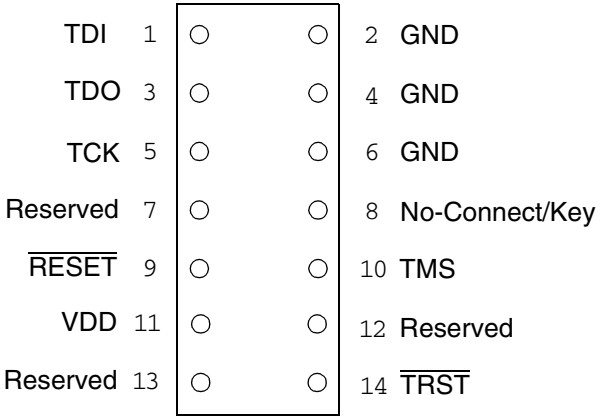


Table D.1 CodeWarrior USB TAP for OnCE Signal Directions

OnCE Pin	Signal Mnemonic	Signal Direction	Description
1	TDI	From USB TAP connector	50mA driver
2	GND	- n/a -	
3	TDO	From target	30pF load
4	GND	- n/a -	
5	TCK	From USB TAP connector	50mA driver
6	GND	- n/a -	
7	Reserved	From USB TAP connector	50mA driver
8	No-Connect/ Key	- n/a -	
9	RESET	Bi-directional	Open-drain. 100Ohm to ground when asserted by USB TAP, 35pF load when not asserted[1]
10	TMS	From USB TAP connector	50mA driver
11	VDD	From target	2MOhm pull-down, plus 0.01uF load
12	Reserved	Bi-directional	Open-drain, 100Ohm to ground when asserted by USB TAP emulator, 35pF load when not asserted[1]
13	Reserved	Bi-directional	Open-drain, 100Ohm to ground when asserted by USB TAP emulator, 35pF load when not asserted[1]
14	$\overline{\text{TRST}}$	From USB TAP connector	50mA driver[2]

[1] 4.7KOhm pull-up to buffered VDD.

[2] 100KOhm pull-up to buffered VDD.

Table D.2 CodeWarrior USB TAP for OnCE Signal Recommendations and Requirements

OnCE Pin	Signal Mnemonic	Requirement
1	TDI	Must be wired to the target processor. The USB TAP emulator drives the TDI output with up to 50 mA. The TDI trace should be kept short and maintain a "two-signal-width" spacing from any other parallel dynamic signal trace. TDI should have an RC termination option at the processor.
2	GND	Must be wired to the USB TAP. GND is connected directly to the USB TAP ground inside the USB TAP.
3	TDO	Must be wired to the target processor. TDO is an output from the target processor and input to the USB TAP emulator. The TDO trace run should be kept short and maintain a "two-signal-width" spacing from any other parallel dynamic signal trace. TDO should have a series termination resistor located near the target processor.
4	GND	Must be wired to the USB TAP. GND is connected directly to the USB TAP ground inside the USB TAP.
5	TCK	Must be wired to the target processor. The USB TAP emulator drives the TCK output with up to 50 mA. The TCK trace run should be kept as short as possible and maintain a “two-signal-width” spacing from any other parallel dynamic signal trace.
6	GND	Must be wired to the USB TAP. GND is connected directly to the USB TAP ground inside the USB TAP.
7	Reserved	Not required for emulation.
8	No-Connect/ Key	Not required for emulation. Pin 8 should be clipped on the target OnCE header.
9	RESET	Must be wired to the target processor. The USB TAP drives the $\overline{\text{RESET}}$ output with 50mA. During reset, the USB TAP drives $\overline{\text{RESET}}$ to ground through a 100Ohm resistor.
10	TMS	Must be wired to the target processor. The USB TAP emulator drives the TCK output with up to 50mA. The TCK trace run should be kept as short as possible and maintain a “two-signal-width” spacing from any other parallel dynamic signal trace.

Table D.2 CodeWarrior USB TAP for OnCE Signal Recommendations and Requirements

OnCE Pin	Signal Mnemonic	Requirement
11	VDD	Must be wired to the target. The USB TAP emulator uses this signal to determine if power is applied to the target. This signal is also used as a voltage reference for the signals driven by the USB TAP emulator (TDI, TCK, TMS, $\overline{\text{RESET}}$, and $\overline{\text{TRST}}$).
12	Reserved	Not required for emulation.
13	Reserved	Not required for emulation.
14	$\overline{\text{TRST}}$	Must be wired to the target processor. The USB TAP emulator drives the $\overline{\text{TRST}}$ output with up to 50 mA. The $\overline{\text{TRST}}$ trace run should be kept short and maintain a “two-signal-width” spacing from any other parallel dynamic signal trace.

E

ARM® JTAG Connector Information

The CodeWarrior USB TAP for ARM® JTAG has a 20-pin connector. The CodeWarrior USB TAP for ARM® JTAG automatically supports target signal levels from 1.8V to 3.3V.

Figure E.1 shows the pin assignments of the CodeWarrior USB TAP for ARM® JTAG.

Table E.1 lists ARM® JTAG signal names, direction, pin numbers, descriptions, and drive capabilities for the CodeWarrior USB TAP for ARM® JTAG.

Table E.2 provides a general description of each ARM® JTAG signal and the CodeWarrior USB TAP for ARM® JTAG operational requirements.

NOTE All ARM® JTAG signals must meet accepted standards for ARM® JTAG signal design. To ensure proper and stable operation between the CodeWarrior USB TAP for ARM® JTAG and the target, the ARM® JTAG signals must meet the requirements listed in Table E.2.

Figure E.1 CodeWarrior USB TAP for ARM® JTAG Connector Pin Assignments

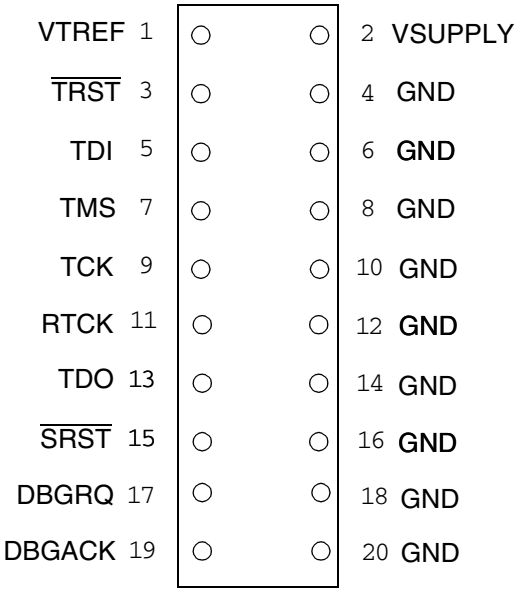


Table E.1 CodeWarrior USB TAP for ARM® JTAG Signal Directions

ARM® JTAG Pin	Signal Mnemonic	Signal Direction	Description
1	VTREF	From target	2MOhm pull-down, plus 0.01µF load
2	VSUPPLY	From target	VSUPPLY is not used by the USBTAP
3	TRST	FROM USB TAP connector	50mA driver[1]
4	GND	- n/a -	
5	TDI	From USB TAP connector	50mA driver
6	GND	- n/a -	
7	TMS	From USB TAP connector	50mA driver
8	GND	- n/a -	50mA driver

Table E.1 CodeWarrior USB TAP for ARM® JTAG Signal Directions (*continued*)

ARM® JTAG Pin	Signal Mnemonic	Signal Direction	Description
9	TCK	From USB TAP connector	50mA driver
10	GND	- n/a -	
11	RTCK	From target	30pF load
12	GND	- n/a -	
13	TDO	From target	30pF load
14	GND	- n/a -	
15	$\overline{\text{SRST}}$	Bi-directional	Open-drain. 100Ohm to ground when asserted by USB TAP, 35pF load when not asserted[2]
16	GND	- n/a -	
17	DBGRQ	From USB TAP connector	50mA driver
18	GND	- n/a -	
19	DBGACK	From target	30pF load
20	GND	- n/a -	

[1]100KOhm pull-up to buffered VDD.

[2]4.7KOhm pull-up to buffered VDD.

Table E.2 CodeWarrior USB TAP for ARM® JTAG Signal Recommendations and Requirements

ARM® JTAG Pin	Signal Mnemonic	Requirement
1	VTREF	Must be wired to the target. The USB TAP emulator uses this signal to determine if power is applied to the target. The signal is also used as a voltage reference for the signals driven by the USB TAP emulator ($\overline{\text{TRST}}$, TDI, TMS, TCK, $\overline{\text{SRST}}$, DBGRQ). VTREF (pin 1) should be connected to target Vcc through a 1KOhm pull-up.
2	VSUPPLY	May be wired to the target processor. The USB TAP emulator does not currently use this signal.
3	TRST	Must be wired to the target processor. The USB TAP emulator drives the $\overline{\text{TRST}}$ output with up to 50 mA. Note, $\overline{\text{TRST}}$ must not be connected to $\overline{\text{SRST}}$. The emulator must be able to independently assert these two signals.
4	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP.
5	TDI	Must be wired to the target processor. The USB TAP emulator drives the TDI output with up to 50 mA. The TDI trace run should be kept as short as possible and maintain a “two-signal-width” spacing from any other parallel dynamic signal trace.
6	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP.
7	TMS	Must be wired to the target processor. The USB TAP emulator drives the TMS output with up to 50 mA. The TMS trace run should be kept as short as possible and maintain a “two-signal-width” spacing from any other parallel dynamic signal trace.
8	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP.
9	TCK	Must be wired to the target processor. The USB TAP emulator drives the TCK output with up to 50 mA. The TCK trace run should be kept as short as possible and maintain a “two-signal-width” spacing from any other parallel dynamic signal trace.

Table E.2 CodeWarrior USB TAP for ARM® JTAG Signal Recommendations and Requirements (*continued*)

ARM® JTAG Pin	Signal Mnemonic	Requirement
10	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP.
11	RTCK	May be wired to the target processor. The USB TAP emulator does not currently use this signal.
12	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP
13	TDO	Must be wired to the target processor. TDO is an output from the target processor and an input to the USB TAP emulator. The TDO trace run should be kept as short as possible and maintain a “two-signal-width” spacing from any other parallel dynamic signal trace. TDO should have a series termination resistor located near the target processor.
14	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP.
15	$\overline{\text{SRST}}$	Must be wired to the target processor. The USB TAP drives the $\overline{\text{SRST}}$ output with up to 50 mA. During reset, the USB TAP drives $\overline{\text{SRST}}$ to ground through a 100Ohm resistor. Note, $\overline{\text{TRST}}$ must not be connected to $\overline{\text{SRST}}$. The emulator must be able to independently assert these tow signals.
16	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP.
17	DBGRQ	May be wired to the target processor. The USB TAP emulator does not currently use this signal.
18	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP.
19	DBGACK	May be wired to the target processor. The USB TAP emulator does not currently use this signal.
20	GND	Must be wired to the target. GND is connected directly to the USB TAP ground inside the USB TAP.

F

USB TAP Firmware (Loader)

This appendix explains the methods for reprogramming the loader image stored in the flash EPROM of the USB TAP. Before reprogramming the flash EPROM, make sure you have already properly installed the USB TAP drivers.

The sections are:

The USB TAP Internal Software Overview

Loader Software

The loader software provides initial USB connectivity to the host, tools for reprogramming the USB TAP loader firmware, and the underlying software framework required to run the debugger.

When the USB TAP is first connected to the host USB port it runs the loader software. This is indicated by the red or green blinking heartbeat of the USB TAP's TX/RX indicator light. To reprogram the loader image stored in the USB TAP flash EPROM, see "Reprogramming the USB TAP Firmware Image".

Dispatcher Software

The USB TAP dispatcher software is basically transparent to the user, and is simply the application that tells the USB TAP how to control the target. It recognizes the specific target processor and debug port interface, and carries out the instructions of the debugger.

Reprogramming the USB TAP Firmware Image

At some point you may be required to reprogram the USB TAP firmware image stored in its flash EPROM. To do this:

1. Make sure the USB TAP is connected to a USB Port on your host computer.
2. Launch CCS and open the CCS command window. The procedure is slightly different on Windows and Linux host machines.

   - For Windows:

   Run the command:

   ```
   <CodeWarrior Installation>\ccs\bin\ccs.exe
   ```

   This will launch CCS and add a CCS icon (Figure F.1) to your taskbar. Double-click that icon in the taskbar to open the command window.

   - For Linux

   Run the command:

   ```
   <CodeWarrior Installation>/ccs/bin/ccs
   ```

   This will launch CCS and open the command window automatically.

Figure F.1 CCS Icon

3. In the CCS Command window, enter the command:

   `updateutap <serial_number>`

   The *<serial_number>* parameter should be the serial number of your USB TAP, which is on the bottom of the device. If you have only one USB TAP connected to your host machine, you do not need to specify the *<serial_number>* parameter. You should see the output displayed in Listing F.1 (although the version number may be different).

Listing F.1 Output Produced by updateutap Command

```
USB TAP Loader current software ver. {1.0}
Sending code to USB TAP - please wait

*** WARNING ***

DO NOT UNPLUG YOUR USB TAP: Doing so will render your USB
TAP non-functional and require factory reprogramming.
```

```
Please wait until the flashing Yellow/Green TGT STATUS light
turns off.

*** WARNING ***
```

4. Before disconnecting your USB TAP, wait for the flashing status light to turn off.

What To Do Next

When you have completed reprogramming the firmware images stored in the USB TAP flash EPROM, configure your debugging environment (if you haven't already done so) as explained in the debugger manual.

G

Troubleshooting

If you are having problems with Linux permissions during the installation of the CodeWarrior software on your Linux operating system, please try the following:

1. If you are not able to connect to the USB TAP emulator, run CCS as root. With root access, run the command:

   `<CodeWarrior Installation>/ccs/bin/ccs`

   Leave this instance of CCS running and try connecting via the CodeWarrior software. If this works then there is probably an issue with the USB TAP emulator's permissions.

2. The CodeWarrior installation includes hotplug scripts which should set the appropriate permissions for the USB TAP emulator. To confirm that they have been installed correctly, ensure that the file `/etc/hotplug/usb/usbtap` exists and that it is executable by root. Also view the file `/etc/hotplug/usb.usermap` and ensure that it contains a usbtap entry.

   On Mandrake®Linux® 10.0, ensure that the file `/usr/lib/hotplug/usbtap/usb.usermap` exists, and is readable by root but not executable by root.

   If the scripts are not installed correctly, please reinstall the CodeWarrior software. Otherwise run this command:

   `grep usb.agent /var/log/messages`

   and send the output to Customer Support with a description of the problem.

Index