MAJIC Windows CE .NET eXDI

User's Manual

Embedded Performance, Inc.

April, 2003

EPI has made every attempt to ensure that the information in this document is accurate and complete. However, EPI assumes no responsibility for any errors, omissions, or for any consequences resulting from the use of the information included herein or the equipment it accompanies. EPI reserves the right to make changes in its products and specifications at any time without notice.

Embedded Performance, Inc.
606 Valley Way
Milpitas, California 95035
USA
Voice: (408) 957-0350
FAX: (408) 957-0307
email: sales@epitools.com
support@epitools.com
URL: http://www.epitools.com

Acknowledgments:

XScale Micro-Architecture, PXA250 are trademarks of Intel®.
ARM, ARM7, and ARM9 are trademarks of ARM Technologies, Inc.
Windows, Windows CE, Platform Builder are trademarks of Microsoft Corporation.
MAJIC, MONICE, and EPI are trademarks of Embedded Performance, Inc.
All other trademarks are trademarks of their respective companies.

Table of Contents

1 Introduction

This chapter provides a brief overview of the scope of this manual, what eXDI is, our MAJIC eXDI Driver / PlugIn, and which Windows CE environments are supported.

About this Manual

This is the user manual for the Embedded Performance eXDI driver and PlugIn. It is aimed at end-users who intend to use the MAJIC with Microsoft's Platform Builder IDE.

Note: Except where explicitly stated to the contrary, the term MAJIC refers to all models of MAJIC (references to MAJIC^{PLUS} and MAJIC^{MX} refer to those respective models).

What is eXDI?

eXDI is Microsoft's Extended Debug Interface. This interface allows EPI to create a driver that is used by Microsoft's Platform Builder's IDE to provide full access to MAJIC's hardware-based debug services. In addition, we offer an eXDI PlugIn which provides extended GUI-based debug services not available within Platform Builder itself. A good example of this is Target CPU Trace services.

Note: Windows CE requires installation on Windows 2000 or Windows XP Professional, consequently the driver has the same restrictions as Windows CE.

Supported Windows CE Environments

At the time of this writing MAJIC eXDI supports the following Windows CE-based development environments. If you don't see your Windows CE-based environment listed, please contact EPI for the latest information. Note that this manual is targeted at users of Windows CE .NET.

Windows CE .Net Full Support

Windows CE 3.0 Access to symbols and source code within the MAJIC eXDI Monitor Window and PlugIn are not supported.

Pocket/PC 2002 Requires installation of Platform Builder 3.0 or .NET for debug via Platform Builder. OS-aware functionality in Platform Builder is not supported due to incompatibility issues with the divergent Windows CE OS used in Pocket/PC. This basically means it can only be used for code linked in with the kernel (nk.exe). Platform Builder cannot determine the load address of anything dynamically loaded (like a DLL or separate EXE file). Note that this is a limitation of Platform Builder, not EPI's eXDI driver.

Pocket/PC 2003 Please contact support@epitools.com for the current road-map.

Related Documentation

Microsoft Windows CE .NET Online Help Describes the Extended Debug Interface (eXDI) supported by Windows CE 3.0 and Windows .Net.

MAJIC User's Manual Tab section in EPI Development Tools for MAJIC: Revision 2.0, Dated: April 2002 or later.

MDI for MAJIC User's Manual Tab section in EPI Development Tools for MAJIC. Documents usage of EPI's Meta Debug Interface Revision 1.0, Dated: May 2001 or later.

Notational Conventions

The following conventions are used in the syntax descriptions of this manual.

Normal — Just Normal Text.

Bold face — Identifies characters that must be entered exactly as shown.

Italic — Indicates a general category of input described in detail in the command operand's section.

Header/Note — Appearing to the left of a text block, indicates a step boundary or special note.

Arial Font — Indicates text appearing on the screen, e.g. menus or dialogs.

2 *Installation*

This chapter describes how to install and configure the MAJIC eXDI Driver and PlugIn provided by EPI.

Getting Started

Step 1: Install the Microsoft Windows CE .Net software package. It is important to do this before installing the EPI Tools package because our installer will update registry entries created by the Microsoft Installer. If you installed the Microsoft Windows CE .Net software package first then run the **"install.bat"** utility found in the C:\Program Files\EPITOOLS\EDTx20\exdi\ directory after installing the EPI Development Tools (EDT) package to update the registry entries.

Step 2: Install the EPI Development Tools package (EDT). Verify that you have a directory named "**`exdi`**" under the installation root of the EDT package. If you do not, then you have not correctly installed the eXDI Driver software.

Step 3: Install any third-party software, such as the Accelent Systems software, or your Board Support Package (BSP).

Step 4: Review the MAJIC Quick Start Guide – Hardware as a starting point for setup and configuration of your MAJIC probe. If you don't have a printed MAJIC Quick Start Guide then please refer to the digital version in the CD's manuals directory. Review the MAJIC Quick Start Guide – Software section and then see either ***step 5*** or ***step 6*** for more specifics on running the MAJIC Setup Wizard.

Step 5: For flash programming with the EPI Flash Programming Utility you will need to set up a shortcut for MONICE, EPI's underlying command-line interface. With this shortcut you can program the bootloader for your WinCE image.

1. Run the MAJIC Setup Wizard found in the Start menu under Start-> Programs-> EPI Tools - EDTx-> MAJIC Setup Wizard.
2. At the first screen select Next.

3. Under Choose your Debugger select MONICE and click Go.
4. Choose a Project Name and Enter a one line description of your project, choose Next.
5. Select your Processor Type and Select your Target's Endianness, click Next.
6. Choose your connection type. If it is your first time connecting to the MAJIC you must select the Serial port. If you are not connected to a network and would like to use Ethernet (cross-over cable) you must choose an IP address. See the MAJIC User's Manual for more information.
7. In the Configuration Files window choose Use Existing Startup Files. Browse to the samples directory and choose a sample directory that does not have an _wince extension. The settings for WinCE will interfere with Flash programming. If there are no sample files for your platform then choose Create New Startup File and Adjust Default Properties.
8. In the next screen choose Reference the existing startup files from their location.
9. Click Perform Actions, and then Done.

A shortcut for MONICE should pop up on your desktop. Power up your target and the MAJIC, then double-click on the desktop shortcut. Verify that you are able to connect to your target board, make sure that you see the message "JTAG connection established" in red letters. Use this connection in the future for flash programming. See the flash programming utility manual in the C:\Program Files\EPITOOLS\EDTx20\manuals directory for more information.

Step 6: To use Platform Builder you will need to setup the appropriate configuration files for your target board. To do this:

1. Run the MAJIC Setup Wizard found in the Start menu under Start-> Programs-> EPI Tools - EDTx-> MAJIC Setup Wizard.
2. In the first screen choose Next.
3. Under Choose your Debugger select Platform Builder (WinCE/PocketPC) and click Go.
4. Choose a Project Name and Enter a one line description of your project, select Next.
5. Select your Processor Type and Select your Target's Endianness, choose Next.
6. Choose your connection type. If it is your first time connecting to the MAJIC you must select Serial port. If you are not connected to a network and would like to use Ethernet (cross-over cable) you must choose IP address. See the MAJIC User's Manual for more information.
7. In the Configuration Files window choose Use Existing Startup Files. Browse to the samples directory and choose a sample directory that matches your target type. If there are no sample files for your platform then choose Create New Startup File and Adjust Default Properties. If you are using an XScale reference platform be sure to choose a folder that has a _wince extension. If there are no sample files for your XScale target with a _wince extension contact support@epitools.com.
9. In the next screen click Perform Actions, and then Done.

Power up your target and the MAJIC, then browse to the C:\Program Files\EPITOOLS\EDTA20\eXDI\ directory and double-click on the majic_exdi_driver.exe file. This should launch the eXDI interface and connect to the target successfully.

Step 7: If you have not already read your MAJIC User's Manual now is the time. This can be found in digital format in the C:\Program Files\EPITOOLS\EDTx20\manuals directory.

Updating the Registry

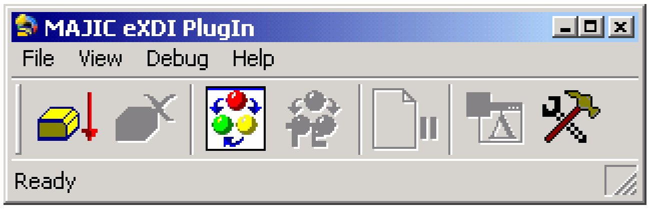
Normally, the Windows registry is setup when you install the EPI Tools from a production CD. If you received your software as a zip file or you are manually moving your EPI Tools installation directory around, then you need to run the script **`./exdi/install.bat`** to update the registry settings. For example:

```
DOS> cd exdi
DOS> install
```

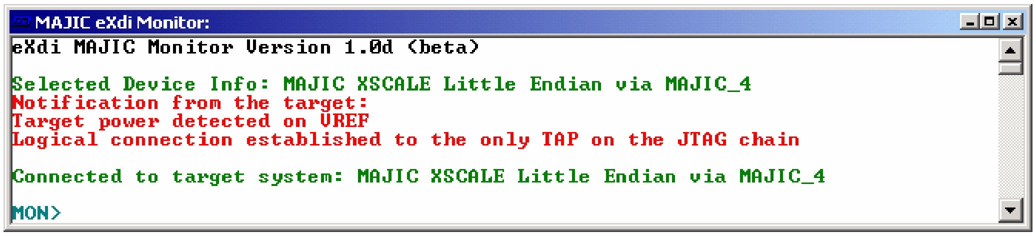
Note: Microsoft's Platform Builder should already installed before you install the EPI Tools CD or run this install batch file. If for some reason you install Platform builder after installing the EPI Tools CD, then simply run this install batch file again.

Test Your Driver Setup/Installation

First, make sure Platform Builder is not running. From the Windows Start menu select Start -> Programs -> EPI Tools - EDTx -> Majic_Exdi_PlugIn. This launches the PlugIn as shown below:



Now select the menu item Debug -> Connect to MAJIC Probe. If your setup is configured correctly, then the driver starts and the MAJIC eXDI Monitor Window pops up as shown below. If it does not start correctly repeat ***step 6:*** in the "Getting Started" section above.



```
eXdi MAJIC Monitor Version 1.0d (beta)

Selected Device Info: MAJIC XSCALE Little Endian via MAJIC_4
Notification from the target:
Target power detected on VREF
Logical connection established to the only TAP on the JTAG chain

Connected to target system: MAJIC XSCALE Little Endian via MAJIC_4

MON>
```

Note: The Driver may pop up a configuration selection dialog if you have multiple configurations defined in your MDI configuration file. If so, simply select the appropriate configuration. The contents of this window will vary slightly, but you should get a valid connection established and a MON> prompt.

Now from the PlugIn Menu select Debug -> Disconnect from MAJIC Probe. This disconnects the PlugIn from the driver and if no other applications are connect to the Driver then the driver shuts down.

eXDI Files

The following files relating to MAJIC eXDI support are provided in the directory where your EPI software is installed.

`./exdi/majic_exdi_driver.exe`
The MAJIC eXDI Driver.

`./exdi/majic_exdi_plugin.exe`
The MAJIC eXDI PlugIn.

`./exdi/eXdi_epips.dll`
A supporting DLL used by the driver.

`./exdi/install.bat`
A DOS batch file that creates or updates eXDI driver and Platform Builder registry entries. Note that normally all the needed registry work is done when you install the tools from the CD. You will need to run the **`install.bat`** file if you manually moved the "**`exdi`**" installation directory or you are installing an engineering version.

`./exdi/uninstall.bat`
Un-installs registry entries setup by **`install.bat`**.

`./manuals/MAJIC_eXDI_UserManual.pdf`
This manual. Installing the manuals is optional, if you did not install the manuals then this file can still be accessed directly from the **`./manuals`** directory on the installation CD.

The following files are not eXDI specific, they are needed regardless of whether you are using an EPI debugger or a third-party debugger.

`./mdi/`…	MDI installation tree. MDI is used by eXDI to talk to your MAJIC probe. Refer to the MDI for MAJIC User's Manual for details

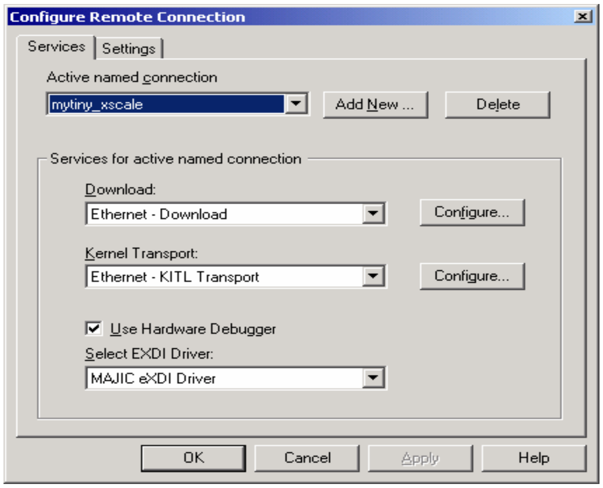
3 Configuring Platform Builder

This chapter explains the steps needed to configure Platform Builder to utilize your MAJIC probe for hardware debug. The steps outlined here are for Windows CE .NET. Similar, but slightly different steps are used by Windows CE 3.0. In either case, the best source of documentation on this aspect is the actual Platform Builders documentation.

Configuring your Platform Builder Project to use MAJIC

Step 1: Launch Platform Builder and select in the workspace/project you wish to configure. You probably want to be using a debug build of your project, but that is not required to utilize MAJIC. The debug build will offer you a view of non-optimized code, while the Release build will be optimized. Note that these configuration changes are workspace-specific and you will need to repeat them for each workspace/project configuration which you wish to enable for debug with MAJIC.

Step 2: Configure Platform Builder to use a hardware debug probe. Select the menu item: Target -> Configure Remote Connection. This brings a dialog as shown below. Click the Use Hardware Debugger checkbox and select MAJIC eXDI Driver from the Select EXDI Driver drop-down list box and click the [OK] button.



Active named connection: mytiny_xscale

Services for active named connection

Download: Ethernet - Download

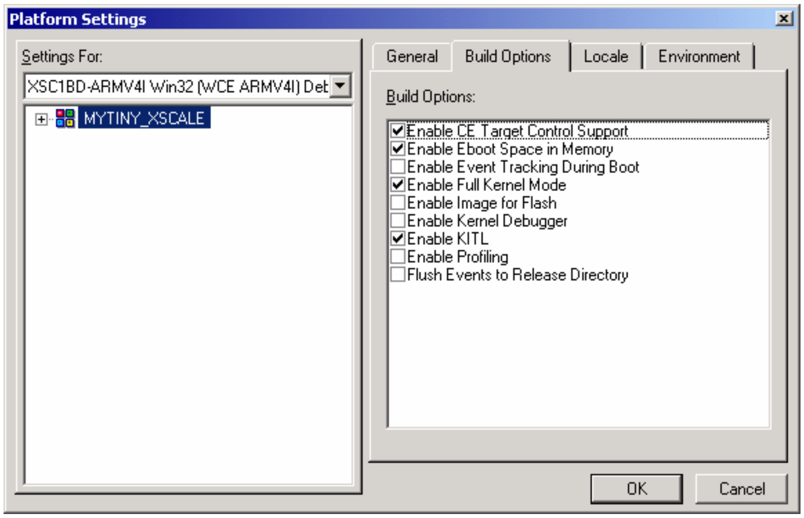
Kernel Transport: Ethernet - KITL Transport

☑ Use Hardware Debugger

Select EXDI Driver: MAJIC eXDI Driver

If you are using a Compaq flash card choose Download: None and Kernel Transport: None. If you are using an alternative device supported by the bootloader, such as an Ethernet PCMCIA card, then choose Download: Ethernet – Download and Kernel Transport: Ethernet – KITL Transport.

Step 3: Next, we need to tell Platform Builder not to include the software-based debug kernel. Select the menu item Platform -> Settings. Now select the Build Options tab and uncheck the Enable Kernel Debugger check box and click the [OK] button.



Settings For: XSC1BD-ARMV4I Win32 (WCE ARMV4I) Det

MYTINY_XSCALE

Build Options:

☑ Enable CE Target Control Support
☑ Enable Eboot Space in Memory
□ Enable Event Tracking During Boot
☑ Enable Full Kernel Mode
□ Enable Image for Flash
□ Enable Kernel Debugger
☑ Enable KITL
□ Enable Profiling
□ Flush Events to Release Directory

Step 4: If you are using an alternative device supported by the bootloader –such as an Ethernet PCMCIA card –for download, you have the option of downloading only to target RAM. This will save time by not copying everything into flash when you connect. This is a good option if you build frequently. To do this, select the menu item Platform -> Settings and choose the Environment tab. You will be adding an environment variable to stop the IDP from saving the downloaded OS image to its internal flash.

Click the New button and for Variable Name: enter IMGINRAM, for Variable Value: enter 1.

Rebuild your Windows CE image, menu item: Build -> Rebuild Platform.

You are now done with your configuration changes. We suggest saving your workspace changes before proceeding on with actual debugging. Select menu item File -> Save Workspace.

Step 4: If you are using a Compaq flash card to transfer the new build to the target, copy the nk.bin file to the Compaq flash card and place the card in the PCMCIA slot of the target. Reboot the target and it will automatically copy the image from the card to the target ROM. If you are using an Ethernet PCMCIA card see Step 4 in “Debugging a Downloadable Image” in Chapter 4.

4 Using the Driver

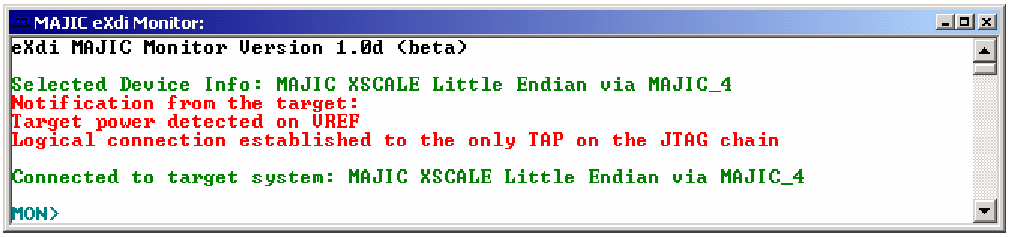
This chapter demonstrates usage of the MAJIC eXDI Driver. It is assumed the user has successfully followed the steps in Chapters 2 and 3 to install the Driver and configure Platform Builder to utilize it.

Debugging a Downloadable Image

For this example we are using an Intel® XScale™ Micro-Architecture (PXA250)-based platform with a working EBOOT installed in flash and Platform Builder set up for download of a new Windows CE image (nk.exe) via EBOOT's download facility.

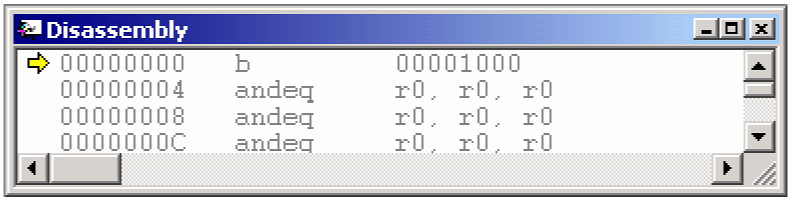
Step 1: First make sure you are disconnected from the target and no Platform Builder target services are running. Do this by selecting the menu item Target -> Disconnect or the Target toolbar button .

Step 2: ***Starting the eXDI Driver and resetting the target system:***
Select the menu item Target -> Connect or the Build Toolbar Go button . This launches or connects up to an already-running MAJIC eXDI Driver. The Driver Monitor Window pops up as shown below and the target is reset and stopped.



```
eXdi MAJIC Monitor Version 1.0d (beta)

Selected Device Info: MAJIC XSCALE Little Endian via MAJIC_4
Notification from the target:
Target power detected on VREF
Logical connection established to the only TAP on the JTAG chain

Connected to target system: MAJIC XSCALE Little Endian via MAJIC_4

MON>
```

Platform Builder takes a few seconds to initialize, read target memory to establish the state of the target OS (if any), and then open up a disassembly window at the current PC location (example shown below).



```
00000000   b        00001000
00000004   andeq    r0, r0, r0
00000008   andeq    r0, r0, r0
0000000C   andeq    r0, r0, r0
```

Step 3: ***Starting the EBOOT Image:***

Now we are ready to start the boot code. Select the menu item Debug -> Go or the Toolbar Go button . EBOOT starts and prompts for an image to download.

Note that before you select the Go operation you can insert breakpoints within your downloaded image –but only at locations that Platform Builder can resolved the address for. This means breakpoints outside of the module nk.exe can be set, but they will not be loaded until later. An unloaded (or unresolved) breakpoint can be seen in a Platform Builder's source window as purple-colored breakpoint. A red-colored breakpoint indicates a resolvable breakpoint that has been loaded.

Step 4: ***Loading your WinCE image:***

Tell Platform Builder to begin the code download by selecting menu item Target -> Download / Initialize or button. The download then proceeds to completion and execution of your image begins.

At any time you can interrupt this process by simple selecting the menu item Debug -> Break or button. You can also set a breakpoint either while stopped or while the image is executing.

Step 5: ***Updating Module/Symbol Information:***

After stopping, if you suspect new modules or DLLs have been loaded, you need to tell Platform Builder to refresh its modules list. This is done by selecting the menu item: Target -> CE Modules and Symbols or button which opens the Modules and Symbols window. If this window was already open, then you must tell it to refresh its data. This is done with the window's refresh button.

If the only module listed is nk.exe, then either you've stopped before any other modules or DLLs were loaded, or your WinCE kernel data format doesn't match the format expect by Platform Builders OsAxs.dll module. This is the reason why source level debugging of Pocket PC 2002 is limited to the statically-linked nk.exe module.

Step 6: ***Ending a Debug Session:***

Be sure when ending a debug session that you not only end your debug session via the menu item Debug -> Stop or button, but also end the Platform Builder target services via Target -> Disconnect or button. Note that you can do both operations at once by choosing Target -> Disconnect first.

The closing of the MAJIC eXDI Monitor Window may take few moments as it cleans up various services. Note that the Driver and Window will remain active if you have a PlugIn running that is actively connected to the Driver.

Debugging a Boot Image

For this example we are using an Intel® XScale™ Micro-Architecture (PXA250)-based platform with EBOOT installed in flash as the image to debug. It could also be a full WinCE image.

Step 1: It is important to make sure your EBOOT ROM image matches your debugging image. This involves building your eboot.bin image file and burning it into flash ROM. Please refer to your Platform Builder documentation on building EBOOT for details.

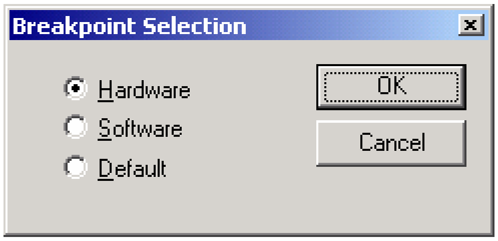
Step 2: After building EBOOT and burning in your ROM image, select into Platform Builder the just built eboot.bin as your workspace. Note that this is not the eboot.bin from your flat release directory, but instead the file from the path MS refers to as:

`%_TARGETPLATROOT%\target\%_TGTCPU%\%WINCEDEBUG%`.

If you need further help on this aspect please consult your Platform Builder documentation or contact Microsoft product support.

Step 3: Make sure you are disconnected from the target and no Platform Builder target services are running. Do this by selecting the menu item Target -> Disconnect or the Target toolbar button .

Step 4: You can either just begin single-stepping into the ROM or use a hardware breakpoint near the location in EBOOT you wish to stop and begin real debugging. For example to stop in main(), first find the location of your EBOOT's main function. The file/location may vary based on your BSP. For the development board used here, it is `main.c` in the directory **`C:\WINCE400\PALTFORM\XSC1BD\EBOOT`**. Select in this file and find the start of the code area for the function(). Set a breakpoint at this location by clicking the cursor on the referenced line and selecting the breakpoint ✋ button. Now select the menu item Edit -> Breakpoints, select the breakpoint from list at the bottom and click the [Hardware] button to bring up a dialog for changing the kind of breakpoint used. Select the Hardware radio button, and then click the [OK] button.

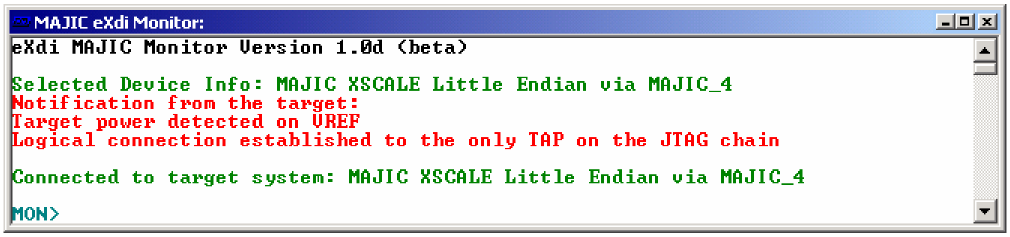


Breakpoint Selection

● Hardware
○ Software
○ Default

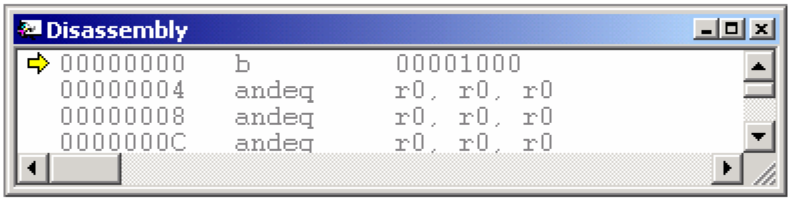
Finally, click the Breakpoints dialogs [OK] button and you are ready to start executing.

Step 5: Next, select the menu item Target -> Connect or the Build Toolbars Go button . This launches or connects-up to an already-running MAJIC eXDI driver. The Driver Monitor Window pops up as shown below and the target is reset and stopped.



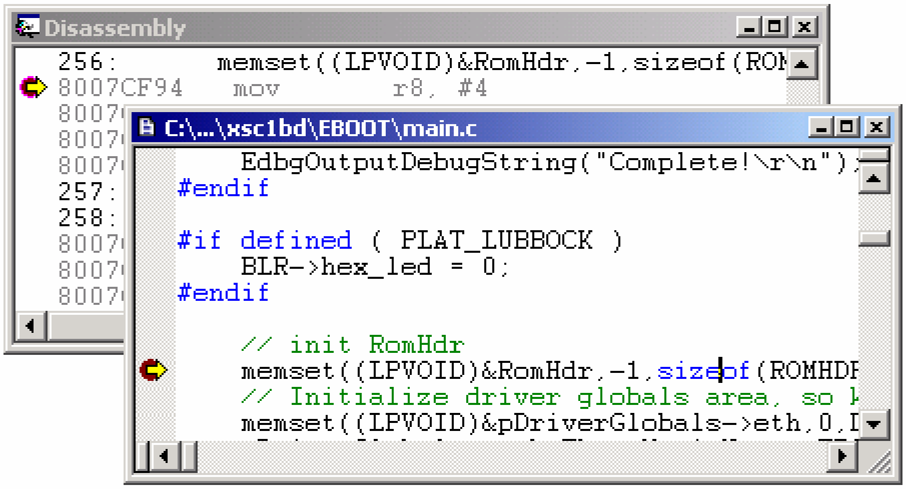
```
eXdi MAJIC Monitor Version 1.0d (beta)

Selected Device Info: MAJIC XSCALE Little Endian via MAJIC_4
Notification from the target:
Target power detected on VREF
Logical connection established to the only TAP on the JTAG chain

Connected to target system: MAJIC XSCALE Little Endian via MAJIC_4

MON>
```

Platform Builder takes a few seconds to initialize, read target memory to establish the state of the target OS (if any), and then open up a disassembly window at the current PC location (example shown below).



```
00000000   b          00001000
00000004   andeq      r0, r0, r0
00000008   andeq      r0, r0, r0
0000000C   andeq      r0, r0, r0
```

Step 6: Now we are ready to start the boot code. Select the menu item Debug -> Go or the Toolbar Go button . The target should then break in main.c similar to below. You can now start debugging.



```
256:      memset((LPVOID)&RomHdr,-1,sizeof(ROM
8007CF94   mov        r8, #4
8007
8007
8007
257:
258:
8007
8007
8007
```

```
    EdbgOutputDebugString("Complete!\r\n");
#endif

#if defined ( PLAT_LUBBOCK )
    BLR->hex_led = 0;
#endif

    // init RomHdr
    memset((LPVOID)&RomHdr,-1,sizeof(ROMHDF
    // Initialize driver globals area, so k
    memset((LPVOID)&pDriverGlobals->eth,0,I
```

Breakpoint Issues

Platform Builder/WinCE is missing a critical link that MS refers to as the "Load Notification Hook". This is an important issue that you need to be aware of when debugging WinCE-based images and applications.

If the hook were implemented, WinCE would be capable of sending notification messages to Platform Builder about the loading of new modules and DLLs. This would allow Platform Builder to gather debug information on the new items and resolve any breakpoints that it might need automatically.

The Problem: I set a breakpoint in my code, but when I run, it doesn't get hit. I know my code is running, so why is this, and how can I make it work?

The Answer: If the breakpoint is in source code that is not linked in directly with nk.exe, then it is a dynamically-loaded module. Platform Builder itself does not know where in memory the WinCE kernel is going to place your code. Consequently, setting a breakpoint before you code is loaded means the breakpoint is unresolved and does not get sent to the eXDI driver. If you believe that your code was loaded prior to setting the breakpoint then you may need to make sure you are keeping the Modules and Symbols window refreshed. See section ***Updating Module/Symbol Information*** on Page 13.

With PB 4.x you'll notice unresolved or unloaded breakpoints shown in the source window are purple. Loaded breakpoints are red.

Work Around 1: Determine the real address of your breakpoint location and then set an address (non-symbolic) based hardware breakpoint via Platform Builders Breakpoints Dialog. The steps to do this are listed below:

Step 1: Start up your system and make sure your code is loaded.

Step 2: Open the Modules and Symbols Window and do a Refresh operation. Make sure your code module is shown as loaded in this window.

Step 3: Bring up your source in a Platform Builder source window. Right-click on the line you wish to set a breakpoint on, and select Go To Disassembly. This brings up the disassembly window and shows you the address of your source line.

Step4: Select Edit -> Breakpoints… and create a new breakpoint using the address from the disassembly window. Select the [Hardware] button and change it to be hardware-based. Close the dialogs by selecting [OK]. Note that if you fail to change the breakpoint to hardware mode, the breakpoint will still be set, but WinCE is likely to overwrite the breakpoint when loading your module.

Step5: End your debug session via Target -> Disconnect and restart. Your breakpoint should now be hit when your code runs.

Note: Once you stop at the breakpoint and Refresh the Modules and Symbols window you should be able set software breakpoints and have them work.

Pitfalls: If you make changes and recompile your code you may need to check the address of your breakpoint to make sure your source code has not moved in memory. Note that WinCE will typically load modules in the same order and therefore at the same virtual address. However, you must make sure you start up in the same order. If you're manually loading multiple applications, be sure to load them in the same order. If this issue becomes problematic for you we suggest you try the workaround listed below.

Work Around 2: Hard-code a special breakpoint directly in your source code. The Microsoft compiler provides a method to embed opcodes directly inline with your source level code. MAJIC recognizes this opcode, stops execution and reports the occurrence as an "Unknown breakpoint". The steps below show this setup in detail

Step 1: Edit the source code where you wish to set your breakpoint and add the following code above the line you wish to stop on. The specific code value you use depends upon your target CPU.

For XScale/ARM V5 chips:

```
        __emit(0xE12cec7e);  // breakpoint
```

For MIPS chips:

```
        __emit(0x70cece3f);  // breakpoint
```

Step 2: Recompile your code and run it. Upon hitting the breakpoint, notice that the program counter (pc register) has been adjusted to the instruction after your hard-coded breakpoint. This allows you to easily continue execution after stopping.

Note: One way to better manage this breakpoint issue is to hard-code your breakpoint at your module's entry point. Once you stop at the breakpoint, you can then decide to set a normal software breakpoint where ever you need one.

Pitfalls: If you run your system without the eXDI driver in place your hard-coded breakpoint will cause your system to hang up. You'll need to disable the breakpoint in your code and recompile to run without the MAJIC eXDI driver loaded.

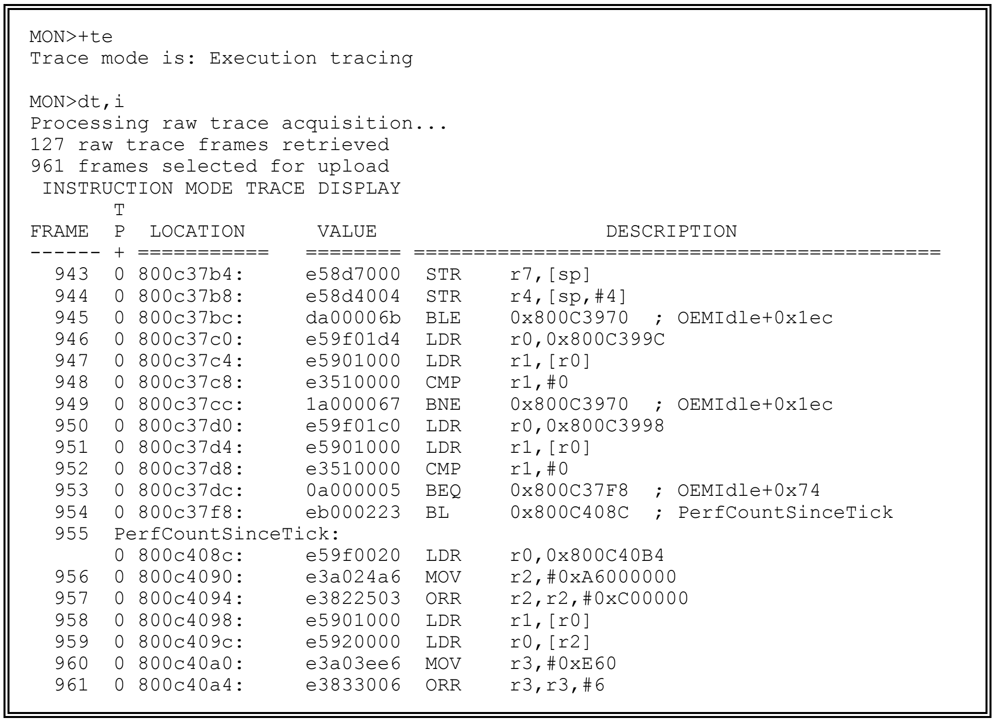
Using the eXDI Monitor Window

The eXDI Driver's Monitor Window contains a subset of our low-level Monice (MON) debugger command interface. It contains a rich set of commands to examine memory and registers, display and control trace acquisition, and configure your MAJIC probe. Please refer to the MAJIC User's Manual for details on the MON command language.

Trace Setup and Display

For some CPUs, trace setup is as easy as enabling trace capture via MON's **+te** command. Others require setup of what kinds of things to trace, when to stop etc. The description of this is beyond the scope of this manual. Please refer to your MAJIC User's Manual chapter on trace control.

Example: Below is a screen shot of enabling trace and displaying the result. Note that between the enable (**+te**) and the display trace (**dt**) command we went back to Platform Builder's menu and told the target Go for a few seconds and then selected the Break button. This allowes MAJIC to capture some trace data for display. Note that the amount of data captured is dependent on your MAJIC model and target CPU. For the CPU used here, the results can vary with the execution stream, but typically about 1000 instructions are captured.



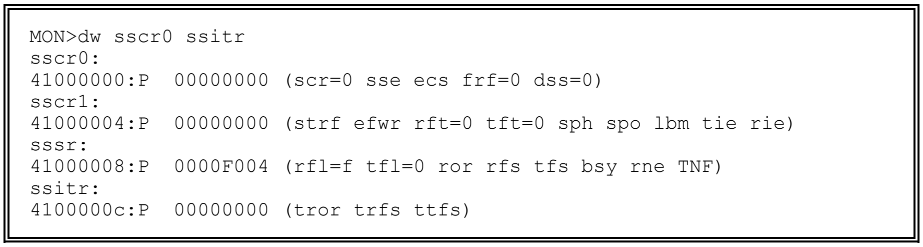
```
MON>+te
Trace mode is: Execution tracing

MON>dt,i
Processing raw trace acquisition...
127 raw trace frames retrieved
961 frames selected for upload
 INSTRUCTION MODE TRACE DISPLAY
       T
FRAME  P  LOCATION      VALUE                   DESCRIPTION
------ + ===========   ======== ============================================
  943  0 800c37b4:     e58d7000  STR    r7,[sp]
  944  0 800c37b8:     e58d4004  STR    r4,[sp,#4]
  945  0 800c37bc:     da00006b  BLE    0x800C3970  ; OEMIdle+0x1ec
  946  0 800c37c0:     e59f01d4  LDR    r0,0x800C399C
  947  0 800c37c4:     e5901000  LDR    r1,[r0]
  948  0 800c37c8:     e3510000  CMP    r1,#0
  949  0 800c37cc:     1a000067  BNE    0x800C3970  ; OEMIdle+0x1ec
  950  0 800c37d0:     e59f01c0  LDR    r0,0x800C3998
  951  0 800c37d4:     e5901000  LDR    r1,[r0]
  952  0 800c37d8:     e3510000  CMP    r1,#0
  953  0 800c37dc:     0a000005  BEQ    0x800C37F8  ; OEMIdle+0x74
  954  0 800c37f8:     eb000223  BL     0x800C408C  ; PerfCountSinceTick
  955  PerfCountSinceTick:
       0 800c408c:     e59f0020  LDR    r0,0x800C40B4
  956  0 800c4090:     e3a024a6  MOV    r2,#0xA6000000
  957  0 800c4094:     e3822503  ORR    r2,r2,#0xC00000
  958  0 800c4098:     e5901000  LDR    r1,[r0]
  959  0 800c409c:     e5920000  LDR    r0,[r2]
  960  0 800c40a0:     e3a03ee6  MOV    r3,#0xE60
  961  0 800c40a4:     e3833006  ORR    r3,r3,#6
```

Note: The trace information above can also be viewed via the PlugIn's Trace Display Window which provides a GUI-based interface and source line information instead of just symbols. Please see the next chapter for details on using the PlugIn.

Coprocessor and Peripheral Register Access

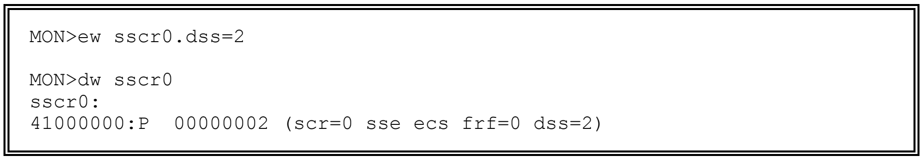
Platform Builder's IDE only provides access to a subset of CPU registers. MON supports access to a full set of CPU and peripheral registers. This is configured via startup files and can be customized by users. For full details on configuration and usage consult the MAJIC User's Manual. Below are some simple examples for the Intel PXA250 demonstrating the power of this feature.

In this screen shot we told the debugger to display, via the Display Word (**`dw`**) command, a bank of peripheral registers for an on-chip serial port. This peripheral has four registers starting at the physical memory address 41000000. To the right of the register's value display is a field breakdown of the individual bits/fields. Multi-bit fields are displayed as ***`fieldname=val`*** and single-bit fields are displayed in uppercase for 1 and lowercase for value 0.



```
MON>dw sscr0 ssitr
sscr0:
41000000:P  00000000 (scr=0 sse ecs frf=0 dss=0)
sscr1:
41000004:P  00000000 (strf efwr rft=0 tft=0 sph spo lbm tie rie)
sssr:
41000008:P  0000F004 (rfl=f tfl=0 ror rfs tfs bsy rne TNF)
ssitr:
4100000c:P  00000000 (tror trfs ttfs)
```

The registers can also be edited via simple commands. The Enter Word (**`ew`**) command (shown below) modifys a field within a register.



```
MON>ew sscr0.dss=2

MON>dw sscr0
sscr0:
41000000:P  00000002 (scr=0 sse ecs frf=0 dss=2)
```

Memory Access

Platform Builder's is limited to examining Virtual Memory, although developers often find it necessary to reach down inside and look at physical memory. MON commands can be used to examine such memory. For example: We know that the boot code on ARM targets start at address 0, but in many virtual memory-based applications the boot code is no longer visible once you have booted up. By using a physical memory address qualifier (**`:p`**) in MON's Display Word command we can see this memory. The "**`L 4`**" after the address says to list a range of four words and the "**`,i`**" says to display the words as instructions.

```
MON>dw 0:p L 4,i
00000000:P    ea0003fe    B      0x1000      ;
00000004:P    00000000    ANDEQ  r0,r0,r0
00000008:P    00000000    ANDEQ  r0,r0,r0
0000000c:P    00000000    ANDEQ  r0,r0,r0
```

Option Configuration

There are many configuration options available for your MAJIC probe. This aspect is covered in detail within the MAJIC User's Manual. As a simple example here we cover the interesting case of using the Vector Catch feature to stop when an exception occurs. Note that is example is specific to ARM and Intel® XScale™ based cores.

By default, the Vector Catch option is configured off for the Windows CE environment. We chose this default because we envision most users in a complex environment, like Windows CE, will want the default behavior of their OS when an exception does occur. MON's Display Option and Enter Option commands allow us to modify the default settings.

```
MON>dov vector_catch
EO vector_catch = 0x0                       // Default: 0x0
//    Valid values:  0x0 - 0x1ff
//
//    Determines which exception conditions are detected and control passed
//    back to the debugger. The bits correspond as follows:
//      Bit#  Field Cause
//        0   0x001 Reset
//        1   0x002 Undefined Instruction
//        2   0x004 Software Interupt
//        3   0x008 Prefetch Abort
//        4   0x010 Data Abort
//        5   0x020 Address Exception
//        6   0x040 IRQ
//        7   0x080 FIQ
//        8   0x100 Error

MON>eo vector_catch = 0x2
```

As you can see from the Enter Option command above, we configured the probe so as to stop execution when an Undefined Instruction exception occurs.

Shuting Down

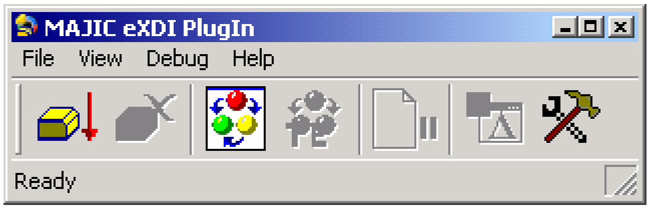
The Driver automatically shuts down as soon as all the connections to it are closed. Unless you manually started the Driver there should be no need to manually shutdown the driver via the window close button. Note that both Platform Builder and the PlugIn are clients of the Driver, so both need to be disconnected (if connected).

5 Using the PlugIn

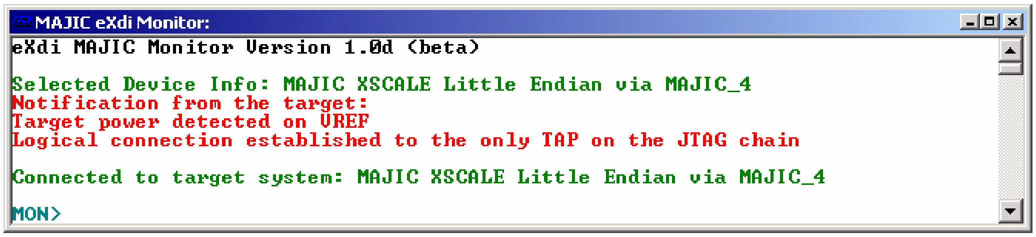
This chapter shows you the capabilities and usage of the MAJIC eXDI PlugIn.

Launching the PlugIn

The PlugIn can be started from the Platform Builder's menu item Tools -> Plug In -> MAJIC eXDI PlugIn. The PlugIn's main window pops up as shown below:



Now select the menu item Debug -> Connect to MAJIC Probe. If your setup is configured correctly the Driver starts up and brings up the MAJIC eXDI Monitor Window (example shown below). Note that when bringing up the PlugIn the MAJIC eXDI Monitor Window is typically already running. In such case, the PlugIn just connects to the existing driver instance.



```
eXdi MAJIC Monitor Version 1.0d (beta)

Selected Device Info: MAJIC XSCALE Little Endian via MAJIC_4
Notification from the target:
Target power detected on VREF
Logical connection established to the only TAP on the JTAG chain

Connected to target system: MAJIC XSCALE Little Endian via MAJIC_4

MON>
```

Note that some of the PlugIn Window's buttons become enabled after successful connection.

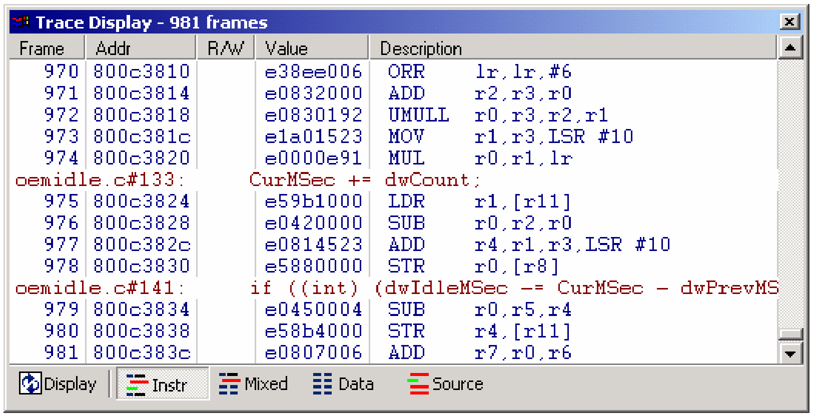
Using the Trace Window

One of the PlugIn's most useful features is the GUI-based Trace Display Window. It provides a convenient scrollable access to any captured trace data in your MAJIC probe. Use the PlugIn's menu item View -> Trace Display or the button shown to the right to open the Trace Window.

Enabling Trace: You can then use the [Display] button to get the trace data from the Driver and display it. However, this assumes you have actually captured some data. To do that you may need to Enable Trace via the menu item Debug -> Trace Enable or the button show to the right, followed by a run / break sequence to actually capture some trace from a real target run.

Note: Some processors with advanced trace features require the setup of a trace specification to determine what is to be captured. Please refer to your MAJIC User's Manual for details on trace setup.

Below is a sample Trace Display window:



```
Frame  Addr      R/W  Value     Description
  970  800c3810       e38ee006  ORR    lr,lr,#6
  971  800c3814       e0832000  ADD    r2,r3,r0
  972  800c3818       e0830192  UMULL  r0,r3,r2,r1
  973  800c381c       e1a01523  MOV    r1,r3,LSR #10
  974  800c3820       e0000e91  MUL    r0,r1,lr
oemidle.c#133:       CurMSec += dwCount;
  975  800c3824       e59b1000  LDR    r1,[r11]
  976  800c3828       e0420000  SUB    r0,r2,r0
  977  800c382c       e0814523  ADD    r4,r1,r3,LSR #10
  978  800c3830       e5880000  STR    r0,[r8]
oemidle.c#141:       if ((int) (dwIdleMSec -= CurMSec - dwPrevMS
  979  800c3834       e0450004  SUB    r0,r5,r4
  980  800c3838       e58b4000  STR    r4,[r11]
  981  800c383c       e0807006  ADD    r7,r0,r6
```

The Window title bar shows the total number of captured frames of trace data and the toolbar indicates the current display mode.

Trace Display Modes

The Trace Display Window's toolbar allows you update the trace data and select the display mode. The available modes are listed below.

Instr Displays valid instruction accesses in disassembled form. If the address of an instruction corresponds to a source line, then the source line is displayed first on its own line. If no source line exists, but a symbolic label corresponds to the address, then the symbol is displayed first on its own line.

Mixed Combination of Instr and Data display modes.

Data Displays valid data accesses in a formatted form. The R/W column displays the kind of access that occurred and the width is implied by the Value field. Note that many CPUs do not support data tracing.

Source Displays only the source or labels associated with valid instruction frames. This allows you to concisely see your execution flow from a source code point of view.

Trace filters and searching are also available via MON commands. Please refer to your MAJIC User's Manual for details.

Miscellaneous Features

Symbol Reload

Symbol/Source information is automatically loaded from Platform Builder upon the first reference. However, because of the dynamic nature of the Windows CE OS new programs might get loaded (or even reloaded at new addresses). In such a case, you must manually tell the Driver to reload symbols from Platform Builder. This is accomplished via the menu item Debug -> Reload Symbols or the toolbar button to the right.

Break

The PlugIn provides a break button that mirrors the behavior of the break button in Platform Builder. Normally there is no reason to use this button, but some times the Platform Builder user interface is so busy starting and stopping the processor for various reasons that it never acts upon a break request. The extra break provided here gives you an easy way to halt the processor if such a case arises. The break facility can used via the menu item Debug -> Break or the button to the right.

Always On Top

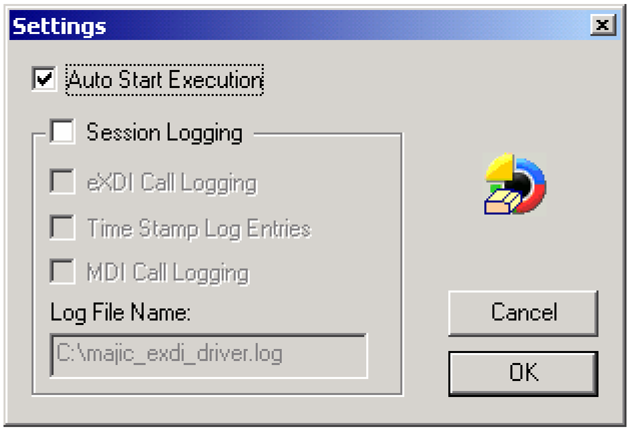
Many times it can be inconvenient to switch back and forth between the PlugIn and Platform Builder Windows. The PlugIn has a feature to allow it to always be a top level window. This allows you to maximize your Platform Builder Window and still see and operate with your PlugIn Windows. The feature is enabled via the menu item: View -> Always On Top.

The Settings Window

The Settings Window provides control of Driver/PlugIn File Logging and the Auto Start Execution feature. The Settings Window can be launched via the toolbar and the menu item View -> Settings.

Auto Start Execution

Some users may find that when starting up the Driver/Probe they don't need the added step of halting at the reset address, but would rather just auto-start execution. This behavior can be accomplished by clicking the checkbox below and selecting the [OK] button.



☑ Auto Start Execution

□ Session Logging

□ eXDI Call Logging

□ Time Stamp Log Entries

□ MDI Call Logging

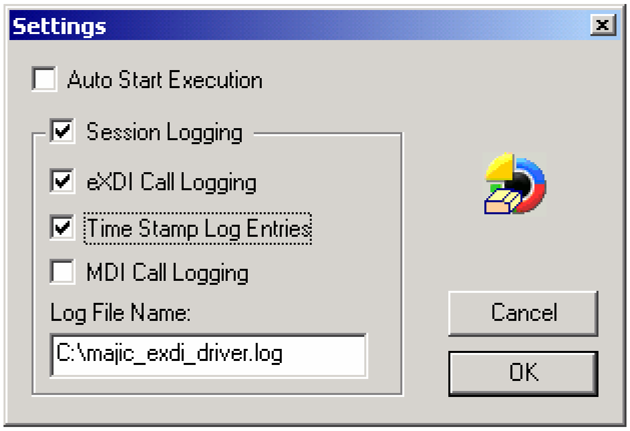
Log File Name: C:\majic_exdi_driver.log

Capturing a Debug Log Session

If you encounter problems that you think related your MAJIC Probe or software we may ask you to capture a debug log session so we can investigate the issue. The log will capture all the communication between your Probe, Driver, PlugIn and Platform Builder. Below are details of how to enable the log capture.

Before you begin a debug session from Platform Builder, start the MAJIC eXDI PlugIn by selecting Platform Builders menu Tools -> Plug In -> MAJIC eXDI PlugIn.

From the PlugIn Toolbar select the Settings ICON or menu item View -> Settings. This brings up the following window:



□ Auto Start Execution

☑ Session Logging

☑ eXDI Call Logging

☑ Time Stamp Log Entries

□ MDI Call Logging

Log File Name: C:\majic_exdi_driver.log

Select the Session Logging check-box. This enables the items contained in the session logging group. Check the eXDI Call Logging and the Time Stamp Log Entries check boxes followed by selecting the [OK] button. We suggest leaving the MDI Call Logging unchecked unless EPI has asked you to enable it. The Log File Name defaults to a location on the Drive C:. You can change this to any filename and/or location you wish. All the Driver captured log information is stored in this file.

You can now shutdown the PlugIn if you wish since it will communicate this setup information to the MAJIC eXDI Driver.

Note: The PlugIn itself also has a log file that is enabled via the Session Logging checkbox. The log file name is `majic_exdi_plugin.log` and it is put in the same directory referenced in the Log File Name edit box. If the issue you are trying to capture involves interactions with the PlugIn then this file can also be relevant.

Session Capture: If you enable Session Logging and leave the rest of the check boxes within Session Logging unchecked what is captured is all the information sent to the eXDI Driver Monitor Window. You might find this useful for recording the output that occurs in this window.

Disconnecting the PlugIn

To disconnect the PlugIn from the Driver simply select the menu item Debug -> Disconnect from MAJIC probe or the button shown to the right.

6 Support/Contact Information

EPI Support

EMail:	`support@epitools.com`
Internet:	`www.epitools.com`
Telephone:	`408 957-0350`

Microsoft Support

Please note that EPI support is specific to the EPI distributed tools, which does not include Platform Builder.   Please contact Microsoft's Support department via the web address below and click on the support link for help with Platform Builder and Win CE .NET specific questions.

`http://www.mswep.com`

Index