

## User's Supplement



Using EST's visionICE Emulator  
with Microtec's XRAY Debugger  
Rev. 2.1

**EST**  
EMBEDDED  
SUPPORT  
TOOLS CORP

EMBEDDED  
SUPPORT  
TOOLS CORP

**HIGH-END EMULATION ON A WHOLE NEW SCALE**

## Contents

Technical Support Information .....	4
Introduction .....	6
Getting Started.....	7
Emulator Configuration .....	7
XRAY Debugger Configuration .....	10
<i>XHM68K.SUP File</i> .....	10
<i>Compilation Options</i> .....	11
Target Configuration.....	12
EST Peripheral Utility Configuration .....	13
General Operation .....	14
Invoking XRAY .....	14
Initializing the Target .....	15
Running the Demo Program .....	16
Building the Demo Program.....	17
Downloading the Demo Program .....	18
Basic Debugging Session .....	19
Feature Overview .....	23
XRAY Monitor .....	24
Multiport Operation .....	24
Miscellaneous Topics and Tech Tips.....	25
Download Options .....	25
XRAY ICE Mode.....	26
XRAY – Loss of Contact with the Monitor.....	27
Secondary Emulator Channel .....	28
Show History Buffer.....	28
Download Problems .....	29
Unexpected Exceptions .....	29
Emulator Communications Port Number.....	30
Emulator's XRAY Command Stack Workspace .....	31
Zombie Processes .....	31
Ping .....	32
Netstat .....	32
Comtap .....	32
Resetting the Emulator Back to ASCII Mode .....	34
ENTRY.C and the STOP Instruction.....	34
NET> to >BKM> and >BKM> to NET> Modes .....	35
Known XRAY Problems .....	35

## **Introduction**

The purpose of this User's Supplement is to describe the operation of EST's visionICE scalable emulation system with Microtec's® XRAY Source Code Debugger. Using XRAY with the visionICE emulator is surprisingly easy and intuitive.

*Note: This supplement assumes that you are familiar with the XRAY development environment and the EST visionICE emulator with visionNET or visionEVENT cards installed. If you are not, consult the following Microtec® and EST Corp. documents:*

- *EST visionICE User's Manual*
- *EST Hardware Reference Manual*
- *Microtec® XRAY Debugger Reference Manual*

Microtec's® XRAY Debugger is available in many different configurations, host platforms, and target platforms. Host platforms supported by EST's implementation include:

- XHM68K (generic monitor version)
- XOM68K (C++ aware version of XHM68K)
- Masterworks or standalone
- PC Host
- UNIX Host

Like XRAY, the visionICE emulator is available in many different configurations – all versions of visionICE (visionNET and visionBDM) may be used with XRAY. The distinction is that the visionBDM emulator contains a serial and parallel communications port and the visionNET emulator contains an Ethernet network connection in addition to a serial and parallel port connection. The network connection provides enhanced performance.

A visionEVENT System is optional. This emulator function provides real-time tracing and events. In addition, an Event System allows the user to debug ROM or Flash resident code in real-time by providing software-to-hardware breakpoint translation.

In order to begin debugging application code with XRAY and the EST visionICE scalable emulation system, you will need to do the following:

1. Install the Microtec® XRAY (XHM68K) release disk(s).
2. Install the EST Utilities disk(s).
3. Complete the emulator, debugger, target, and peripheral utility configuration.
4. Optionally run the EST-supplied pre-built demo programs to familiarize yourself with the XRAY/visionICE environment.
5. Exit the demo program and begin debugging your application.

## **Getting Started**

Prior to operating the Microtec® XRAY debugger with the EST visionICE emulator, it is necessary to configure the emulator, the debugger, the target, and any peripheral utilities you will be using to ensure proper functionality.

### **Emulator Configuration**

There are a number of emulator settings that affect the visionICE/XRAY combination:

- Communications Parameters
- High-Level Language Setting
- Breakpoint Translation
- MRI Command Stack Workspace

#### **Communications Parameters**

Before XRAY can communicate with visionICE, the emulator and host must be set up for a common communications scheme. Typically, communications between XRAY and visionICE consist of either a serial port connection or an Ethernet connection.

If the emulator and host are to be connected via a network, the emulator's IP Address must be specified. The network version of the emulator is divided into two areas:

- The Network Card
- The Emulator Card

To initially set up the emulator's network parameters, a serial connection must be made to the Network Console Port. When properly connected to this port, a NET> prompt is displayed. A serial connection can be implemented via the Terminal Window of the EST Utilities Panel if a PC is available or a TIP Connection if a workstation is in use.

Once a serial port connection is established, the IP address, port numbering, and routing parameters can be changed. For visionICE, use the emulator's ethsetup command; use of the ethsetup command is fully described in the *EST visionICE Users' Manual*.

If the emulator and host are to be connected via a serial port connection, the emulator's baud rate is set using the CF BDR {rate} command. In order to set the baud rate on the emulator, you first need a serial port connection. The default baud rate is set to 9600 bits/second. The CF (Configuration) command is fully described in the *EST visionICE User's Manual*.

### **High-Level Language Setting**

The High-Level Language (HLL) setting is an option in the Configuration Options menu that allows you to specify the type of source code debugger connected to the emulator. The emulator's firmware simulates the XRAY Monitor, which is normally a small piece of code provided by Microtec® that gets linked with the application being debugged. In addition to this monitor, the emulator can also simulate pROBE and a number of other debug monitors. The type of monitor emulation is specified using the emulator's CF HLL command and is issued in the Terminal Window as a low-level command:

```
CF HLL [XRAY, XRAY+, SDS, SPECTRA, EST, ... ]
```

When using the XRAY debugger, the CF HLL menu option is set to XRAY. An emulator IN (Initialization) command should be issued after changing the debugger type in order for it to take effect.

### **Breakpoint Translation**

Software-to-Hardware Breakpoint Translation is an optional emulator setting, which affects how software breakpoints are handled. Normally, software

breakpoints replace the opcode at the breakpoint address with a special BGND debug instruction. The emulator provides a translation feature which translates software breakpoints to hardware breakpoints. This feature is available only with an EST visionEVENT System.

Anytime XRAY issues a Breakpoint command, the breakpoint defaults to a software breakpoint. This is usually fine as long as the software being debugged is in writeable RAM. If the code being debugged is located in Flash or ROM, an opcode swap cannot be performed. In this case, the breakpoint translation setting becomes very useful.

The Software-to-Hardware Breakpoint Translation configuration option is specified using the CF SB command. The default setting for this entry is CF SB SB, which indicates normal software breakpoint operation. To specify breakpoint translation, issue the following command in the emulator's Terminal Window:

```
CF SB HBC
```

An emulator IN command should be issued after the CF SB HBC command in order for it to take effect.

There are some restrictions that apply when breakpoint translation is selected. Normally, an unlimited number of software breakpoints can be inserted. When hardware breakpoints are traded in place of software breakpoints, the number of breakpoints is limited to the number of hardware comparitors supported by visionEVENT—typically eight.

It should be noted that if an Event System is not present, you can still debug ROM/Flash resident code—the emulator uses a special Assertion Mode that runs in non-Real-Time Mode. Instead of inserting software breakpoints into the instruction stream, the emulator single steps the code and performs comparison operations after each step.

### **MRI Command Stack Workspace**

The MRI Command Stack Workspace can be relocated using the emulator's MRI command.

The XRAY Monitor is simulated internal to the emulator; however, a small MRI Command Stack Workspace is located on the target. XRAY requires up to 16 bytes of RAM in order to implement certain high-level functions, which typically involve moving the stack pointer and evaluating return values of functions.

By default, the emulator locates this temporary data space at 500 Hex. To relocate the command stack space, use the emulator's MRI command:

```
MRI <CR>           // Displays the current workspace  
  
MRI value <CR>     // Relocates the workspace (value is  
                    in hex)
```

This area must be located in readable/writable memory. Most of the operations that use this workspace occur when the debugger is stopped. XRAY uses this memory as a temporary stack and performs non-destructive accesses to this area, returning the original values before allowing the code to run.

## **XRAY Debugger Configuration**

XRAY does not require any special configuration options to work properly with the EST visionICE emulator. The only two areas that need to be addressed are the entries in the XHM68.SUP file and the debug compilation options.

### **XHM68K.SUP File**

The XHM68K.SUP file should be placed in the directory that XRAY is invoked from for UNIX and in the /mri/master/bin subdirectory for PC users. This file contains entries that specify how the host portion of XRAY should communicate with the XRAY Monitor – in this case, the emulator resident simulated monitor. There are only four lines required in the .SUP file:

```
C: or S: Communication Scheme  
B:  
G:XRAY  
W:0, 0, 1000
```

*Note:* The .SUP file command entries are fully described in Microtec's® XRAY In-Circuit Debugger Monitor Specification.

**C: or S: Communication Scheme:** This line contains a description of the communication scheme used between the host system and the XRAY Monitor. This entry specifies the various serial port parameters or Ethernet connection types. The following examples illustrate the required parameters:

```
C: Is, 192.9.200.60, 1235
```

This communications entry instructs XRAY to use a TCP type Ethernet connection to 192.9.200.60 on port 1235. The *Is* part of the entry indicates the TCP connection.

`C:Id,192.9.200.60,1235`

This communications entry instructs XRAY to use a UDP type Ethernet connection to 192.9.200.60 on port 1235. The *Id* part of the entry indicates the UDP connection. This is the normal visionNET connection type used when communicating with XRAY. In general, TCP communications are preferred over UDP. The *S:* communications entry is used when a serial port connection is desired.

`S:com2,19200,8,n,2`

This entry is used on a PC to specify COM2 as the communications port using 19200 baud, 8 bits per char, no parity, and 2 stop bits.

`S:/dev/ttya,9600,8,n,1`

This entry specifies *ttya* as the serial communications port on a UNIX workstation.

**B: [bad command]:** The *B:* entry specifies the Clear Buffer command. With no 'bad command' argument specified, the XRAY host will not send any Buffer Clear command to the monitor; this is the setting required by visionICE.

**G:XRAY:** The *G:* entry specifies the Auxiliary Monitor Go command. The Auxiliary Monitor is considered the ICE capability of visionICE. After XRAY issues a single-line ICE command (e.g., 'ice in' to initialize the emulator via the XRAY command line), it will terminate the host-to-monitor transaction by sending the string specified in the *G:* command. In this case, the string must be "XRAY".

**W:0,0,1000:** The *W:* entry is used to specify the character, line, and command delay times. The first entry specifies the delay in milliseconds between characters sent from XRAY and the monitor. The second parameter specifies the line delay in milliseconds, and the third entry specifies the command delay in milliseconds.

### Compilation Options

In order to use XRAY to debug a program, the program must be compiled with debug information. This is typically done by using the `-g` compiler and assembly options:

```
mcc68k testfile.c -o testfile.obj -g <other options>
```

## Target Configuration

Before you can begin a debug session, the emulator and target must be configured to match the desired target configuration. Assuming the target is an EST SBC360 evaluation board and the standard @5000 demo program is to be used, the target requires the following setup considerations:

- Target Board Switches
- Emulator CS and SC Register Setup

Once the target hardware is set up, the emulator must be configured to mirror the hardware. The emulator contains an internal NVRAM that can be set up to mirror the SIM60 module. This internal storage of the target parameter is used to inform the emulator of the target configuration and also to provide an initialization mechanism for resetting and initializing the target.

Using the emulator's CS (Chip Select) and SC (System Configuration) commands, the chip select and system configuration registers can be setup to reflect the target. In the case of an SBC360, a shortcut for configuring the emulator is to enter the following commands at the >BKM> prompt:

```
>BKM> CF TAR 360  
>BKM> SC DEFAULT
```

The first emulator command sets the target processor type to a 68360 and the second sets the register set to the default 68360 register set -- in this case, the register set is for an SBC360.

Whenever the emulator/target combination is initialized via an emulator IN command, the target will be in a state that should reflect its hardware configuration.

If an EST evaluation board is not the target board, setting up the SIM Register is a bit more laborious. In this case, EST's visionSIM Register Configuration Utility can greatly reduce the configuration task. The alternative is working your way through a data book and using the emulator's SC and CS commands to enter all critical register values.

Refer to the *EST visionICE User's Manual* for additional information on target configuration and options.

## **EST Peripheral Utility Configuration**

The term Peripheral Utility is used to refer to any of the EST utilities that can be used in conjunction with the XRAY/visionICE combination; in particular:

- visionXP UNIX Utility Panel
- EST Utilities Panel
- visionEVENT Tools for Windows Toolbar

These convenience utilities add functionality to other applications such as third-party debuggers. When used with XRAY, these utility panels provide:

- A terminal interface into the visionICE emulator
- Flash programming capabilities
- Hardware breakpoints
- Events
- Trace

All of these features are described in detail in the *EST visionXP User's Manual* and the *EST visionICE User's Manual*.

When configuring these utilities, keep in mind that they must communicate with visionICE via a separate channel from the one that XRAY is communicating with. Some possible scenarios might be:

- XRAY-to-emulator over network and Emulator-to-Utility over serial port
- XRAY-to-emulator over serial port and Emulator-to-Utility over parallel port
- XRAY-to-emulator over network and Emulator-to-Utility over network

The last combination might seem contradictory, but in the case of the network, different communication port numbers can be used. For example, when working with visionXP, which operates on port 1234 and XRAY, which typically operates on port 2 for TCP, the following restrictions apply:

- visionXP only supports a network connection
- XRAY does not support parallel communications