Debug with printf in real-time

IN THIS DOCUMENT

The xCORE debugger lets you suspend execution of a program in order to analyze its internal state. However, if your program contains timing-critical behavior, for example due to it implementing a real-time communication protocol, the act of suspending the program may cause other system components to fail, preventing further debugging.

An alternative approach to debugging is to add trace statements to your program that are used to observe its internal behavior at run-time (sometimes referred to as `printf` debugging). By printing the results of intermediate calculations, you can quickly isolate where errors occur in your program.

In a traditional debugging environment, outputting data using a standard such as JTAG results in interrupts that block core execution, slowing your program down considerably. xTIMEcomposer lets you redirect the standard streams `stdout` and `stderr` to an xTAG debug adapter, where the data is buffered until it can be output to the host.

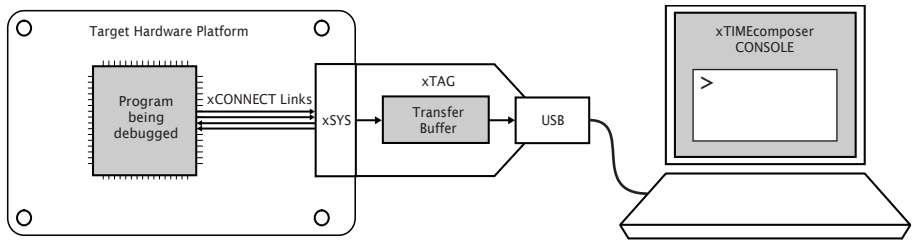

Figure 1: xTAG debug configuration with I/O redirection

In this configuration, calls to output routines such as `printf` complete as soon as the data has been output on an xCONNECT Link, minimizing the effect on the program's timing characteristics. This allows debugging statements to be added to many timing-critical code sections and viewed in a console during execution. In the case of a program crash, all remaining contents in the xTAG buffer is forwarded to the PC, ensuring important information is not lost.

If you are using a legacy FTDI or xTAG-1 debug adapter, or if the XSYS connector on your target hardware does not provide an xCONNECT Link, you can output data over your adapter's UART interface instead (see §3). Note that the UART interface offers significantly reduced performance.

Publication Date: 2013/11/12 Document Number: X1093B

1 Redirect `stdout` and `stderr` to the xTAG

The program below redirects standard output to the xTAG.

```
#include <stdio.h>
#include <xscope.h>

port receive;
port transmit;

int process(int);

void xscope_user_init(void) {
  xscope_register(0);
  xscope_config_io(XSCOPE_IO_BASIC);
}

int main() {

  while (1) {
    int dataIn, dataOut;

    receive :> dataIn;
    dataOut = process(dataIn);

    /* Debug Information */
    if (dataOut < 0)
      printf("%d %d", dataIn, dataOut);

    transmit <: dataOut;
  }
}
```

In the constructor `xscope_user_init`, the call to `xscope_register` initializes the xTAG interface, and the call to `xscope_config_io` redirects the streams `stdout` and `stderr` to this interface.

The main program inputs data from a port, performs a computation on it and outputs the result to another port. It uses the standard output function `printf` to log instances where the computed result is less than zero.

You can use the C standard I/O functions on any core at the same time. This usage results in a single channel end being allocated on each tile on which data is output.

You can timestamp the output data by calling `xscope_config_io` with the option `XSCOPE_IO_TIMED`. This causes the output timestamp to be displayed with the data in the console. Note that this also reduces the amount of data that can be buffered at any time.

2 Run a program with xTAG output enabled

To redirect standard output to the xTAG and display it in the console, you must build and run your program with the xSCOPE instrumentation library. To build and run your program, follow these steps:

1. Open the Makefile for your project.
2. Locate the `XCC_FLAGS_`*config* variable for your build configuration, for example `XCC_FLAGS_Release`.
3. Add the option `-fxscope`.
4. If you are developing using xTIMEcomposer Studio, create a Run Configuration for your target device (see XM-000963-PC). In the **xSCOPE** tab, select **Offline mode**. Click **Run** to save and run the configuration.

   xTIMEcomposer loads your program, displaying data received from the xTAG in the console.
5. If you are developing using the command-line tools, pass the option `--xscope` to XRUN, for example:

   ▶ `xrun --xscope myprog.xe`

   XRUN loads your program and remains attached to the xTAG adapter, displaying data received from it in the terminal. XRUN terminates when the program performs a call to `exit`.

3 Output using the UART interface

If you are using a legacy FTDI or XTAG-1 debug adapter, or if the XSYS connector on your target hardware does not provide an xCONNECT Link, you can output data over the UART interface provided by your adapter.

To use the UART interface, you must provide the xSCOPE library with a 1-bit UART TX port that has been initialized with the pin connected to the UART-TX pin on your debug adapter. An example initialization is shown below.

```
#include <platform.h>
#include <xscope.h>

port uart_tx = PORT_UART_TX;

void xscope_user_init(void) {
  xscope_register(0);
  xscope_config_uart(uart_tx);
  xscope_config_io(XSCOPE_IO_BASIC);
}
```

To run your program in xTIMEcomposer Studio, create a Run Configuration for your target device (see XM-000963-PC) and select the option **Run UART Server**.

To run your program using the command-line tools, pass the option `--uart` to XRUN, for example:

```
▶ xrun --uart --xscope myprog.xe
```

Because the UART interface uses a port instead of an xCONNECT Link, you can use the C standard I/O functions on a single tile only.