

Abstract

This application note demonstrates how to redirect the Instrumentation Trace Macrocell (ITM) debug output to a third-party application from within μ Vision. This is helpful when data must be displayed in a way not offered by μ Vision, or the target uses a special debug protocol, which needs data processing. A Cortex-M3 processor-based device is used in the example for demonstration purposes.

Contents

Abstract.....	1
Components	1
Introduction.....	1
Application Setup	2
Creating the Pipe Server	2
Connecting μ Vision as a Client to the Server	3
Testing the Example	4
Notes on Performance.....	5
Revision History	6

Components

The application note consists of:

- This document APNT240.pdf
- A zip-file APNT240.zip containing the μ Vision project and the Microsoft Visual Studio solution project.

Introduction

Cortex-M3 processor-based devices feature the Instrumentation Trace Macrocell (ITM). With the help of ITM, a program can send data easily to the debugger, while μ Vision does all the processing to capture the device data during a debugging session. 32 ITM channels can be addressed in total. Some channels have dedicated default functionality. For example, the RTX kernel uses channel 31 to report task events. Channel 0 is a general purpose text output channel, which gets routed into the Debug (printf) Viewer.

Except for channel 31, μ Vision offers a logging feature for ITM output. The debug command “ITMLOG” can redirect an ITM channel output to a file or to a Windows named pipe. While redirecting to a file might not be suitable for reading data instantly into another application, Windows named pipe makes it possible to send the debug output directly to another application for post processing. In this case, μ Vision plays the client role, while the third-party application is the named pipe server.

Application Setup

The application note consists of a Visual Studio 2010 solution that sets up the Windows named pipe and a μ Vision project that sends ITM data to the pipe.

<i>Folder Name</i>	<i>Description</i>
itmpipe	Base folder for the application note
VS2010_sln	Visual Studio 2010 solution. Contains the file <code>itmpipe.c</code> with the C source code.
VS2010_sln\Release	Precompiled executable <code>itmpipe.exe</code> that creates the Windows named pipe
ITM_Test	μ Vision project <code>ITM_Rate.proj</code> sends data to the pipe server

Creating the Pipe Server

The small console application (Visual Studio project) attached to this application note demonstrates the setup of the named pipe server. The functions used are well described in the Windows API documentation and are not discussed in here. The code in the file `itmpipe.c` creates the named pipe:

```

PipeName    = "\\.\pipe\myitmpipe";
PipeHandle  = CreateNamedPipe(
                PipeName,
                ...
            );

```

Note the special naming convention for the named pipe, and that we have to escape the back-slash character ('\`\`') in the C-string. Now we have a valid handle to the pipe and can wait for the client (μ Vision issuing the `ITMLOG` command) to connect:

```

Connected = ConnectNamedPipe( PipeHandle, NULL );

```

Thereafter, the pipe waits for data to arrive:

```

Success = ReadFile( PipeHandle, Buffer, BUFFERSIZE - 1 , &BytesRead, NULL );

```

We can call this in a loop to listen permanently on the pipe for additional data. When the call returns 0, it means in most cases, the pipe was closed by the remote side. For example, when the debugging session finished or the logging was shut down.

Basically, this is all that is needed to setup the pipe. Once the pipe has been closed, a proper shutdown of the pipe server has to follow. This is not discussed here.

You can build and run the project with the included Visual Studio project. The server must be started prior to connecting μ Vision as a client.

Connecting μ Vision as a Client to the Server

Use the μ Vision project attached to the application note to start a debugging session. The test example executes the `ITMLOG` command saved in the file `itmlog_to_pipe.ini` and redirects output to channel 1. The command parameters are the channel number and the named pipe. The pipe name must be the one used by the server.

```
ITMLOG 1 >> "\\.\pipe\myitmpipe"
```

This call succeeds when the server is already running; otherwise it fails as the named pipe is not available.

The `main.c` program has to be instrumented to use the defined channel for its output. This can be done with a slightly modified version of the CMSIS-compliant function `ITM_SendChar()` from the file `core_cm3.h`. By default, this function is used to send data via ITM channel 0. In the test project, the channel number is a parameter:

```
__STATIC_INLINE uint32_t ITM_SendCharX (uint32_t Channel, uint32_t Data)
{
    if ((ITM->TCR & ITM_TCR_ITMENA_Msk) &&          /* ITM enabled */
        (ITM->TER & (1UL << Channel) ) )            /* ITM Port X enabled */
    {
        while (ITM->PORT[Channel].u32 == 0);
        ITM->PORT[Channel].u8 = (uint8_t) Data;
    }
    return (Data);
}
```

Testing the Example

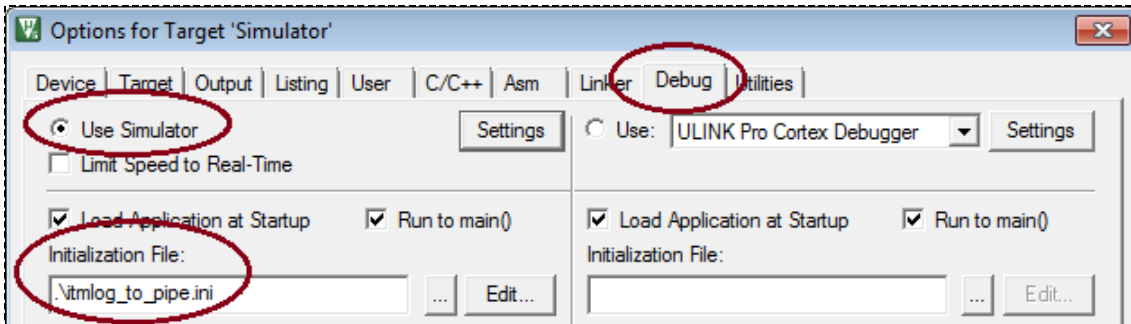
1. Extract the archive file on your hard disk.
2. Run the prebuilt executable `\VS2010_sln\Release\itmpipe.exe`
The message “pipe handle ok” is printed on the Command prompt. (Leave the window open.)



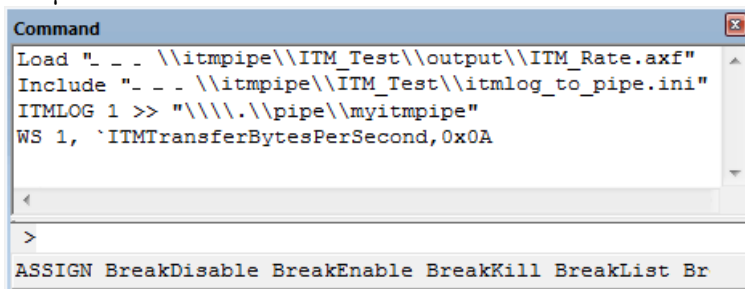
3. Open the μ Vision project `\ITM_Test\ITM_Rate.uvproj`. The project is built to use the Simulator.
The command shown below is saved in the initialization file `itmlog_to_pipe.ini` and gets executed at the start of each debugging session.

```
ITMLOG 1 >> \\\\.\\pipe\\myitmpipe
```

The initialization file is configured in the dialog **Options for Target – Debug**.



4. Build the application. Menu **Project – Build target**.
5. Start the debugging session. Menu **Debug – Start/Stop Debug Session**.
The μ Vision **Command** window shows that the `ITMLOG` command was executed.



The pipe server reports “client connected”.



6. Run the program.
Alphabet characters get printed in the DOS console window. The characters are sent by the embedded application via ITM channel 1.

You can use other project targets to connect an MCBSTM32E board to the ULINK2 or ULINKpro debug probe. Build the project and start the debug session.

You may use another ITM channel as well. For this:

- Modify the initialization file `itmlog_to_pipe.ini`
- Set the ITM output channel in `main.c`

```
ITM_SendCharX( 1, TransferChar );
```
- When using an evaluation board, enable the ITM output channel in the dialog **Options for Target – Debug – Trace – ITM Stimulus Port**.
(Refer to the chapter *Trace Configuration* in the ULINK2 or ULINKpro User's Guide for details.)

Notes on Performance

Up to the μ Vision version 4.60, the ITM output was buffered. While this ensures the maximum performance, buffering has some disadvantage when redirecting the output. The buffers get flushed at certain times. If the application on the other end of pipe waits for a single byte, that application might have to wait until the debug session ends.

Starting with μ Vision 4.60, the type of the ITMLOG output file is checked. If a pipe is detected, the output buffers will be bypassed, so that each byte is immediately outputted. This adds a small performance penalty on the actual output rate (of about 10% on a STM32F103 target connected via ULINK2).

Revision History

- March 2013: Initial Version
- November 2013: Configuration Pictures added