GCC for MSP430™ Microcontrollers

This document describes:

- How to use the open-source GCC compiler for MSP430™ microcontrollers to build an example for an MSP430 target device
- How to debug software with the GDB Agent and the GNU Debugger tools

Experience with a command terminal on either the Microsoft® Windows® operating system or the GNU Linux® operating system is necessary.

Contents

1 Prerequisites

This document assumes that a version of the GNU Make utility is installed on the system and that it is available on the system path.

The placeholder **INSTALL_DIR** refers to the directory where you installed the GCC MSP430 package. The directory **INSTALL_DIR/bin** should be on the system path.

As this document cannot go into much detail on how to use the GNU Compiler and Debugger, see *GDB: The GNU Project Debugger* [1] and *Using the GNU Compiler Collection* [2].

2 Quick Start: Blink the LED

2.1 Building With a Makefile

1. In the command terminal, go to the INSTALL_DIR\examples directory.
2. There are examples for Windows and Linux. They are located in the corresponding sub-directories. Choose one of the examples suitable for your Operating System and MSP430 target device.
3. Change to the directory and type `make`.
4. The binary can now be downloaded and debugged on the target hardware.

2.2 Building Manually With gcc

To build one of the examples manually, open a terminal and change to the example suitable for your target device and operating system. The compiler executable **msp430-elf-gcc** must be available on your system path.

```
msp430-elf-gcc -I <Path to MSP430 Support Files> -L <Path to MSP430 Support Files>
-T DEVICE.ld -mmcu=DEVICE -O2 -g blink.c -o blink.o
```

The placeholder **<Path to MSP430 Support Files>** is the directory that contains the MSP430 Support Files (header files and linker scripts to support the different MSP430 devices).

The placeholder **DEVICE** tells the compiler and linker to create code for your target device. The command line argument `-T DEVICE.ld` is optional, as the compiler automatically selects the correct linker script from the `-mmcu=DEVICE` argument.

Example:

```
msp430-elf-gcc -I=../../../include -L=../../../include -T msp430fr5969.ld -
mmcu=msp430fr5969 -O2 -g blink.c -o blink.o"
```

2.3 Debugging

2.3.1 Using the GDB Agent

2.3.1.1 Introduction

The GDB Agent is a tool to connect the GDB with the target hardware to debug your software. The GDB Agent uses the MSP430 debug stack to connect to the hardware and provides an interface to the GDB.

On Windows, both a console and a GUI application version of the GDB agent is provided. Only the console application is supported on Linux.

2.3.1.2 Console Application

If you use the console application, invoke it from a command terminal using following syntax:

Linux:

```
INSTALL_DIR/bin/gdb_agent_console INSTALL_DIR/msp430.dat
```

Windows:

```
INSTALL_DIR\bin\gdb_agent_console INSTALL_DIR\msp430.dat
```

The console application opens a TCP/IP port on the local machine. It displays the port number in the console. By default, this port number is 55000.

2.3.1.3 GUI Application

After you start the GUI application, configure the GUI and then start the GDB server.

1. Click the *Configure* button and, in the Select board configuration file window, select the msp430.dat file. If successfully configured, an MSP430 device is displayed in the <Targets> list. The TCP/IP port for the GDB Agent is displayed when the MSP430 device is selected from the list.
2. To start the GDB server, click the <Start> button when the MSP430 device is selected.

2.3.1.4 Attaching the Debugger

After starting the debugger and to attach to the GDB server, use the *target remote [<host ip address>]:<port>* command, where <port> is the TCP/IP port from Section 2.3.1.3. If the GDB Agent runs locally, you can omit the host IP address.

2.3.1.5 Configuring the Target Voltage

To configure the target voltage for your device, open the file msp430.dat in a text editor. To change the voltage, modify the key msp430_vcc. By default, this value is set to 3.3 volts.

2.3.2 Starting GDB Agent

On Microsoft Windows, you can start the GDB Agent either as a small GUI application or on the command line. On GNU Linux, only the command line version is available.

2.3.2.1 Using the GUI

Open the **INSTALL_DIR/bin** directory and double-click gdb_agent_gui.

1. After the program is started, click the button *Configure*, select *msp430.dat*, and click *Open*.
2. Click on the button *Start* under the Panel *Controls*.
3. The "Log" window now contains the status message *Waiting for client*.
4. Leave the window open until the end of the debugging process.

2.3.2.2 Using the Command Line

Open a command terminal, change to INSTALL_DIR and type:

Linux:

```
./bin/gdb_agent_console msp430.dat
```

Windows:

```
.\bin\gdb_agent_console msp430.dat
```

2.3.3 Debugging With GDB

2.3.3.1 Running a Program in the Debugger

1. In the command terminal, go to the INSTALL_DIR\examples\[Selected example], and type the command `make debug`.
2. This command starts the GDB, and it waits for commands. This is indicated by the prompt <gdb>.
3. To connect GDB to the GDB Agent, type the command `target remote :55000` and press enter.
4. To load a program binary to the MSP430 target device, type `load`.
5. Typing the command `continue` (short version: c) tells GDB to run the loaded program.
6. The LED on the target board will blink.

2.3.3.2 Setting a Breakpoint

1. Connect the GDB to the GDB Agent as previously described, and load a program to the device.
2. To set a breakpoint on a function, type: `break function name`.
3. To set a breakpoint on a source line, type: `break filename:line`.
4. When you run the program, the program execution stops at the entry to the specified function or stops at the specified line.

2.3.3.3 Single Stepping

1. Connect the GDB to the GDB Agent as previously described, and load a program to the device.
2. After the debugger has stopped the program at a breakpoint, you can step through the code:
   - To execute the source line, type `next`.
     `next` does not step into functions, it executes the complete function and stops on the line following the function call.
   - To execute the next source line and step into functions, type `step`.
   - To execute the next instruction, type `nexti`.
   - To execute the next instruction and step into functions, type `stepi`.

2.3.3.4 Stopping or Interrupting a Running Program

1. Connect the GDB to the GDB Agent as previously described, and load a program to the device.
2. To stop a running program and get back to the GDB command prompt, type `Ctrl+C`. This currently applies only on Linux.

3 Creating a New Project

1. Create a directory for your project.
2. Copy one of the makefiles from the example project into your project directory.
3. Open the copied makefile, and set the variable DEVICE to the target device you are using.
4. Set the variable GCC_DIR to point to the directory where you installed the GCC MSP430 package.
5. Include all of your project source files (that is, the *.c files) as a dependency for the first target of the makefile.
6. You can now go to the project directory in a terminal and type `make` to build the project or `make debug` to start debugging the project.

```
OBJECTS=blink.o

GCC_DIR =  ../../../bin
SUPPORT_FILE_DIRECTORY = ../../../include

# Please set your device here
DEVICE  = msp430X
CC      = $(GCC_DIR)/msp430-elf-gcc
GDB     = $(GCC_DIR)/msp430-elf-gdb

CFLAGS = -I $(SUPPORT_FILE_DIRECTORY) -mmcu=$(DEVICE) -O2 -g
LFLAGS = -L $(SUPPORT_FILE_DIRECTORY) -T $(DEVICE).ld

all: ${OBJECTS}
    $(CC) $(CFLAGS) $(LFLAGS) $? -o $(DEVICE).out

debug: all
    $(GDB) $(DEVICE).out
```

4 Troubleshooting

4.1 Missing libexpat

If msp430-elf-gdb displays the following error message when trying to debug under Linux, even though libexpat is installed:

```
../../i686-msp430-gcc/bin/msp430-elf-gdb: error while loading shared libraries:
libexpat.so.0: cannot open shared object file: No such file or directory
make: *** [debug] Error 127
```

Go to the directory where libexpat*.so is installed (probably /usr/lib/i386-linux-gnu) and create a link with the name that msp430-elf-gdb expects:

```
sudo ln -s libexpatw.so.1 libexpat.so.0
```

4.2 Could Not Initialize MSP430 (TIUSB)

If GDB Agent exits with the following error message after msp430-elf-gdb tries to connect under Linux:

```
Could not initialize MSP430 (TIUSB)
MSP430 Error :Could not find MSP-FET430UIF on specified COM port
Looking for MSP430 devices:1 devices detected.
Device ttyACM0: status is Available
Failed to connect to target...exiting
```

The current user probably does not have the necessary privileges to access the UIF hardware. Try starting GDB Agent with root privileges:

```
sudo ./gdb_agent_console msp430.dat
```

4.3 GDB Time-out

To avoid time-out errors in GDB, use the command `set remotetimeout 120` to increase the default time-out (2 s) threshold for remote commands to complete. Time-outs are most common when connecting, particularly when connecting to an MSP430 device that requires a firmware upgrade of the debug probe.

5 References

1. *GDB: The GNU Project Debugger*, Free Software Foundation, Inc. (https://sourceware.org/gdb/current/onlinedocs/)
2. *Using the GNU Compiler Collection*, Richard M. Stallman (http://gcc.gnu.org/onlinedocs/gcc.pdf)

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from March 26, 2016 to July 26, 2016	Page
• Removed all instances of "Red Hat" including in document title	1

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265