Application Report
SLAA341–November 2006

A Flash Monitor for the MSP430

Jim Patterson *Americas Field Applications*

ABSTRACT

A small monitor program for the MSP430F1xx and MSP430F4xx microcontrollers that incorporate at least one integrated USART provides a facility for examining the device memory and updating the flash contents with new application code via a universal synchronous/asynchronous receive/transmit (USART) peripheral interface. A personal computer (PC) and a terminal emulation package are all that is required to access the device memory. The monitor allows field updates of system firmware without the need to provide access to the JTAG signals or to the GPIO pins used by the bootstrap loader (BSL).

Contents

List of Figures

List of Tables

1 Introduction

The monitor is a small program that provides a facility for examining and modifying memory via a MSP430 universal synchronous/asynchronous receiver/transmitter (USART) port. A PC with a terminal emulation program, such as TeraTerm, HyperTerminal®, or minicom, can be used to communicate with the MSP430-based system. The monitor occupies two segments of the MSP430 main flash memory. The monitor is designed to have minimal impact on the operation of the end application. The monitor uses no MSP430 interrupts and reserves no processor resources beyond the two 512-byte flash segments.

There are several options for the monitor that can be selected by editing a header file. These include:

- Which MSP430 family device the monitor is to run on
- How interrupts are handled
- Which USART port is used for communication
- Whether the monitor or the user application runs first following a system reset
- Whether or not a password is required to erase flash memory
- Whether or not the monitor can dump MSP430 memory contents
- Whether or not commands and data received via USART are echoed to the terminal

Section 3 contains instructions for selecting the monitor build options.

The monitor code as provided assumes that a 32,768-Hz crystal is connected to the LFXT1 oscillator, as ACLK is the source for the USART baud rate clock (BRCLK).

The monitor and sample application provided were built using IAR Embedded Workbench™ for MSP430 (KickStart edition).

2 Monitor Operation

2.1 Configuring the Terminal Emulation Program

The terminal emulation must be configured for 9600-bps operation with eight data bits, one stop bit, no parity, and no flow control. The terminal program should also be configured to insert at least a 1-ms inter-line and to send line end (Ctrl-M, 0x0D) with line feed. The monitor normally echoes each received character back to the terminal, but this can be changed as a build option.

2.2 Starting the Monitor

The monitor can be built to run following a device reset or to run when it is invoked by the application. This is a build-time selectable option.

When the monitor starts, it disables the watchdog timer (WDT), configures the flash timing generator, and configures the USART for 9600-bps operation with eight data bits, one stop bit, and no parity. The next operation depends upon whether the monitor or the application runs first.

If the monitor is built to run at system reset, Timer_A is configured to overflow in approximately 2 seconds and is started. The monitor next transmits a ? prompt on the USART and enters a loop that checks the USART receive flag and the Timer_A overflow flag.

If any character is received on the USART before Timer_A overflows, control is transferred to the monitor main loop, in which it waits for a character to be received from the terminal. If Timer_A overflows before a character is received on the USART, the monitor stops Timer_A and attempts to transfer control to a user application. It first checks the application's reset vector to see if it has been programmed. If the vector has been programmed, control transfers to the instruction that is addressed by the reset vector. If the vector is blank, control returns to the monitor main loop.

If application is configured to run at reset, the monitor is started when the application transfers control to the monitor. In this case, Timer_A is not configured or started, and the monitor enters the main loop.

2.3 User Commands

The monitor responds to five commands, as shown in Table 1. Each command is a single upper-case character. Any characters that are not valid commands are echoed to the terminal program but are ignored. All commands and hexadecimal values must be entered in upper case. When hexadecimal values are entered, anything other than a valid hexadecimal character (0–9 or A–F) is treated as a delimiter.

Table 1. Monitor Commands

COMMAND	PARAMETER	ACTION
C	None	**Checksum:** Calculate a checksum across the user available main memory
D	Start address and count (hexadecimal)	**Display memory:** Dump memory contents beginning at the specified address
E	Password, start address, stop address	**Erase flash:** Erase user flash memory
G	None	**Go:** Execute loaded application.
I	Segment	**Erase info memory:** Erase information memory segment A or B
U	Text file	**Update flash:** Program the flash memory using the contents of an MSP430-TXT formatted program file

2.3.1 Generate a Flash Checksum

The C (checksum) command instructs the monitor to generate a modulo-65536 sum of the contents of the main flash memory that are available for the application. The checksum is transmitted to the terminal as a four-character hexadecimal number and can be used to verify the validity of the loaded program.

2.3.2 Display MSP430 Memory

The D (display memory) command begins a dump of memory contents to the terminal. The command syntax is: D <start address> <word count>.

Both the start address and count are entered as hexadecimal values. The data are returned as the hexadecimal address of the first word followed by the contents of eight consecutive words presented as four-character hexadecimal values, as shown in Figure 1.

The D command is followed by the starting address. If a valid 16-bit address is not received, the address defaults to the start of the available flash memory. If no word count is entered, the monitor displays a line of eight words and pauses until another character is received. If the received character is Q (quit) the monitor returns to the command prompt. If it is any other character, the monitor transmits the contents of the next eight words to the terminal. The monitor also returns to the command prompt if the memory address reaches the end of the available flash memory.

Figure 1 shows a request to display eight words starting at address 0x1500.

```
? D 1500 8
1500 E0F2 0003 0021 4130 0020 4031 0A00 12B0
```

Figure 1. Memory Dump Command Example

The memory address range 0x0000–0x0200, which includes the special function registers and peripherals, cannot be displayed.

2.3.3 Erasing Flash Memory

The E (erase memory) command can erase the entire main flash memory except for the two segments that contain the monitor program itself, or it can erase only a specified range of memory. The erase command can be password protected to reduce the chance of inadvertently erasing the flash memory.

2.3.3.1 Password Protection Disabled

If the password protection is not enabled, upon receiving an E command, the monitor looks for a start address and end address for the memory to be erased. If a valid start address is not entered, the monitor erases the entire user flash. If a valid start address is entered without a valid end address, the monitor erases from the start address to the end of user flash.

2.3.3.2 Password Protection Enabled

Upon detecting the E command, the monitor prompts for the password, which is a 16-bit value stored in the first word before the application's interrupt vector table. If the next four characters received are not the ASCII representation of the password, the monitor returns to the command prompt without erasing the flash. When the flash is fully erased, the password value is always 0xFFFF.

If the received password matches the calculated password, the monitor prompts for a start address and an end address. If the addresses entered do not represent a valid address range for the main flash, the monitor returns to the command prompt. The start address must be greater than or equal to the start address of the main flash available for applications. The end address must be less than or equal to the end address of the available flash and must be greater than or equal to the start address.

The flash memory is erased by segment, so that when valid addresses are entered, the segment containing the start address, the segment containing the end address, and any segments in between are erased.

2.3.4 Running the Loaded Application

The G (Go) command has no argument. It transfers control to the entry point of a loaded application by making an indirect branch to the instruction addressed by the application's reset vector located at 0xFBFE. If the vector is blank (0xFFFF), the monitor returns to the main loop and issues another prompt.

2.3.5 Erasing the Information Flash

The I (Erase Information Flash) command erases one of the 256-byte segments of information flash, either segment A or B. The only valid argument is the segment name. The segment that is erased is echoed to the terminal.

2.3.6 Updating Flash Memory

The U (Update) command initiates programming of the flash memory using data received via the USART. The update function expects to receive flash data in the MSP430-TXT format produced by the IAR linker. Instructions on preparing an application for use with the monitor program are provided in Section 3.

While executing the update function, the monitor responds to only two commands, which are described in Table 2. Note that these commands are a part of the MSP430-TXT file and are automatically generated by the IAR linker.

Table 2. Flash Update Commands

COMMAND	DESCRIPTION	OPERATION
@	Receive address	Sets the destination address in flash
q	Quit	Terminate flash programming

2.3.6.1 Setting Flash Address

Upon recognizing the U command, the monitor enters the flash programming function and waits to receive a flash address, delimited by the @ character.

If the next character received is not @, the monitor returns to the main loop and issues a prompt. If an @ is received, the monitor scans the input for a 16-bit hexadecimal flash address. If an invalid character (anything other than 0–9 or A–F) is received as one of the next four characters, the flash programming function terminates, and the monitor returns to the main loop.

If a valid hexadecimal number is received, it is checked against the bounds of flash memory available for the application, which ranges from the top of segment 2 (0xFBFF) to the beginning of segment n. The boundaries are determined based upon the processor variant specified in the TargetDefs.h header file. If the received address is outside the range of available flash, the flash programming terminates, and the monitor returns to the main loop.

2.3.6.2 Receiving Flash Data

Once a valid address has been received, the monitor scans the input from the USART for hexadecimal-encoded byte data. Two valid hexadecimal characters (0–9, A–F) must be received consecutively to be recognized as a valid byte. Any number of non-hexadecimal characters other than @ or q may be received between bytes and are echoed to the terminal and ignored. Byte data can also be sent with no delimiting characters.

As each byte is received, it is stored in a buffer in RAM. When the monitor detects a linefeed character, the contents of the buffer memory are copied to flash.

The monitor tests each byte immediately before writing a new value. If the byte is not blank, the monitor issues a NB error message and terminate programming. The remaining file contents are ignored.

The monitor also checks each byte after it is written. If the value read back does not match the value written, the monitor issues an ERR error message and terminate programming. The remaining file contents are ignored.

If an @ is received, the monitor expects a new flash address. If a valid address is received following the @, the monitor sets the new destination address and continues writing received values to flash. If a valid address is not received, the flash programming terminates, and the monitor returns to the main loop.

2.3.6.3 Terminating Flash Update

If the monitor receives a q (0x61) at any time during the flash programming procedure, the programming is terminated, and the monitor returns to the main loop.

2.3.6.4 Loading an Application Using HyperTerminal

The following procedure can be used to load an application into flash using the HyperTerminal program that is supplied with most versions of the Windows® operating systems.

1. Start HyperTerminal.
2. From the HyperTerminal menu, select File→Properties. In the pop-up dialog box, select the correct serial port in the Connect Using dropdown list, then click the Configure button.
3. In the pop-up dialog box, configure the serial port for 9600 bits/second, eight data bits, no parity, one stop bit, and Xon/Xoff flow control. Click OK.
4. Ensure that the PC serial port is connected to the MSP430 USART with the appropriate cable and line circuits. Power up or reset the MSP430 microcontroller. If the flash monitor is loaded and the terminal is correctly connected and configured, a ? prompt appears on the terminal screen.
5. Ensure that the text transfer settings match what is expected by the monitor program. To do that, click on ASCII Setup on the Settings dialog page. Check the box labeled Send line ends with line feeds, and also specify a delay of 50 milliseconds in the Line Delay input field.
6. Make sure that the memory ranges used by the application to be loaded have been erased. If necessary, erase the flash.
7. At a ? prompt, enter U. Then from the HyperTerminal menu, select Transfer→Send Text File.
8. In the pop-up dialog box, select the MSP430-TXT formatted file to be loaded and click OK.

The content of the file is echoed back to the terminal as it is received. When the flash update is complete, the monitor returns to the main loop and issues a prompt.

3 Building the Monitor and Applications

The monitor was designed to have minimum impact on the application to be loaded. The monitor does not reserve any processor resources other than two flash segments.

There are several options for building the monitor and applications:

- The monitor can be built for any MSP430F1xx or MSP430F4xx family variant with at least one USART.
- For target devices with two USARTs, either USART0 or USART1 may be used.
- The monitor can occupy flash segments 0 and 1, or it can be placed in the first two 512-byte segments in flash.
- The application and monitor can be built separately or simultaneously.

3.1 Monitor Options

The source files required to build the monitor are described in Table 3.

Table 3. Monitor Source Files

FILE NAME	DESCRIPTION
flash_monitor.s43	Assembly language source code for the monitor program
TargetDefs.h	Header file used to set the build options for building the monitor for a particular target board. It is used to select: • MSP430 target device • Serial port used for communication with the host • Whether or not memory contents can be dumped to the terminal • Whether the monitor or application starts first • Whether or not a password is required to erase user flash
flash_monitor.h	Contains macros to calculate memory boundaries for the target device selected, and for selection of the serial port

The monitor build options are controlled by #define statements in the header file TargetDefs.h (see Figure 2). The following paragraphs discuss the control of the options.

```
// TargetDefs.h for setting monitor build options

#define CHIP          F169  // Target is MSP430F169
#define MEMDUMP       1     // Allow memory dump
#define SERIAL_PORT_1       // Use USART1
#define ECHO          1     // Echo all USART input to terminal
#define DELAY_START   1     // If defined, the monitor runs first at reset
#define PASSWORD      1     // Enable password protection
#define DIRECT_INTERRUPTS   // if defined, the monitor is placed in low flash
                            // if undefined, the monitor is placed in high flash
```

Figure 2. Sample TargetDefs.h

The major options are the target device, the USART to be used, and where the monitor is to be placed in memory. The minor options include whether or not the monitor is able to display MSP430 memory, whether the monitor or the application runs first at startup, whether or not the flash erase is password protected, whether or not terminal input is echoed on the USART, and whether or not a startup banner is displayed.

3.1.1 Major Options

3.1.1.1 Selecting the Target Device

The target device is selected by the `#define CHIP` statement in TargetDefs.h. CHIP should be defined as the generic part number suffix as defined in TargetDefs.h. For example, to build the monitor to run on the MSP430F169, the chip definition line should be `#define CHIP F169`.

3.1.1.2 Selecting the USART

The USART that is used is selected by the `#define SERIAL_PORT_x` statement. Defining SERIAL_PORT_0 selects USART0 and defining SERIAL_PORT_1 selects USART1.

3.1.1.3 Memory Placement

The monitor uses two 512-byte segments of the MSP430's main flash. These can be either segments 0 and 1, which are at 0xFFC0–0xFFFF in all MSP430 devices, or at the lowest two 512-byte segments, typically segments n and n – 1, where n depends upon the total flash memory size.

Each approach has advantages and disadvantages. When the monitor is located in high memory, the interrupt vector table is in a common segment with the monitor program. Because the flash can only be erased in segments, this means the interrupt vector table must contain fixed values. To accommodate this, an extra level of indirection is required to respond to an interrupt. The interrupt vectors point to a table of indirect jumps located at the bottom of segment 1. Instructions in this table branch to the actual interrupt service routines (ISRs) by loading the PC from a secondary interrupt vector table located at the end of segment 2 (0xFBE0–0xFBFF). This adds an extra three CPU clock cycles to the interrupt response time. The advantage of this approach is that the monitor cannot erase the interrupt vector table, because the primary vector table is collocated with the monitor in flash segment 0. The monitor cannot erase its own program segments, so it cannot be rendered inaccessible by loading a faulty application, assuming that the application itself does not erase the flash memory. With this approach, the monitor should always be accessible by way of a reset.

When the monitor is located in low memory, the interrupt vector table at 0xFFE0–0xFFFF points directly to the application's ISRs. This eliminates the additional three-cycle interrupt response time, but it requires that the interrupt vector table be reprogrammed each time a new application is loaded. If an application is loaded with an invalid vector table, or if the application cannot transfer control to the monitor program, the only way to reprogram the processor may be via the ROM bootloader or via the JTAG port.

To place the monitor in low memory, the TargetDefs.h should contain the line `#define DIRECT_INTERRUPTS`. If this line is deleted or commented out, the monitor is placed in segments 0 and 1. This also sets the bounds of the flash available to the application.

3.1.2 Minor Options

3.1.2.1 Enabling Display of MSP430 Memory

If MEMDUMP is non-zero, the user is able to display MSP430 memory contents. If MEMDUMP is undefined or defined as zero, the memory display is disabled.

3.1.2.2 Terminal Echo

If ECHO is defined as non-zero, the monitor transmits back each character received on the USART. To disable this feature, leave ECHO undefined or defined as zero.

3.1.2.3 Password Protection for Flash Erase

The password-protection scheme provides some protection against unintentionally erasing the flash. It does not provide flash security, nor is it intended to be hacker proof. It does, however, make it less likely that an operator will inadvertently erase the flash memory.

When password protection is enabled, the monitor prompt for a 16-bit password when the E (erase flash) command is entered. The received password is checked against a 16-bit value stored immediately before the application's interrupt vector table. If the entered value does not match the stored value, the monitor returns to the command prompt.

3.2 Building Applications

Applications that are to be run or loaded by the monitor can be written either in assembly language or C. The monitor is designed to have minimum impact on the application. The only consideration that needs to be made in coding is whether the monitor is to be entered from the application. If the monitor does not run at reset, the application must be able to start the monitor. This can be done by branching to the monitor's reset vector. This is defined as a public symbol (Reset), so that its absolute address can be found in the linker map file generated when the monitor is built.

The main requirements when building an application that is loaded by the flash monitor are:

- Do not use (or erase) the flash segments occupied by the monitor.
- Generate an output file in the MSP430-TXT format.
- Provide a way to enter the monitor from the application, if the monitor does not run first at reset.

3.2.1 Applications Written in C

For C applications, the required modifications can be made by modifying the linker command file to change the available memory limits and the placement of the interrupt vector table.

For the IAR linker, the changes are made in the section of the linker command file following this comment:

```
// -------------------------------------------------------------------
// ROM memory (FLASH)
// -------------------------------------------------------------------
```

Range of flash memory must be changed to reserve the two main flash segments occupied by the monitor and, if the monitor is to reside in high memory, the application's interrupt vector program must be placed at the end of the user flash. Figure 3 shows the IAR linker command file for the MSP430F169, msp430f169.xcl, modified to build an application for use with the flash monitor located in segments 0 and 1.

```
//  Code

-Z(CODE)CSTART=1100-FBDF
-Z(CODE)CODE=1100-FBDF


// Constant data

-Z(CONST)DATA16_C,DATA16_ID,DIFUNCT,CHECKSUM=1100-FBDF


// Interrupt vectors

-Z(CONST)INTVEC=FBE0-FBFF
-Z(CONST)RESET=FBFE-FBFF
```

Figure 3. Example Linker Command File

The –Z(CODE) statements define memory segments that the linker will use for placing code. For the MSP430F169, the available memory range is from 0x1100–0xFBDF.

The –Z(CONST) statement defines memory segments that the linker will use for placing data.

The 32-byte INTVEC segment is used for the interrupt vectors, and the 2-byte RESET segment is used for the reset vector.

After modifying the linker command file, the program is built with memory usage that is compatible with the monitor program.

Be sure not to overwrite the IAR-provided linker command file with the modified linker command file.

It is also necessary to change the default project options to use the modified linker command file and to generate the MSP430-TXT formatted output file. If the generate extra output option is selected, the linker generates both a binary executable file that the debugger can load into flash via the JTAG emulator and the text formatted file that the monitor can use.

- In the Linker category, select the Extra Output tab. Check the Generate Extra Output File box and the Override Default box. Enter the desired output file name. Select MSP430-TXT from the Output Format dropdown list.
- Select the Config tab. In the Linker Command File area, check the Override Default box and enter the pathname for the modified linker command file.

3.2.2 Applications Written in Assembly Language

When building assembly language applications, it is common practice to define the interrupt vectors using the assembler ORG and DW directives. When building applications that are compatible with the monitor, it is only necessary to place the interrupt vectors at the top of flash segment 2 (0xFBE0–0xFBFF), rather than at the top of flash segment 0.

4 Building the Monitor Program

The monitor program and application can be built separately or together. For initial flash programming, it is usually convenient to build the monitor and the application as a single binary file, which can be stored in the flash using the JTAG emulator. Then the monitor can be used to erase the application flash and load updates via the USART.

In order to build the application and monitor as a single program, it is only necessary to add the monitor source file (flash_monitor.s43) to the project and make sure that the monitor header files (TargetDefs.h and flash_monitor.h) are in the include path.

If the monitor program is built and stored in the flash as a stand-alone program, it will always run at reset.

5 Demonstration Application

A demonstration program is included in the source files for the monitor program. This program is based on the FET demonstration program FET140_UART15_9600.c. As provided, it is targeted for the MSP430F169 and runs as is on the SoftBaugh DIr169 evaluation board.[4]

The demonstration program initializes the USART and Timer_A and enters LPM3 until an interrupt occurs.

Timer_A is configured to generate interrupts at a 0.5-Hz rate. The Timer_A ISR toggles Port1 bit 0. On most of the TI MSP-FET target boards and on the the SoftBaugh DIr169 board, this turns an LED on or off.

The USART ISR copies each received character from the receive buffer to the transmit buffer. It also checks each received character to determine if it is a ^C character (0x03 or ETX). When an ETX character is received, the application transfers control to the monitor via an inline assembly statement that indirectly loads the PC from the WarmStart address, in this case 0xFC40.

The files required to build the demonstration application are described in Table 4.

Table 4. Demonstration Application

FILE NAME	DESCRIPTION
monitor_demo_app.c	C source file modified based on fet140_uart15_9600.c
lnk430f169_boot.xcl	IAR MSP430F169 linker command file modified as described in Section 3.2.1 for compatibility with the resident monitor

6 References

1. *MSP430x1xx Family User's Guide* (TI literature number SLAU049)
2. *MSP430 IAR Assembler Reference Guide* (IAR EW Help Menu)
3. *IAR Linker and Library Tools Reference Guide* (IAR EW Help Menu)
4. SoftBaugh, Inc. DIr169 demonstration board schematic (http://www.softbaugh.com/ProductPage.cfm?strPartNo=DIr169)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

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

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI 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 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. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

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

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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