MSP430™ Programming Via the JTAG Interface

User's Guide

TEXAS INSTRUMENTS

Literature Number: SLAU320C
July 2010–Revised October 2011

Contents

1.2.2 JTAG Access Macros

To keep descriptions of the JTAG functions in the following sections simple, high-level macros have been used to describe the JTAG access. This document does not detail the basic JTAG functionality; rather, it focuses on the MSP430-specific implementation used for memory access and programming. For the purpose of this document, it is important to show the instructions that must be loaded into the JTAG instruction register, as well as when these instructions are required. Section 1.2.2.1 summarizes the macros used throughout this document and their associated functionality. See the accompanying software for more information.

Table 1-5. JTAG Communication Macros

Macro Name	Function
IR_SHIFT (8-bit Instruction)	Shifts an 8-bit JTAG instruction into the JTAG instruction register. At the same time, the 8-bit value is shifted out through TDO.
DR_SHIFT16 (16-bit Data)	Shifts a 16-bit data word into a JTAG data register. At the same time, the 16-bit value is shifted out through TDO.
DR_SHIFT20 (20-bit Address)	Shifts a 20-bit address word into the JTAG Memory Address Bus register. At the same time, the 20-bit value is shifted out through TDO. Only applicable to MSP430X architecture devices.
MsDelay (time)	Waits for the specified time in milliseconds
SetTCLK	Sets TCLK to 1
ClrTCLK	Sets TCLK to 0
TDOvalue	Variable containing the last value shifted out on TDO

1.2.2.1 Macros for 4-Wire JTAG Interface

1.2.2.1.1 IR_SHIFT (8-bit Instruction)

This macro loads a desired JTAG instruction into the JTAG instruction register (IR) of the target device. In the MSP430, this register is eight bits wide with the least significant bit (LSB) shifted in first. The data output from TDO during a write to the JTAG instruction register contains the version identifier of the JTAG interface (or JTAG ID) implemented on the target device. Regardless of the 8-bit instruction sent out on TDI, the return value on TDO is always the JTAG ID. Each instruction bit is captured from TDI by the target MSP430 on the rising edge of TCK. TCLK should not change state while this macro is executed (TCLK = TDI while the TAP controller is in the Run-Test/Idle state). Figure 1-2 shows how to load the ADDR_16BIT instruction into the JTAG IR register. See Section 1.2.4 for a complete list of the JTAG interface communication instructions used to access the target device flash memory module.

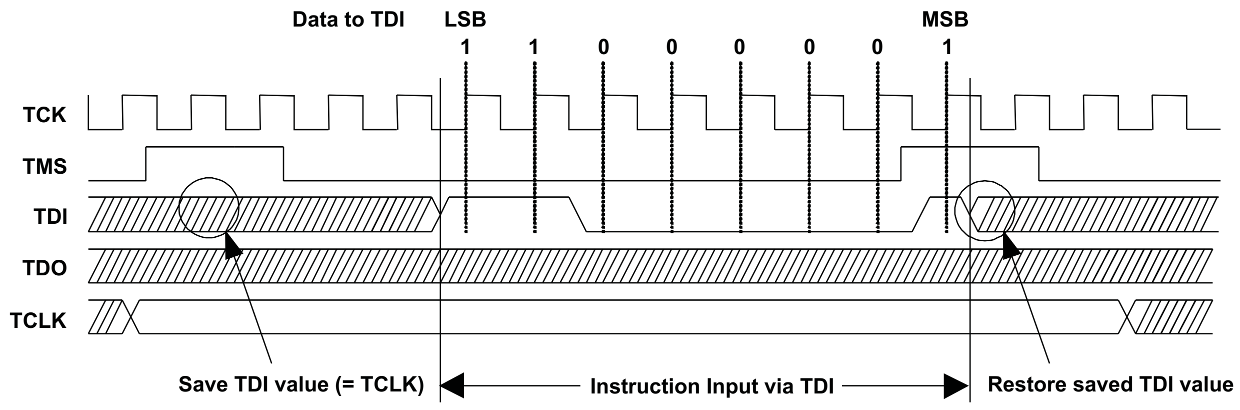

Figure 1-2. Timing Example for IR_SHIFT (0x83) Instruction

1.2.2.1.2 DR_SHIFT16 (16-bit Data)

This macro loads a 16-bit word into the JTAG data register (DR) (in the MSP430 devices, a data register

1.2.4 JTAG Communication Instructions

Selecting a JTAG register and controlling the CPU is done by shifting in a JTAG instruction using the IR_SHIFT macro described in Section 1.2.2.1.1. The following instructions that can be written to the JTAG IR are used to program the target flash memory. All instructions sent to the target MSP430 via the JTAG register are transferred LSB first.

Table 1-6. Memory Access Instructions

Instruction Name	8-Bit Instruction Value (Hex)
Controlling the Memory Address Bus (MAB)	
IR_ADDR_16BIT	0x83
IR_ADDR_CAPTURE	0x84
Controlling the Memory Data Bus (MDB)	
IR_DATA_TO_ADDR	0x85
IR_DATA_16BIT	0x41
IR_DATA_QUICK	0x43
IR_BYPASS	0xFF
Controlling the CPU	
IR_CNTRL_SIG_16BIT	0x13
IR_CNTRL_SIG_CAPTURE	0x14
IR_CNTRL_SIG_RELEASE	0x15
Memory Verification Via Pseudo Signature Analysis (PSA)	
IR_DATA_PSA	0x44
IR_SHIFT_OUT_PSA	0x46
JTAG Access Security Fuse Programming	
IR_Prepare_Blow	0x22
IR_Ex_Blow	0x24
JTAG Mailbox System	
IR_JMB_EXCHANGE	0x61

NOTE: Do not write any unlisted values to the JTAG instruction register. Instruction values written to the MSP430 JTAG register other than those listed above may cause undesired device behavior.

NOTE: When a new JTAG instruction is shifted into the JTAG instruction register, it takes effect with the UPDATE-IR state of the TAP controller. When accessing a JTAG data register, the last value written is captured with the CAPTURE-DR state, and the new value shifted in becomes valid with the UPDATE-DR state. In other words, there is no need to go through Run-Test/Idle state of the JTAG TAP controller to shift in instructions or data. Be aware of the fact that clocking TCLK is only possible in the Run-Test/Idle state. This is why the provided software example application exclusively makes use of the JTAG macros described in Section 1.2.2, which always go through Run-Test/Idle state.

1.3.3 Accessing Non-Flash Memory Locations With JTAG

1.3.3.1 Read Access

To read from any memory address location (peripherals, RAM, or flash), the R/W signal must be set to READ using the JTAG control signal register (bit 0 set to 1). The MSP430 MAB must be set to the specific address to be read using the IR_ADDR_16BIT instruction while TCLK is 0. To capture the corresponding value of the MSP430 MDB, the IR_DATA_TO_ADDR instruction must be executed. After the next rising edge of TCLK, the data of this address is present on the MDB. The MDB can now be captured and read out via the TDO pin using a 16-bit JTAG data access. When TCLK is set low again, the address of the next memory location to be read can be applied to the target MAB. Following is the flow required to read data from any memory address of a target device. Implementations for both the MSP430 and MSP430X architectures are shown.

- MSP430 architecture, Reference function: ReadMem

Set CPU to stopped state (HaltCPU)	
`ClrTCLK`	
`IR_SHIFT("IR_CNTRL_SIG_16BIT")`	
`DR_SHIFT16(0x2409)         : Read Memory`	
`IR_SHIFT("IR_ADDR_16BIT")`	**Yes**
`DR_SHIFT16("Address")      : Set desired address`	
`IR_SHIFT("IR_DATA_TO_ADDR")`	
`SetTCLK`	
`ClrTCLK`	
`DR_SHIFT16(0x0000)         : Memory value shifted out on TDO`	
Read again?	
No	
ReleaseCPU should now be executed, returning the CPU to normal operation.	

- MSP430X architecture, Reference function: ReadMem_430X

Set CPU to stopped state (HaltCPU)	
`ClrTCLK`	
`IR_SHIFT("IR_CNTRL_SIG_16BIT")`	
`DR_SHIFT16(0x2409)         : Read Memory`	
`IR_SHIFT("IR_ADDR_16BIT")`	**Yes**
`DR_SHIFT20("Address")      : Set desired address`	
`IR_SHIFT("IR_DATA_TO_ADDR")`	
`SetTCLK`	
`ClrTCLK`	
`DR_SHIFT16(0x0000)         : Memory value shifted out on TDO`	
Read again?	
No	
ReleaseCPU should now be executed, returning the CPU to normal operation.	

- MSP430Xv2 architecture, Reference function: ReadMem_430Xv2

CPU must be in the Full-Emulation-State prior to the following sequence.	
`ClrTCLK`	**Yes**
`IR_SHIFT("IR_CNTRL_SIG_16BIT")`	
`DR_SHIFT16(0x2409)          : Read Memory`	
`IR_SHIFT("IR_ADDR_16BIT")`	
`DR_SHIFT20("Address")      : Set desired address`	
`IR_SHIFT("IR_DATA_TO_ADDR")`	
`SetTCLK`	
`ClrTCLK`	
`DR_SHIFT16(0x0000)          : Memory value shifted out on TDO`	
`SetTCLK`	
`ClrTCLK`	
`SetTCLK`	
Read again?	
No	
CPU is now again in Full-Emulation-State.	

1.3.3.2 Write Access

To write to a memory location in peripherals or RAM (but not flash), the R/W signal must be set to WRITE using the JTAG control signal register (bit 0 set to 0). The MAB must be set to the specific address using the IR_ADDR_16BIT instruction while TCLK is low. The MDB must be set to the data value to be written using the IR_DATA_TO_ADDR instruction and a 16-bit JTAG data input shift. On the next rising edge of TCLK, this data is written to the selected address set by the value on the MAB. When TCLK is asserted low, the next address and data to be written can be applied to the MAB and MDB. After completion of the write operation, it is recommended to set the R/W signal back to READ. Following is the flow for a peripheral or RAM memory address write. Implementations for both the MSP430 and MSP430X architectures are shown.

- MSP430 architecture, Reference function: WriteMem

Set CPU to stopped state (HaltCPU)	
`ClrTCLK`	
`IR_SHIFT("IR_CNTRL_SIG_16BIT")`	
`DR_SHIFT16(0x2408)          : Write Memory`	
`IR_SHIFT("IR_ADDR_16BIT")`	**Yes**
`DR_SHIFT16("Address")      : Set desired address`	
`IR_SHIFT("IR_DATA_TO_ADDR")`	
`DR_SHIFT16("Data")          : Send 16-bit Data`	
`SetTCLK`	
Write again?	
No	
ReleaseCPU should now be executed, returning the CPU to normal operation.	

- Released from JTAG control

Target device is released from JTAG control (free running)		
`IR_SHIFT("IR_JB_EXCHANGE")`		
`DR_SHIFT16(0x0001)`	`: Send input request to JTAG mailbox`	**Yes**
`DR_SHIFT16("Data")`	`: Shift 16 bit word into JTAG mailbox`	
`DR_SHIFT16(0x0000)`	**No**	
Bit 0 of TDOword = 1 ?		
Yes		
Write Another Flash Address?		
No		
Get target device in Full-Emulation-State		

1.3.5 Erasing the Flash Memory (Using the Onboard Flash Controller)

1.3.5.1 Function Reference for 1xx/2xx/4xx Families

Reference function: EraseFLASH

This section describes how to erase one segment of flash memory (ERASE_SGMT), how to erase the device main memory (ERASE_MAIN), and how to perform an erase of the complete flash memory address range including, main and info flash segments (ERASE_MASS). This method requires the user to provide a TCLK signal at a frequency of 350 kHz ± 100 kHz while the erase cycle is being executed, as is also the case when programming the flash memory. The following tables show the segment and mass erase flows, respectively, and the minimum number of TCLK cycles required by the flash controller to perform each action (FCTL2 register bits 0 to 7 = 0x40).

1.3.5.1.1 Flow to Erase a Flash Memory Segment

Switch CPU to stopped state (HaltCPU)	
`ClrTCLK`	
`IR_SHIFT("IR_CNTRL_SIG_16BIT")`	
`DR_SHIFT16(0x2408)`	`: Set RW to Write`
`IR_SHIFT("IR_ADDR_16BIT")`	
`DR_SHIFT16(0x0128)` [1]	`: Point to FCTL1 Address`
`IR_SHIFT("IR_DATA_TO_ADDR")`	
`DR_SHIFT16(0xA502)`	`: Enable FLASH segment erase`
`SetTCLK`	
`ClrTCLK`	
`IR_SHIFT("IR_ADDR_16BIT")`	
`DR_SHIFT16(0x012A)` [1]	`: Point to FCTL2 Address`
`IR_SHIFT("IR_DATA_TO_ADDR")`	
`DR_SHIFT16(0xA540)`	`: Source is MCLK, divider by 1`
`SetTCLK`	
`ClrTCLK`	
`IR_SHIFT("IR_ADDR_16BIT")`	
`DR_SHIFT16(0x012C)` [1]	`: Point to FCTL3 Address`
`IR_SHIFT("IR_DATA_TO_ADDR")`	
`DR_SHIFT16(0xA500)` [2]	`: Clear FCTL3 Register`

[1] Replace with DR_SHIFT20("Address") when programming an MSP430X architecture device.
[2] Substitute 0xA540 for '2xx devices for Info-Segment A programming.

SetTCLK		
ClrTCLK		
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(“EraseAddr”) [1]	: Set Address for Erase [3]	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0x55AA)	: Write Dummy Data for Erase Start	
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x2409)	: Set RW to Read	
SetTCLK		*Repeat 4819 times* [4]
ClrTCLK		
IR_SHIFT("IR_CNTRL_SIG_16BIT")		
DR_SHIFT16(0x2408)	: Set RW to Write	
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(0x0128) [5]	: Point to FCTL1 Address	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA500)	: Disable FLASH Erase	
SetTCLK		
ClrTCLK		
IR_SHIFT("IR_ADDR_16BIT")		
DR_SHIFT16(0x012C) [5]	: Point to FCTL3 Address	
IR_SHIFT("IR_DATA_TO_ADDR")		
DR_SHIFT16(0xA500) [6]	: Disable FLASH Write Access	
SetTCLK		
ReleaseCPU should now be executed, returning the CPU to normal operation.		

[3] The EraseAddr parameter is the address pointing to the flash memory segment to be erased.
[4] Correct timing required. Must meet min/max TCLK frequency requirement of 350 kHz ±100 kHz.
[5] Replace with DR_SHIFT20("Address") when programming an MSP430X architecture device.
[6] Substitute 0xA540 for '2xx devices for Info-Segment A programming.