京微齐力
Hercules Microelectronics

HME-M5 Family JTAG Programming User Guide

Oct 2018

Hercules microelectronics Co., Ltd

Revision History

The table below shows the revision history for this document.

Date	Version	Revision
July. 2014	CME-M5JTAGPUGE01	Initial release.
Oct. 2018	HME-M5JTAGPUGE02	Update logo

Contents

1 JTAG Schematic

The JTAG schematic is shown below.

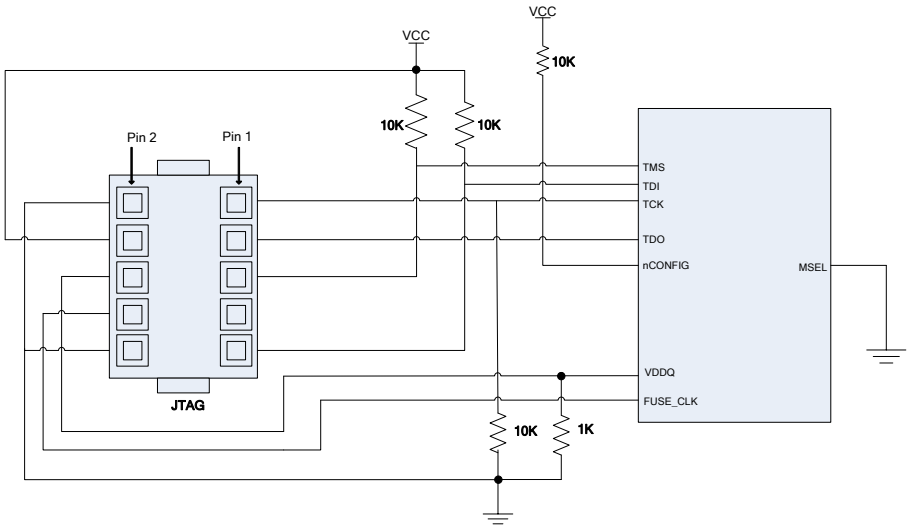


Figure 1 JTAG schematic

Table 1 JTAG interface definition

Name	Description	Type(1)	Width
TCK	JTAG clock	I	1
TMS	JTAG mode select	I	1
TDI	JTAG data input	I	1
TDO	JTAT data output	O	1
VDDQ	Efuse Program voltage	O	1
FUSE_CLK	Efuse Program clock	O	1

2 FP JTAG and MCU JTAG Cascading

There are two JTAG devices inside HME-M5, one is for fabric debugging and configuration, another is for OCDS of MCU, in order to make it convenient and to decrease cost, these two JTAG devices are cascaded into one according to the IEEE standard.

As OCDS can access all the memory space of MCU, it is essential to provide a mechanism to protect memory space of MCU, to achieve this goal, a virtual JTAG TAP controller is designed, a MUX controlled by the security bit (prot_flagn) is added to prevent data flow into MCU OCDS when the security bit is LOW active, and another MUX controlled by the security bit is added to select OCDS's TDO pin when the security bit is HIGH inactive.

With the above described mechanism, once the security bit is active, and then TDI pin of MCU OCDS is tied to HIGH so that MCU OCDS can only receive BYPASS instruction, and meanwhile only the TDO pin of virtual JTAG is MUXed outside. So when the security bit is LOW active, OCDS is blocked to prevent accessing by outside JTAG host.

When the security bit is inactive, the TDI pin of MCU OCDS is MUXed to TDO pin of FP JTAG, and the TDO pin of MCU OCDS is MUXed outside, so MCU OCDS can be accessed.

Seeing from outside JTAG interface, there are always two JTAG devices cascaded together regardless of the status of the security bits. With method, the difficulty of verification and outside software design will be reduced with the security capability guaranteed.

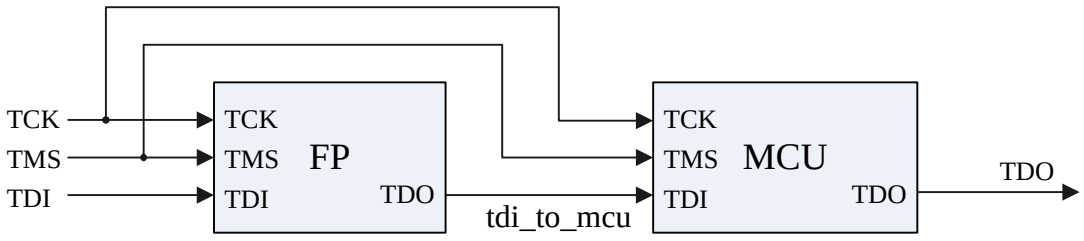


Figure 2 JTAG Cascading

3 Bitstream Format

Five instructions are used for JTAG operation, which are SEND_HEADER, SEND_DATA, SEND_TAIL, RD_HDR and RD_DATA.

Each of these instructions is composed of basic instructions: LOAD_IR, LOAD_DR and/or READ_DR.

3.1 SEND_HEADER (32'h[header])

$$SEND\_HEADER\begin{cases}\text{LOAD\_IR}(4'b0110)\\ \text{LOAD\_DR}(32'h[header])\\ \text{LOAD\_IR}(4'b0100)\\ \text{LOAD\_DR}(8'h00)\end{cases}$$

3.2 SEND_DATA (32'h[data])

$$SEND\_DATA\begin{cases}LOAD\_IR(4'b0110)\\ LOAD\_DR(32'h[data])\\ LOAD\_IR(4'b0100)\\ LOAD\_DR(8'h01)\end{cases}$$

3.3 SEND_TAIL (32'h[tail])

$$SEND\_TAIL\begin{cases}LOAD\_IR(4'b0110)\\ LOAD\_DR(32'h[tail])\\ LOAD\_IR(4'b0100)\\ LOAD\_DR(8'h02)\end{cases}$$

3.4 RD_HDR (32'h[read_header])

$$RD\_HDR\begin{cases}LOAD\_IR(4'b0110)\\ LOAD\_DR(32'h[read\_header])\\ LOAD\_IR(4'b0100)\\ LOAD\_DR(8'h03)\end{cases}$$

3.5 RD_DATA (32'h[read_data])

$$RD\_DATA\begin{cases} LOAD\_IR(4'b0100) \\ LOAD\_DR(8'h04) \\ LOAD\_IR(4'b0110) \\ READ\_DR(32'h[read\_data]) \end{cases}$$

4 Waveform for Each Basic Instruction

In this section, basic instructions, which include GOTO_IDLE, LOAD_IR and LOAD_DR, are illustrated.

Note: TCK shall be stopped at either '0' or '1' when JTAG is in IDLE state, and meanwhile, TMS and TDI shall be pulled up to stay HIGH.

4.1 GOTO_IDLE

In the beginning this instruction shall be executed, see figure below.

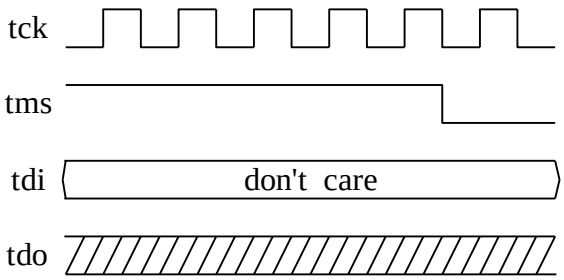


Figure 3 Waveform for instruction GOTO_IDLE

4.2 LOAD_IR

LOAD_IR is used to transmit 4-bit instruction into the chip.

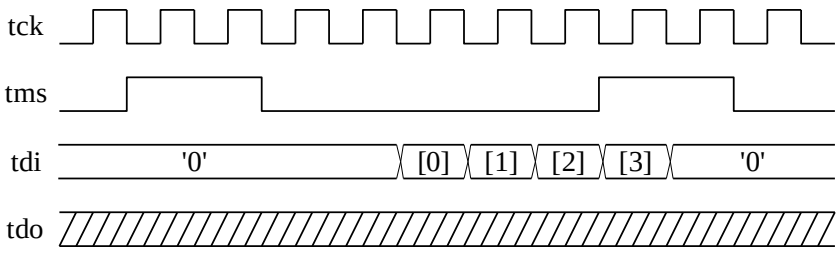


Figure 4 Waveform for the LOAD_IR (4 bit) instruction

4.3 LOAD_DR (8-bit)

LOAD_DR is used to transmit 8/32-bit data into the chip (see figure below). Different data width is differentiated by the argument of LOAD_IR executed last time (Please refer to section 1).

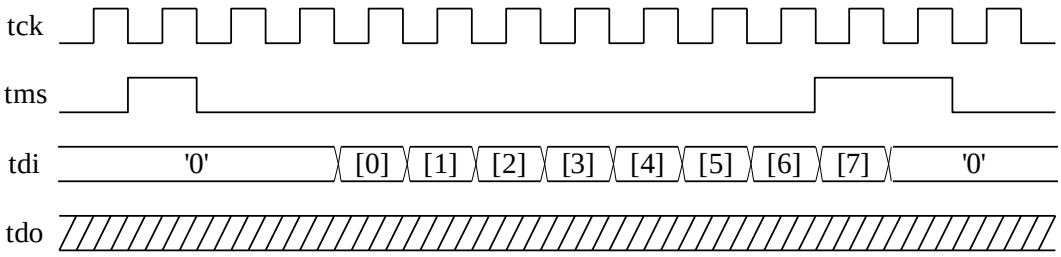


Figure 5 Waveform for the LOAD_DR (8 bit) instruction

4.4 LOAD_DR (32-bit)

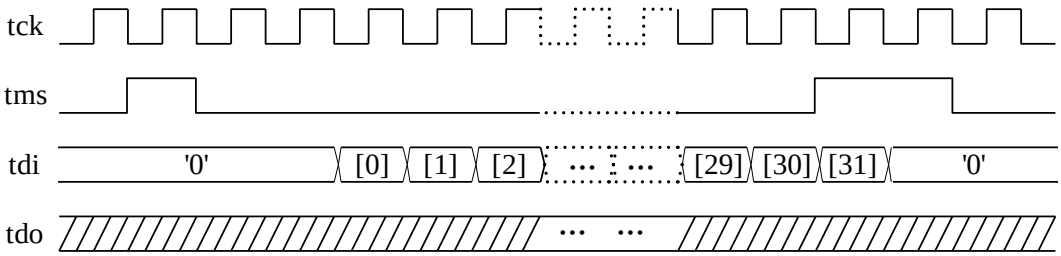


Figure 6 Waveform for the LOAD_DR (32 bit) instruction

4.5 READ_DR (32-bit)

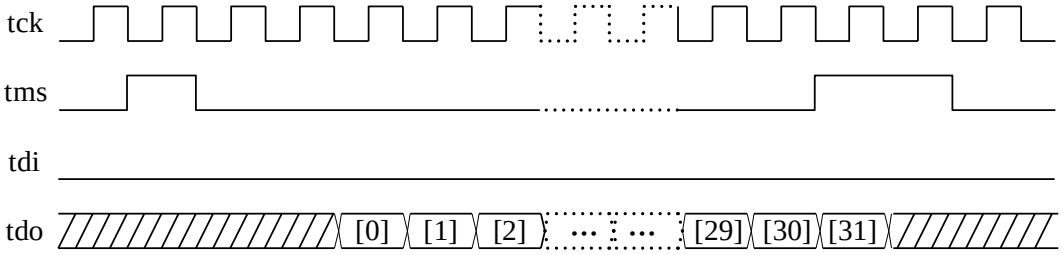


Figure 7 Waveform for the READ_DR (32 bit) instruction

5 TAP FSM

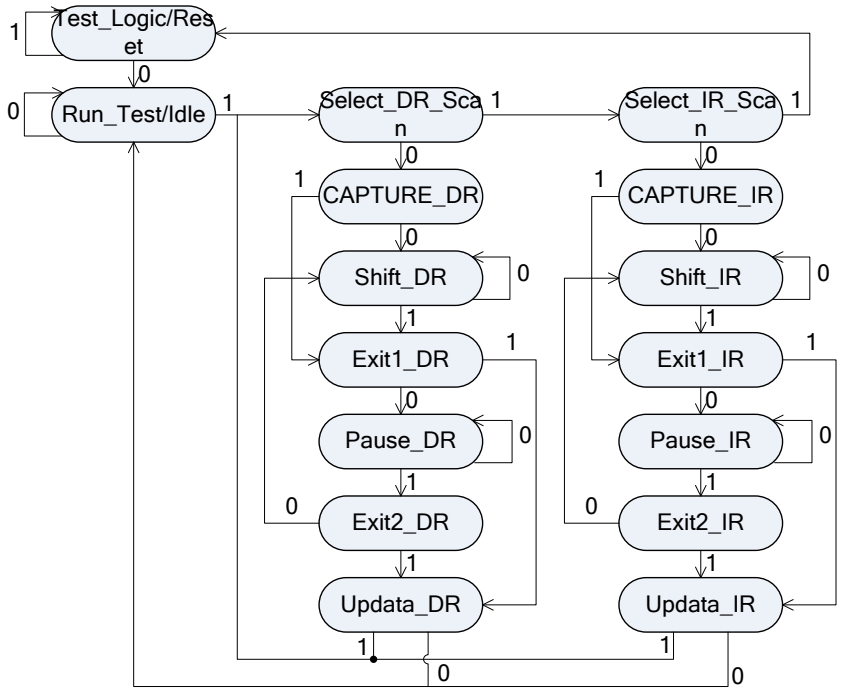


Figure 8 FSM of TAP controller

Table 2 TAP_FSM State's Function

State Name	State Function
Test Logic Reset	The Test Logic Reset controller state disables the test logic and reset the instruction TAP ports. The instruction that selects the device ID registers. If the device ID register is not present, Test Logic Reset selects the bypass register. When the TMS signal keeps high for at least 5 TCK clocks, the TAP controller transitions to the Test Logic Reset controller state.
Run-Test/Idle	The Run-Test/Idle controller state retains the last state of the Test Logic. During this state, the TAP controller can execute an internal test such as MBIST, previously selected by the instruction register.
Select DR-Scan Select IR-Scan	The Select DR Scan and Select IR Scan are temporary controller states, which keeps the last state of the Test Logic. During this state, if the TMS signal is low, the TAP controller initiates the scan sequence for either the selected test data or the instruction register.
CAPTURE_DR Capture_IR	The CAPTURE_DR and Capture_IR controller states parallel load data into either a selected test data register or the instruction register on the rising edge of the TCK signal.
Shift_DR Shift_IR	The Shift_DR and Shift_IR controller states shift either the selected test data register or the instruction register on stage towards its serial output.

Exit1_DR Exit1_IR	The Exit1_DR and Exit1_IR temporary controller states, where all the test data and instruction registers keep their previous state. During this state, if the TMS signal is high the TAP controller terminates the scanning process. If the TMS signal is low, the TAP controller transitions the selected test data register or instruction register corresponding Pause controller state.
Pause_DR Pause_IR	The Pause_DR and Pause_IR controller states temporarily halt the shifting of either the selected test data register or the instruction register. The TAP controller remains paused until the TMS signal goes high.
Exit2_DR Exit2_IR	The Exit2_DR and Exit2_IR are temporarily controller states, where all the test data registers and instruction register retain their previous state. During this state, if the TMS goes high the TAP controller terminates the scanning process. If the TMS goes low, the TAP controller returns to either selected test data register or the instruction register to the corresponding Shift controller state.
Update_DR Update_IR	The Update_DR and Update_IR controller states transfer data from each shift register stage into the corresponding parallel output latch on the falling edge of the TCK signal.

As defined in IEEE standard, all operation shall be start from RUN_TEST_IDLE state. The TMS signal is captured on the rising edge of TCK, and the FSM state transition shall only occur on the rising edge of TCK.

All inputs of the TAP including TDI, TMS shall be captured on the rising edge of TCK, and the host shall generate these signals on the falling edge of TCK to avoid timing violation.
All output of TAP including TDO and other internal signals shall be registered on the falling edge of TCK, and the host or the next module shall capture these signals on the rising edge of TCK if belongs to the same clock domain.

6 Instructions

6.1 Instruction Register

As defined in IEEE 1149.1, IR is shift-register based and able to hold instruction data, and parallel input is an optional.

In the SHIFT_IR state, TDI will first input LSB of the expected instruction.
IR is composed of two sections, TAP_INSTR and INSTR_REG, as shown in below.

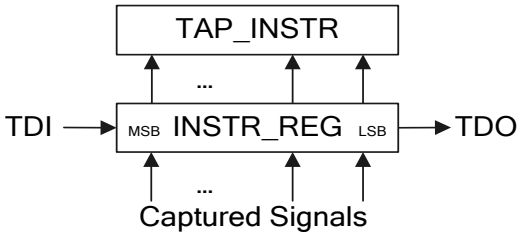


Figure 9 IR architecture

TAP_INSTR is initialed to be IDCODE, if IDCODE is not supplied, then TAP_INSTR is initialed to be BYPASS. It shall be updated to be the value of INSTR_REG when UPDATE_IR state.

INSTR_REG is selected to connect between TDI and TDO in SHIFT_IR state, it capture specified bits in CAPTURE_IR state, see table below.

Table 3 IR capture values

Bit Map	Captured Signal	Description
9	prot_flagn	Indicate the status of security protection, LOW active
8	read_disable	Chain reading via JTAG is disabled
7	crc_correct	Indicate if CRC of e-fuse input key is correct
6	key_cmp_flag	Compare result between input data and the reading result of e-fuse 1: correct 0: wrong
5	wfuse_done	Indicate e-fuse programming is done
4	rfuse_done	Indicate e-fuse 0/1 reading finished
3	isc_enabled	This two signals indicate current ISC state (see section 4.4.2.3)
2	isc_done	
1:0	{2'b01}	Specified by the IEEE standard

The two least significant bits of INSTR_REG shall capture “01”pattern for self-testing. Before using JTAG to test the whole board, the JTAG logic should be tested and be sure of fault free by adopting the following three steps.

Step 1: Apply the sequence to TMS, which causes each device to place the IR between TDI and TDO. At this stage, there is a serial shift register that starts at the board TDI input and ends at the board TDO output and which is made up of the various IRs in the devices — an IR chain.

Step 2: Apply an additional sequence to TMS to cause each IR to capture the hardwired 01 into the least two significant positions of the IR shift register. Higher-order bits capture what they are set up to capture. These values are not mandated by the Standard. The captured 01 values constitute a checkerboard “flush” test for the serial IR chain.

Step 3: Clock the captured values out of the IR chain to the board’s TDO output while clocking in the instruction code sequence 11110...0.
If sequence “100010001000” emerges on TDO, then we can reasonably sure of that TMS, TDI, TDO and TCK are all connect correctly, and that is capable of responding correctly on TMS to capture and shift IR.

Instructions supported in this design are listed in table below.

Table 4 instructions supported by M5 device

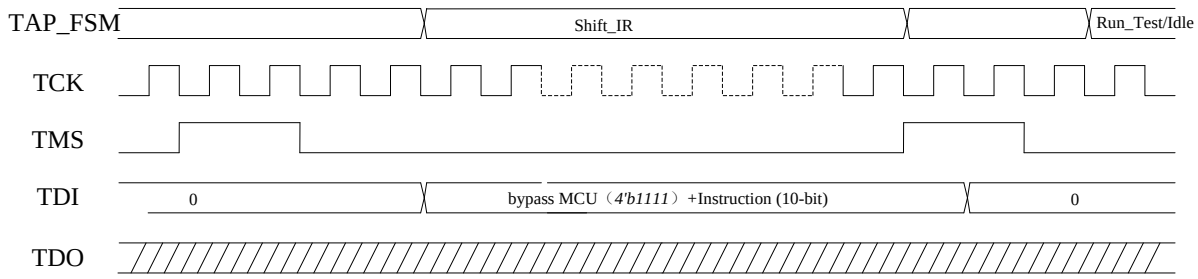
Instruction	CODE	Selected Register	Function
IDCODE	1111110000	DEVICE_ID_REG	Read the fixed Device_ID
USERCODE	1111110001	DEVICE_ID_REG	Read the user-programmable ID
SAMPLE	1111110010	BSR	Sample signals to/from on-chip logic into shift registers of BSR and shift out via TDO to verify
PRELOAD	1111110011	BSR	Sample signals to/from on-chip logic into shift registers of BSR and shift out via TDO to verify, and preload defined data into parallel output register of BSR
EXTEST	1111110100	BSR	Drive external pins with data stored in the parallel output register of BSR
TESTCTRL	1111110101	TSTCTRL_REG	Control the test control bits.
CLAMP	1111110110	BSR	Used as re-EXTEST and drive component pins from boundary-scan register
HIGH_Z	1111110111	BSR	Place all system logic outputs at high impedance to avoid risks to other chips.
ISC_ENABLE	1111111000	BYPASS_REG	Make the chip enter ISC_ACCESSED state for the forthcoming programming. All other ISC instruction shall be preceded by this instruction.

ISC_DISABLE	1111111001	BYPASS_REG	Make the chip leave ISC_ACCESSED state after programming done.
ISC_PROGRAM	1111111010	ISC_DATA	Program the chip cooperate with ISC_SETUP instruction.
ISC_NOOP	1111111011	BYPASS_REG	Select bypass register with no operation for multi-chip configuration.
ISC_SETUP	1111111100	ISC_CONFIG	Register control bits for ISC_PROGRAM and ISC_RDATA instructions
ISC_RDATA	1111111101	ISC_DATA	Read-back data of configuration memory (Efuse, flash, or regfile).
ISC_ERASE	1111111110	BYPASS_REG	To erase the Configure Done signal
BYPASS	1111111111	BYPASS_REG	To bypass data shift in
JTAG_ENABLE	1111100000	BYPASS_REG	To generate JTAG_REQ signal
JTAG_DISABLE	1111100001	BYPASS_REG	To Stop JTAG_REQ singal
JTAG_NOOP	1111100010	BYPASS_REG	The same as BYPASS instruction
JTAG_SETUP	1111100011	ISC_CONFIG	Select the operating target
JTAG_PROGRAM	1111100100	ISC_DATA	Program target selected by JTAG_SETUP instruction
JTAG_RDATA	1111100101	ISC_DATA	Read target selected by JTAG_SETUP instruction
FL_INIT	1111100110	BYPASS_REG	Initial protection status of flash, including updating the cipher and initializing flash status to locked

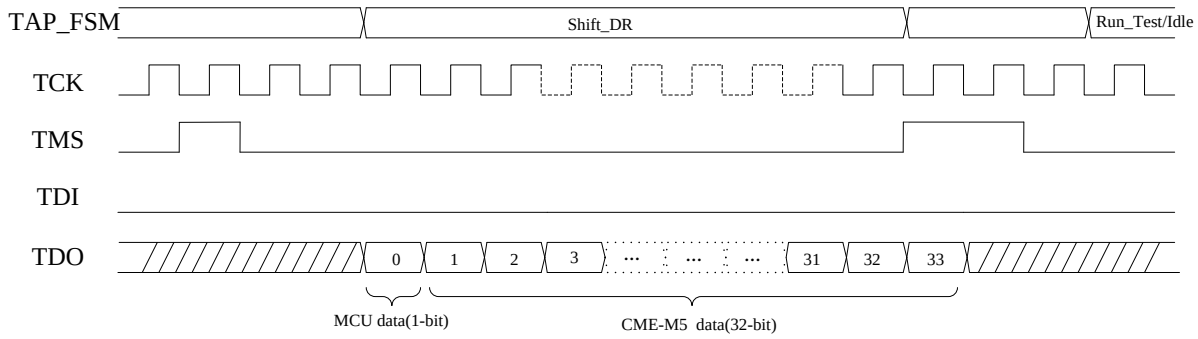
6.2Bypass the MCU

Because of the cascade of the two JTAG devices inside HME-M5, when debugging and configuring the HME-M5 device, another JTAG device for OCDS of MCU should be is bypassed firstly.

The instruction register of the MCU is a 4-bit width shift register, so the bypass instruction code is 4’b1111. The figure below shows the operation of sending instruction to HME-M5.



The data register of the MCU is a 1-bit width shift register, which should also be bypassed. So the operation of receiving data from HME-M5 is shown in the figure below.



6.3 IDCODE

The IDCODE instruction is mandatory if device identification register is provided.
The IDCODE instruction selects device identification register (DEVICE_ID_REG) to connect between TDI and TDO in the SHIFT_DR state.

When the IDCODE instruction is selected, the vendor identification code shall be loaded into the device identification register on the rising edge of TCK after entry into the CAPTURE_DR controller state.

Table 5 Device ID of M5

Device ID	Description
0000_06CB	M5C06N3
0000_16CB	M5C03N3
0020_06CB	M5P06N3
0020_16CB	M5P03N3

6.4 USERCODE

The USERCODE instruction is mandatory only when device identification register is provided and the component is user-programmable.

The USERCODE instruction selects device identification register (DEVICE_ID_REG) to connect between TDI and TDO in the SHIFT_DR state.

When the USERCODE instruction is selected, the user programmable identification code shall be loaded into the device identification register on the rising edge of TCK in the CAPTURE_DR state.

6.5 SAMPLE

SAMPLE, PRELOAD, EXTEST, SAMP/PRELD, CLAMP and HIGH_Z are all boundary scan related instruction, and the former three instructions are mandatory to every component compliant

to JTAG standard.

SAMPLE, PRELOAD, EXTEST, SAMP/PRELD select BSR to connect between TDI and TDO in SHIFT_DR state, while CLAMP and HIGH_Z would select BYPASS_REG to connect between TDI and TDO.

When the SAMPLE instruction is selected, the test logic shall have no effect on operation of the on-chip system logic and pins.

When the SAMPLE instruction is selected, all signals flowing through BSR should be captured into shift register of BSR on the rising edge of TCK in CAPTURE_DR state. As permitted, parallel output register can/may load the data held in the associated shift register on the falling edge of TCK in UPDATE_DR state.

6.6 PRELOAD

When the PRELOAD instruction is selected, the test logic shall have no effect on operation of the on-chip system logic and pins.
When the PRELOAD instruction is selected, parallel output register should load the data held in the associated shift register on the falling edge of TCK in UPDATE_DR state. And as permitted, all signals flowing through BSR can/may be captured into shift register of BSR on the rising edge of TCK in CAPTURE_DR state.

6.7 EXTEST

The EXTEST instruction is employed to test the off chip connection with other component on a board.

When the EXTEST instruction is selected, all outputs should be completely defined by the data held in the parallel output register of BSR and change only on the falling edge of TCK in UPDATE_DR state.

When the EXTEST instruction is selected, all signals received from the input pins shall be captured into shift register of BSR on the rising edge of TCK in CAPTURE_DR state, after which the captured data can be shifted out via TDO to verify.

In order to drive determinate data to output pins from parallel output registers, PRELOAD can be executed to load determinate data into parallel output registers preceded to instruction EXTEST.

Note: In ALPS and Fragrant, all pads are bidirectional which is controlled by signal “en”, care should be taken to the generation of test vector. If one pad is defined to be output pins, then the corresponded en should be defined to be ‘0’, else ‘1’.

6.8 TESTCTRL

The TESTCTRL instruction is employed to control test mode enable signals.

When the TESTCTRL instruction is selected, data on TDI shift into the register TSTCTRL_REG from MSB to LSB in SHIFT_DR tap state

When the TESTCTRL instruction is selected, data held in the register TSTCTRL_REG is updated into TSTCTRL in UPDATE_DR tap state.

Table 4 illustrates the bit map of TSTCTRL_REG.

6.9 CLAMP

The CLAMP instruction is employed to speed up the test process. If there are number of ICS need to be test of the interconnection on a board, then test data can be loaded into parallel output register of BSR with instruction PRELOAD, after that, CLAMP can be executed to test each chip without change or shift the data held in the parallel output registers of BSR.

When the CLAMP instruction is selected, all output should be completely defined by the data held in the parallel output registers and will not change while CLAMP.

6.10 HIGH_Z

The HIGH_Z instruction is employed to test component that doesn't comply with IEEE1149.1 with test logic (e.g. an ATE). When the HEGH_Z instruction is executed, the test logic will be active while other components will be inactive, then the test logic would be able to test the component that doesn't comply with IEEE1149.1, otherwise, test logic will be in an inactive state.

The instruction HIGH_Z is identical to CLAMP except that all out put should be placed in an inactive Drive state (e.g. high impedance) immediately after instruction HIGH_Z is selected.

6.11 ISC_ENABLE

When the ISC_ENABLE instruction is executed, BYPASS register is selected to connect between TDI and TDO in SHIFT_DR state.

The signal isc_enabled is asserted once the ISC_ENABLE instruction is executed, and is de-asserted once ISC_DISABLE is executed.

The two instructions ISC_ENABLE and ISC_DISABLE are mandatory as defined in standard IEEE 1532 which are used to indicate the beginning and end of ISC operations.

6.12 ISC_DISABLE

When the ISC_DISABLE instruction is executed, BYPASS register is selected to connect between TDI and TDO in SHIFT_DR state.

ISC_DONE is cleared by successful execution of this instruction.

6.13 ISC_PROGRAM

When the ISC_PROGRAM instruction is executed, ISC_DATA register is selected to connect between TDI and TDO in SHIFT_DR state.

The ISC_PRAGAM and ISC_SETUP work corporately to specify the configuration target.

6.14 ISC_NOOP

When the ISC_NOOP instruction is executed, BYPASS_REG register is selected to connect between TDI and TDO in SHIFT_DR state.

In a multi-device configuration board, it is very possible that not all the chip need the same configuration period, ISC_NOOP instruction shall be inserted for “small” device, select bypass register with no operation and stay in ISC_ACCESSED system modal state.

6.15 ISC_SETUP

When the ISC_SETUP instruction is executed, ISC_CONFIG register is selected to connect between TDI and TDO in SHIFT_DR state.

The ISC_SETUP instruction together with ISC_PROGRAM and ISC_READ instructions can finish programming and reading of specified memory/registers, the ISC_CONFIG register which targeted by ISC_SETUP specify the program/read target and corresponding instruction listed in table below.

The ISC_SETUP instruction shall be executed prior to the ISC_PROGRAM and ISC_READ instructions to specify the out coming operating target.

Table 6 ISC_CONFIG register

Content	Instruction	Description
0x00	CFG_FP_REG	Select Fabric and register as current operating target
0x01	EFUSE_SEL0	Select EFUSE 0 to operate
0x02	EFUSE_SEL1	Select EFUSE 1 to operate

Content	Instruction	Description
0x04	DFF_SCAN_NUM	DFF scan chain number to be tested
0x05	DFF_SCAN	Test DFF scan chain
0xF0	TOTAL_NUM	Send total number of JTAG device
0xF1	EFUSE_KEY	Send key for PGM or temporary PGM
0xF2	PGM_KEY	Trigger key programming
0xF3	PGM_KEY_CNT	Indicate to input count value for programming
0xF4	PGM_TEMP_KEY	Trigger temporary key programming
0xF5	READ_EFUSE1	Read EFUSE1 serially
0xF6	EFUSE_CRC	Send CRC for input EFUSE_KEY
0xF7	EFUSE_CMP	Compare if the input key is equal to the key for AES
0xF8	SELECT_FLASH	Select Flash as current operating target
0xFE	ISC_CFG_DONE	Indicate to activate ISC_DONE

6.16 ISC_RDATA

When the ISC_RDATA instruction is executed, ISC_DATA register is selected to connect between TDI and TDO in SHIFT_DR state.

The ISC_RDATA and ISC_SETUP work corporately to specify the read-data type.

6.17 ISC_ERASE

Currently, the ISC_ERASE instruction is used to erase the JTAG_CFG_DONE indication signal.

Flash erasing can be implemented by transport SE/BE instruction to flash, and configuration erasing can be implemented by active re_init to initialize all chain to 0s.

6.18 BYPASS

The BYPASS instruction is mandatory to every component compliant to JTAG standard.
The BYPASS instruction selects BYPASS_REG to connect between TDI and TDO in the SHIFT_DR state, it allows more rapid movement of test data to/from one component that are required to perform test operations.

The BYPASS instruction is defined to be all 1s in binary code as permitted in IEEE 1149.1 standard and all other undefined codes should be interpreted to be BYPASS.

BYPASS is force into parallel output of IR in TEST_LOGIC_RESET state if no device identification is provided, however, in our design, device identification is provided, so IDCODE is forced into parallel output of IR in TEST_LOGIC_RESET state.

6.19 JTAG_ENABLE

When the JTAG_ENABLE instruction is executed, BYPASS register is selected to connect between TDI and TDO in SHIFT_DR state.

The signal jtag_req is asserted once the JTAG_ENABLE instruction is executed, and is de-asserted once JTAG_DISABLE is executed.

6.20 JTAG_DISABLE

When the JTAG_ENABLE instruction is executed, BYPASS register is selected to connect between TDI and TDO in SHIFT_DR state.

The signal jtag_req is asserted once the JTAG_ENABLE instruction is executed, and is de-asserted once JTAG_DISABLE is executed.

6.21 JTAG_NOOP

The instruction JTAG_NOOP is the similar as BYPASS instruction and can be altered with new instruction.

6.22 JTAG_SETUP

When the JTAG_SETUP instruction is executed, ISC_CONFIG register is selected to connect between TDI and TDO in SHIFT_DR state.

The JTAG_SETUP instruction is similar as ISC_SETUP which is used to select the operating target.

6.23 JTAG_PROGRAM

When the JTAG_PROGRAM instruction is executed, ISC_DATA register is selected to connect between TDI and TDO in SHIFT_DR state.

6.24 JTAG_RDATA

When the JTAG_RDATA instruction is executed, ISC_DATA register is selected to connect between TDI and TDO in SHIFT_DR state.

6.25 FL_INIT

When the FL_INIT instruction is executed, BYPASS register is selected to connect between TDI and TDO in SHIFT_DR state.

The FL_INIT instruction is employed to initial flash monitor’s status, and to retrieve the flash cipher for the flash monitor module.

7 Register

The table below list all the register implemented in reg file.

Note that register 4 CTRL_MASK is used to mask register 5 CTRL. For example, if CTRL_MASK is 32'h8000_0000, only bit [31] would be modified while writing CTRL register.

Table 7 The register list in reg file

Register Name	Address	R/W	Default value	Descriptions
DEVICE ID	0	R/W	Device ID	Chip revision id. When write a check will be performed and mismatch error is generated.
RECFG	1	R/W	0x00000000	[31:8]: Flash start address for RECFG [0]: recfg
USERID	3	R/W	0xFFFFFFFF	USERID required by JTAG
CTRL_MASK	4	R/W	0xFFFFFFFF	CTRL register control mask, can be used for certain bit configuration and not change other bit value
CTRL	5	R/W	0x00030000	[31:30]: 2'h0 [29]: cfgdone_mcu_en [28]: cfgdone_fp_en [27:22]: 6'h0 [21]: update_io [20]: update_gclk1 [19]: update_gclk0 [18]: mcu_reset [17:16]: isolation_ctrl [15]: read_disable0 [14]: cf_rstn [13:11]: fusectrl [10:8]: oscctrl [7]: vcfg_sel [6]: vcfg_enable [5]: spmbist_en [4]: 1'b0 [3]: soft_rst [2]: cfgdone_fp_flg [1]: cfgdone_mcu_flg [0]: mcu_mode

Register Name	Address	R/W	Default value	Descriptions
STATUS	6	RO	0x00000000	[31:16]:16'h0000 [15]:read_disable [14]:read_disable0 [13:12]:cfg_domn [11]:clk_end [10]:spmbist_fail_high [9]: spmbist_fail_low [8]: spmbist_done_high [7]: spmbist_done_low [6]: dec_error [5]: chksum_err [4]: wrong_id [3]: mcu_mode [2]: usermode_mcu [1]: usermode_fp [0]: cfgdone

8 Flash

Internal Flash can be accessed either by the SPI controller or by the JTAG which is selected by a clock MUX. Once JTAG assert JTAG_ENABLE to begin JTAG operation, the MUX will select JTAG control signals into flash so that JTAG can operate flash directly.

To operate flash shall obey the instruction format of SPI interface. The instruction format is composed of four parts: Command (8-bits) + Address (24-bits) + Dummy (8-bits) + Data, see table below.

Table 8 Flash instructions

Command	One-Byte Command Code	Address Bytes	Dummy Bytes	Data Bytes
READ	03H (0000 0011)	3	0	1 to ∞
FAST_READ	0BH (0000 1011)	3	1	1 to ∞
WREN	06H (0000 0110)	0	0	0
SE	D8H (1101 1000)	3	0	0
BE	C7H (1100 0111)	0	0	0
PP	02H (0000 0010)	3	0	1 to 256
RDSR	05H (0000 0101)	0	0	1 to ∞
WRSR	01H (0000 0001)	0	0	1
WRDI	04H (0000 0100)	0	0	0
RDID	9FH (1001 1111)	0	0	1 to 3
READ_ID	90H (1001 0000)	3	0	1 to ∞

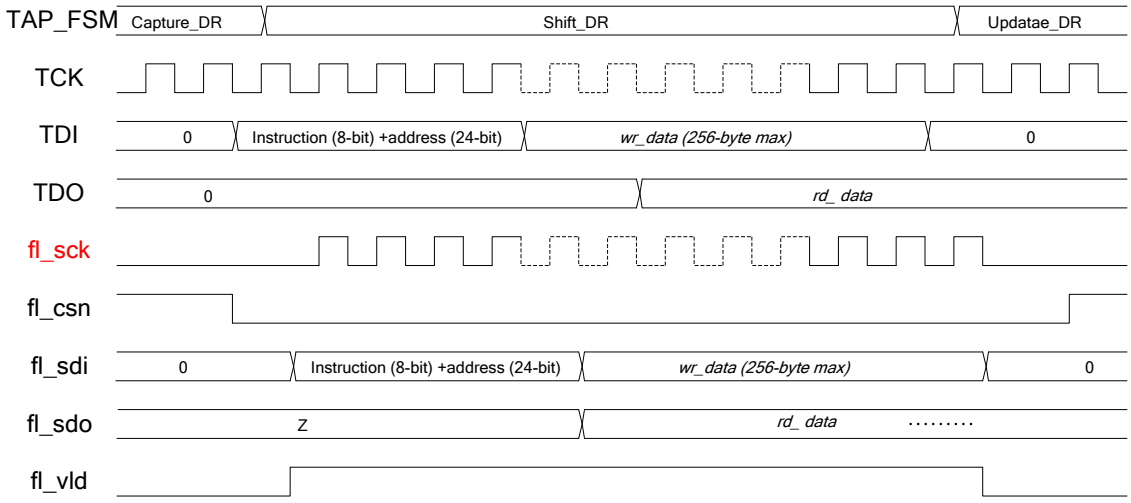


Figure 10 Waveform of Flash operation

Flash Operation via JTAG is protected by the security bit. User can define it and store inside flash, by default it is 32'hFFFF_FFFF. So when one wants to operate flash via JTAG interface, it shall first check the security bit.

The following steps shall be followed to check the security bit. If the security bit is LOW inactivated, then only a limit number of flash instructions can be executed, otherwise, all flash instructions can be executed.

The table below shows how to check security bit.

Table 9 Steps to read security bit

Step	JTAG instruction	Description
1	JTAG_ENABLE	Send JTAG require
2	ISC_ENABLE	Send ISC_ENABLE
3	ISC_SETUP(TOTAL_NUM)	Select TOT_NUM register as target
4	ISC_PROGRAM(number)	Send total number of JTAG device
5	ISC_SETUP(SELECT_FLASH)	Select flash as target
6	ISC_PROGRAM	Send flash WRSR command(wrsr value is 8'h00)
7	ISC_PROGRAM	Send flash RDSR command
8	ISC_PROGRAM	Send flash READ command and address('d28), and make sure the read cycle is 32
Now check the security bit ()		
……	……	……

ISC_SETUP is employed to select flash as out coming operation target.

ISC_PROGRAM is employed to write {Command (8-bits) + Address (24-bits) + Dummy (8-bits)} to flash and to write WR_Data to the flash or read RD_Data from flash. When write data to flash, data shall be put on the TDI, and when read data from flash (RDSR, RDID, READ and etc..), JTAG host can get the expected data after send over command [and address [and dummy bytes]].

The table below listed flash accessing flow using ISC_SETUP and ISC_PROGRAM.

Table 10 Operation steps of flash

No	Steps	Description
1	ISC_SETUP(TOTAL_NUM)	Select TOT_NUM register as operating target
2	ISC_PROGRAM(number)	Send total number of JTAG device
3	ISC_SETUP(SELECT_FLASH)	Select flash as operating target
4	ISC_PROGRAM	Program command, address and program/read data
……	……	……

Note: Flash is accessed by JTAG directly, as required by the IEEE 1149.1 standard, the LSB of

data is transferred first, so when execute ISC_PROGRAM, the programming data shall be reversed so that the MSB of command is transferred first. The security bit is stored in SPI flash at address 'd28.

9 E-FUSE Programming with CRC

As the E-fuse is one-time-programmable, more attention shall be paid to its programming.

This sub-module is designed to assure the data prepared for E-fuse programming is correct checked using a CRC mechanism. The CRC mechanism is implemented to check the CRC value of input data, if the CRC value is the same as input CRC value, then e-fuse programming can be triggered by sending specified JTAG instructions.

9.1 E-fuse

There are totally 2 128bit E-fuses embedded in M5, E-fuse can be accessed via JTAG when prot_flagn is HIGH inactivated.

Before you operate E-fuse, the instructions listed in Table IR capture values shall be executed previously.

Table below list all operation support in M5.

Table 11 E-fuse operation

Operation	E-fuse 0	E-fuse 1	Description
Program E-fuse	√	√	Program input data E-fuse
Read E-fuse (to internal logic)	√	√	Read E-fuse content to internal logic
Write E-fuse register	√	√	Write input data into E-fuse register
Compare	√	√	Compare input data with the value on E-fuse pins Q[127:0]
Read out E-fuse data	×	√	Read out E-fuse data outside via JTAG

9.1.1 Program E-fuse

As E-fuse is OTP based, it shall be carefully when programming E-fuse.

And programming E-fuse require strict programming time, so a count value shall be transfer into the inside programming FSM.

The table below lists the programming steps, which include sending count value, sending CRC value, sending input 128-bit data and triggering programming.

Table 12 E-fuse Programming steps

Steps	JTAG instruction composition	Description
Select E-fuse to operate	ISC_SETUP (EFUSE_SEL0/EFUSE_SEL1)	Select which E-fuse to program
Send count value	ISC_SETUP(PGM_KEY_CNT)	Send a count value which will be used by internal program logic
	ISC_PROGRAM(count[7:0])	
Send CRC	ISC_SETUP(EFUSE_CRC)	Send CRC value which will be used to assure the following input data is correct
	ISC_PROGRAM(CRC[31:0])	
Send 128-bit data	ISC_SETUP(EFUSE_KEY)	Send 128 bit data to program into the E-fuse select by step 1
	ISC_PROGRAM(data[127:0])	
Trigger programming	ISC_SETUP(EFUSE_KEY)	Trigger internal programming logic to start programming
	ISC_PROGRAM(data[127:0])	
	Check “wfuse_done”, see ***Table 3***.	

9.1.2 Read out E-fuse 1 data

Reading out E-fuse data via JTAG is only supported by E-fuse 1, the table below lists steps.

Table 13 Reading out E-fuse steps

Steps	JTAG instruction composition	Description
Reading out	Check “rfuse_done1”	Reading out data Data[127:0] is expected data
	SETUP(READ_EFUSE1)	
	PROGRAM_RDLD(data[128:0])	

9.1.3 Efuse usage

E-fuse 0 is used to store the cipher key for AES decryption, and E-fuse 1 is used to store product information, package information and other control bits as shown in the table below.

Table 14 Bit map for E-fuse 1

Bit	Description	
[127:8]	Reserved	
[7:6]	Package information	
[5]	Decryption flag	
[4]	JTAG & OCDS disable	
[3]	FUSE0/1 Lock, once enabled, both the two E-fuse can’t be programmed	
[2]		
[1]		
[0]		

9.2 Special note

1) The Efuse program interface schematic refers to **JTAG Schematic**.
2) The power information of VDDQ when electrical fuse in programming mode

Table 15 Power information of VDDQ

Mode	Symbol	Description	WC	TC	BC	Unit
VDDQ	Ivddq_prog	The VDDQ current flow when VDDQ is set at high.	8.62	0.71	0.20	mA

3) VDDQ ramp-up slew rate must be slower than 2.5V/250us to avoid unintentional program.

Figure 11 VDDQ ramp-up timing

4) The count = 5us/ Tfuse_clk. The frequency of fuse_clk is from 15M~25M.
5) FUSE_CLK should be on when JTAG starts to program and stop when program is finished.