# SpartanMC

## *JTAG-Controller*

# Table of Contents

# List of Figures

# List of Tables

# JTAG-Controller

The JTAG-Controller for the SpartanMC is a JTAG-Master. It can communicate with JTAG-Slaves, which are connected through the 4 JTAG-Pins described in the following Table. If you need a TRST-Port you can use a PortOut Component of the SpartanMC. This component implements an extra feature for MSP430 micro controllers to control the internal clock signal.

**Table 1: JTAG Basics**

| Pin | Description |
|---|---|
| TCK | Test Clock - this pin is the clock signal used for ensuring the timing of the boundary scan system. The TDI shifts values into the appropriate register on the rising edge of TCK. The selected register contents shift out onto TDO on the falling edge of TCK. |
| TDI | Test Data Input - Test instructions shift into the device through this pin. |
| TDO | Test Data Output - This pin provides data from the boundary scan registers, i.e. test data shifts out on this pin. |
| TMS | Test Mode Select - This input which also clocks through on the rising edge of TCK determines the state of the TAP controller. |
| TRST | This is an optional active low test reset pin. It permits asynchronous TAP controller initialization without affecting other device or system logic. |

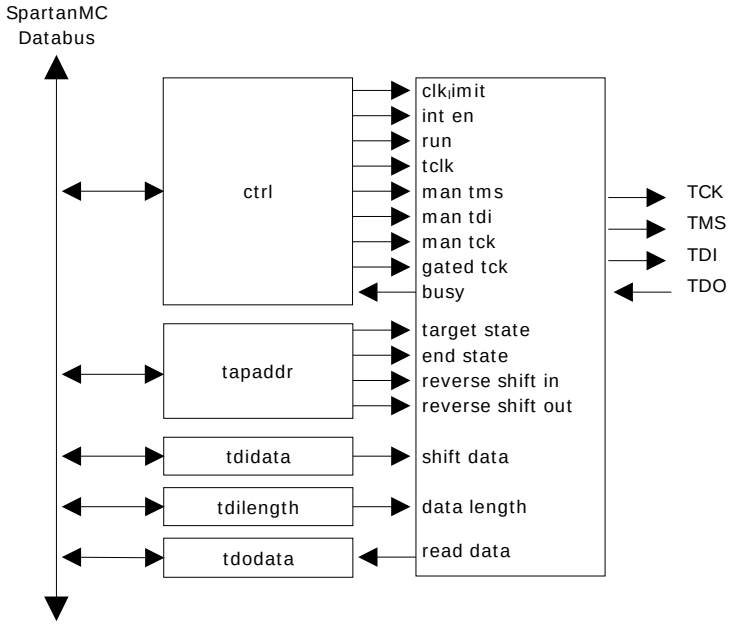


**Figure 1: JTAG block diagram**

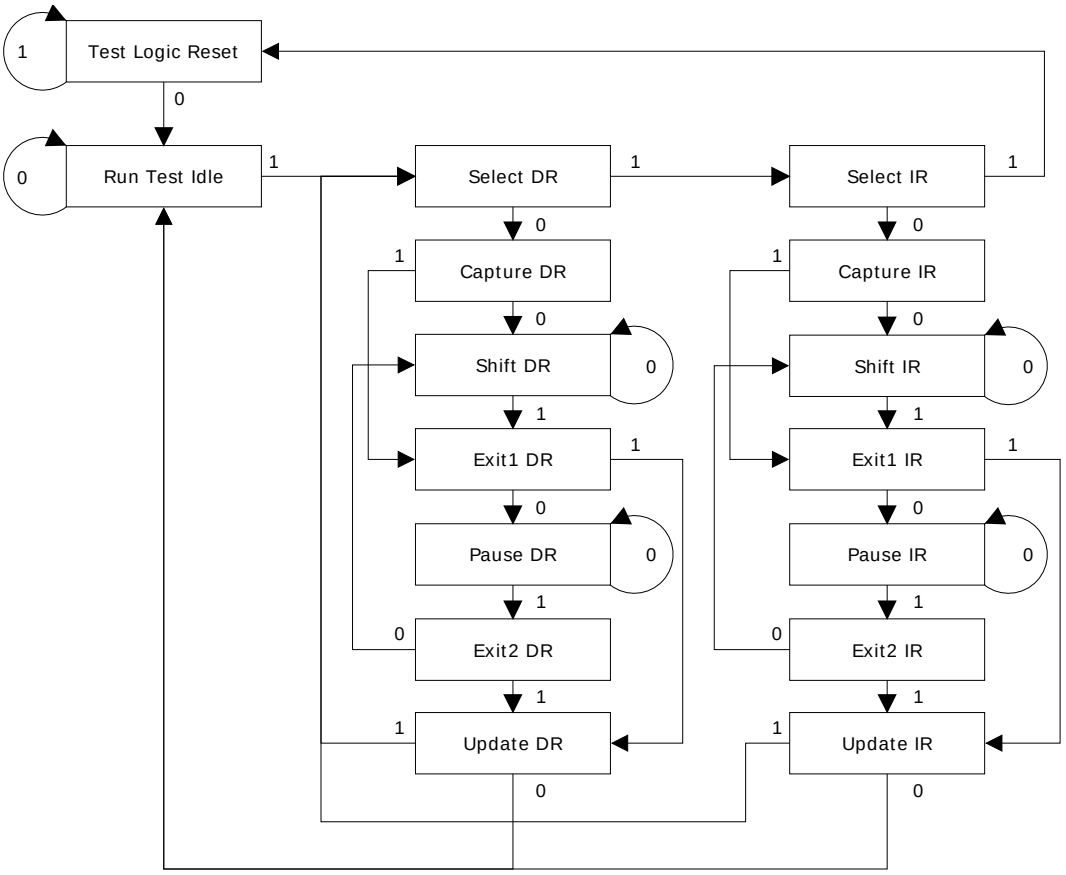


**Figure 2: JTAG TAP Controller State Machine**

# 1. Communication

To shift data to a connected device, you have to bring the TAP Controller in the device into the Run-Test-Idle-State. This can be done by resetting the Controller (clocking TMS=1 6 times) manually and clocking one TMS=0 in to go to the RTI-State. Now the Run-Bit (Bit 11 in the ctrl-register) can be set one and the controller is in automatic mode. Shifting data is very simple. Set target state in the TAP-Control-Register (maybe SHIFT_DR = 4) and set the end state (maybe RUN-TEST-IDLE). Then you put the data into the tdidata register and set the length of this data. Now the controller drives the TAP-Controller in your connected device into the target state (here: 1-0-0), shifts the data and generates a TMS-sequence to drive the TAP-Controller into the end state (here 1-1-0). When the process is done a interrupt is generated when interrupts are enabled. You can also poll the control-register and check the busy-bit (bit 18) to know when the process is done.

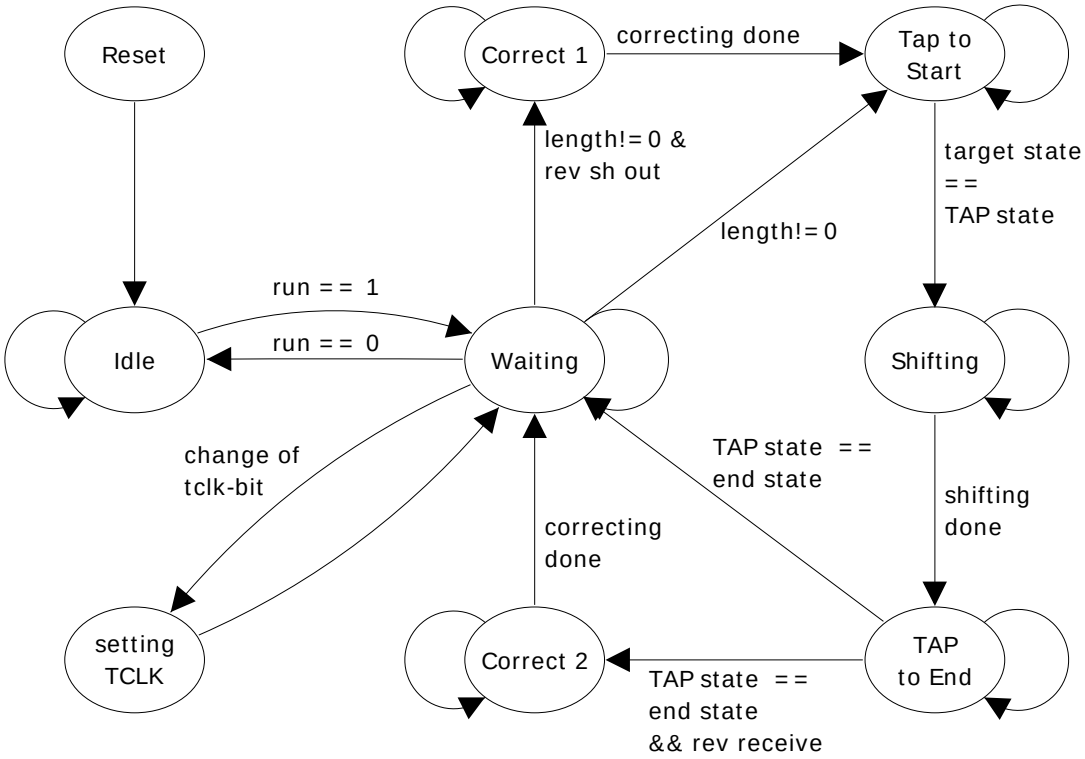


**Figure 3: JTAG State machine**

# 2. Module parameters

This module does not have Synthesis-Parameter

# 3. Peripheral Registers

## 3.1. JTAG Register Description

The JTAG peripheral provides 5 18 bit registers which are mapped to the SpartanMC address space.

**Table 2: JTAG registers**

| Offset | Name | Access | Description |
|---|---|---|---|
| 0 | ctrl | read/ write | Contains the current JTAG setting e.g. clock divider, irq settings, jtag signal levels and generates a busy signal. |
| 1 | tapaddresses | read/ write | Register for setting the target- and end state of the shifting process. Also configuring reverse send and reverse receive. |
| 2 | tdidata | read/ write | Register for data which should be shifted out. |
| 3 | tdilength | read/ write | Register for the length of the data. Writing this registers starts a shift process. |
| 4 | tdidata | read | Register for received data. |

## 3.2. JTAG Control Register (ctrl)

**Table 3: JTAG control register layout**

| Bit | Name | Access | Default | Description |
|---|---|---|---|---|
| 0-9 | EN | r/w | 1 | Clock devive register. |
| 10 | IRQ_EN | r/w | 0 | Enable sending irqs. |
| 11 | Run | r/w | 0 | Enables the automatic mode of the Controller. Before you enable the automatic mode you have to put the TAP controller in your connected device into the RUN-TEST-IDLE mode using the bits for TMS and TCK in this register. |
| 12 | TCLK | r/w | 0 | This bit is a special control bit for Ti-MSP430 microcontroller. It controls the internal clk signal of this microcontroller. Setting TDI to Logic One and holding the TCK for 3 cycles high, set the internal clock signal to one. Setting TDI to zero and holding TCK for three cycles high, set the internal clock signal to zero. The Value of this bit will be set to the clock signal in the MSP430. |
| 13 | man TMS | r/w | 0 | Logical level of the TMS Pin when not in auto mode. |

| Bit | Name | Access | Default | Description |
|---|---|---|---|---|
| 14 | man TDI | r/w | 0 | Logical level of the TDI Pin when not in auto mode. |
| 15 | man TCK | r/w | 0 | Logical level of the TCK Pin when not in auto mode. |
| 16 | gated clk | r/w | 0 | If this Bit is set to one, the controller does not generate the tck-signal when the controller is idle. |
| 17 | busy | r | 0 | This Bit is set to one when the controller performing a shift operation or changes the tclk-signal (MSP430-feature) |

**Table 3: JTAG control register layout**

## 3.3. JTAG TAP Control Register (tapaddr)

**Table 4: JTAG TAP control register layout**

| Bit | Name | Access | Default | Description |
|---|---|---|---|---|
| 0-3 | TAP target | r/w | 0 | In this state of the TAP-Controller the provided data will be shifted in. |
| 4-7 | TAP end | r/w | 0 | After shifting the data, the controller will drive the TAP-Controller in the connected device into this state. |
| 8 | reverse send | r/w | 0 | If this bit is set to one, the controller will shift out the MSB of the given data first else the LSB will be shifted out first. |
| 9 | reverse receive | r/w | 0 | If this bit is set to one, the controller will receive the data as MSB else as LSB in the receive shift register. |
| 10-17 | not used | - | - | not used |

**Table 4: JTAG TAP control register layout**