



# Opal RISC-V SoC Data Sheet

---

DS-OPAL-v1.6  
December 2021  
[www.elitestek.com](http://www.elitestek.com)



# Contents

<b>Introduction.....</b>	<b>4</b>
VexRiscv RISC-V Core.....	4
<b>Features.....</b>	<b>5</b>
<b>Functional Description.....</b>	<b>6</b>
Address Map.....	7
Flash Address.....	7
Clocks.....	7
Interrupts.....	8
Resets.....	8
APB3 Interface.....	8
JTAG Interface.....	9
GPIO Peripheral Interface.....	10
Input Register: 0x0000_0000.....	10
Output Register: 0x0000_0004.....	11
Output Enable Register: 0x0000_0008.....	11
Interrupt Rise Enable Register: 0x0000_0020.....	12
Interrupt Fall Enable Register: 0x0000_0024.....	12
Interrupt High Enable Register: 0x0000_0028.....	12
Interrupt Low Enable Register: 0x0000_002C.....	12
I <sup>2</sup> C Peripheral Interface.....	13
txData Register: 0x0000_0000.....	14
txAck Register: 0x0000_0004.....	14
rxData Register: 0x0000_0008.....	15
rxAck Register: 0x0000_000C.....	15
Interrupt Register: 0x0000_0020.....	16
Interrupt Clears Register: 0x0000_0024.....	17
Timeout Register: 0x0000_002C.....	17
Sampling Clock Divider Register: 0x0000_0028.....	17
tsuData Register: 0x0000_0030.....	17
Master Status Register: 0x0000_0040.....	18
tLow Register: 0x0000_0050.....	18
tHigh Register: 0x0000_0054.....	18
tBuf Register: 0x0000_0058.....	18
Filtering Status Register: 0x0000_0080.....	19
Hit Context Register: 0x0000_0084.....	19
PLIC Peripheral Interface.....	20
SPI Master Peripheral Interface.....	21
Cmd Register: 0x0000_0000.....	22
RSP Register: 0x0000_0004.....	22
Config Register: 0x0000_0008.....	22
Interrupt Register: 0x0000_000C.....	23
clockDivider Register: 0x0000_0020.....	23
ssSetup Register: 0x0000_0024.....	23
ssHold Register: 0x0000_0028.....	23
ssDisable Register: 0x0000_002C.....	24
ssActiveHigh Register: 0x0000_0030.....	24
UART Peripheral Interface.....	24
Data Register: 0x0000_0000.....	24
Status Register: 0x0000_0004.....	25
Clock Divider Register: 0x0000_0008.....	25

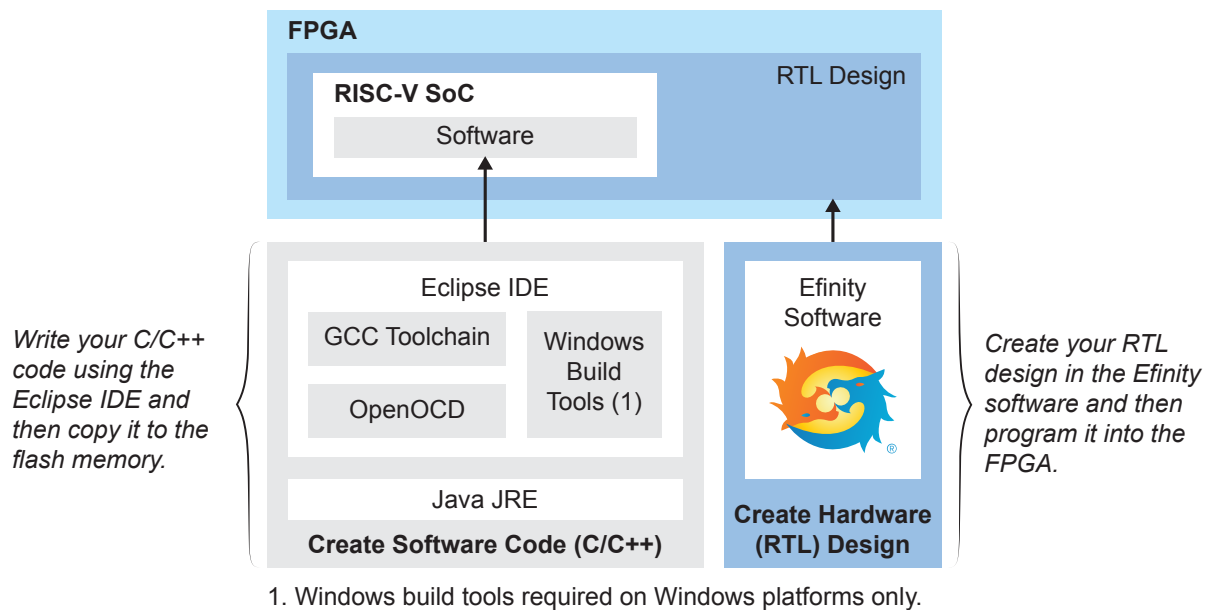
Config Register: 0x0000_000C.....	25
Control and Status Registers.....	26
Machine-Level ISA.....	26
Hart ID Register (mhartid): 0xF14.....	26
Machine Status Register (mstatus): 0x300.....	27
Machine Trap-Vector Base-Address Register (mtvec): 0x305.....	27
Machine Interrupt Enable Register (mie): 0x304.....	27
Machine Exception Program Counter (mepc): 0x341.....	28
Machine Cause Register (mcause): 0x342.....	28
Machine Trap Value Register (mtval): 0x343.....	29
Machine Interrupt Pending Register (mip): 0x344.....	29
<b>Revision History.....</b>	<b>30</b>

# Introduction

易灵思 provides the ultra-light-weight soft RISC-V SoC, Opal, which has a small resource footprint. This SoC is ideal for applications that require embedded compute capability such as system monitoring or remote configuration and control. Some example applications for the Opal SoC are low-power, small form factor IOT, or handheld devices. This core is similar to the open-source SaxonSOC, but it has been optimized for 钛金系列 and Trion FPGAs. The Opal SoC comes in several variations:

- For 钛金系列 FPGAs and Trion T13 and larger FPGAs. This variation is included with the Efinity<sup>®</sup> IP Manager.
- For the Trion T8 BGA81 FPGA. This variation has a slower  $f_{MAX}$  than the regular Opal SoC and uses a soft JTAG block and FTDI cable for debugging instead of a regular USB cable. A complete example is available for download in the Support Center Example Designs page.
- For the Xyloni Development Board. This variation is configured to take advantage of the resources on the Xyloni Development Board and is available on GitHub.

Figure 1: Opal RISC-V SoC Design Flow



**Learn more:** For details on developing RTL designs or creating software, refer to the [Opal RISC-V Hardware and Software User Guide](#), [Exploring the Opal SoC and T8 FPGA using the Trion T8 BGA81 Development Board](#), and [Opal \(Xyloni\) RISC-V Hardware and Software User Guide](#).

## VexRiscv RISC-V Core

The Opal SoC is based on the VexRiscv core created by Charles Papon. The VexRiscv core is a 32-bit CPU using the ISA RISC-V32I with M and C extensions and has five pipeline stages (fetch, decode, execute, memory, and writeback).

In the Opal SoC, the cacheless VexRiscv core supports an APB3 bus interface.

The VexRiscv core won first place in the RISC-V SoftCPU contest in 2018.<sup>(1)</sup>

## Features

- VexRiscv processor with 5 pipeline stages (fetch, decode, execute, memory, and write back), interrupts and exception handling with machine mode
- 20 - 300 MHz or 20 MHz (T8 BGA81 only) system clock frequency
- 4 - 512 KB on-chip RAM with boot loader for SPI flash
- APB3 peripherals:
  - 8 or 16 GPIOs
  - 1 I<sup>2</sup>C master and slave
  - Machine timer
  - PLIC
  - 1 or 2 SPI flash master(s) with a maximum clock frequency of 25 MHz (10 MHz maximum for T8 BGA81 only)
  - 1 UART with 115,200 baud rate
  - 1 slave user peripheral

### FPGA Support

The Opal SoC supports all 钛金系列 FPGAs and all Trion FPGAs (except the T4).

### 钛金系列 Resource Utilization and Performance

FPGA	Logic/Adders	FlipFlops	Memory Blocks	DSP48 Blocks	f <sub>MAX</sub> (MHz)	Efinity Version
Ti60 F225 C4	4,439	2,295	14	4	348	2021.2

### Trion Resource Utilization and Performance

FPGA	Logic Utilization (LUTs)	Memory Blocks	f <sub>MAX</sub> (MHz)	Efinity Version
T20 BGA256 C4	5,951	18	83	2021.2
T8 BGA81 C2	5,196	16	23.793	2019.3

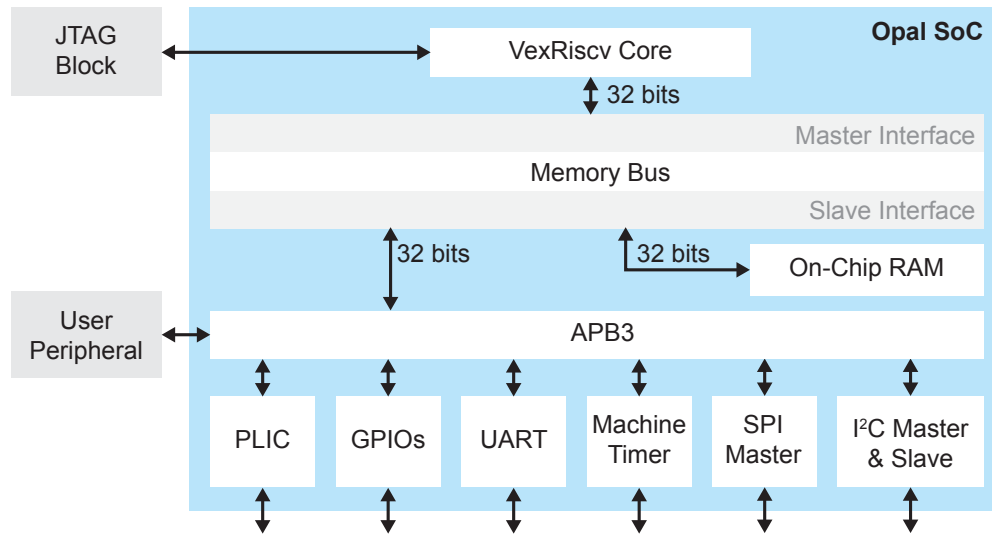
<sup>(1)</sup> <https://www.businesswire.com/news/home/20181206005747/en/RISC-V-SoftCPU-Contest-Winners-Demonstrate-Cutting-Edge-RISC-V>

# Functional Description

The Opal SoC incorporates a 32-bit RISC-V processor, 4 - 512 KB of on-chip RAM, and a variety of peripherals (including 1 APB3 slave peripheral). It runs at operating frequencies of 20 - 300 MHz.

You can customize the SoC using the IP Manager in the software v2020.2 and higher.

Figure 2: Opal SoC Block Diagram



The Opal SoC comes in variations that have slightly different peripheral sets.

Table 1: Opal SoC Peripheral Sets

Peripheral	Opal	Opal (Xyloni)
GPIO	8	16
I <sup>2</sup> C	1	1
PLIC	1	1
Machine timer	1	1
SPI master	1	2
UART	1	1
User peripheral (APB3 bus)	1	1

## Address Map



**Note:** Because the address range might be updated, 易灵思 recommends that you always refer to the parameter name when referencing an address in firmware, not by the actual address. The parameter names and address mappings are defined in `soc.h`.

*Table 2: Default Address Map, Interrupt ID, and Cached Channels*

Device	Parameter	Size	Interrupt ID	Region
GPIO	SYSTEM_GPIO_0_IO_APB	4K	[0]: 12 [1]: 13	I/O
I <sup>2</sup> C	SYSTEM_I2C_0_IO_APB	4K	8	I/O
Machine timer	SYSTEM_MACHINE_TIMER_APB	4K	31	I/O
PLIC	SYSTEM_PLIC_APB	4K	-	I/O
SPI master 0	SYSTEM_SPI_0_IO_APB	4K	4	I/O
SPI master 1 <sup>(2)</sup>	SYSTEM_SPI_1_IO_APB	4K	5	I/O
UART	SYSTEM_UART_0_IO_APB	4K	1	I/O
User peripheral	IO_APB_SLAVE_0_APB	64K	-	I/O
On-chip BRAM	SYSTEM_RAM_A_BMB	4 - 512 KB	-	Cache
External interrupt	-	-	25	I/O



**Note:** The RISC-V GCC compiler does not support user address spaces starting at 0x0000\_0000.

## Flash Address

When the FPGA boots up, the Opal SoC copies your binary application file from a SPI flash device to the on-chip memory, and then begins execution. The SPI flash binary address starts at 0x0038\_0000.

For the Xyloni Development Board, the address starts at 0x000E\_0000.

## Clocks

*Table 3: Clock Ports*

Port	Direction	Description
io_systemClock	Input	Provides a 20 - 300 MHz clock for the SoC.

<sup>(2)</sup> The open-source Opal SoC available on Github has 2 SPI masters. Other variations only have 1.

## Interrupts

**Table 4: Interrupt Ports**

Port	Direction	Description
userInterruptA	Input	Provides an external interrupt.

## Resets

The Opal SoC has a master reset signal, `io_asyncReset` that triggers a system reset. Your RTL design should hold `io_asyncReset` for 10 ns to reset the whole SoC system completely. When you assert `io_asyncReset`, the SoC asserts:

- `io_systemReset`, which resets the RISC-V processor, on-chip memory, and peripherals

Once `io_systemReset` goes low, the user binary code is executed.

**Table 5: Reset Ports**

Port	Direction	Description
<code>io_asyncReset</code>	Input	Active-high asynchronous reset for the entire system.
<code>io_systemReset</code>	Output	Synchronous active-high reset for <code>systemClk</code> .

## APB3 Interface

The following table shows the ports for the APB3 user slave peripheral. Refer to the AMBA APB Protocol Specification for APB port descriptions and handshake information.

**Table 6: APB3 Ports**

Port	Direction	Description
<code>io_apbSlave_0_PADDR[15:0]</code>	Output	User address.
<code>io_apbSlave_0_PSEL</code>	Output	User select.
<code>io_apbSlave_0_PENABLE</code>	Output	User enable.
<code>io_apbSlave_0_PREADY</code>	Input	User ready.
<code>io_apbSlave_0_PWRITE</code>	Output	User direction.
<code>io_apbSlave_0_PWDATA[31:0]</code>	Output	User write data.
<code>io_apbSlave_0_PRDATA[31:0]</code>	Input	User read data.
<code>io_apbSlave_0_PSLVERROR</code>	Input	User transfer failure.



## JTAG Interface

The Opal SoC uses the JTAG User TAP interface block to communicate with the OpenOCD debugger. The example designs for the T20 BGA256 Development Board and the Xyloni Development Board use this interface block.

**Table 7: JTAG Ports**

Port	Direction	Description
jtagCtrl_enable	Input	Indicates that the user instruction is active for the interface.
jtagCtrl_capture	Input	TAP controller is in the capture state.
jtagCtrl_shift	Input	TAP controller is in the shift state.
jtagCtrl_update	Input	TAP controller in the update state.
jtagCtrl_reset	Input	TAP controller is in the reset state.
jtagCtrl_tdi	Input	JTAG TDI for debugging.
jtagCtrl_tdo	Output	JTAG TDO for debugging.
jtagCtrl_tck	Input	JTAG TCK for debugging.

For T8 BGA81 designs, the Opal SoC has a soft JTAG interface to communicate with the OpenOCD debugger. This method sends JTAG signals through regular GPIO pins to the JTAG debugger.

**Table 8: JTAG Ports**

Port	Direction	Description
io_jtag_tms	Input	JTAG TMS for soft debug core.
io_jtag_tdi	Input	JTAG TDI for soft debug core.
io_jtag_tdo	Output	JTAG TDO for soft debug core.
io_jtag_tck	Input	JTAG TCK for soft debug core.



**Learn more:** For information on example designs, refer to the [Opal RISC-V Hardware and Software User Guide](#) or [Opal \(Xyloni\) RISC-V Hardware and Software User Guide](#).

## GPIO Peripheral Interface

Use the `SYSTEM_GPIO_0_IO_APB` parameter to reference the GPIO interface.

**Table 9: GPIO Ports**

Port	Direction	Description
<code>system_gpio_0_io_read[7:0]</code>	Input	GPIO input.
<code>system_gpio_0_io_write[7:0]</code>	Output	GPIO output.
<code>system_gpio_0_io_writeEnable[7:0]</code>	Output	GPIO output enable.

**Table 10: GPIO Register Map**

Address Offset	Register Name	Privilege	Width
<code>0x0000_0000</code>	Input	Read/Write	32
<code>0x0000_0004</code>	Output	Read/Write	32
<code>0x0000_0008</code>	Output Enable	Read/Write	32
<code>0x0000_0020</code>	Interrupt Rise Enable	Read/Write	32
<code>0x0000_0024</code>	Interrupt Fall Enable	Read/Write	32
<code>0x0000_0028</code>	Interrupt High Enable	Read/Write	32
<code>0x0000_002C</code>	Interrupt Low Enable	Read/Write	32

### Input Register: `0x0000_0000`

The Opal SoC provided with the Efinity® IP Manager has 8 GPIO. The version of the Opal SoC provided on Github for the Xyloni development board has 16.

#### Opal (IP Manager)

31	8 7	0
Reserved		Input

Bits	Field	Description	Privilege
0-7	Input	Set GPIO pin as an input (8 pins).	Read/Write
8-31	Reserved	Reserved.	N/A

#### Opal (Xyloni)

31	16 15	0
Reserved		Input

Bits	Field	Description	Privilege
0-15	Input	Set GPIO pin as an input (16 pins).	Read/Write
16-31	Reserved	Reserved.	N/A

*Output Register: 0x0000\_0004*

The Opal SoC provided with the Efinity® IP Manager has 8 GPIO. The version of the Opal SoC provided on Github for the Xyloni development board has 16.

## Opal (IP Manager)

31	8	7	0
Reserved		Output	

Bits	Field	Description	Privilege
0-7	Output	Set GPIO pin as an output (8 pins).	Read/Write
8-31	Reserved	Reserved.	N/A

## Opal (Xyloni)

31	16	15	0
Reserved		Output	

Bits	Field	Description	Privilege
0-15	Output	Set GPIO pin as an output (16 pins).	Read/Write
16-31	Reserved	Reserved.	N/A

*Output Enable Register: 0x0000\_0008*

The Opal SoC provided with the Efinity® IP Manager has 8 GPIO. The version of the Opal SoC provided on Github for the Xyloni development board has 16.

## Opal (IP Manager)

31	8	7	0
Reserved		OE	

Bits	Field	Description	Privilege
0-7	OE	Enable GPIO output pin (8 pins).	Read/Write
8-31	Reserved	Reserved.	N/A

## Opal (Xyloni)

31	16	15	0
Reserved		OE	

Bits	Field	Description	Privilege
0-15	OE	Enable GPIO output pin (16 pins).	Read/Write
16-31	Reserved	Reserved.	N/A

*Interrupt Rise Enable Register: 0x0000\_0020*

31	2	1	0
Reserved			IntRiseEn

Bits	Field	Description	Privilege
0-1	IntRiseEn	Enable a rise interrupt on GPIO pins 0 and 1.	Read/Write
2-31	Reserved	Reserved.	N/A

*Interrupt Fall Enable Register: 0x0000\_0024*

31	2	1	0
Reserved			IntFallEn

Bits	Field	Description	Privilege
0-1	IntFallEn	Enable a fall interrupt on GPIO pins 0 and 1.	Read/Write
2-31	Reserved	Reserved.	N/A

*Interrupt High Enable Register: 0x0000\_0028*

31	2	1	0
Reserved			IntHighEn

Bits	Field	Description	Privilege
0-1	IntHighEn	Enable a high interrupt on GPIO pins 0 and 1.	Read/Write
2-31	Reserved	Reserved.	N/A

*Interrupt Low Enable Register: 0x0000\_002C*

31	2	1	0
Reserved			IntLowEn

Bits	Field	Description	Privilege
0-1	IntLowEn	Enable a low interrupt on GPIO pins 0 and 1.	Read/Write
2-31	Reserved	Reserved.	N/A

## I<sup>2</sup>C Peripheral Interface

The Opal SoC has one I<sup>2</sup>C master/slave peripheral. Use the `SYSTEM_I2C_0_IO_APB` parameter to refer to the interface.

**Table 11: I<sup>2</sup>C Peripheral Ports (User)**

Port	Direction	Description
<code>system_i2c_0_io_sda_write</code>	Output	SDA output for user device.
<code>system_i2c_0_io_sda_read</code>	Input	SDA input for user device.
<code>system_i2c_0_io_scl_write</code>	Output	SCL output for user device.
<code>system_i2c_0_io_scl_read</code>	Input	SCL input for user device.

**Table 12: I<sup>2</sup>C Register Map**

Address Offset	Register Name	Privilege	Width
<code>0x0000_0000</code>	txData	Read/Write	32
<code>0x0000_0004</code>	txAck	Read/Write	32
<code>0x0000_0008</code>	rxData	Read/Write	32
<code>0x0000_000C</code>	rxAck	Read/Write	32
<code>0x0000_0020</code>	Interrupt	Read/Write	32
<code>0x0000_0024</code>	Interrupt Clears	Read/Write	32
<code>0x0000_0028</code>	Sampling Clock Divider	Read/Write	32
<code>0x0000_002C</code>	Timeout	Write	32
<code>0x0000_0030</code>	tsuData	Write	32
<code>0x0000_0040</code>	Master Status	Read/Write	32
<code>0x0000_0050</code>	tlow	Read/Write	32
<code>0x0000_0054</code>	tHigh	Read/Write	32
<code>0x0000_0058</code>	tBuf	Read/Write	32
<code>0x0000_0080</code>	Filtering Status	Read/Write	32
<code>0x0000_0084</code>	Hit Context	Read/Write	32
<code>0x0000_0088</code>	Filtering Configuration	Read/Write	32

*txData Register: 0x0000\_0000*

31	12	11	10	9	8	7	0
Reserved			DisableDataConflict	repeat	enable	valid	value

Bits	Field	Description	Privilege
0-7	value	Transmit data value.	Write
8	valid	Transmit data valid bit.	Read/Write
9	enable	Transmit data enable.	Read/Write
10	repeat	Transmit data repeat bit.	Write
11	DisableDataConflict	Disable transmit data conflict.	Write
12-31	Reserved	Reserved.	N/A

*txAck Register: 0x0000\_0004*

31	12	11	10	9	8	7	1	0
Reserved			DisableDataConflict	repeat	enable	valid	Reserved	
							value	

Bits	Field	Description	Privilege
0	value	Transmit acknowledge bit.	Write
1-7	Reserved	Reserved.	N/A
8	valid	Transmit acknowledge valid bit.	Read/Write
9	enable	Transmit acknowledge enable.	Read/Write
10	repeat	Transmit acknowledge repeat bit.	Write
11	DisableDataConflict	Disable transmit acknowledge conflict.	Write
12-31	Reserved	Reserved.	N/A

*rxData Register: 0x0000\_0008*

31	10	9	8	7	0
Reserved			listen	valid	value

Bits	Field	Description	Privilege
0-7	value	Received data.	Read
8	valid	Receive data valid.	Read
9	listen	Start listen data.	Write
10-31	Reserved	Reserved.	N/A

*rxAck Register: 0x0000\_000C*

31	10	9	8	7	1	0	
Reserved			listen	valid	Reserved		value

Bits	Field	Description	Privilege
0	value	Received acknowledge.	Read
1-7	Reserved	Reserved.	N/A
8	valid	Receive acknowledge valid.	Read
9	listen	Start listen acknowledge.	Write
10-31	Reserved	Reserved.	N/A

## Interrupt Register: 0x0000\_0020

31	22	21	20	19	18	17	16	15	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved			filterFlag	clockGenBusyFlag	Reserved		filterEnable	clockGenBusyEnable	Reserved			dropFlag	endFlag	restartFlag	startFlag	dropEnable	endEnable	restartEnable	startEnable	txAckEnable	txDataEnable	rxAckEnable	rxDataEnable

Bits	Field	Description	Privilege
0	rxDataEnable	Receive data interrupt enable	Read/Write
1	rxAckEnable	Receive acknowledge interrupt enable	Read/Write
2	txDataEnable	Transmit data interrupt enable	Read/Write
3	txAckEnable	Transmit acknowledge interrupt enable	Read/Write
4	startEnable	Start interrupt enable	Read/Write
5	restartEnable	Restart interrupt enable	Read/Write
6	endEnable	End interrupt enable	Read/Write
7	dropEnable	Drop interrupt enable	Read/Write
8	startFlag	Start interrupt flag	Read
9	restartFlag	Restart interrupt flag	Read
10	endFlag	End interrupt flag	Read
11	dropFlag	Drop interrupt flag	Read
12-15	Reserved	Reserved.	N/A
16	clockGenBusyEnable	Master clock generation interrupt enable.	Read/Write
17	filterEnable	Slave address filtering hit interrupt enable	Read/Write
18-19	Reserved	Reserved.	N/A
20	clockGenBusyFlag	Master clock generation interrupt flag.	Read
21	filterFlag	Slave address filtering hit interrupt flag.	Read
22-31	Reserved	Reserved.	N/A



*Interrupt Clears Register: 0x0000\_0024*

31	12	11	10	9	8	7	0		
Reserved				dropFlagClear	endFlagClear	restartFlagClear	startFlagClear	Reserved	

Bits	Field	Description	Privilege
0-7	Reserved	Reserved.	N/A
8	startFlagClear	Clear start flag.	Write
9	restartFlagClear	Clear restart flag.	Write
10	endFlagClear	Clear end flag.	Write
10	dropFlagClear	Clear drop flag.	Write
12-31	Reserved	Reserved.	N/A

*Timeout Register: 0x0000\_002C*

31	20	19	0
Reserved		value	

Bits	Field	Description	Privilege
0-19	value	Inactive timeout setting.	Write
20-31	Reserved	Reserved.	N/A

*Sampling Clock Divider Register: 0x0000\_0028*

31	10	9	0
Reserved		samplingClockDividerWidth	

Bits	Field	Description	Privilege
0-9	samplingClockDividerWidth	Clock divider width. Controls the rate at which the I <sup>2</sup> C controller reads SCL and SDA.	Read/Write
10-31	Reserved	Reserved.	N/A

*tsuData Register: 0x0000\_0030*

31	6	5	0
Reserved		value	

Bits	Field	Description	Privilege
0-5	value	Data setup time.	Write
6-31	Reserved	Reserved.	N/A

*Master Status Register: 0x0000\_0040*

31	7	6	5	4	3	1	0
Reserved				drop	stop	start	Reserved
							isBusy

Bits	Field	Description	Privilege
0	isBusy	Master busy.	Read
1-3	Reserved	Reserved.	N/A
4	start	Master start.	Read/Write
5	stop	Master stop.	Read/Write
6	drop	Master drop.	Read/Write
6-31	Reserved	Reserved.	N/A

*tLow Register: 0x0000\_0050*

31	12	11	0
Reserved		value	

Bits	Field	Description	Privilege
0-11	value	SCL low period.	Write
12-31	Reserved	Reserved.	N/A

*tHigh Register: 0x0000\_0054*

31	12	11	0
Reserved		value	

Bits	Field	Description	Privilege
0-11	value	SCL high period.	Write
12-31	Reserved	Reserved.	N/A

*tBuf Register: 0x0000\_0058*

31	12	11	0
Reserved		value	

Bits	Field	Description	Privilege
0-11	value	Start and stop bus free time.	Write
12-31	Reserved	Reserved.	N/A

*Filtering Status Register: 0x0000\_0080*

31	8	7	6	5	4	3	2	1	0		
Reserved				hit_7	hit_6	hit_5	hit_4	hit_3	hit_2	hit_1	hit_0

Bits	Field	Description	Privilege
0	hit_0	Filtering hit bit 0.	Read
1	hit_1	Filtering hit bit 1.	Read
2	hit_2	Filtering hit bit 2.	Read
3	hit_3	Filtering hit bit 3.	Read
4	hit_4	Filtering hit bit 4.	Read
5	hit_5	Filtering hit bit 5.	Read
6	hit_6	Filtering hit bit 6.	Read
7	hit_7	Filtering hit bit 7.	Read
8-32	Reserved	Reserved.	N/A

*Hit Context Register: 0x0000\_0084*

31	1	0
Reserved		rw

Bits	Field	Description	Privilege
0	rw	Hit context read.	Read
1-31	Reserved	Reserved.	N/A

## PLIC Peripheral Interface

Use the `SYSTEM_PLIC_APB` parameter to reference the interface PLIC interface.

Use the `SYSTEM_PLIC_CTRL` parameter to reference the interface PLIC interface.

**Table 13: RISC-V PLIC Operation Parameters**

Defines	Description
Interrupt priorities registers	The interrupt priority for each interrupt source.
Interrupt pending bits registers	The interrupt pending status of each interrupt source.
Interrupt enables registers	Enables the interrupt source of each context.
Priority thresholds registers	The interrupt priority threshold of each context.
Interrupt claim registers	The register to acquire interrupt source ID of each context.
Interrupt completion registers	The register to send interrupt completion message to the associated gateway.

The `soc.h` file contains a number of PLIC parameters to specify the interrupt ID for the various peripherals.

**Table 14: PLIC Interrupt ID Parameters**

Where  $n$  is the peripheral number and  $m$  is the interrupt ID.

Parameter	Refer to
<code>SYSTEM_PLIC_SYSTEM_I2C_n_IO_INTERRUPT m</code>	<b>Interrupt Register: 0x0000_0020</b> on page 16 <b>Interrupt Clears Register: 0x0000_0024</b> on page 17
<code>SYSTEM_PLIC_SYSTEM_GPIO_n_IO_INTERRUPTS_0 m</code>	<b>Interrupt Low Enable Register: 0x0000_002C</b> on page 12 <b>Interrupt High Enable Register: 0x0000_0028</b> on page 12 <b>Interrupt Fall Enable Register: 0x0000_0024</b> on page 12 <b>Interrupt Rise Enable Register: 0x0000_0020</b> on page 12
<code>SYSTEM_PLIC_SYSTEM_AXI_A_INTERRUPT</code>	<b>Interrupts</b> on page 8
<code>SYSTEM_PLIC_SYSTEM_SPI_n_IO_INTERRUPT m</code>	<b>Interrupt Register: 0x0000_000C</b> on page 23
<code>SYSTEM_PLIC_SYSTEM_UART_n_IO_INTERRUPT m</code>	<b>Status Register: 0x0000_0004</b> on page 25
<code>SYSTEM_PLIC_USER_INTERRUPT_A_INTERRUPT</code> <code>SYSTEM_PLIC_USER_INTERRUPT_B_INTERRUPT</code>	<b>Interrupts</b> on page 8

## SPI Master Peripheral Interface

The SPI master peripheral interface supports traditional 4-wire SPI as well as quad-SPI mode, which sends 4 data bits per clock cycle. When implementing the SPI peripheral in traditional dual-line mode, use the `data_0` ports as MOSI and and the `data_1` ports as MISO.

Use these parameters to reference the interface:

- SPI master 0—`SYSTEM_SPI_0_IO_APB`
- SPI master 1—`SYSTEM_SPI_1_IO_APB`



**Note:** The Opal SoC configured for the Xyloni Development Board as 2 SPI masters. Other variations only have 1.

**Table 15: SPI Master Ports**

Where  $n$  is 0 or 1

Port	Direction	Description
<code>system_spi__io_sclk_write</code>	Output	SPI SCK.
<code>system_spi__io_data_0_writeEnable</code>	Output	SPI output enable for data 0.
<code>system_spi__io_data_0_read</code>	Input	SPI input for data 0.
<code>system_spi__io_data_0_write</code>	Output	SPI output for data 0.
<code>system_spi__io_data_1_writeEnable</code>	Output	SPI output enable for data 1.
<code>system_spi__io_data_1_read</code>	Input	SPI input for data 1.
<code>system_spi__io_data_1_write</code>	Output	SPI output for data 1.
<code>system_spi__io_data_2_writeEnable</code>	Output	SPI output enable for data 2.
<code>system_spi__io_data_2_read</code>	Input	SPI input for data 2.
<code>system_spi__io_data_2_write</code>	Output	SPI output for data 2.
<code>system_spi__io_data_3_read</code>	Input	SPI input for data 3.
<code>system_spi__io_data_3_write</code>	Output	SPI output for data 3.
<code>system_spi__io_data_3_writeEnable</code>	Output	SPI output enable for data 3.
<code>system_spi__io_ss</code>	Output	SPI SS.

**Table 16: SPI Master Register Map**

Address Offset	Register Name	Privilege	Width
<code>0x0000_0000</code>	Cmd	Read/Write	32
<code>0x0000_0004</code>	RSP	Read	32
<code>0x0000_0008</code>	Config	Write	32
<code>0x0000_000C</code>	Interrupt	Read/Write	32
<code>0x0000_0020</code>	ClockDivider	Write	32
<code>0x0000_0024</code>	ssSetup	Write	32
<code>0x0000_0028</code>	ssHold	Write	32
<code>0x0000_002C</code>	ssDisable	Write	32
<code>0x0000_0030</code>	ssActiveHigh	Write	32

*Cmd Register: 0x0000\_0000*

31	12	11	10	9	8	7	0
Reserved			SS	RD	WR	data	

Bits	Field	Description	Privilege
0-7	data	FIFO data value transmit/receive.	Read/Write
8	WR	Write trigger.	Write
9	RD	Read trigger.	Write
10	Reserved	Reserved.	N/A
11	SS	SPI chip select.	Read/Write
12-31	Reserved	Reserved.	N/A

*RSP Register: 0x0000\_0004*

31	16	15	0
fifoOccupancy		fifoAvailability	

Bits	Field	Description	Privilege
0-15	fifoAvailability	FIFO Availability.	Read
16-32	fifoOccupancy	FIFO Occupancy.	Read

*Config Register: 0x0000\_0008*

31	2	1	0
Reserved		cpha	cpol

Bits	Field	Description	Privilege
0	cpol	Clock polarity setting.	Write
1	cpha	Clock phase setting.	Write
2-31	Reserved	Reserved.	N/A

*Interrupt Register: 0x0000\_000C*

31	10	9	8	7	2	1	0
Reserved				rsplnt	cmdlnt	Reserved	
						rsplntEnable	cmdlntEnable

Bits	Field	Description	Privilege
0	cmdlntEnable	Command FIFO empty interrupt enable.	Read/Write
1	rsplntEnable	Read FIFO not empty interrupt enable.	Read/Write
2-7	Reserved	Reserved.	N/A
8	cmdlnt	Command FIFO empty interrupt pending.	Read/Write
9	rsplnt	Read FIFO not empty interrupt pending.	Read/Write
10-31	Reserved	Reserved.	N/A

*clockDivider Register: 0x0000\_0020*

31	clockDivider	0
----	--------------	---

Bits	Field	Description	Privilege
0-31	clockDivider	SPI frequency = FCLK / (2 * clockDivider)	Write

*ssSetup Register: 0x0000\_0024*

31	ssSetup	0
----	---------	---

Bits	Field	Description	Privilege
0-31	ssSetup	Time between the chip select enable and the next byte.	Write

*ssHold Register: 0x0000\_0028*

31	ssHold	0
----	--------	---

Bits	Field	Description	Privilege
0-31	ssHold	Time between the last byte transmission and the chip select disable.	Write

*ssDisable Register: 0x0000\_002C*

31	0
ssDisable	

Bits	Field	Description	Privilege
0-31	ssDisable	Time between the chip select disable and the chip select enable.	Write

*ssActiveHigh Register: 0x0000\_0030*

31	0
ssActiveHigh	

Bits	Field	Description	Privilege
0-31	ssActiveHigh	These bits correspond to the hardware SPI chip select. 0: Chip select is active low. 1: Chip select is active high.	Write

## UART Peripheral Interface

The UART peripheral runs at 115200 baud and supports 8 data bits, no parity, and 1 stop bit. Use the `SYSTEM_UART_0_IO_APB` parameter to reference the interface.

*Table 17: UART Ports*

Port	Direction	Description
system_uart_0_io_txd	Output	UART 0 transmit.
system_uart_0_io_rxd	Input	UART 0 receive.

*Table 18: SPI Master Register Map*

Address Offset	Register Name	Privilege	Width
0x0000_0000	Data	Read/Write	32
0x0000_0004	Status	Read/Write	32
0x0000_0008	Clock divider	Read/Write	32
0x0000_000C	Config register	Read/Write	32

*Data Register: 0x0000\_0000*

31	0
data	

Bits	Field	Description	Privilege
0-31	data	Stores data that is transmitted or received.	Read/Write



*Status Register: 0x0000\_0004*

31	24	23	16	15	2	1	0	
readOccupancy			writeAvailability		Reserved		RXInterrupt	TXInterrupt

Bits	Field	Description	Privilege
0	TXInterrupt	Interrupt when TX FIFO is empty.	Read/Write
1	RXInterrupt	Interrupt when RX FIFO is not empty.	Read/Write
2-15	Reserved	Reserved.	N/A
16-23	writeAvailability	UART FIFO availability.	Read/Write
24-31	readOccupancy	UART FIFO occupancy.	Read/Write

*Clock Divider Register: 0x0000\_0008*

31	0
DividerFactor	

Bits	Field	Description	Privilege
0-31	DividerFactor	Divider factor for the UART baud rate. Baudrate = SystemClk/(Data Length * DividerFactor)	Read/Write

*Config Register: 0x0000\_000C*

31	17	16	15	9	8	7	0
Reserved			Stop	Reserved		Parity	DataLength

Bits	Field	Description	Privilege
0-7	DataLength	Data length.	Read/Write
8	Parity	Parity bit number.	Read/Write
9-15	Reserved	Reserved.	
16	Stop	Stop bit number.	Read/Write
17-32	Reserved	Reserved.	N/A

## Control and Status Registers

The following tables show the machine-level CSR implementation.

*Table 19: Machine Information Register*

Address	Register Name	Privilege	Description	Width
0xF14	mhartid	Read	Hardware thread ID.	32

*Table 20: Machine Trap Registers*

Address	Register Name	Privilege	Description	Width
0x300	mstatus	Read/Write	Machine status register.	13
0x304	mie	Read/Write	Machine interrupt enable register.	12
0x305	mtvec	Read/Write	Machine trap handler base address.	32

*Table 21: Machine Trap Handling Registers*

Address	Register Name	Privilege	Description	Width
0x341	mpec	Read/Write	Machine exception program counter.	32
0x342	mcause	Read	Machine trap cause.	32
0x343	mtval	Read	Machine bad address or instruction.	32
0x344	mip	Read/Write	Machine interrupt pending.	12

## Machine-Level ISA

### *Hart ID Register (mhartid): 0xF14*

The `mhartid` CSR is a 32-bit read-only register containing the integer ID of the hardware thread running the code. This register must be readable in any implementation. Hart IDs might not necessarily be numbered contiguously in a multiprocessor system, but at least one hart must have a hart ID of zero. Hart IDs must be unique.

31	0
Hart ID	

Bits	Field	Description	Privilege
0-31	Hart ID	Hardware thread ID.	Read

### Machine Status Register (*mstatus*): 0x300

The *mstatus* register is a 13-bit read/write register formatted. The *mstatus* register keeps track of and controls the hart's current operating state. Restricted views of the *mstatus* register appear as the *sstatus* and *ustatus* registers in the S-level and U-level ISAs, respectively.

12	11	10	9	8	7	6	5	4	3	2	1	0
MPP	Reserved			MPIE	Reserved			MIE	Reserved			

Bits	Field	Description	Privilege
0-2	Reserved	Reserved.	N/A
3	MIE	Machine interrupt enable register.	Read/Write
4-6	Reserved	Reserved.	N/A
7	MPIE	Machine previous interrupt enable.	Read/Write
8-10	Reserved	Reserved.	N/A
11-12	MPP	Machine Previous privilege mode.	Read/Write

### Machine Trap-Vector Base-Address Register (*mtvec*): 0x305

The *mtvec* register is a 32-bit read/write register that holds trap vector configuration, consisting of a vector base address (*base*) and a vector mode (*mode*).

31											2	1	0
base											mode		

Bits	Field	Description	Privilege
0-1	mode	Vector mode. 0: Direct. All exceptions set pc to BASE 1: Vectored. Asynchronous interrupts set pc to BASE + 4xcause ≥ 2: Reserved	Read/Write
2-31	base	Vector base address.	Read/Write

### Machine Interrupt Enable Register (*mie*): 0x304

The *mie* register is a 12-bit read/write register containing interrupt enable bits.

11	10	9	8	7	6	5	4	3	2	1	0
MEIE	Reserved			MTIE	Reserved			MSIE	Reserved		

Bits	Field	Description	Privilege
0-2	Reserved	Reserved.	N/A
3	MSIE	Machine software interrupt enable.	Read/Write
4-6	Reserved	Reserved.	N/A
7	MTIE	Machine timer interrupt enable.	Read
8-10	Reserved	Reserved.	N/A
11	MEIE	Machine external interrupt enable.	Read

### Machine Exception Program Counter (mepc): 0x341

mepc is a 32-bit read/write register. The low bit of mepc (mepc[0]) is always zero. On implementations that support only IALIGN=32, the two low bits (mepc[1:0]) are always zero.

31	0
mepc	

Bits	Field	Description	Privilege
0-31	mepc	Machine exception program counter.	Read/Write

### Machine Cause Register (mcause): 0x342

The mcause register is a 32-bit read-write register. When a trap is taken into M-mode, mcause is written with a code indicating the event that caused the trap. Otherwise, mcause is never written by the implementation, though it may be explicitly written by software.

31	30	0
Interrupt	Exception Code	

Bits	Field	Description	Privilege
0-30	Exception code	See <a href="#">Table 22: Machine Cause Register (mcause) Values after Trap</a> on page 28.	Read
31	Interrupt	mcause interrupt bit.	Read

**Table 22: Machine Cause Register (mcause) Values after Trap**

Interrupt	Exception Code	Description
1	0	Reserved.
1	1	Supervisor software interrupt.
1	2	Reserved.
1	3	Machine software interrupt.
1	4	User timer interrupt.
1	5	Supervisor timer interrupt.
1	6	Reserved.
1	7	Machine timer interrupt.
1	8	User external interrupt.
1	9	Supervisor external interrupt.
1	10	Reserved.
1	11	Machine external interrupt.
1	≥12	Reserved.
0	0	Instruction address misaligned.
0	1	Instruction access fault.
0	2	Illegal instruction.
0	3	Breakpoint.

Interrupt	Exception Code	Description
0	4	Load address misaligned.
0	5	Load access fault.
0	6	Store/AMO address misaligned.
0	7	Store/AMO access fault.
0	8	Reserved.
0	9	Reserved.
0	10	Reserved.
0	11	Environment call from M-mode.
0	12	Instruction page fault.
0	13	Load page fault.
0	14	Reserved.
0	15	Store/AMO page fault.
0	≥16	Reserved.

### Machine Trap Value Register (mtval): 0x343

The `mtval` register is a 32-bit register. When a trap is taken into M-mode, `mtval` is either set to zero or written with exception-specific information to assist software in handling the trap. Otherwise, `mtval` is never written by the implementation, though it may be explicitly written by software. The hardware platform will specify which exceptions must set `mtval` informatively and which may unconditionally set it to zero.

31	0
mtval	

Bits	Field	Description	Privilege
0-31	mtval	Machine trap value register bit.	Read/Write

### Machine Interrupt Pending Register (mip): 0x344

The `mip` register is a 12-bit read/write register containing information on pending interrupts.

11	10	9	8	7	6	5	4	3	2	1	0
MEIP	Reserved			MTIP	Reserved			MSIP	Reserved		

Bits	Field	Description	Privilege
0-2	Reserved	Reserved.	N/A
3	MSIP	Machine software interrupt pending.	Read/Write
4-6	Reserved	Reserved.	N/A
7	MTIP	Machine timer interrupt pending.	Read
8-10	Reserved	Reserved.	N/A
11	MEIP	Machine external interrupt pending.	Read

# Revision History

**Table 23: Revision History**

<b>Date</b>	<b>Version</b>	<b>Description</b>
December 2021	1.6	Updated resource numbers.
September 2021	1.5	The SoC minimum frequency changed to 20 MHz. (DOC-544)
July 2021	1.4	Updated for the Efinity v2021.1 release. Updated the SoC $f_{MAX}$ range. Updated the GPIO register descriptions. (DOC-475)
March 2021	1.3	Updated SoC operating frequency. Updated on-chip RAM size.
November 2020	1.2	Added UART register descriptions. Updated the address map to show parameters instead of address ranges. Updated to support Opal SoC variation for Xyloni Development Board.
August 2020	1.1	User peripheral address size changed to 64K. io_apbSlave_PADDR size changed to 15:0. Corrected typos.
June 2020	1.0	Initial release.