DIGITAL INDIA RISC-V (DIR-V) PROGRAM

THEJAS32 USER GUIDE

VEGA PROCESSOR

vega@cdac.in
www.vegaprocessors.in

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

THEJAS32 is a SoC based on VEGA ET1031 Microprocessor. It connects different peripherals like SPI, I2C, PWM, TIMER, GPIO etc. with the processor and provides a complete embedded system environment.

VEGA ET1031 is a 32-bit CPU core based on RISC-V Instruction Set Architecture designed and developed by Centre for Development of Advanced Computing. Its usage mainly centers on low power embedded applications. The core has a highly optimized 3-stage in-order pipeline with machine mode support and has the capability to boot FreeRTOS and run baremetal applications. The pipeline supports the RISC-V RV32IM extensions.

The following sections describes the THEJAS32 SoC and each peripheral in detail.

1.1 ACRONYMS &ABBREVIATIONS

APB	-	Advanced Peripheral Bus
CPU	-	Central Processing Unit
HDVL	-	Hardware Design and Verification Language
PC	-	Program Counter
SoC	-	System on Chip

1.2 DEFINITIONS

RISC-V - RISC-V is an open standard instruction set architecture (ISA) based on established Reduced Instruction Set Computer (RISC) principles. Unlike most other ISA designs, the RISC-V ISA is provided under open-source licenses that do not require fees to use.

1.3 REFERENCES

1. The RISC-V Instruction Set Manual Volume I: User-Level ISA
2. The RISC-V Instruction Set Manual Volume II: Privileged Architecture

2 THEJAS32 – DETAILS

The THEJAS32 is a low-power SoC for robust embedded applications such as Sensor Fusion, Smart Metering, Wearable electronics, IoT devices, etc. The SoC integrates the power of the VEGA ET1031 32 bit Microprocessor with various communication/ peripheral IPs in LQFP 128 package. THEJAS32 SoC includes VEGA ET1031 Microprocessor, 256KB internal SRAM, Three UARTs, Four SPIs, Three TIMERs, Eight PWMs, Three I2C interfaces, 32 GPIOs and External interrupts capable GPIOs (GPIO0-GPIO11). The THEAJS32 ASIC operates at a frequency of 100MHz. The block diagram of the SoC is given below.

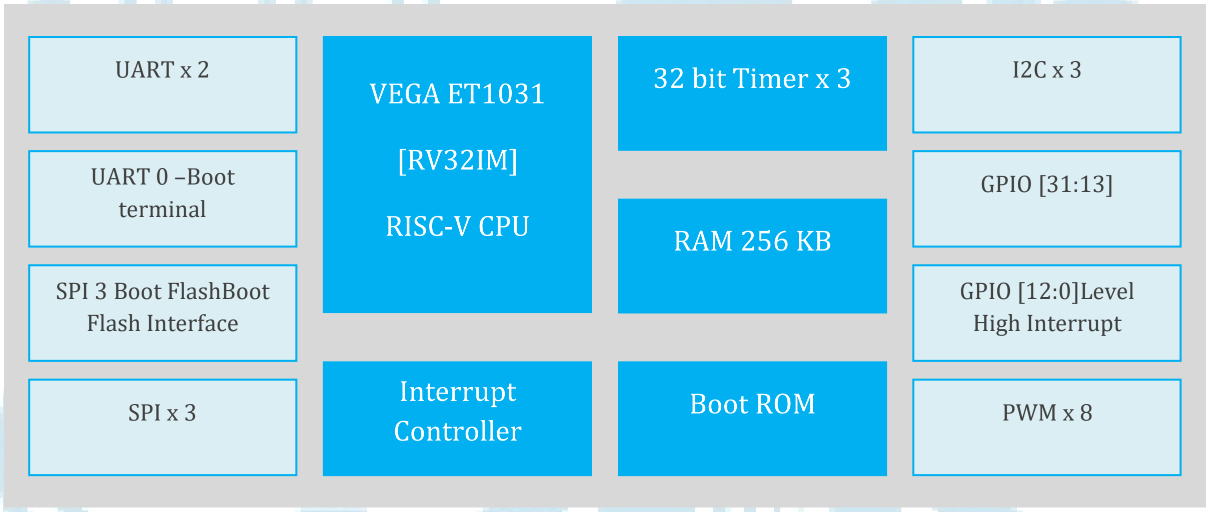


Figure 1-1: THEJAS32 Block diagram

THEJAS32 Key features:

- VEGA ET1031 Microprocessor
    - 32-bit RISC-V ISA
    - 3-stage in-order pipeline
    - Harvard architecture
    - High-performance multiply / divide unit
    - Low interrupt latency
    - Vectored interrupt support
- Clock frequency up to 100MHz
- Supply Voltages
    - Core Voltage - 1.2 V
    - IO Voltage - 3.3 V
- Up to 2MB Boot Flash via SPI
- 256 KB SRAM
- 4x SPI Communication Interfaces
- 3x I2C Communication Interfaces
- 3x UART Communication Interfaces
- 3x 32-bit TIMERs
- 8x PWMs
- 32x GPIOs
- 128-Pin LQFP Package
- External interrupt capable GPIOs

Applications

- Sensor fusion
- Smart Meter
- System supervisors
- Remote sensors
- Small IoT devices
- Wearable devices
- Motor drives
- Hand held devices
- GPS platforms
- Electronic education devices
- PLCs
- Inverters

- Printers & Scanners
- Industrial networking
- Legacy 8/16-bit applications

2.1 THEJAS32 – ADDRESS MAP

The table below provides the address map of the peripherals connected in THEJAS32.

Peripheral	Start Address	End Address	Interrupt Number
RAM	0x0020_0000	0x0023_FFFF	
UART0	0x1000_0100	0x1000_01FF	0
UART1	0x1000_0200	0x1000_02FF	1
UART2	0x1000_0300	0x1000_03FF	2
SPI0	0x1000_0600	0x1000_06FF	3
SPI1	0x1000_0700	0x1000_07FF	4
I2C0	0x1000_0800	0x1000_08FF	5
I2C1	0x1000_0900	0x1000_09FF	6
TIMER0	0x1000_0A00	0x1000_0A10	7
TIMER1	0x1000_0A14	0x1000_0A24	8
TIMER2	0x1000_0A28	0x1000_0A38	9
TIMERSINTSTATUS	0x1000_0AA0		
TIMERSEOI	0x1000_0AA4		
TIMERSRAWINT STATUS	0x1000_0A2A8		
I2C2	0x1000_1000	0x1000_1FFF	22
GPIO	0x1008_0000	0x101C_0000	10-21
SPI2	0x1020_0100	0x1020_01FF	23
SPI3	0x1020_0200	0x1020_02FF	
PWM	0x1040_0000	0x1040_00FF	24 - 31
PLIC	0x2001_0000	0x2001_FFFF	

Table 1: THEJAS32 Address Map

2.2 THEJAS32 PIN DESCRIPTION

Pin Diagram

LQFP128 – 128 Pin Low Profile Quad Flat Package (Top View)

THEJAS32

Pin #	Pin Name	Pin Description
1	GPIO1(3)	General purpose IO.
2	GPIO1(2)	General purpose IO.
3	PVSSIOC23	Ground reference for IO pins.
4	PVDDIO23	Positive supply for IO pins. Connect to 3.3V supply.
5	GPIO1(1)	General purpose IO.
6	GPIO1(0)	General purpose IO.
7	SPI3MOSI	SPI 3 Master Out Slave In.
8	PVDDC18	Positive supply for logic. Connect to 1.2V supply.
9	PVSSC18	Ground reference for logic.
10	SPI3MISO	SPI 3 Master In Slave Out.
11	SPI3CLK	SPI 3 Clock.
12	SPI3CSN	SPI 3 Chip Select.
13	PVSSIOC21	Ground reference for IO pins.
14	PVDDIO21	Positive supply for IO pins. Connect to 3.3V supply.
15	BOOT	Boot select.
16	PROCBT	Heart beat signal.
17	TEDTUPD	Connect to GND.
18	PVDDC17	Positive supply for logic. Connect to 1.2V supply.
19	PVSSC17	Ground reference for logic.
20	TSTCLK	Connect to GND through a 1K resistor.

21	TJTAGTDO	JTAG TDO. Left unconnected.
22	TJTAGTMS	JTAG TMS. Connect to GND through a 1K resistor.
23	TJTAGTDI	JTAG TDI. Connect to GND through a 1K resistor.
24	PVSSIOC19	Ground reference for IO pins.
25	PVDDIO19	Positive supply for IO pins. Connect to 3.3V supply.
26	PVDDC16	Positive supply for logic. Connect to 1.2V supply.
27	PVSSC16	Ground reference for logic.
28	TJTAGTCK	JTAG TCK. Connect to GND through a 1K resistor
29	TJTAGTRST	JTAG TRST. Connect to GND through a 1K resistor
30	TSTMODE	Test mode select. Connect to GND through a 1K resistor.
31	IIC2SDA	I2C 2 Serial Data.
32	IIC2SCL	I2C 2 Serial Clock.
33	IIC0SCL	I2C 0 Serial Clock.
34	IIC0SDA	I2C 0 Serial Data.
35	PVSSC14	Ground reference for logic.
36	PVDDC14	Positive supply for logic. Connect to 1.2V supply.
37	PVDDIO17	Positive supply for IO pins. Connect to 3.3V supply.
38	PVSSIOC17	Ground reference for IO pins.
39	SPI1CSN	SPI 1 Chip Select.
40	SPI1CLK	SPI 1 Clock.
41	SPI1MISO	SPI 1 Master In Slave Out.
42	SPI1MOSI	SPI 1 Master Out Slave In.
43	RSTIN	Reset.
44	CLKSYS	System Clock.
45	URT1SOUT	UART 1 Serial Out / Transmit.
46	PVDDIO15	Positive supply for IO pins. Connect to 3.3V supply.
47	PVSSIOC15	Ground reference for IO pins.
48	PVSSC12	Ground reference for logic.
49	PVDDC12	Positive supply for logic. Connect to 1.2V supply.
50	URT1SIN	UART 1 Serial In / Receive.
51	GPIO0(15)	General purpose IO.
52	GPIO0(14)	General purpose IO.
53	GPIO0(13)	General purpose IO.
54	GPIO0(12)	General purpose IO.
55	GPIO0(11)	General purpose IO.
56	PVSSC11	Ground reference for logic.
57	PVDDC11	Positive supply for logic. Connect to 1.2V supply.
58	GPIO0(10)	General purpose IO.
59	PVDDIO13	Positive supply for IO pins. Connect to 3.3V supply.

60	PVSSIOC13	Ground reference for IO pins.
61	GPIO0(9)	General purpose IO.
62	GPIO0(8)	General purpose IO.
63	GPIO0(7)	General purpose IO.
64	GPIO0(6)	General purpose IO.
65	GPIO0(5)	General purpose IO.
66	GPIO0(4)	General purpose IO.
67	PVSSC9	Ground reference for logic.
68	PVDDC9	Positive supply for logic. Connect to 1.2V supply.
69	PVDDIO11	Positive supply for IO pins. Connect to 3.3V supply.
70	PVSSIOC11	Ground reference for IO pins.
71	GPIO0(3)	General purpose IO.
72	GPIO0(2)	General purpose IO.
73	GPIO0(1)	General purpose IO.
74	GPIO0(0)	General purpose IO.
75	PWM(7)	Pulse Width Modulation.
76	PWM(6)	Pulse Width Modulation.
77	PWM(5)	Pulse Width Modulation.
78	PVSSC7	Ground reference for logic.
79	PVDDC7	Positive supply for logic. Connect to 1.2V supply.
80	PWM(4)	Pulse Width Modulation.
81	PWM(3)	Pulse Width Modulation.
82	PWM(2)	Pulse Width Modulation.
83	PVDDIO8	IO Power Supply VDD pin.
84	PVSSIOC8	Ground reference for IO pins.
85	PWM(1)	Pulse Width Modulation.
86	PWM(0)	Pulse Width Modulation.
87	SPI0MOSI	SPI 0 Master Out Slave In.
88	PVSSC6	Ground reference for logic.
89	PVDDC6	Positive supply for logic. Connect to 1.2V supply.
90	SPI0MISO	SPI 0 Master In Slave Out.
91	SPI0CLK	SPI 0 Clock.
92	SPI0CSN	SPI 0 Chip Select.
93	PVDDIO6	Positive supply for IO pins. Connect to 3.3V supply.
94	PVSSIOC6	Ground reference for IO pins.

95	IIC1SDA	I2C 1 Serial Data.
96	IIC1SCL	I2C 1 Serial Clock.
97	SPI2MOSI	SPI 2 Master Out Slave In.
98	SPI2MISO	SPI 2 Master In Slave Out.
99	PVDDC4	Positive supply for logic. Connect to 1.2V supply.
100	PVSSC4	Ground reference for logic.
101	SPI2CLK	SPI 2 Clock.
102	SPI2CSN	SPI 2 Chip Select.
103	PVSSIOC4	Ground reference for IO pins.
104	PVDDIO4	Positive supply for IO pins. Connect to 3.3V supply.
105	URT2SIN	UART 2 Serial In / Receive.
106	URT2SOUT	UART 2 Serial Out / Transmit.
107	URT0SIN	UART 0 Serial In / Receive.
108	URT0SOUT	UART 0 Serial Out / Transmit.
109	GPIO1(15)	General purpose IO.
110	GPIO1(14)	General purpose IO.
111	GPIO1(13)	General purpose IO.
112	PVDDC2	Positive supply for logic. Connect to 1.2V supply.
113	PVSSC2	Ground reference for logic.
114	PVSSIOC2	Ground reference for IO pins.
115	PVDDIO2	Positive supply for IO pins. Connect to 3.3V supply.
116	GPIO1(12)	General purpose IO.
117	GPIO1(11)	General purpose IO.
118	GPIO1(10)	General purpose IO.
119	GPIO1(9)	General purpose IO.
120	GPIO1(8)	General purpose IO.
121	GPIO1(7)	General purpose IO.
122	GPIO1(6)	General purpose IO.
123	PVSSIOC0	Ground reference for IO pins.
124	PVDDIO0	Positive supply for IO pins. Connect to 3.3V supply.
125	PVDDC0	Positive supply for logic. Connect to 1.2V supply.
126	PVSSC0	Ground reference for logic.
127	GPIO1(5)	General purpose IO.
128	GPIO1(4)	General purpose IO.

PVDDIO = 3.3V **PVDDC = 1.2V**

3 VEGA ET1031 – CPU

ET1031 is a 32-bit Three stage pipelined processor designed and developed by Centre for Development of Advanced Computing. ET1031 is well suited Microprocessor for low power embedded applications.

3.1 ET1031 FEATURES

- 32-bit RISC-V(RV32IM)Instruction Set Architecture
  - User level ISA Version 2.2
  - Privileged Architecture Version 1.10
- 3 stage In-Order pipeline
- Support for 55 instructions
- Support for privileged Instruction set
- 32-bit load/store architecture
- Pipelined Harvard architecture
- Byte, half-word and word memory access
- One cycle bubble for Jump and Branch Instructions
- Software Interrupt support
- Timer Interrupt support
- External Interrupt support
- Vectored Interrupts
- Processor Modes – Machine

3.2 INSTRUCTIONS SUPPORTED

ET1031 supports RISC-V Base integer and Multiply/Divide instruction set (RV32IM).

3.3 REGISTERS

3.3.1 INTEGER REGISTERS

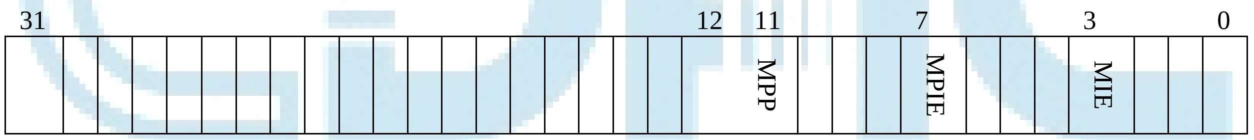
ET1031 has thirty-two, 32-bit wide registers and Register x0 is statically bound to 0.

Sl. no.	Instruction	Sl. no.	Instruction
1	LUI	29	BGE
2	AUIPC	30	BLTU
3	JAL	31	BGEU
4	JALR	32	LB
5	ADDI	33	LBU
6	SLTI	34	LH
7	SLTIU	35	LHU
8	XORI	36	LW
9	ORI	37	SB
10	ANDI	38	SH
11	SLLI	39	SW
12	SRLI	40	CSRRS
13	SRAI	41	CSRRC
14	ADD	42	CSRRSI
15	SUB	43	CSRRCI
16	SLT	44	CSRRW
17	SLTU	45	CSRRWI
18	XOR	46	FENCE.I
19	OR	47	FENCE
20	AND	48	MUL
21	SLL	49	MULH
22	SRL	50	MULHSU
23	SRA	51	MULHU
24	ECALL	52	DIV
25	MRET	53	DIVU
26	BEQ	54	REM
27	BNE	55	REMU
28	BLT		

3.3.2 CSR REGISTERS

Sl. nol	CSR name	CSR address	R/W	Description
1	misa	0x301	RO	Machine ISA Register (RV32IM), 0x40001100
2	mvendorid	0xF11	RO	Machine Vendor ID
3	marchid	0xF12	RO	Machine Architecture ID
4	mimpid	0xF13	RO	Machine Implementation ID
5	mhartid	0xF14	RO	Hart ID , Always zero
6	mstatus	0x300	RW	Machine Status
7	mtvec	0x305	RW	Machine Trap-Vector Base Address
8	mepc	0x341	RW	Machine Exception Program Counter
9	mtval	0x343	RW	Machine bad address or instruction
10	mip	0x344	RW	Interrupt Pending Register
11	mie	0x304	RW	Interrupt Enable Register
12	mcycle	0xB00 0xB80	RW	64-bit Machine cycle counter. 0xB00 – mcyclel (Lower 32-bit) 0xB80 – mcycleh (Upper 32-bit)
13	minstret	0xB02 0xB82	RW	64-bit Machine instructions-retired counter 0xB02- minstretl (Lower 32-bit) 0xB82- minstreth (Upper 32-bit)
14	mscratch	0x340	RW	Machine mode scratch register
15	mcause	0x342	RW	Machine Cause register

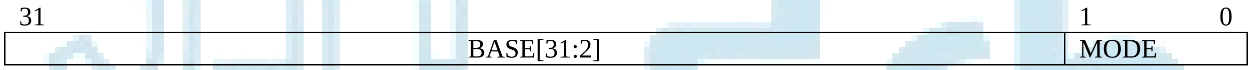
3.3.2.1 MACHINE STATUS REGISTER (MSTATUS)



Bit #	R/W	Name	Description
3	RW	MIE	Machine mode Interrupt Enable: Set this bit to enable the interrupt
7	RW	MPIE	Previous Interrupt Enable: When an exception is encountered, MPIE will be set to MIE. When the MRET instruction is executed, the value of MPIE will be restored to MIE.
12-11	R	MPP	11 and cannot be altered (read-only).

3.3.2.2 MACHINE TRAP-VECTOR BASE-ADDRESS REGISTER (MTVEC)

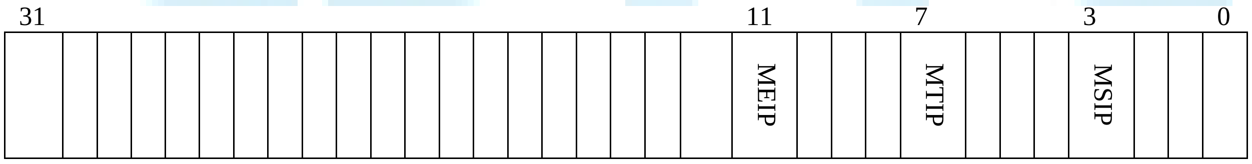
The MTVEC register is a 32-bit read/write register that holds trap vector configuration, consisting of a vector base address (BASE) and a vector mode (MODE). The value in the BASE field must always be aligned on a 4-byte boundary. When MODE=Direct, all traps into machine mode cause the PC to be set to the address in the BASE field. When MODE=Vectored, all synchronous exceptions into machine mode cause the PC to be set to the address in the BASE field, whereas interrupts cause the PC to be set to the address in the BASE field plus four times the interrupt cause number. For example, a machine-mode timer interrupt causes the PC to be set to BASE+0x1c



Bit #	R/W	Name	Description
1-0	RW	MODE	Vector mode ''00''- All exceptions set PC to BASE. ''01''- Asynchronous interrupts set PC to BASE+4×cause "1X"- RESERVED
31-2	RW	BASE	Trap vector Base address

3.3.2.3 MACHINE INTERRUPT REGISTERS (MIP)

The MIP register is a 32-bit read/write register containing information on pending interrupts.



Bit #	R/W	Name	Description
3	RW	MSIP	This read-only bit that indicates Machine software interrupt pending cleared by writing zero to this bits
7	R	MTIP	This bit that indicates Machine timer interrupt pending MTIP bit is read-only and is cleared by writing to the memory-mapped machine-mode timer compare register

11	R	MEIP	Machine External interrupt pending This read-only bit that indicates a machine-mode external interrupt is Pending. MEIP is set and cleared by a platform-specific interrupt controller.

3.3.2.4 MACHINE INTERRUPT REGISTERS (MIE)

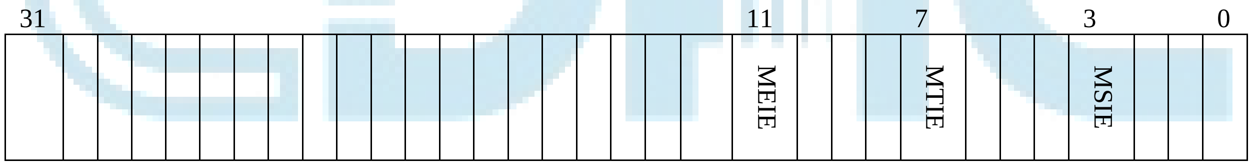
The MIE register is a 32-bit read/write register containing interrupt enable bits corresponding to MIP bits.

Bit #	R/W	Name	Description
3	RW	MSIE	Machine software interrupt Enable
7	RW	MTIE	Machine timer interrupt Enable
11	RW	MEIE	Machine External interrupt Enable

3.3.2.5 MACHINE CAUSE REGISTER (MCAUSE)

The mcause register is a 32-bit read-write register. When a trap is encountered, mcause is written with a code indicating the event that caused the trap. The Interrupt bit in the mcause register is set if the trap was caused by an interrupt. The Exception Code field contains a code identifying the last exception.

31	30 0
Interrupt	Exception Code



Interrupt	Exception Code	Description
1	3	Machine software interrupt
1	7	Machine timer interrupt
1	11	Machine external interrupt
1	>11	Reserved for future use
0	1	Reserved for future use
0	2	Illegal instruction
0	4	Reserved for future use
0	5	Reserved for future use
0	6	Reserved for future use

0	7	Reserved for future use
0	11	Environment call from M-mode
0	>11	Reserved for future use

3.3.2.6 **MACHINE TRAP VALUE (MTVAL) REGISTER**

When a trap is encountered mtval is either set to zero or written with exception-specific information to assist software in handling the trap. When a trap occurs, mtval is written with the faulting virtual address. On an illegal instruction trap, mtval may be written with faulting instruction. For other traps, mtval is set to zero

31	0
MTVAL	

3.3.2.7 **MACHINE EXCEPTION PROGRAM COUNTER (MEPC)**

31	0
MEPC	

When an exception is encountered, the current program counter is saved in MEPC, and the core jumps to the exception address. When an MRET instruction is executed, the value from MEPC replaces the current program counter. MEPC is written with the virtual address of the instruction that was interrupted or that encountered the exception

3.3.2.8 MACHINE SCRATCH REGISTER (MSCRATCH)

31	0
MSCRATCH	

The mscratch register is a 32-bit read/write register dedicated for use by machine mode. Typically, it is used to hold a pointer to a machine-mode hart-local context space and swapped with a user register upon entry to an M-mode trap handler.

3.4 EXCEPTIONS AND INTERRUPTS

ET1031 supports interrupts and exceptions.

3.4.1 INTERRUPTS

Interrupts can only be enabled/ disabled on a global basis and not individually. There is an event/interrupt controller outside of the core that performs masking and buffering of the interrupt lines. The global interrupt enable is done via the CSR register MSTATUS.

Multiple interrupts requests are assumed to be handled by event/interrupt controller.

3.4.2 EXCEPTIONS

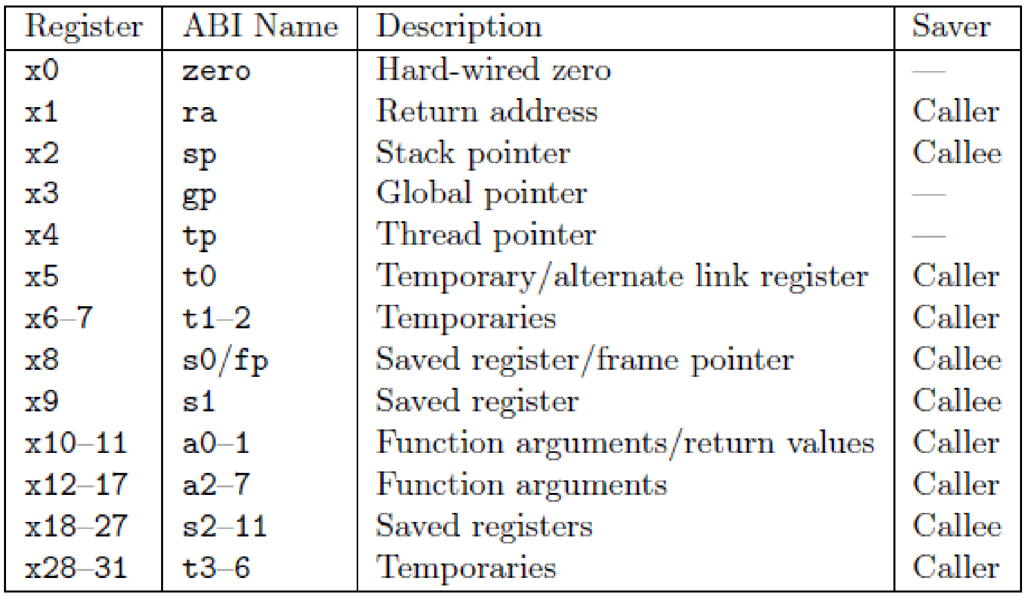
The illegal instruction exception and ECALL instruction exceptions cannot be disabled and are always active.

3.4.3 HANDLING

ET1031 does support nested interrupt/exception handling. Exceptions inside interrupt/exception handlers cause another exception, thus exceptions during the critical part of the exception handlers, i.e. before having saved the MEPC and MSTATUS register, will cause those register to be overwritten. Interrupts during interrupt/exception handlers are disabled by default, but can be explicitly enabled if desired.

Upon executing an MRET instruction, the core jumps to the program counter saved in the CSR register MEPC and restores the MPIE value of the register MSTATUS to MIE. When entering an interrupt/exception handler, the core sets MEPC to the current program counter and saves the current value of MIE in MPIE of the MSTATUS register.

3.5 CALLING CONVENTION



Register	ABI Name	Description	Saver
x0	zero	Hard-wired zero	—
x1	ra	Return address	Caller
x2	sp	Stack pointer	Callee
x3	gp	Global pointer	—
x4	tp	Thread pointer	—
x5	t0	Temporary/alternate link register	Caller
x6–7	t1–2	Temporaries	Caller
x8	s0/fp	Saved register/frame pointer	Callee
x9	s1	Saved register	Callee
x10–11	a0–1	Function arguments/return values	Caller
x12–17	a2–7	Function arguments	Caller
x18–27	s2–11	Saved registers	Callee
x28–31	t3–6	Temporaries	Caller

4 IIC MASTER CONTROLLER

The IIC-bus protocol uses two wires, serial data (SDA) and serial clock (SCL), which carry information between the devices connected to the bus. Each device is recognized by a unique address and can operate as either a transmitter or receiver, depending on the function of the device. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is a device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave. Data transfer on the SDA bus line is always a byte long. To start a transfer the initiating master issues the clock and sets a START sequence on the bus as shown in figure. The data is shifted out with MSB first and for every byte an acknowledge signal is expected from the slave/master in transmit/receive mode. The acknowledge takes place after every byte. The acknowledge bit allows the receiver to signal the transmitter that the byte was successfully received, and another byte may be sent. The master generates all clock pulses, including the acknowledge ninth clock pulse. The Acknowledge signal (ACK) is defined as follows: the transmitter releases the SDA line during the acknowledge clock pulse so the receiver can pull the SDA line LOW and it remains stable LOW during the HIGH period of this clock pulse. When SDA remains HIGH during this ninth clock pulse, this is defined as the Not Acknowledge signal (NACK). The master can then generate either a STOP condition to abort the transfer, or a repeated START condition to start a new transfer.

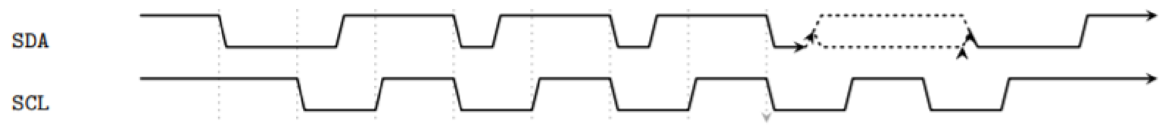


Figure 4-1:IIC data transfers

Some of the features of the IIC Master controller includes:

- Configurable SCK frequency to support different IIC standards.

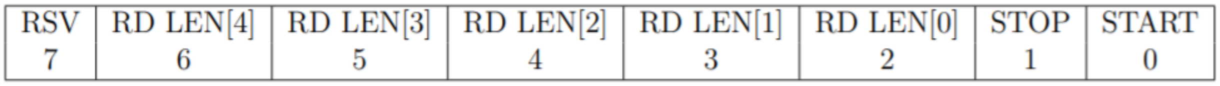
- Separate Receive/Transmit FIFO for easy read/write from the Bus.
- Fully configurable interrupt for Receive/Transmit operation.
- Auto STOP generation while detecting NACK condition on the bus.
- Simple start and stop procedure to animate repeated start condition on the bus.

4.1 REGISTER DESCRIPTIONS

This portion provides details of the registers used in IIC.

4.1.1 CONTROL REGISTER(CR) [WRITE ONLY]

ADDRESS - 0X00



RSV	RD LEN[4]	RD LEN[3]	RD LEN[2]	RD LEN[1]	RD LEN[0]	STOP	START
7	6	5	4	3	2	1	0

Figure 4-2:Bit mappings of CR

- **START** - Set to '1' to send a start sequence on the bus. This bit is not cleared in hardware and need not be cleared in software.
- **STOP** - Set to '1' to send a stop sequence on the bus. This bit is not cleared in hardware and need not be cleared in software.
- **RD LEN [4:0]** - Set required read length in this field. A zero value would indicate a write operation. This should be set along with the START bit only.
- **RSV** - reserved fields.

Reading from this register will return undefined data.

NOTE: Even though read length is 5 bits which signifies 31 bytes can be read, the RxFF is only 16 byte deep. Care must be taken in the software to read maximum 16 bytes during a single read transfer i.e., read length can be maximum 16(b"10000"). For example, to read 5 bytes set read length as 5 (b"00101") and to read 1-byte set read length as 1 (b"00001").

4.1.2 STATUS REGISTER 0 (SR0) [READ ONLY]

ADDRESS - 0X01

RxC	RxFF empty	RxFF full	TxC	TxFF empty	TxFF full	STPS	STAS
7	6	5	4	3	2	1	0

Figure 4-3:Bit mappings of SR0

- **STAS** - Start condition sent on the bus, asserted HIGH after a successful start has been initiated. Cleared (LOW) when stop condition is sent on the bus.
- **STPS** - Stop condition sent on the bus, asserted HIGH after a successful stop condition has been sent on the bus. Also indicates bus is free. Cleared (LOW) when next start command is received.
- **TxFF full** - Transmit FIFO full - HIGH when Transmit FIFO is full else LOW.
- **TxFF empty** - Transmit FIFO empty - HIGH when Transmit FIFO is empty else LOW.
- **TxC** - Transmit complete - Asserted HIGH once every byte written in TxFF is transmitted else LOW.
- **RxFF full** - Receive FIFO full - HIGH when receive FIFO is full else LOW.
- **RxFF empty** - Receive FIFO empty - HIGH when Receive FIFO is empty else LOW.
- **RxC -** Receive complete flag - asserted HIGH when the requested read is complete else LOW.
- Writing to this register has no effect.

NOTE: Between a STAS and STPS some active transaction is being done on the IIC bus. This can be used in conjunction with RxFF/TxFF flags to signify that bus is busy.

4.1.3 STATUS REGISTER 1 (SR1) [READ ONLY]

ADDRESS - 0X02

RSV 7	RSV 6	RSV 5	RSV 4	RSV 3	RX INTR 2	TX INTR 1	NACK 0

Figure 4-4: Bit mappings of SR1

- **NACK** – Negative acknowledge received. This can be due to slave not able to accept write transfer or not able to service read request. The controller automatically sends the STOP condition on the bus and also sets the STPS bit in Status register 0. The programmer should read this flag on every TxC check during write operation and ensure that no NACK is received. If a NACK is received, then the programmer should discard the transfer and start again the entire sequence (including sending START sequence). This bit is auto cleared on next start sequence.
- The data in the Transmit FIFO is stale in the event of NACK and should be cleared in software. The dedicated register TXCLR can be used to clear the Transmit FIFO data in event of a NACK.
- **TX INTR** – Transmit interrupt - an active HIGH interrupt is generated signifying that the TX FIFO can accept at least one byte of DATA (i.e., not full). This can be disabled using the GIE or TXI EN bits in the Interrupt enable register. During operation, this interrupt keeps on occurring until there is no space in TxFF.
- **RX INTR** – Receive interrupt - an active HIGH interrupt is generated for every byte received in the RX FIFO. This signifies that there is at least one byte to be read in the RX FIFO, this interrupt is cleared when the RX FIFO is empty.
- Writing to this register has no effect.

4.1.4 INTERRUPT ENABLE REGISTER (IER) [READ/WRITE]

ADDRESS - 0X03

RSV	RSV	RSV	RSV	RSV	RXI EN	TXI EN	GIE
7	6	5	4	3	2	1	0

Figure 4-5: Bit mappings of IER

- **GIE** – Global interrupt enable – enables transmit and receive interrupt. This bit should be set to ’1’ by software for any interrupt operations.
- **TXI EN** – Transmit interrupt enable - Writing ’1’ enables the transmit interrupt and ’0’ disables it. A Transmit interrupt is generated whenever there is space (at least one byte) in TxFF.
- **RXI EN** – Receive interrupt enable - Writing ’1’ enables the receive interrupt and ’0’ disables it. A Receive interrupt is generated whenever there is some data in RxFF to be read.
- GIE must be enabled prior to using TXI EN or RXI EN

4.1.5 TX DATA FIFO (TXFF) [WRITE ONLY]

ADDRESS - 0X04

TXD[7]	TXD[6]	TXD[5]	TXD[4]	TXD[3]	TXD[2]	TXD[1]	TXD[0]
7	6	5	4	3	2	1	0

Figure 4-6: Bit mappings of TXFF

- For a write transfer the data/address to be transmitted is written into this register. The data FIFO is 16 entry deep. Care must be taken by the software to write the TxFF after checking the TxFF flags from the status register.
- Reading this register would return zero.

4.1.6 RX DATA FIFO (RXFF) [READ ONLY]

ADDRESS - 0X05

RXD[7]	RXD[6]	RXD[5]	RXD[4]	RXD[3]	RXD[2]	RXD[1]	RXD[0]
7	6	5	4	3	2	1	0

Figure 4-7: Bit mappings of RXFF

- For a read transfer, the data requested is available in this FIFO. The data FIFO is 16 entry deep. Care must be taken by the software to read the RxFF after checking the RxFF flags from the status register.
- Writing to this register has no effect.

4.1.7 CLOCK PERIOD HALF REGISTER LOW (CHL) [READ/WRITE]

ADDRESS - 0X06

- This register is used to set the lower byte of half time period of IIC clock frequency.

4.1.8 CLOCK PERIOD HALF REGISTER HIGH (CHH) [READ/WRITE]

ADDRESS - 0X07

- This register is used to set the higher byte of half time period of IIC clock frequency.

4.1.9 CLOCK PERIOD HALF HALF REGISTER LOW (CHHL) [READ/WRITE]

ADDRESS - 0X08

- This register is used to set the lower byte of one quarter time period of IIC clock frequency.

4.1.10 CLOCK PERIOD HALF HALF REGISTER HIGH (CHHH) [READ/WRITE]

ADDRESS - 0X09

- This register is used to set the higher byte of one quarter time period of IIC clock frequency.

All the timing registers have been internally set to a default value corresponding to 25MHz system clock and 100KHz SCL frequency.

4.1.11 TX FIFO CLEAR REGISTER (TXCLR) [WRITE ONLY]

ADDRESS - 0X0A

- Writing any value into this register would result in Transmit FIFO being cleared and all the data in the TX FIFO will be lost. This can be used to clear out the TX FIFO in case of NACK received. The Transmit FIFO empty flag will be set to indicate a successfully clear.
- Reading this register would return zero.

4.2 CALCULATION

# CHL = lower byte of (system frequency/ (2 ∗ IIC clock frequency))

# CHH = higher byte of (system frequency/ (2 ∗ IIC clock frequency))

# CHHL = lower byte of (system frequency/ (4 ∗ IIC clock frequency))

# CHHH = higher byte of (system frequency/ (4 ∗ IIC clock frequency))

Example: For 100KHz mode of operation if the system frequency is

25Mhz, corresponding values of the timing registers are:

# CHL = lower byte of ((25 ∗ 1000000)/ (2 ∗ 100 ∗ 1000))

CHL = 125d

# CHH = higher byte of ((25 ∗ 1000000)/ (2 ∗ 100 ∗ 1000))

CHH = 0d

# CHHL = lower byte of ((25 ∗ 1000000)/ (4 ∗ 100 ∗ 1000))

CHHL = 62d

# CHHH = higher byte of ((25 ∗ 1000000)/ (4 ∗ 100 ∗ 1000))

CHHH = 0d

4.3 TIMING SPECIFICATIONS

- **$T_{SU;STA}$** : Setup time for start condition. The SCL must be high for this much time before SDA goes low. This is internally set to IIC clock half period.
- **$T_{HD;STA}$** : Hold time for start condition. The SCL must be high for this much time after SDA goes low. This is internally set to IIC clock half period. This is depicted as T1 in the figure.
- **$T_{SU;DAT}$** : Setup time for data before active SCL clock. This is internally set to IIC clock quarter period. This is depicted as T2 in the figure.

- **$T_{HD;DAT}$** : Hold constraint for data after active SCL clock. This is internally set to IIC clock quarter period. This is depicted as T3 in the figure.
- **$T_{VD;DAT}$** : Maximum data valid time after SCL clock. This is used to ensure that next data must change during and SCL low phase. This is internally set to IIC clock quarter period. This is same as $T_{HD;DAT}$.
- **$T_{SU;STO}$** : Setup time for stop condition, minimum time SCL should be high before SDA goes high. This is internally set to IIC clock half period. This is depicted as T4 in the figure.
- **T_{BUF}** : Bus free time between a stop and start condition. This is set internally to one half clock period. This is depicted as T5 in the figure.

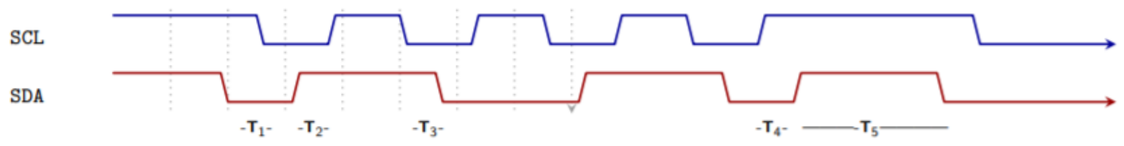


Figure 4-8: IIC Timing Specifications

4.4 SOFTWARE ENVIRONMENT

4.4.1 WRITE OPERATION

Write operation steps.

1. Check for Tx FIFO empty (no data in FIFO), if not empty, clear by writing into TXCLR register.

2. Set CHL, CHH, CHHL, CHHH values.

3. Set start bit in control register by writing 0x01 to Control register (address - 0x00).

4. Enable interrupt by writing into IE register (address - 0x03). (optional)

5. Write data in Tx FIFO (address - 0x04). The address of slave (slave address(7 bit) + R̃/W) is written first.

6. Continue writing data into Tx data safely while checking the Tx full flag in status register 0(3rd bit).

7. Check if any NACK received from status register 1(address -0x02). If NACK received redo the entire sequence. In event of a NACK the controller automatically sends the STOP sequence on the bus. The Transmit complete TxC and stop sent (STPS) flag will also be set. The software should repeat the entire sequence from sending START sequence.

8. Set stop bit in control register by writing 0x02 in CR (address - 0x00).

4.4.2 READ OPERATION

The controller is capable of reading a maximum of 16 bytes from a single read operation. For reads greater than 16 bytes the software should repeat the read sequence with each read count set to a maximum of 16bytes. The read procedure for each 16bytes read may or may not involve sending the address byte (depends on the device specification). During a read operation the ACK is provided by the controller on every byte read except the last byte.

Read operation steps:

1. Check for Tx FIFO empty (no data in FIFO), if not empty, clear by writing into TXCLR register.

2. Set CHL, CHH, CHHL, CHHH values.

3. Set start bit and read length in control register (address - 0x00).

4. Enable interrupt by writing into IE register (address - 0x03, optional).

5. Write slave address (with read bit) in TX register (address 0x04).

- Slave address (MSB 7 bits) with LSB bit low to signify read.

6. Wait for Tx complete.

7. Check NACK, if NACK wait for stop sent flag to be set and repeat the sequence from Step 1.

8. If ACK wait for RX complete.

9. Read data from RX register (address - 0x05) till RX FIFO is empty.

10. Wait for stop sent flag to be set. (This is to ensure that the software doesn't issue another start transaction while the bus is sending the start sequence).

4.4.3 **INTERRUPT**

The IIC master controller provides a single interrupt for both Transmission and Reception operation. The interrupt has to be enabled using the IER register. The controller interrupts in the following cases:

4.4.3.1 **TX INTERRUPT**

Tx Interrupt is generated:

1. When the controller has transmitted every byte in the TX FIFO i.e., TX FIFO is not full.

2. During start up after reset.

3. Can be cleared by using GIE or TXI EN in the IER register.

4.4.3.2 **RX INTERRUPT**

Rx Interrupt is generated:

1. When there exists at least one byte in the RX FIFO (this signifies that there is some valid data in the RX FIFO).

2. Reading the Rx FIFO till RX FIFO becomes empty clears this interrupt.

4.4.4 READ WRITE OPERATION

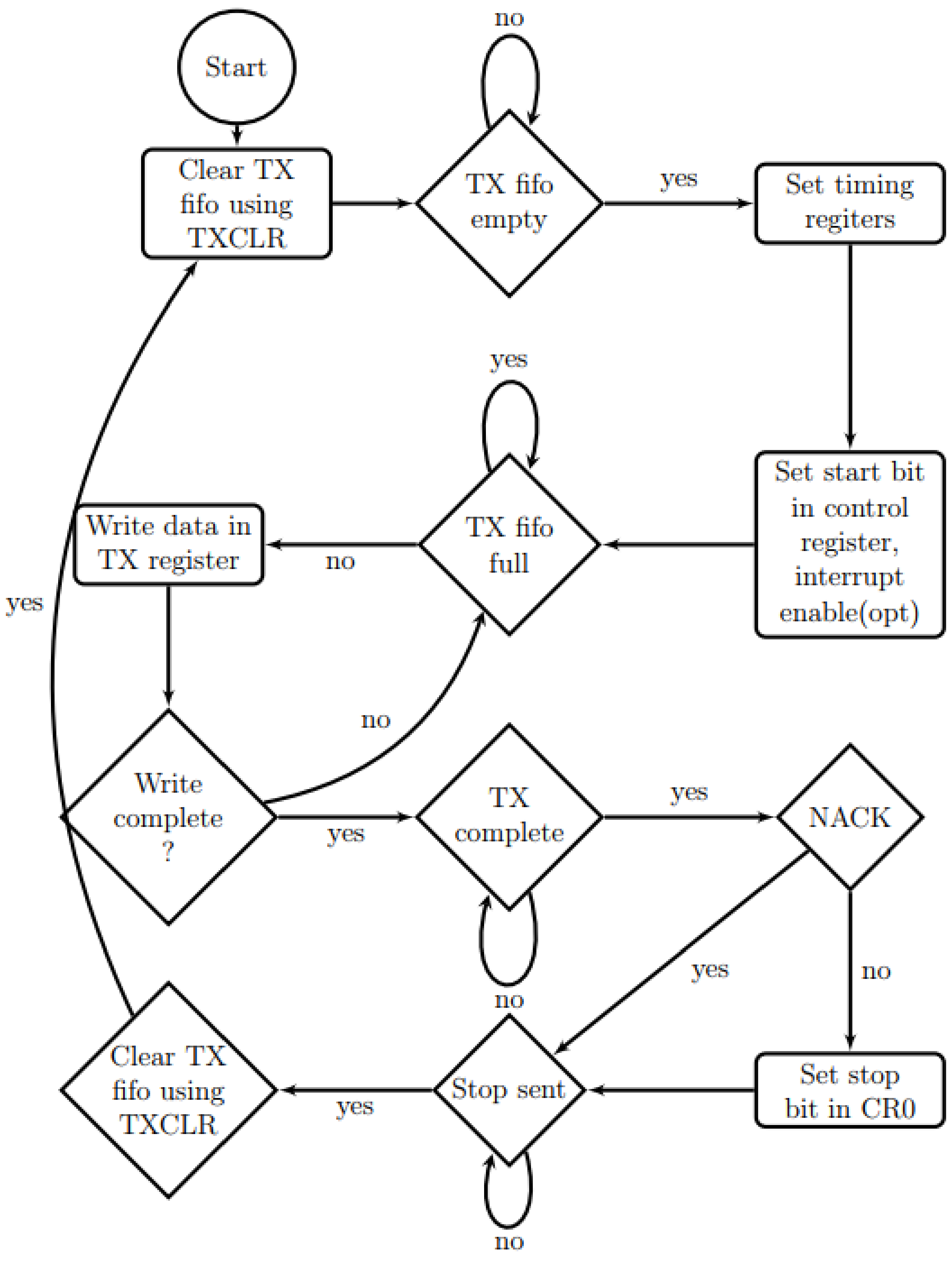


Figure 4-9: Write Operation

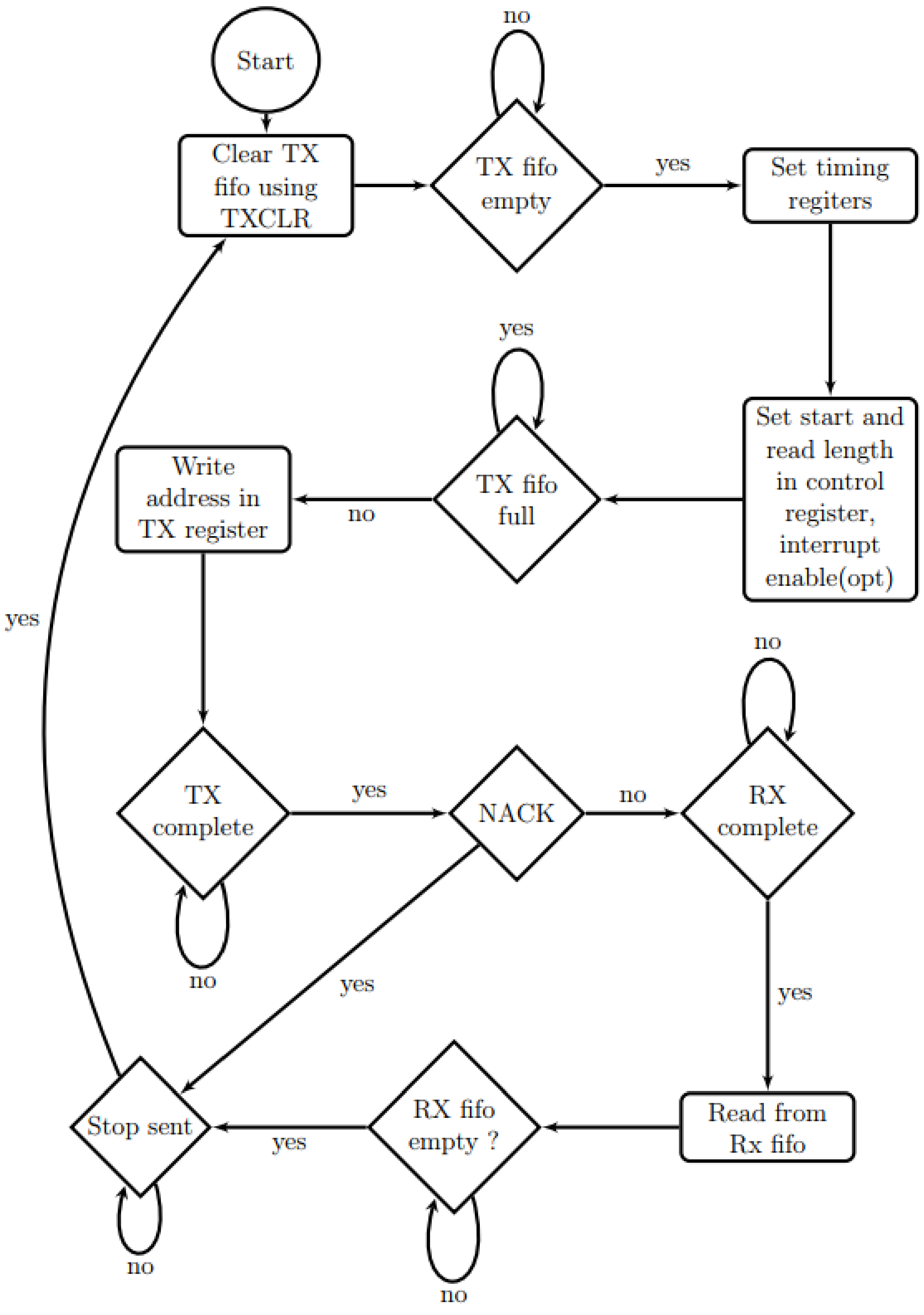


Figure 4-10: Read Operation

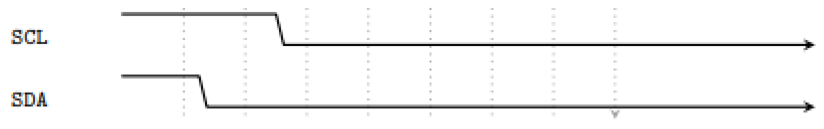


Figure 4-11: Start Condition

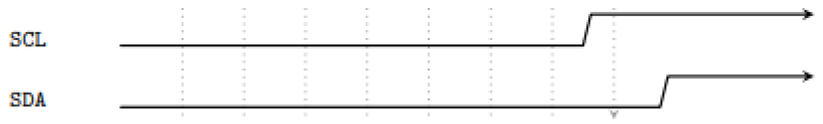


Figure 4-12: Stop Condition

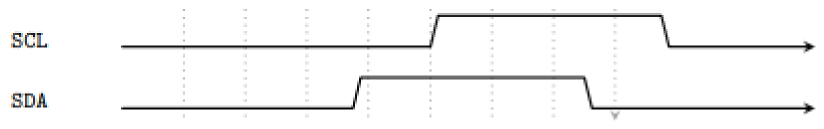


Figure 4-13: Repeated start condition

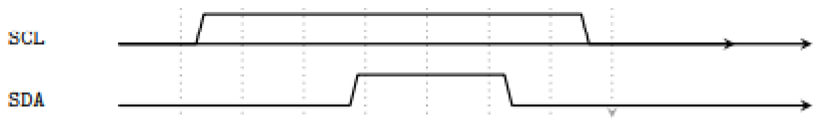


Figure 4-14: Repeated Start condition using start and stop condition

5 GPIO - GENERAL PURPOSE INPUT/OUTPUT CONTROLLER

The GPIO provides Sixteen dedicated programmable general-purpose pins that can be configured as either inputs or outputs. When configured as an output, an internal register can be written to control the state driven on the output pin. When configured as an input, the state of the input pins can be detected by reading the state of an internal register. The GPIO has the provision of bit masking in both read and write operations.

At the system reset, GPIO lines default to input.

5.1 FEATURES

- Sixteen individually programmable Input/Output pins
- Provision for bit masking in both read and write operations through address lines

5.2 ARCHITECTURE OVERVIEW

The GPIO provides sixteen programmable inputs or outputs that are controlled through software. In software control mode, data and control for these lines are provided by a Data Register and a Data Direction Register. On reads, the data register contains the current status of the GPIO pins, whether they are configured as input or output. The Data Direction Register is used to control the enable signal of the I/O pad.

The GPIODIR register serves the purpose of configuring the external I/O pads as input or output depending on its setting.

The address [15:0] bit serves as the WRITE mask bit for GPOUT [15:0] register bits in the respective order. The GPOUT register gets updated with write data [15:0] depending on the status of address [15:0] bits. Address bits which correspond to logic ‘1’ affects the

GPOUT register values. The remaining bits of GPOUT register will be unchanged. The GPOUT register is connected to the GPOUT port of the GPIO Controller.

It should be noted that the content of the GPOUT register doesn't depend on the status of GPIODIR register; instead the value in the GPOUT register gets updated based on the logic state of address [15:0] bits for each valid write transaction. For each write, only those bits of GPOUT corresponding to the logic '1' value of address [15:0] will get updated with the write data bits.

For GPIO READ operation, the status of the GPIO input pins are interfaced to master through read data [15:0] bits. The GPIO data register [15:0] bits reflect the logic state of those GPIO input pins, for which the corresponding address [15:0] bits are at logic '1'. The remaining GPIO data register [15:0] bits will be at logic '0', which is the default value. The GPOUT register will not be affected by the READ operation.

5.3 REGISTER DEFINITION

The table below lists the registers in GPIO.

Name	Address (ADDR[3:0])	Type	Default Value	Description
GPIODIR	1000	R/W	32'b0	Data Direction Register. 0 indicates input 1 indicates output
GPIODATA	0000-1111	R/W	32'b0	GPIO data register

Table 2: Register definition

GPIODIR register should be configured for input/output direction setting.

5.4 FUNCTIONAL DESCRIPTION

5.4.1 GPIO WRITE OPERATION

In WRITE mode, the address [16] should be necessarily set to ‘0’. The write data [15:0] will be written into the GPOUT register depending on address [15:0] bits. In order to write into the GPOUT register; the corresponding bits in the ‘mask bits’ resulting from the address [15:0] bits must be high. Otherwise the bit values in GPOUT register remains unchanged during the WRITE.

During a write, if the address bit associated with that data bit is HIGH; the value of the GPOUT register is altered. If it is LOW, it is left unchanged.

For example:

Writing to GPIO address 0x0098 (address [15:0]) the value 0’x AAAA

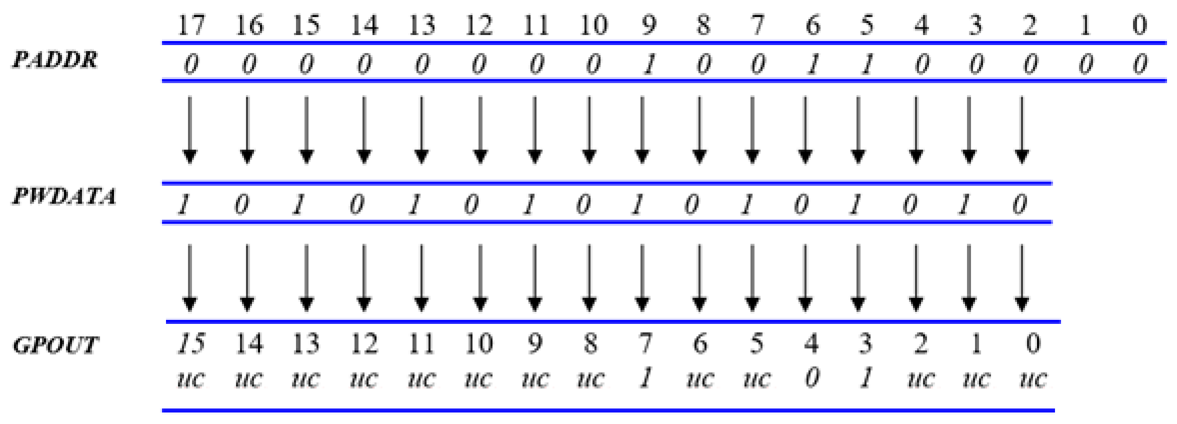


Figure 5-1: GPIO write

5.4.2 GPIO READ OPERATION

In READ mode, the address [16] should be necessarily set to ‘0’. The GPIN [15:0] will be read depending on the status of address [15:0] bits. In order to read from the GPOUT register; the corresponding bits in the ‘mask bits’ resulting from the address [15:0] bits must be high. Otherwise the bit values in read data [15:0] remain ‘0’ during the READ.

During a read, if the address bit associated with that data bit is HIGH; the value of the read data [15:0] bits is altered. If it is LOW, the associated read data bit will be at logic '0'.

For example:

Reading from GPIO address 0x0098 (address [15:0]) the value 0'x 0505

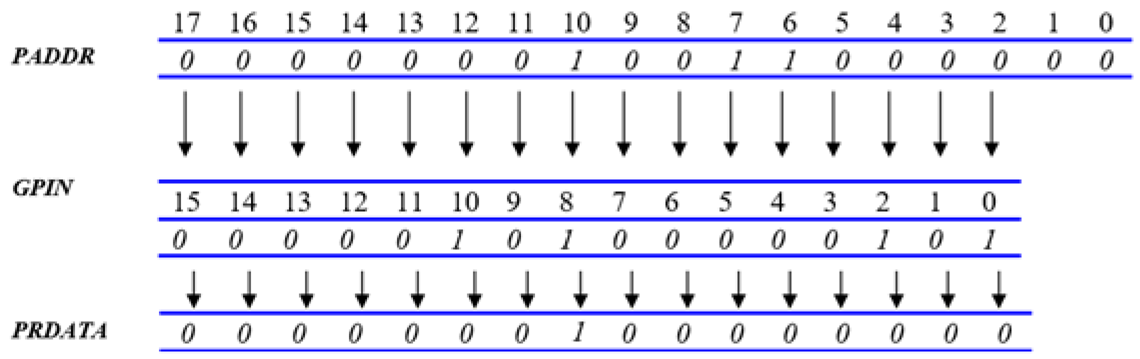


Figure 5-2: GPIO read

6 PWM - PULSE WIDTH MODULATOR

Pulse Width Modulation (PWM) controller provides eight individually programmable PWM Outputs.

6.1 FEATURES

- 32-bit address width and data width.
- PWM output channels.
- Separate interrupt enable/disable for each channel.
- Fully programmable, the shape of the output waveform can be easily modified.
- 32-bit Period counter and ON/OFF counter controls the shape of the PWM signal
- Ability to generate right aligned and left aligned PWM signals.
- Support for one shot and continuous mode of operation.
- Power saving facility by switching off each channel via special control bits.
- Status recording to facilitate debug. After configuring PWM channel, if PWM output is not being generated then programmer can check whether PWM is running or not by checking status bit.
- Interrupt Pending status for each channel.

6.2 PROCESSING

The register block of PWM Controller (PWMC) consists of a global control register, control register, status register, period register and ON/OFF register. Control and status register contains all the controls and status associated with all the PWM output channels.

Each register is 32-bit wide. Global control register provides global enables for PWM output channels and PWM interrupt signals.

PWMC core generates PWM signal based on the period counter and ON/OFF time counter. Period counter decides the PWM cycle duration and ON/OFF counter decides the duration of first phase in a PWM cycle, either ON (Left Aligned) or OFF (Right Aligned). Pulse alignment control configures left or right aligned PWM generation.

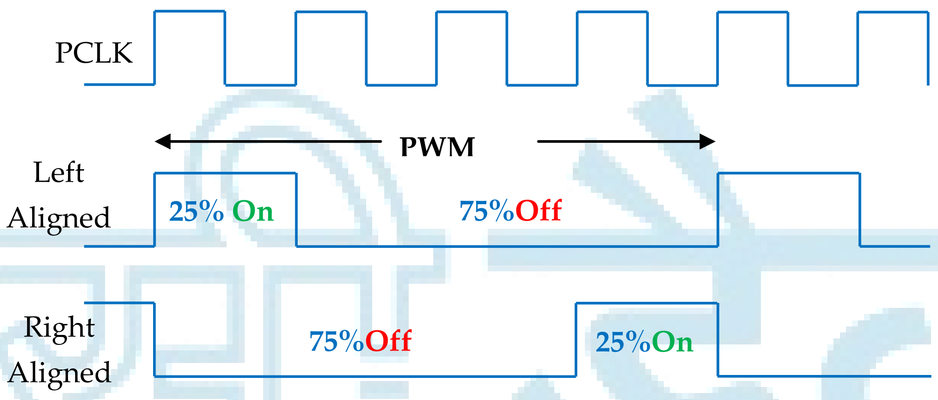


Figure 6-1:Left Aligned and Right Aligned PWM signals at 25% duty cycle

Address Range	Description
0x00 – 0x0C	PWMC Channel 1 Registers
0x10 – 0x1C	PWMC Channel 2 Registers
0x20 – 0x2C	PWMC Channel 3 Registers
0x30 – 0x3C	PWMC Channel 4 Registers
0x40 – 0x4C	PWMC Channel 5 Registers
0x50 – 0x5C	PWMC Channel 6 Registers
0x60 – 0x6C	PWMC Channel 7 Registers
0x70 – 0x7C	PWMC Channel 8 Registers
0x80	PWMC Global Control Register (GCR)

Table 3: PWMC Address Summary

When multiple PWM outputs are enabled address map for next PWM channel is obtained by adding an offset of 0x10. Control register, status register, period register and ON/OFF register are associated with each PWM channel. PWMC Global Control Register (GCR) is a single global control register used to control all PWM generators in the Core.

PWMC Generation logic always generates PWM signal based on shadow register block. Shadow register block contain Shadow Period Register, Shadow ON/OFF Register, Shadow Control Register and Shadow global control register. Programmer can initiate shadow register loading in two ways, (1) Write to global control register and (2) write to control register.

Address Offset of PWMC Channel Number 1 to 8	PWMC Channel Number (N)							
	1	2	3	4	5	6	7	8
PWMC_N_CR	0x00	0x10	0x20	0x30	0x40	0x50	0x60	0x70
PWMC_N_SR	0x04	0x14	0x24	0x34	0x44	0x54	0x64	0x74
PWMC_N_PR	0x08	0x18	0x28	0x38	0x48	0x58	0x68	0x78
PWMC_N_ON/OFF	0x0C	0x1C	0x2C	0x3C	0x4C	0x5C	0x6C	0x7C
PWMC_GCR	0x80							

Table 4: PWMC Full Address Map

6.3 MODES OF OPERATION

PWMC has three modes of operation, namely Idle, One shot and Continuous. Operating mode can be configured using 'Mode' bits in the Control register.

Idle mode: In this mode, PWMC does not generate PWM signal at the output port. PWM output will be driven high or low based on the output control (OPC) bit in the control register. Interrupt Control bits (global and local) and other control bits have no effect in this mode. PWMC status shows idle in the status register.

One short mode: In this mode, PWMC generates PWM signal at the output port for a specified number of PWM cycles defined by repeat count field in the control register.

Global PWM Enable bit in PWMC Global Control Register should also be set for generating PWM signal. Once Global PWM Enable bit is programmed, write to PWMC Period register, ON/OFF Register and Control Register have no effect on generated PWM signal. Those registers should be programmed prior to Global PWM Enable bit. After generating repeat (count + 1) PWM cycles, output goes to high or low based on the output control (OPC) in the control register. If global interrupt control bit in PWMC Global Control Register and local interrupt control bit in PWMC Control Register are enabled, then interrupt pending (IP) bit in PWMC Status Register goes high after repeat (count + 1) cycles. Interrupt can be cleared by reading channel specific PWMC Status Register. PWM in one shot mode stops after generating repeat (count + 1) cycles. It can also be stopped immediately by clearing Global PWM Enable bit in PWMC Global Control Register. If the programmer is setting Global PWM Enable bit again in the middle of one shot mode PWM generation, PWM output changes to the value of output control (OPC) bit in the control register, then start one shot mode again from the beginning. PWMC Global Control Register also shows interrupt pending status.

Continuous mode: In this mode, PWMC generates PWM signal at the output port continuously. Global PWM Enable bit in PWMC Global Control Register should also be set for generating PWM signal. Once Global PWM Enable bit is programmed, write to PWMC Period Register and ON/OFF Register have no effect on current PWM Cycle. Any changes made in those registers will take effect in next PWM cycle, if there was a write to Control Register previously. If Period Register or ON/OFF Register is modified again, the PWMC will change period and duty cycle only after the next Control Register write. PWM generation can be stopped either by resetting Global PWM Enable bit (Stops immediately) or by changing mode to Idle (Stop after current PWM cycle). If Global Interrupt control in PWMC Global Control Register and local interrupt control in PWMC Control Register are enabled, then the interrupt pending (IP) bit in PWMC Status Register goes high after a PWM Cycle. The interrupt can be cleared by reading channel specific PWMC Status Register. PWMC Global Control Register also shows interrupt pending status.

6.4 PWM ALIGNMENT CONTROL

PWMC has a 2-bit Alignment Control (AC) in the control register. Support Left alignment and right alignment.

Left Alignment: In this alignment scheme, the value programmed to ON/OFF Register is considered as the on time of the PWM output in a PWM Cycle. PWM generates signal at the output as shown in the figure.

Right Alignment: In this alignment scheme, the value programmed to ON/OFF Register is considered as the off time of the PWM output in a PWM Cycle. PWM generates signal at the output as depicted in the figure.

PWM PR decides the PWM cycle duration and PWM ON/OFF Register decides the duration of first phase of the PWM signal. Alignment control decides the first phase PWM output state. Below example shows, how we can generate left aligned and right aligned PWM signal having 25% duty cycle with Period Register value of '4'.

For left aligned PWM generation PWM ON/OFF =1, for right Aligned with same period and duty cycle PWM ON/OFF = 3.

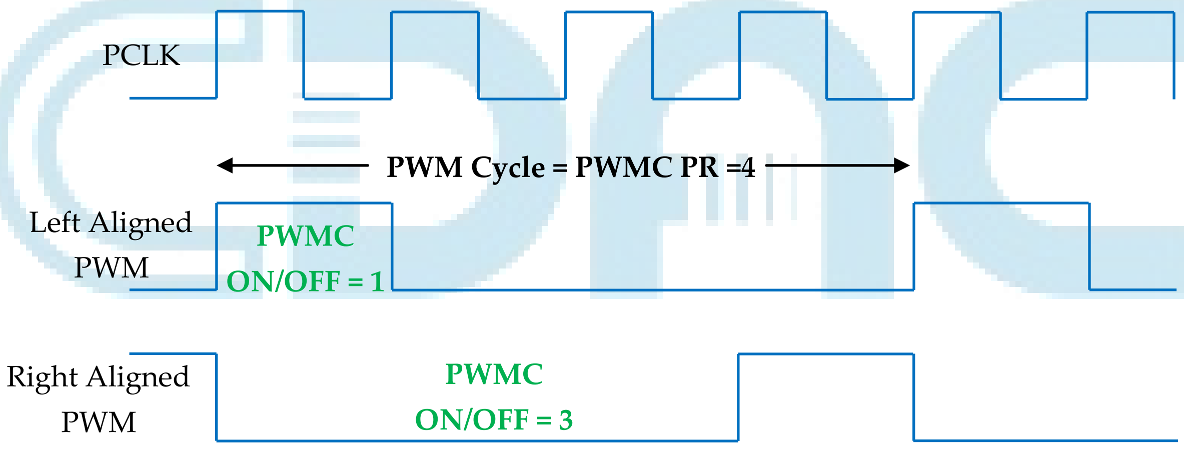


Figure 6-2:Left Aligned and Right Aligned Example

6.5 REGISTER BLOCK

This section provides the details of registers used in PWM.

6.5.1 PWMC_N CONTROL REGISTER (PWMC_CR) (32-BIT)

31 22	21 6	5	4	3 2	1 0
Reserved	**Repeat Count**	**OPC**	**IE**	**AC**	**Mode**

- Read/Write Register
- <u>Mode</u>
  - 00 : PWM Idle (Default)
  - 01: One shot mode
    - PWM signal is generated for a specified number of PWM cycles as defined by the repeat count field in PWMC control register.
  - 10: Continuous mode
    - PWM signal is generated continuously.
  - 11: Reserved for future use
    - Writing reserved value leads to default mode setting.
- <u>AC (Alignment control)</u>
  - Control the position of pulse in the PWM cycle
  - 00: Left alignment (Default)
  - 01: Right alignment
  - 10,11: Reserved for future use

- Writing reserved values will lead to default alignment setting.

- <u>IE (Interrupt Enable)</u>
  - Local interrupt enable bit
  - 0: Interrupt disabled, 1: Interrupt enabled
- <u>OPC (Output Control)</u>
  - Define the PWM output signal level when PWM is in idle state.
  - 0: Output level is Low in idle state
  - 1: Output level is High in idle state
- <u>Repeat count</u>
  - No: of pulses for one-shot mode of operation
- <u>Reserved</u>
  - Reserved for future use. Return zeros on read and write has no effect.

6.5.2 **PWMC_N STATUS REGISTER (PWMC_SR) (32-BIT)**

31 … 2	1	0
Reserved	**IP**	**Status**

- Read only Register.
- <u>Status</u>
  - 0: PWM in idle state, 1: PWM in running state
- <u>IP (Interrupt Pending)</u>
  - 0: No PWM interrupt pending, 1: PWM interrupt is pending

- Reserved
  - Reserved for future use. Return zeros on read and write has no effect.

6.5.3 PWMC_N PERIOD REGISTER (PWMC_PR) (32-BIT)

31 24	23 16	15 8	7 0
Value [31:24]	**Value [23:15]**	**Value [15:8]**	**Value [7:0]**

- Read/Write Register
- Value [31:0]
  - Capable of keeping 32-bit Period Value
  - Minimum value supported is 2. Maximum value supported 2^32 – 1.

6.5.4 PWMC_N ON/OFF REGISTER (PWMC_ON/OFF_R) (32-BIT)

31 24	23 16	15 8	7 0
Value [31:24]	**Value [23:15]**	**Value [15:8]**	**Value [7:0]**

- Read/Write Register
- Value [31:0]
  - Capable of keeping 32-bit on/off Time Value.
  - Minimum value supported is 1.
  - Maximum value supported is one less than maximum value supported by period register, i.e. (2^32 – 1) -1.

- When Alignment register is set for left alignment this register represents on time or high time of the PWM signal.
- When Alignment register is set for right alignment this register represents off time or low time of the PWM signal.

6.5.5 PWMC GLOBAL CONTROL REGISTER (PWMC_GCR) (32-BIT)

31 – 10	9 – 2	1	0
Reserved	**GIP_N**	**GIE**	**GPE**

- GPE (Global PWM Enable)
  - Global Control bit for starting PWM generation
  - Accessibility : Read and Write
  - 0: PWM channels are disabled, 1: PWM Channels are enabled
- GIE (Global Interrupt Enable)
  - Global interrupt enable control for PWM channels
  - Accessibility : Read and Write
  - 0: Interrupts in all channels are disabled
  - 1: Interrupts are enabled for PWM channels
- GIP_N (Global Interrupt Pending)
  - Global interrupt pending status for each PWM channels
  - Here N varies from 1 to 8. GPI_1 to GPI_8 from bit position 2 to 9.
  - Accessibility : Read Only
  - GPIN : 0, No interrupt is pending on Nth Channel

- GPIN : 1, Interrupt is pending on Nth Channel

- <u>Reserved</u>
  - Reserved for future use. Return zeros on read and write has no effect.

6.6 OUTPUTS

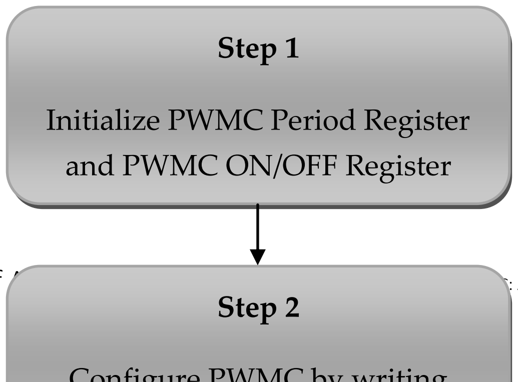
The output channel consists of 8-PWM output signals and 8-PWM interrupt signals.

6.7 PROGRAMMING CONSIDERATIONS

This section describes the programmable features of the PWMC peripheral. In order to avoid potential synchronization problems when initializing, loading, and enabling a PWM channel, basic procedure outlined in PWMC Usage Flow should be followed. The PWMC module is little-endian. All PWMC channels are disabled on reset and are enabled by writing "1" to the Global PWM enable bit of the PWMC Global Control Register. PWMC contains both channel specific registers and a Global Control Register.

6.7.1 PWMC USAGE FLOW

The procedure illustrated in Figure is a basic flow to follow when programming PWMC peripheral. Step 1 and step 2 are interchangeable. Step 2 can also be done prior to Step1. Once the Global Control register is programmed changes made to the Period Register, ON/OFF Register and Control Register will not take effect in current PWM cycle. PWM generation begins when Global PWM Enable bit is set to '1'.


Step 1
Initialize PWMC Period Register and PWMC ON/OFF Register
Step 2
Configure PWMC by writing

6.7.2 ONE SHOT MODE OF OPERATION:

- PWM generation stops after generating repeat count + 1 PWM cycles. PWM generation can be resumed by writing ‘1’ to Global PWM Enable bit again. If the programmer wants to run PWM again after completing the current repeat + 1 cycles, the programmer should make sure that the PWM stopped running by reading status register, then issue write to Global PWM Enable bit.
- When Global Interrupt Enable bit in the global control register and Interrupt Enable bit in Channel specific control register is set to ‘1’, interrupt is generated. Interrupt is indicated by interrupt pending bit in channel specific PWMC Status Register.
- Interrupt is cleared by reading channel specific PWMC Status Register.
- When PWM generation is active in one shot mode, write to Global Control Enable bit either restart (if value written is ‘1’) or stop (if value written is ‘0’) PWM generation.

- Writing 0 to Global Control Enable bit: Stop PWM generation immediately
- Writing 1 to Global Control Enable bit: Stop PWM generation immediately and restart one shot mode PWM generation again. Shadow Registers get reloaded again and any changes made to PWMC Control Register, Period Register and ON/OFF Register will take effect.

6.7.3 CONTINUOUS MODE OF OPERATION:

- PWM generation continues forever. It can be stopped by clearing Global PWM Enable bit in Global Control register. PWM can also be stopped by writing mode field to idle mode.
- Global Enable reset immediately stops PWM generation
- Mode reset stops PWM after the current PWM cycle.
- When Global Interrupt Enable bit in the global control register and the Interrupt Enable bit in channel specific control register is set to ‘1’, interrupt is generated after one PWM cycle. Interrupt is indicated by interrupt pending bit in channel specific PWMC Status Register.
- Interrupt is cleared by reading channel specific PWMC Status Register.
- When PWM generation is active in continuous mode, we can update Period Register and ON/OFF Register with new values. Write to PWMC CR make new values of Period Register and ON/OFF Register take effect from the next PWM cycle. If there is no write to Control register, then PWM will continue to work with old Period Register and ON/OFF Register values.

When multiple channels are active simultaneously and PWM channels can be disabled or enabled individually by writing mode in channel specific PWMC Control Register.

<u>Notes:</u>

- For proper PWM generation, minimum value in Period Register should be 2 and minimum value in ON/OFF Register should be 1. Always keep Period Register value higher than ON/OFF Register value. If these constraints are not met, then behaviour of PWM channel is undefined.
- Programmer can start multiple PWM channels simultaneously by writing PWM GCR after configuring channel specific registers.
- If the programmer wants to start a channel after other channels, programmer can set idle mode initially in that particular channel and write PWMC GCR. Later programmer can start idle channel by writing one shot or continuous mode in the channel specific control register. In this scenario values in PWM Period and ON/OFF Register need to be modified prior to writing the mode field.
- PWM Channels can be disabled simultaneously by writing 0 to PWMC Global PWM enable bit.
- PWM Channels can be disabled individually by changing mode field to idle in channel specific control register.

7 SPIM - SPI MASTER CONTROLLER

The Serial Peripheral Interface (SPI) bus is a synchronous, serial communication link used for short distance communication, primarily in embedded systems. SPI bus consists of four signals: master out slave in (MOSI), master in slave out (MISO), serial clock (SCK), and active-low chip select (CS). The serial clock line [SCK] synchronizes the shifting and sampling of the information on the two serial data lines. The master places the information onto the MOSI line a half-cycle before the clock edge that the slave device uses to latch the data. The master can latch the data from the slave through MISO line.

The SPIM controller core implements a single-lane Serial Peripheral Interface bus, which can operate as a master. The Serial Peripheral Interface allows synchronous serial data transfers between microprocessors and peripheral devices.

SPIM works with a wide variety of SPI-bus variants, the core supports different serial transfer frame formats, such as bit-width of frame and most-significant bit position in a frame are all software programmable. The core can control up to four slaves. A software-controllable clock generator derives the serial clock for the SPI slave.

The SPIM core interacts with the processor via 32-bit APB slave interface. It allows access to the transfer data, control, and states registers. It will generate an interrupt to indicate availability of received data or availability to transmit new data.

7.1 FEATURES

- Full-duplex synchronous serial operation
- Programmable clock rate
- Double-buffered data register
- Programmable polarity and phase
- Software programmable operation

- Maskable interrupt and status registers for status reporting
- 8 to 16-bit frames
- MSB-first and LSB-first frames
- Support up to four slaves under Master control
- Variable or fixed peripheral select

7.2 ARCHITECTURE OVERVIEW

The SPIM is essentially a shift register that serially transmits data bits to other SPIs. The SPI requires two control lines (CS_n and SCLK) and two data lines MOSI and MISO. The active low CS_n signal selects the peripheral device with which the master wants to communicate. In the unselected state the MOSI lines are high impedance.

The chip select signal and clock signal have to be generated by the master device when the data exchange has been processed. The SPIM operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s) connected to the SPI bus.

In addition to setting the clock frequency, the master must also configure the clock polarity and phase with respect to the data. During an SPI transfer, data is simultaneously transmitted and received. The serial clock line [SCK] synchronizes shifting and sampling of the information on the two serial data lines. The master places the information onto the MOSI line a half-cycle before the clock edge that the slave device uses to latch the data. The master can latch the data from the slave through MISO line.

The SPI bus parameters are highly configurable via registers. Core features include configurable word length and bit ordering. It supports all SPI modes.

The SPIM features two data holding registers. SPI_TDR holds the data to be transmitted through SPI and SPI_RDR holds the received data through SPI. The core also contains internal transmit and receive shift registers. A new data transfer through SPI begins when the data written on SPI_TDR is transferred to the internal transmit shift register. While data in the Transmit Shift Register is shifted on the MOSI line, the MISO line is sampled to take data from the SPI slave and stored into the Receive Shift Register. After the exchange of a specified number of data bits between master and slave, the received data is transferred from Receive Shift Register to SPI_RDR.

The SPI has four modes of operation, 0 through 3. These modes essentially control the way data is clocked in or out of an SPI device. The configuration is done by two bits in the SPI_CR. The clock polarity is specified by the CPOL control bit, which selects an active high or active low clock. The clock phase (CPHA) control bit selects one of the two fundamentally different transfer formats. To ensure a proper communication between master and slave both devices have to run in the same mode.

7.3 REGISTER DESCRIPTIONS

Name	Address (Range)	Size (byte)	Description
SPIM_CR	0x00	2	Control Register
SPIM_SR	0x04	1	Status Register
SPIM_BRR	0x008	1	Baud Rate Register
SPIM_TDR	0x0C	4	Transmit Data Register
SPIM_RDR	0x010	4	Receive Data Register

SPIM Registers

7.3.1 **SPIM_CR**

This register provides the user control over SPI modes of data transfer. The bit assignments in this register are described in table.

Bit	Name	Type	Default value	Description
[15:13]	Reserved	R/W	"000"	Reserved for future expansion
[12:9]	DBITS	R/W	"0000"	Bits per transfer. See table 14.
8	CSAAT	R/W	'0'	Chip Select Active After Transfer When this bit is set, the chip select line remains active low until transfer to another peripheral is required, or this bit is cleared.

7	SPTIE	R/W	‘0’	SPI Transmit Interrupt Enable ‘0’= Interrupt disabled ‘1’= Interrupt enabled If enabled, interrupt is generated when data is loaded from SPI_TDR to Transmit Shift Register
6	SPRIE	R/W	‘0’	SPI Receive Interrupt Enable ‘0’= Interrupt disabled ‘1’= Interrupt enabled If enabled, interrupt is generated when data is loaded from Receive Shift Register to SPI_RDR
5	CPOL	R/W	‘0’	Clock Polarity. See table 15.
4	CPHA	R/W	‘0’	Clock Phase. See table 15.
3	LSB/MSB First	R/W	‘0’	This bit selects LSB/MSB first data transfer format. ‘0’ = MSB first transfer format ‘1’ = LSB first transfer format
2	PS	R/W	‘0’	Fixed peripheral-0: Variable peripheral-1. See table 16.
1,0	PCS1, PCS0	R/W	‘0’	Peripheral Chip select. Active only if PS is active. See table 16.

Table 6:SPIM_CR Register

DBITS SPIM_CR (12:9)	Bits per transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16

Table 7:DBITS: Bits per transfer

SPI-mode	CPOL SPIM_CR (5)	CPHA SPIM_CR (4)
0	0	0
1	0	1
2	1	0
3	1	1

Table 8:SPI Clock configuration Modes

PCS1 SPIM_CR (1)	PCS0 SPIM_CR (0)	CS_n
0	0	1110
0	1	1101
1	0	1011
1	1	0111

Table 9:Fixed Peripheral Selection when PS=0

7.3.2 SPIM_SR

The SPIM_SR is a read-only register that gives the programmer visibility of the status of some aspects of the SPIM Core. The bit assignments in this register are described in table.

Bit	Name	Type	Default value	Description
7	TxE	R	‘0’	Transmit data register empty ‘1’ transmit data register empty ‘0’ transmit data register not empty

				This bit is set when the data from the transmit data register is loaded into an internal shift register for transmission. This bit cleared when data is loaded into transmit data register through APB interface
6	RxC	R	‘0’	SPI Receive Complete ‘1’ = New data available on receive data register ‘0’= Reception of new data is not complete This bit is set when received from the slave are available in Receive Data Register. This bit cleared by reading Receive Data Register.
5	OV	R	‘0’	Overrun Error 0 = No overrun error has been detected since the last read from Receive Data Register. 1 = An overrun has occurred since the last read from Receive Data Register. An overrun occurs when Receive Data Register. is loaded at least twice before reading through APB interface
4	TxB	R	‘0’	Transmitter busy ‘1’ –busy,‘0’- idle This bit is set when an SPI transfer is in progress
3	TxI	R	‘0’	Transmit Interrupt Status
2	RxI	R	‘0’	Receive Interrupt Status
1-0	-	R	‘00’	Reserved

Table 10:SPIM_SR Register

7.3.3 SPIM_BRR

This register can be loaded through APB interface and the value available in this register determines the data transfer rate through SPI. The bit assignments in this register are described in table.

Bit	Name	Type	Default value	Description
7-4	Baud[7:4]	R/W	'0'	Baud rate register 0000 to 1111 for selection of frequencies from 25 MHz to 50 KHz for clock division by powers of 2
3-0	Reserved	--	'0'	

Table 11:SPIM_BRR Register

BAUD SPI_BRR(7:4)	Clock Frequency Divisor	SCLK
0000	4	25MHz
0001	8	12.5MHz
0010	16	5.25MHz
0011	32	3.125MHz
0100	64	1.5625MHz
0101	128	700KHz
0110	256	400KHz
0111	512	200KHz
1000	1024	100KHz
1001	2048	50KHz
1010-111	--	Reserved

Table 12:Serial Clock Frequency.

7.3.4 SPIM_TDR

This register is loaded through APB interface with the data to be transmitted on the SPI bus. The bit assignments in this register are described in table.

Bit	Name	Type	Default value	Description
15-0	Data	R/W	‘0’	Data to be transmitted to slave
17-16	CS1, CS0	R/W	‘0’	Slave Select bits in variable peripheral mode.
31-18	Reserved	--	‘0’	--

Table 13:SPIM_TDR

CS1 SPIM_TDR(17)	CS0 SPIM_TDR(16)	CS_n
0	0	1110
0	1	1101
1	0	1011
1	1	0111

Table 14:Variable Peripheral Selection when PS=1

7.3.5 SPIM_RDR

This register is loaded through APB interface with the data to be transmitted on the SPI bus. The bitassignments in this register are described in table.

Bit	Name	Type	Default value	Description
15-0	Data	R	‘0’	Data received from slave
17-16	CS1, CS0	R/W	‘0’	
31-18	Reserved	--	‘0’	--

Table 15:SPIM_RDR

7.4 FUNCTIONAL DESCRIPTION

7.4.1 SPI CONFIGURATION

SPI master can be configured in four modes depending on the CPOL and CPHA bits in the control register. The polarity and the phase of the clock can be adjusted. A positive polarity clock results in latching data at the rising edge of the clock. However, data is put on the data line already at the falling edge in order to stabilize.

If the phase of the clock is zero, i.e. CPHA = 0, data is latched at the rising edge of the clock with CPOL = 0, and at the falling edge of the clock with CPOL = 1. If the phase of the clock is zero CPHA = 1, the polarities are reversed. CPOL = 0 means falling edge, CPOL = 1 rising edge.

7.4.2 SPI DATA TRANSFER FORMAT

The figure below depicts SPI data transfer format.

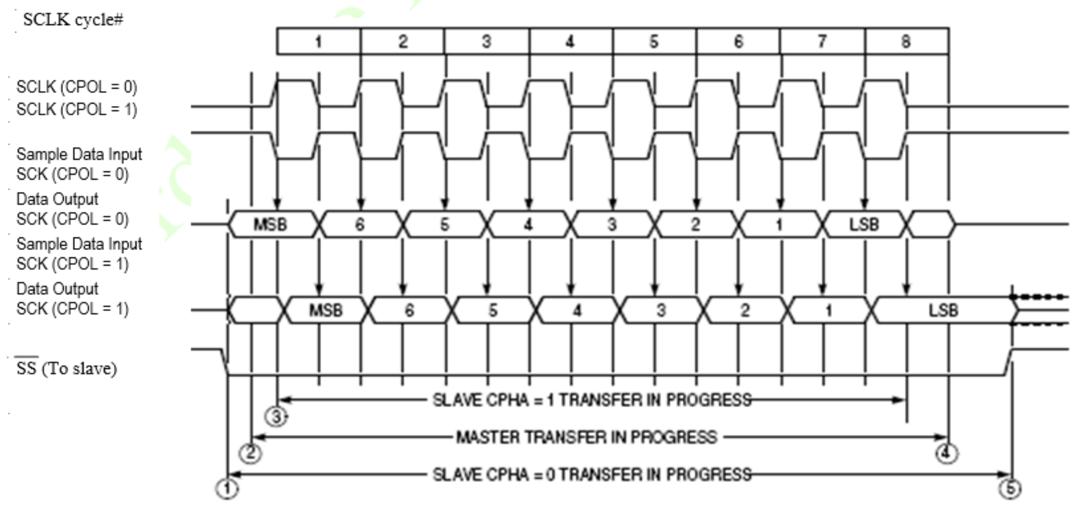

SCLK cycle#
1
2
3
4
5
6
7
8
SCLK (CPOL = 0)
SCLK (CPOL = 1)
Sample Data Input SCK (CPOL = 0)
Data Output SCK (CPOL = 0)
Sample Data Input SCK (CPOL = 1)
Data Output SCK (CPOL = 1)
SS (To slave)
MSB
6
5
4
3
2
1
LSB
SLAVE CPHA = 1 TRANSFER IN PROGRESS
MASTER TRANSFER IN PROGRESS
SLAVE CPHA = 0 TRANSFER IN PROGRESS
1
2
3
4
5

Figure 7-1:SPI Data Transfer Format

7.4.3 SPI PERIPHERAL SELECTION

The serial peripherals are selected through the assertion of the CS_n signals.

The peripheral selection can be performed in two different ways:

- Fixed Peripheral Select: SPI exchanges data with only one peripheral.
- Variable Peripheral Select: Data can be exchanged with more than one peripheral.

Fixed Peripheral Select is activated by writing the PS bit to zero in SPI_CR. In this case, the current peripheral is defined by the PCS field in SPI_CR. Variable Peripheral Select is activated by setting PS bit to one. This means that the peripheral selection can be defined for each new data by setting the CS1, CS0 bits in SPI_TDR appropriately.

7.4.4 SPI PERIPHERAL DESELECTION

When operating normally, as soon as the transfer of the last data written in SPI_TDR is completed, the CS line will rise. This might lead to runtime error if the processor is too long in responding to an interrupt, and thus might lead to difficulties for interfacing with some serial peripherals requiring the chip select line to remain active during a full set of transfers. To facilitate interfacing with such devices, the SPI_CR can be programmed with the CSAAT bit (Chip Select Active After Transfer) at 1. This allows the chip select line to remain in their current state (active low) until transfer to another peripheral is required.

7.4.5 SPI INTERRUPT

SPI master controller supports transmit hold register empty interrupt and receive complete interrupt. Active low interrupt is generated and is cleared by reading status register. Interrupt is generated whenever any one of the following interrupt types has an active high condition and is enabled via the control register.

7.4.6 SPI MASTER INITIALIZATION

The SPI Master Initialization steps are listed in the following table.

NO:	STEPS	CONTROL BITS
1.	Choose SPI clock frequency	Bits 7-4 of SPIM_BRR
2.	Configure the clock	CPOL and CPHA in SPIM_CR
3.	Enable interrupts(if desired)	SPTIE and SPRIE in SPIM_CR
4.	Select Data order	MSB/LSB bit of SPIM_CR
5.	Select Variable or Fixed Slave	PS and PCS bits in SPIM_CR
6.	If Variable Slave	CS0 and CS1 bits of SPIM_TDR

Table 16: SPI Initialization

8 TIMER

The TIMER implements three identical but separately programmable timers. Timers count down from a programmed value and generate an interrupt when the count reaches zero. Each timer has an independent clock input, which can be connected to the APB clock or to an external clock source. The width of the timer is 32 bits. Supports both free-running and user-defined count modes.

8.1 FEATURES

- Three timers
- Supports free-running and user-defined count modes
- Provides independent clocking of individual timers
- Support both asynchronous and synchronous clocking for APB clock and timer clock
- Configurable polarity for each individual interrupt
- Configurable option for a single or combined interrupt output
- Configurable option to support both asynchronous and synchronous clocking for APB clock and timer clock
- Configurable option to include timer toggle output, which toggles whenever timer counter reloads
- Configurable option to extend the internal interrupt signal up to three timer clock cycles

Note : TIMER_NO = 3 and TIMER_WIDTH=32

8.2 ARCHITECTURE OVERVIEW

The initial value for each timer, that is, the value from which it counts down is loaded into the timer using the appropriate Load_Count register. Timers count down from the programmed value and generate an interrupt when the count reaches zero. Two events can cause a timer to load the initial count from its TimerNLoadCount register:

- Timer is enabled after being reset or disabled
- Timer counts down to 0

The COMB_INTR parameter (Single or Combined Interrupt) is used to create a single combined interrupt, which is active whenever any of the individual timer interrupt is active.

In user mode, the timer starts counting from a programmed value and when the value reaches 'zero' an interrupt is generated. After that, the timer starts counting from the programmed value. But in free running mode, after the initial interrupt, a value of 'FFFF' is used by the timer to count down instead of the programmed value.

8.3 REGISTER MEMORY MAP

TIMER implements three timers. There are several registers with names specific to the number of timers that you choose (where N is from 1 to 3). TIMER have five individual registers for each of the timers and all other registers control their respective functions for all active timers, rather than for individual timers.

- TimerNLoadCount – TimerN load count register
- TimerNCurrentValue – TimerN current value register
- TimerNControlReg – TimerN control register
- TimerNEOI – TimerN end-of-interrupt register
- TimerNIntStatus – TimerN interrupt status register

The following table contains information about the register memory map. Table list the address ranges of the registers of each timer.

Address Range (Base+)	Function
0x00 to 0x10	Timer 1 Registers
0x14 to 0x24	Timer 2 Registers
0x28 to 0x38	Timer 3 Registers

Figure 8-1: Address descriptions of TIMER

8.3.1 GENERAL REGISTERS AND FUNCTIONS

Table lists registers associated with Timer 1; use this table as an example for timers 2-8.

Name	Address Offset	Width	R/W	Description
Timer1 LoadCount	0x00 0x01 0x02 0x03	See Description	R/W	Value to be loaded into Timer1 Width: TIMER_WIDTH-1 Range: 0 to [2 $^{\text{TIMER\_WIDTH}}$-1] Default value: TIMER_WIDTH-1'b0
Timer1 CurrentValue	0x04 0x05 0x06 0x07	See Description	R	Current Value of Timer1 Width: TIMER_WIDTH-1 Range: 0 to to [2 $^{\text{TIMER\_WIDTH}}$-1] Default value: TIMER_WIDTH-1'b0
Timer1 ControlReg	0x08	3-bits	R/W	Control Register for Timer1 Default Value: 3'b0
Timer1EOI	0x0c	1-bit	R	Clears the Interrupt from Timer1. Default value:1'b0
Timer1	0x10	1-bit	R	Contains Interrupt status for

IntStatus				Timer1 Default value:1'b0
Timers IntStatus	0xa0	See Description	R	Contains the interrupt status of all timers in the component Width: TIMER_NO Default value:TIMER_NO'b0
TimersEOI	0xa4	See Description	R	Returns all zeros (0) and clears all active interrupts. Width: TIMER_NO Default value:TIMER_NO'b0
Timers RawIntStatus	0xac	See Description	R	Contains the unmasked interrupt status of all timers in the component. Width: TIMER_NO Default value:TIMER_NO'b0

Table 17 TIMER Registers

Table lists the complete memory map of TIMER Registers.

Address Offset of Timer Number timer 1 to timer8	**Timer Number**		
	1	**2**	**3**
timerNLoadValue	**0x00**	**0x 14**	**0x 28**
	0x 01	**0x 15**	**0x 29**
	0x 02	**0x 16**	**0x 2A**
	0x 03	**0x 17**	**0x 2B**
timerNCurrentValue	**0x04**	**0x 18**	**0x 2C**
	0x 05	**0x 19**	**0x 2D**
	0x 06	**0x 1A**	**0x 2E**
	0x 07	**0x 1B**	**0x 2F**
timerNControlReg	**0x 08**	**0x 1C**	**0x 30**
timerNEOI	**0x 0C**	**0x 20**	**0x 34**

timerNIntStatus	**0x 10**	**0x 24**	**0x 38**

Table 18 TIMER Memory Map

The following sections contain the memory diagrams and field descriptions for the individual registers.

8.3.1.1 **TIMERNLOADCOUNT**

- Name : TimerNLoadCount Register
- Size: TIMER_WIDTH
- Read/write access: Read/write
- Address Offset:

  for N=1, 0x00

  for N=2, 0x14

  for N=3, 0x28

  for N=4, 0x3C

  for N=5, 0x50

  for N=6, 0x64

  for N=7, 0x78

  for N=8, 0x8C

Name	Bits	R/W	Description
TimerNLoadCount Register	TIMER_WIDTH-1:0	R/W	Values to be loaded into TimerN. This is the value from which counting commences. Any value written to this register is loaded into the associated timer.

8.3.1.2 **TIMERNCURRENTVALUE**

- Name : TimerNCurrent Value Register
- Size: TIMER_WIDTH
- Read/write access: Read
- Address Offset:

  for N=1, 0x04

  for N=2, 0x18

  for N=3, 0x2C

  for N=4, 0x40

  for N=5, 0x54

  for N=6, 0x68

  for N=7, 0x7C

  for N=8, 0x90

Name	Bits	R/W	Description
TimerNCurrentValue Register	TIMER_WIDTH-1:0	R	Current value of TimerN. This register is supported only when timer_N_clk is synchronous to pclk. Reading this register when using independent clocks results in an

			undefined value.

8.3.1.3 TIMERNCONTROLREG

- Name : TimerNControl Register
- Size: 3- bit
- Read/write access: Read/write
- Address Offset:

  for N=1, 0x08

  for N=2, 0x1C

  for N=3, 0x30

  for N=4, 0x44

  for N=5, 0x58

  for N=6, 0x6C

  for N=7, 0x80

  for N=8, 0x94

Name	Bits	R/W	Description
Reserved	31:3		read as zero
Timer Interrupt Mask	2	R/W	Timer interrupt mask for TimerN. 0: not masked 1: masked
Timer Mode	1	R/W	Timer mode for TimerN 0: free-running mode 1: user-defined count mode
Timer Enable	0	R/W	Timer enable bit for TimerN 0: disable 1: enable

8.3.1.4 **TIMERNEOI**

- Name : TimerNEndOfInterrupt Register
- Size: 1bit
- Read/write access: Read
- Address Offset:

  for N=1, 0x0C

  for N=2, 0x20

  for N=3, 0x34

  for N=4, 0x48

  for N=5, 0x5C

  for N=6, 0x70

  for N=7, 0x84

  for N=8, 0x98

Name	Bits	R/W	Description
Reserved	31:1		read as zero
TimerNEnd-of-Interrupt	0	R	Reading from this register returns all zeros (0) and clears the interrupt from TimerN.

8.3.1.5 **TIMERNINTSTATUS**

- Name : TimerNInterrupt Status Register
- Size: 1bit
- Read/write access: Read
- Address Offset:

  for N=1, 0x10

  for N=2, 0x24

  for N=3, 0x38

  for N=4, 0x4c

  for N=5, 0x60

  for N=6, 0x74

  for N=7, 0x88

  for N=8, 0x9C

Name	Bits	R/W	Description
Reserved	31:1		read as zero
TimerNInterrupt Status	0	R	Contains the Interrupt Status for TimerN.

8.3.1.6 3.3.6 TIMERSINTSTATUS

- Name : Timers Interrupt Status Register
- Size: Timer_No
- Read/write access: Read
- Address Offset: A0

Name	Bits	R/W	Description
Timers Interrupt Status Register	TIMER_NO-1:0	R	Contains the Interrupt status of all timers in the component. If a bit of this register is zero, then the corresponding timer interrupt is not active and the corresponding interrupt may be active high or active low depending on the interrupt polarity chosen. Similarly, if a bit of this register is 1 then the corresponding interrupt bit has been set in the relevant interrupt bus. In both cases the status reported is the status after the interrupt mask has been applied. Reading from this register does not clear any active interrupt. 0: Interrupt is not active after masking. 1: Interrupt is active after masking.

8.3.1.7 **TIMERSEOI**

- Name : Timers End Of Interrupt Register
- Size: Timer_No
- Read/write access: Read
- Address Offset: A4

Name	Bits	R/W	Description
Timers End Of Interrupt Register	TIMER_NO-1:0	R	Reading this register returns all zeros (0) and clears all active interrupts.

8.3.1.8 **TIMERSRAWINTSTATUS**

- Name : Timers Raw Interrupt Status Register
- Size: Timer_No
- Read/write access: Read
- Address Offset: A8

Name	Bits	R/W	Description
Timers Raw Interrupt Status Register	TIMER_NO-1:0	R	This register contains the unmasked interrupt status of all timers in the component. 0: Interrupt is not active prior to masking. 1: Interrupt is active prior to masking.

8.3.2 TIMER USAGE FLOW

The procedure illustrated in Figure is a basic flow to follow when programming the TIMER.

- Initialize the timer through the TimerN Control_Reg register (where N is in range 1 to 8):
    - Disable the timer by writing a "0" to the timer enable bit (bit 0). Before writing to a TimerNLoadCount register, you must disable the timer by writing a "0"to the timer enable bit of TimerNControlReg in order to avoid potential synchronization problems.
    - Program the timer mode; user-defined or free-running by writing a "0" or "1" respectively, to the timer mode bit (bit 1).
        - TimerNLoadCount register must be set to all 1s before enabling the timer in free-running mode.
    - Set the interrupt mask as either masked or not masked by writing a "0" or "1" respectively, to the timer interrupt mask bit (bit 2).
- Load the timer counter value into the TimerNLoad_Count register (where N is in the range 1 to 8).
- Enable the Timer by writing a "1" to bit 0 of TimerNControl_Reg register.

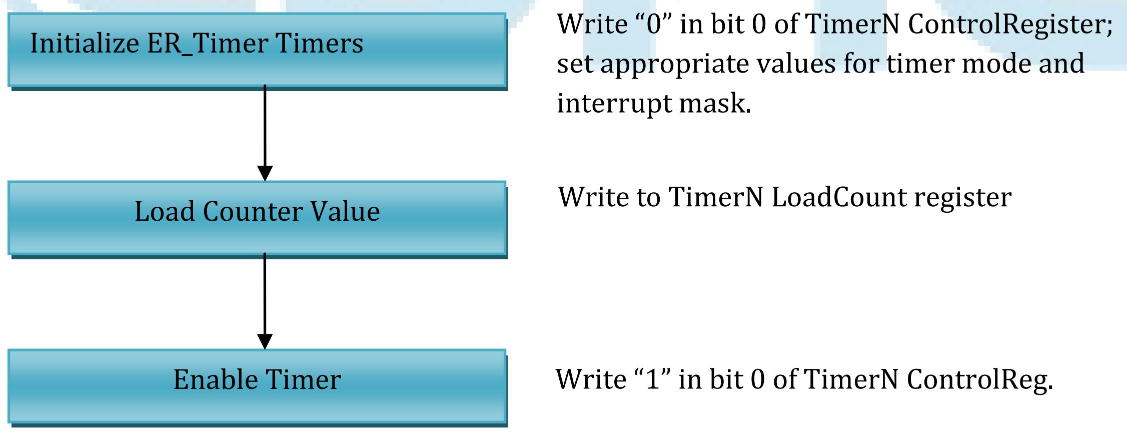

Initialize ER_Timer Timers
Write "0" in bit 0 of TimerN ControlRegister; set appropriate values for timer mode and interrupt mask.
Load Counter Value
Write to TimerN LoadCount register
Enable Timer
Write "1" in bit 0 of TimerN ControlReg.

Figure 8-2:TIMER USAGE FLOW

9 MEMORY

The SoC integrates 256KB RAM.

10 UART

The UART performs serial-to-parallel conversion on data characters received from a peripheral device or a parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions (parity, overrun or framing). The UART includes a programmable baud rate generator that is capable of dividing the timing reference clock input by divisors of 1 to (2^{16}-1) and producing a 16X clock for driving the internal transmitter logic. Interrupts can be programmed to the user's requirements, minimizing the computing required to handle the communications link.

10.1 FEATURES

- Programmable word length, stop bits and parity
    - 5-, 6-, 7-, or 8-bit characters
    - Even, odd, or no-parity bit generation and detection
    - 1-, 11/2-, or 2-stop bit generation.
    - Baud generation (DC to 1.5M baud).
    - Programmable baud rate generator
- Loop-back mode
- Scratch register
- In the FIFO mode, transmitter and receiver are each buffered with 16 byte FIFO's to reduce the number of interrupts presented to the CPU.
- Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from the serial data.
- Independently controlled transmit, receive, line status, and data set interrupts.
- Programmable baud generator divides any input clock by 1 to (2^{16}-1) and

generates the 16X clock.

- Full prioritized interrupt system controls.

10.2 REGISTER ADDRESSES

A table of registers and their addresses is shown below. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the Line Control Register, affects the selection of certain UART registers. The DLAB must be set high by the system software to access the Baud Generator Divisor Latches.

DLAB	A2	A1	A0	Register
0	0	0	0	ReceiverBuffer(read) TransmitterHolding Register(write)
0	0	0	1	Interrupt Enable
x	0	1	0	Interrupt Identification(read)
x	0	1	0	FIFO Control(write)
x	0	1	1	Line Control
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

Table 19: UART register map

10.3 REGISTERS

The system programmer may access any of the UART registers via the CPU. These registers control UART operations including transmission and reception of data. Each register bit and reset state is shown in below table.

10.3.1 LINE CONTROL REGISTER

The system programmer specifies the format of the asynchronous data communications exchange and set the Divisor Latch Access bit via the Line Control Register (LCR). The programmer can also read the contents of the Line Control Register. The read capability simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. Table shows the contents of the LCR.

Bits 0 and 1: These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows

Bit1	Bit0	Character Length
0	0	5Bits
0	1	6Bits
1	0	7Bits
1	1	8Bits

Bit 2: This bit specifies the number of Stop bits transmitted and received in each serial character. If bit 2 is a logic 0, one Stop bit is generated in the transmitted data. If bit 2 is a logic 1 when a 5-bit word length is selected via bits 0 and 1, one and a half Stop bits are generated. If bit 2 is logic 1, when a 6-, 7-, or 8-bit word length is selected, two Stop bits are generated. The receiver checks the first Stop- bit only, regardless of the number of Stop bits selected.

Bit 3: This bit is the Parity Enable bit. When the bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed.)

Bit 4: This bit is the Even Parity Select bit. When the bit 3 is a logic 1 and the bit 4 is a logic 0, an odd number of logic 1s are transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of logic 1s are transmitted or checked.

Bit 5: This bit is the Stick Parity bit. When bits 3, 4 and 5 are logic 1, the Parity bit is transmitted and checked as a logic 0. If bits 3 and 5 are 1 and bit 4 is a logic 0, then the Parity bit is transmitted and checked as a logic 1. If bit 5 is a logic 0, Stick Parity is disabled.

Bit 6: Reserved.

Note: This feature enables the CPU to alert a terminal in a computer communications system.

If the following sequence is followed, no erroneous or extraneous characters will be transmitted because of the break.

1. Load an all 0s, pad character, in response to THRE.
2. Set break after the next THRE.
3. Wait for the transmitter to be idle, (TEMT=1), and clear break when normal transmission has to be restored.

During the break, the Transmitter can be used as a character timer to accurately establish the break duration.

Bit 7: This bit is the Divisor Latch Access Bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the Baud Generator during a Read or Write operation. It must be set low (logic 0) to access the Receiver Buffer, the Transmitter Holding Register, or the Interrupt Enable Register.

10.3.2 PROGRAMMABLE BAUD GENERATOR

The UART contains a programmable Baud Generator that is capable of taking any clock input and dividing it by any divisor from 2 to 216-1. The output frequency of the Baud Generator is 16 X the Baud [divisor # = (frequency input)%(baud rate x16)].

Two 8-bit latches store the divisor in a 16-bit binary format. These divisor latches must be loaded during initialization to ensure proper operation of the Baud Generator. Upon loading

either of the divisor latches, a 16-bit Baud counter is immediately loaded.

10.3.3 LINE STATUS REGISTER

This register provides status information to the CPU concerning the data transfer.

Bit 0: This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register or the FIFO. Bit 0 is reset to logic 0 by reading all of the data in the Receiver Buffer Register or the FIFO.

Bit 1: This bit is the Overrun Error (OE) indicator. It indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. The OE indicator is set to a logic 1 upon detection of an overrun condition and reset whenever the CPU reads the contents of the Line Status Register. If the FIFO mode data continues to fill the FIFO beyond the trigger level, an overrun error will occur only after the FIFO is full and the next character has been completely received in the shift register. OE is indicated to the CPU as soon as it happens. The character in the shift register is overwritten, but it is not transferred to the FIFO.

Bit 2: This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity- select bit. The PE bit is set to a logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU reads the contents of the Line Status Register. In the FIFO mode, this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO.

Bit 3: This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a logic 0 bit (Spacing level). The FE indicator is reset whenever the CPU reads the contents of the Line Status Register. In the FIFO mode, this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. The UART will try to resynchronize after a framing error. To do this it assumes that the framing error was due to the next start bit, so it samples this “start” bit twice and then takes in the "data".

Bit 4: This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits +Parity +Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status Register. In the FIFO mode, this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. When break occurs only one zero character is loaded into the FIFO. The next character transfer is enabled after SIN goes to the marking state and receives the next valid start bit.

Note: Bits 1 through 4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected and the interrupt is enabled.

10.3.4 INTERRUPT IDENTIFICATION REGISTER (IIR)

When the CPU accesses the IIR, the UART freezes all interrupts and indicates the highest priority pending interrupt to the CPU. While this CPU access is occurring, the UART records new interrupts, but does not change its current indication until the access is complete.

Bit 0: This bit can be used in a prioritized interrupt environment to indicate whether an interrupt is pending. When bit 0 is logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is logic 1, no interrupt is pending.

Bits 1 and 2: These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in Table.

Registers (Continued)
Interrupt Control Functions

FOFO Mode Only	**Interrupt Identification Register**			**Interrupt Set and Reset Functions**			
Bit 3	**Bit 2**	**Bit 1**	**Bit 0**	**Priority Level**	**Interrupt Type**	**Interrupt Source**	**Interrupt Reset Control**
0	0	0	1	--	None	None	--
0	1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
0	1	0	0	Second	Received Data Available	Receiver Data Available or Trigger Level Reached	Reading the Receiver Buffer Register or the FIFO Drops Below the Trigger Level
1	1	0	0	Second	Character Timeout Indication	No Characters Have Been Removed From or Input to the RCVR FIFO During the Last 4 Char. Times and There is at Least 1 Char. In It During This Time	Reading the Receiver Buffer Register
0	0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register

Bit 3: In the Normal Mode, this bit is 0. In the FIFO mode, this bit is set along with bit 2 when a timeout interrupt is pending.

Bits 4 and 5: These two bits of the IIR are always logic 0.

Bits 6 and 7: These two bits are set when FCR0= 1.

10.3.5 INTERRUPT ENABLE REGISTER

This register enables the five types of UART interrupts. Each interrupt can individually activate the interrupt (INTR) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the Interrupt Enable Register (IER). Similarly, setting bits of the IER register to logic 1, enables the selected interrupt(s). Disabling an interrupt prevents it from being indicated as active in the IIR and from activating the INTR output signal. All other system functions operate in their normal manner, including the set- ting of the Line Status and MODEM Status Registers.

Bit 0: This bit enables the Received Data Available Interrupt (and timeout interrupts in the FIFO mode) when set to logic1.

Bit 1: This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1.

Bit 2: This bit enables the Receiver Line Status Interrupt when set to logic 1.

Bit 3: Reserved.

Bits 4 through 7: These four bits are always logic 0.

10.3.6 **SCRATCHPAD REGISTER**

This 8-bit Read/Write Register does not control the UART in anyway. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

10.3.7 **FIFO INTERRUPT MODE OPERATION**

When the RCVR FIFO and receiver interrupts are enabled.

(FCR0=1, IER0=1) RCVR interrupts will occur as follows

A. The receive data available interrupt will be issued to the CPU when the FIFO has reached its programmed trigger level; it will be cleared as soon as the FIFO drops below its programmed trigger level.

B. The IIR receive data available indication also occurs when the FIFO trigger level is reached, and like the interrupt it is cleared, when the FIFO drops below the trigger level.

C. The receiver line status interrupt (IIR=06), as before, has higher priority than the received data available (IIR=04) interrupt.

D. The data ready bit (LSR0) is set as soon as a character is transferred from the shift register to the RCVR FIFO. It is reset when the FIFO is empty.

When RCVR FIFO and receiver interrupts are enabled, RCVR FIFO timeout interrupts will occur as follows

A. A FIFO timeout interrupt will occur if the following conditions exist

- at least one character is in the FIFO
- the most recent serial character received was longer than 4 continuous character times ago (if 2 stop bits are programmed the second one is included in this time delay).
- the most recent CPU read of the FIFO was longer than 4 continuous character times ago.

The maximum time between a received character and a timeout interrupt will be 160 ms at 300 baud with a 12-bit receive character (i.e., 1 Start, 8 Data, 1 Parity and 2 Stop Bits).

B. Character times are calculated by using the RCLK input for a clock signal (this makes the delay proportional to the baud rate).

C. When a timeout interrupt has occurred, it is cleared and the timer reset when the CPU reads one character from the RCVR FIFO.

D. When a timeout interrupt has not occurred, the timeout timer is reset after a new character is received or after the CPU reads the RCVR FIFO.

When the XMIT FIFO and transmitter interrupts are enabled

(FCR0=1, IER1=1), XMIT interrupts will occur as follows

A. The transmitter holding register interrupt (02) occurs when the XMIT FIFO is empty; it is cleared as soon as the transmitter holding register is written to (1 to 16 characters may be written to the XMIT FIFO while servicing this interrupt) or the IIR is read.

10.3.8 FIFO POLLED MODE OPERATION

With FCR0=1 resetting IER0, IER1, IER2, IER3 or all to zero puts the UART in the FIFO Polled Mode of operation. Since the RCVR and XMITTER are controlled separately either one or both can be in the polled mode of operation.

In this mode the user's program will check RCVR and XMITTER status via the LSR. As stated previously:

- LSR0 will be set as long as there is one byte in the RCVR FIFO.
- LSR1 to LSR4 will specify which error(s) has occurred. Character error status is handled the same way as when in the interrupt mode, the IIR is not affected since IER 2=0.
- LSR5 will indicate when the XMIT FIFO is empty.
- LSR6 will indicate that both the XMIT FIFO and shift register are empty.
- LSR7 will indicate whether there are any errors in the RCVR FIFO.
- There is no trigger level reached or timeout condition indicated in the FIFO Polled Mode, however, the RCVR and XMIT FIFOs are still fully capable of holding characters.

11 INTERRUPT CONTROLLER

The interrupt controller used is a simple controller the registers are

Address	Register	Description
20010000	Raw interrupt	Give the status of the interrupt pin from each peripheral before masking
20010004	Interrupt enable	Interrupt Enable/disable bit 1: Enable, 0: Disable
20010008	Interrupt status	Status of the interrupt after making

Peripheral	Interrupt Number
UART0	0
UART1	1
UART2	2
SPI0	3
SPI1	4
I2C0	5
I2C1	6
TIMER0	7
TIMER1	8
TIMER2	9
GPIO	10-21
I2C2	22
SPI2	23
PWM	24 - 31

12 RESET AND CLOCK

- A low external signal on the RSTIN pin resets the THEJAS32 device
- THEJAS32 clock input can be go upto 100 MHz with 50% duty cycle

13 BOOT PROCEDURE

THEJAS32 bootloader is designed to transfer firmware binaries to and from UART or SPI flash to the internal RAM. The bootloader is the first code that is executed upon a system reset. It resides at the start address of the SoC which is at 0x10000

Boot select	Boot mode
1	UART
0	SPI flash

UART Boot

The Bootstrap begins a UART0 boot. UART must be configured with the following parameters to get the boot messages. UART 0 boot uses XMODEM (1K) protocol to transfer data blocks with checksums to Internal SRAM. In this mode boot loader waits for input binary file and the boot loader transfers data blocks byte by byte from the start address. When the total number of bytes is received, the boot code exits by jumping to the entry point of the transferred code.
UART Settings to get the bootloader message;

- **Data bits : 8**
- **Parity : None**
- **Stop bits : 1**
- **Speed : 115200**

Uploading Using UART

Uploading using **UART** method uses **XMODEM protocol** to upload program.

1. Open **minicom**, **reset** board and make sure Transfer mode is **UART XMODEM**.
2. Use **CTRL+A S** to enter file sending menu and select **xmodem** by pressing **Enter**.

```
+-[Upload]--+
| zmodem    |
| ymodem    |
|>xmodem<   |
| kermit    |
| ascii     |
+-----------+
```

3. In the next window, with the **space bar** select the **.bin** file to be transferred, by pressing **Enter**, the transfer process starts.
4. Wait until the process is completed. The screen should display how much data has been transferred.

After completing the transfer the Program will start to execute.

Note: Please press an enter after completion of each file transfer to transfer more files.

SPI Boot

In this mode data is transferred from SPI memory to SRAM. The bootloader reads and copies 250KB. When the total number of bytes is transferred, the boot code exits by jumping to the entry point of the transferred code.

Note: While the boot code executes, it reserves upper 6KB of IRAM space from 0x23E800 to 0x23FFFF for stack, variables and other software functions. The boot code loads user binary as a single contiguous block and if the specified code size is greater than 250 KB it will corrupt the IRAM space reserved by the boot code and possibly fail. This limits the size of user boot code loaded by the bootstrap to 250KB or less from UART boot. This limit does not apply to other boot options.

14 DEVICE INFORMATION

PART NUMBER	PACKAGE	BODY SIZE#
THEJAS32	LQFP 128 pins	16 mm x 16 mm

# the size shown here are approximations.

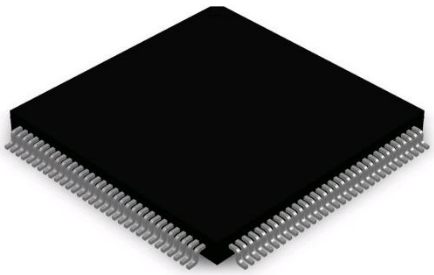
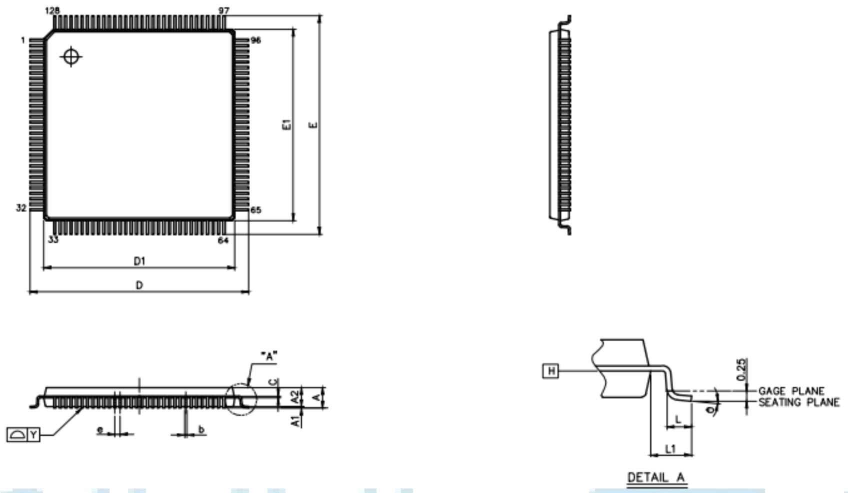


Figure 3: THEJAS32 SoC LQFP128 Package

15 PACKAGE DETAILS – LQFP128



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	--	--	1.60
A1	0.05	--	0.15
A2	1.35	1.40	1.45
b	0.13	--	0.23
c	0.09	--	0.20
D	16.00 BSC		
D1	14.00 BSC		
E	16.00 BSC		
E1	14.00 BSC		
e	0.40 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		
θ	0°	3.5°	7°
Y	0.08		

16 CONTACT DETAILS

For any further information and technical support, please contact us at

Hardware Design Group

C-DAC, Vellayambalam

Thiruvananthapuram, Kerala – 695033

Phone: 0471- 2723333 (Ext: 347)

Mob: 9037569219

E-Mail: vega@cdac.in

www.vegaprocessors.in