

MM SOC Specification

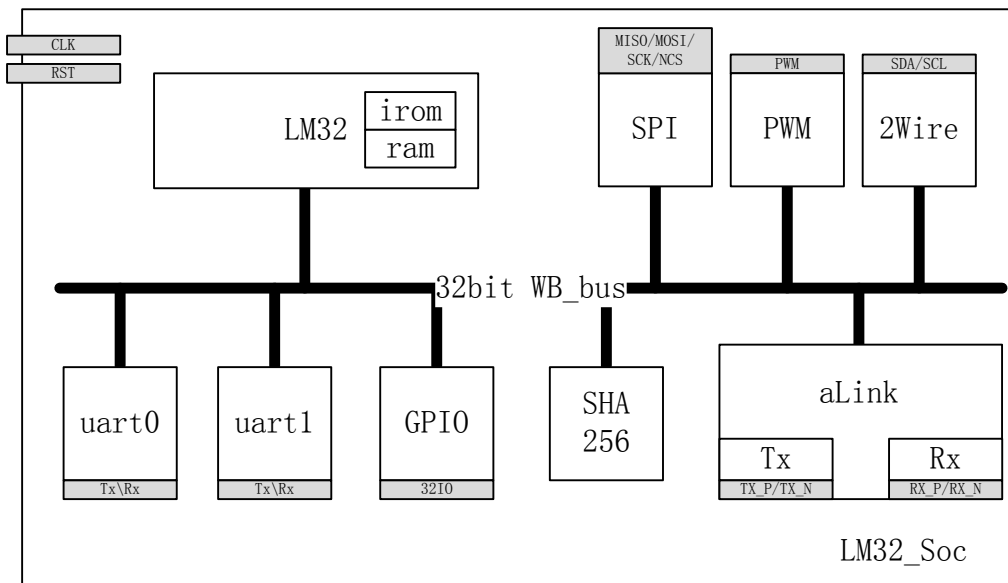
Date 2014.01.12

This is free and unencumbered document released into the public domain.
For details see the <http://unlicense.org/>.

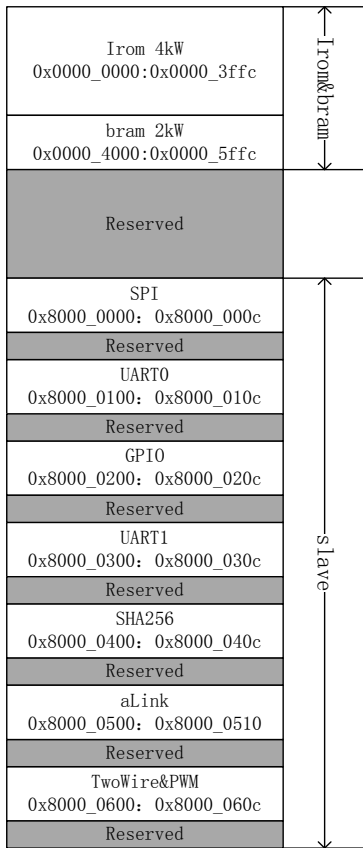
1. GENERAL DESCRIPTION

- LM32: 32-bit Harvard architecture “soft” microprocessor core.
- Support Serial peripheral interface.
- Two universal asynchronous receiver-transmitter(UART) used to interface to RS232 serial devices.
- 32bit GPIO.
- SHA256 Acceleration.
- aLink Controller.
- TwoWire(TWI) Interface.
- PWM and WatchDog Controller.
- 8bit Shifter Controller.
- Second level Timer.
- Fan Speed Reader*2.

2. SYSTEM ARCHITECTURE

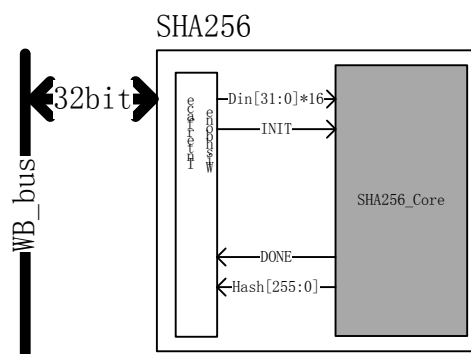


3. MEMARY MAP



4. SHA256

4.1 Architecture



4.2 Register Description(BaseAddress:0x80000400)

Register Name	Address	Address Offset within Register Word	Description
CMD	0x00	SHA256 Controller Command[W] [0] INIT, wire 1 to initial Algorithm core,	SHA256 Command Register

		<p>WriteOnly.</p> <p>[1] DONE: SHA_HASH is valid. This bit will clear by write DIN or CMD.DBL automatically [R].</p> <p>[2]RST: Set H0~7 to Default Value. MUST set RST before INIT;</p> <p>[3]DBL: Double-SHA256, When DONE=1, set 1 to this bit, then wait DONE return to 1.(i.e. sha256(sha256(data))); The 2ed-sha256's input will be {sha256(data),0x80000000,0x00000000,0x00000000,0x00000100}.</p>	
DIN	0x04	Hash Data Input (message), [W];	Hash Data Input
HASH	0x08	Hash Data Output, 4Words, ReadOnly; SHA256(message) = {HASH0,HASH1,HASH2,...,HASH6,HASH7};	SHA256 Data Output
HI	0x0c	Initial Hash Value [W] when sent INIT the HI will load to SHA256 algorithm core : HI = {H0,H1,H2,H3,H4,H5,H6,H7} Default Value: H0 = 6a09e667 H1 = bb67ae85 H2 = 3c6ef372 H3 = a54ff53a H4 = 510e527f H5 = 9b05688c H6 = 1f83d9ab H7 = 5be0cd19.	Initial Hash Value
PRE	0x10	The Pre-Calculation Result [R] {a0,a1,a2,e0,e1,e2} Only Valid when CMD.DONE=1.	Pre-Calculation Result

4.3 Test Case

4.3.1 Single Sha256

(1)Set CMD.INIT = 1;

(2)Input Data

H0 = 6a09e667

H1 = bb67ae85
H2 = 3c6ef372
H3 = a54ff53a
H4 = 510e527f
H5 = 9b05688c
H6 = 1f83d9ab
H7 = 5be0cd19.

The words of the padded message block are then assigned to the words W0,...,W15 of the message schedule:

DIN0 = 61626380//the first word write to DIN Register
DIN 1 = 00000000
DIN 2 = 00000000
DIN 3 = 00000000
DIN 4 = 00000000
DIN 5 = 00000000
DIN 6 = 00000000
DIN 7 = 00000000
DIN 8 = 00000000
DIN 9 = 00000000
DIN 10 = 00000000
DIN 11 = 00000000
DIN 12 = 00000000
DIN 13 = 00000000
DIN 14 = 00000000
DIN 15 = 00000018//the last word to DIN Register

(3)Then, wait CMD.DONE = 1;

(4)Output Data(SHA256)

HASH0 = ba7816bf// the first word read from HASH Register
HASH1 = 8f01cfea
HASH2 = 414140de
HASH3 = 5dae2223
HASH4 = b00361a3
HASH5 = 96177a9c
HASH6 = b410ff61
HASH7 = f20015ad

Output PRE(SHA256)

A0 = 0x5d6aebcd;
A1 = 0x5a6ad9ad;
A2 = 0xc8c347a7;
E0 = 0xfa2a4622;
E1 = 0x78ce7989;

E2 = 0xf92939eb;

4.3.2 Double Sha256

(1)Set CMD.INIT = 1;

(2)Input Data

H0 = 6a09e667

H1 = bb67ae85

H2 = 3c6ef372

H3 = a54ff53a

H4 = 510e527f

H5 = 9b05688c

H6 = 1f83d9ab

H7 = 5be0cd19.

The words of the padded message block are then assigned to the words W0,...,W15of the message schedule:

DIN0 = 61626380//the first word write to DIN Register

DIN 1 = 00000000

DIN 2 = 00000000

DIN 3 = 00000000

DIN 4 = 00000000

DIN 5 = 00000000

DIN 6 = 00000000

DIN 7 = 00000000

DIN 8 = 00000000

DIN 9 = 00000000

DIN 10 = 00000000

DIN 11 = 00000000

DIN 12 = 00000000

DIN 13 = 00000000

DIN 14 = 00000000

DIN 15 = 00000018//the last word to DIN Register

(3)Then, wait CMD.DONE = 1;

(4)Set CMD.DBL = 1;

(5)Then, wait CMD.DONE = 1;

(6)Output Data(Double SHA256)

HASH0 = 4f8b42c2

HASH1 = 2dd3729b

HASH2 = 519ba6f6

HASH3 = 8d2da7cc

HASH4 = 5b2d606d

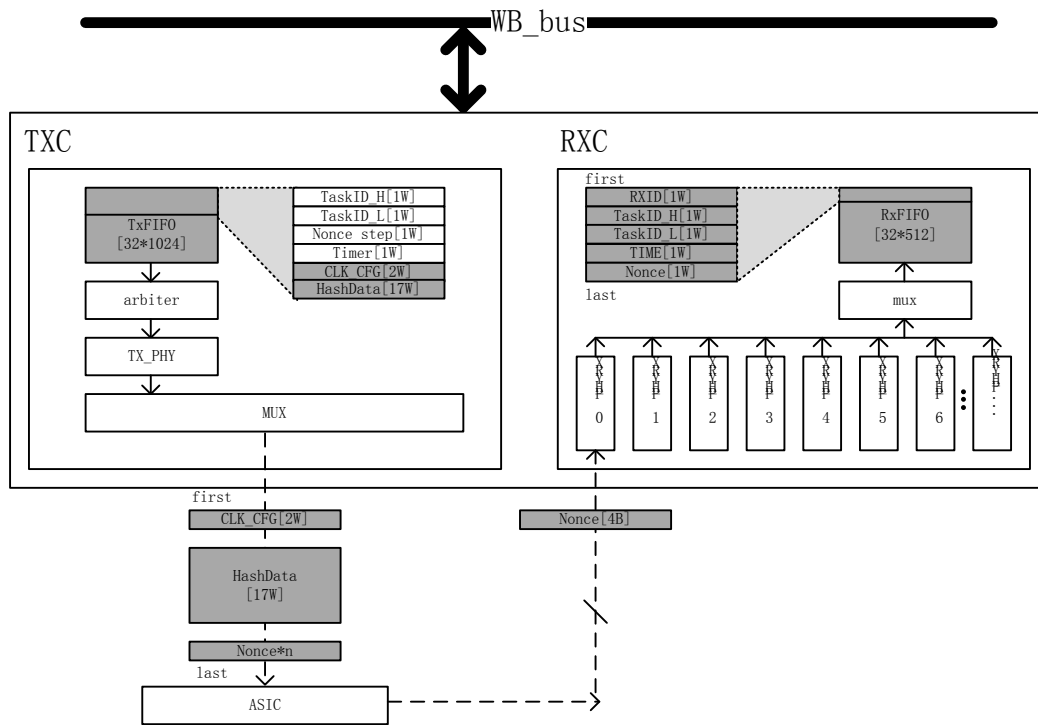
HASH5 = 05daed5a

HASH6 = d5128cc0

HASH7 = 3e6c6358

5. aLink

5.1 Architecture



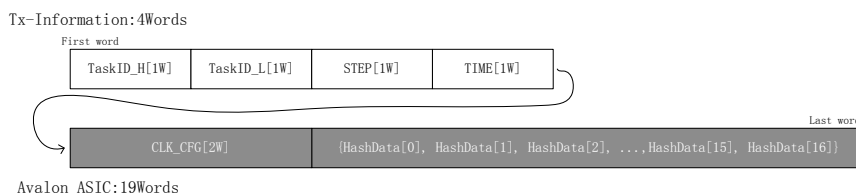
5.2 Register Description(BaseAddress:0x80000500)

Register Name	Address	Address Offset within Register Word	Description
TXFIFO	0x00	TxFIFO Data Input, [W]. A Message Block ^[1] Include 4words of Tx-Information and 19words of Avalon Chip Configure Data.	TxFIFO Data Input
STATE	0x04	TXC part:[R] [0] TxFULL , ReadOnly; 0: TxFIFO can receive 1(or more) Message Block(5+19Words); 1: TxFIFO Full, can NOT receive any Message Block. [1] FLUSH FIFO , write 1 to FLUSH TxFIFO & RxFIFO, WriteOnly; [14:4] TxCNT , TxFIFO Counter, Debug Interface, ReadOnly;	aLink State Register

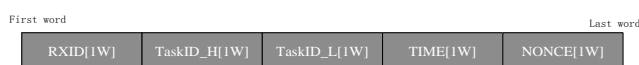
		<p>RXC part:</p> <p>[16] RxEMPTY, ReadOnly. 0: RxFIFO have 1(or more) Receive Block(5Words) to be read; 1: RxFIFO Empty.</p> <p>[29:20] RxCNT, RxFIFO Counter, Debug Interface, ReadOnly</p> <p>[30] Reserved.</p> <p>[31] Chip scan mode enable;</p> <p style="text-align: center;">Scan-mode Com-mode</p> <p>msg_blk[1] <i>nonce</i> <i>taskid_l</i></p> <p>msg_blk[0] <i>chip-num</i> <i>taskid_h</i></p> <p>chip-num^[3] [7:0] chip number in the chain: from 0 to 2[^]8-1.</p>	
Enable	0x08	<p>[9:0]PHYx Enable, [W]</p> <p>If Enablex=1, means the coordinate PHY will be enable. Write&Read. After power on, MUST delay 60ms.</p> <p>[31:10] Reserved.</p>	PHY Enable Register
BUSY	0x0c	<p>PHY Busy State: [R]</p> <p>0: the coordinate PHY IDLE. 1: the coordinate PHY BUSY.</p>	PHY Busy State
RXFIFO	0x10	<p>RxFIFO Data output, [R].</p> <p>A Receive Block^[2] =</p> <p>{RXID, TaskID_H, TaskID_L, TIME, NONCE}.</p>	Rx FIFO Output

NOTE

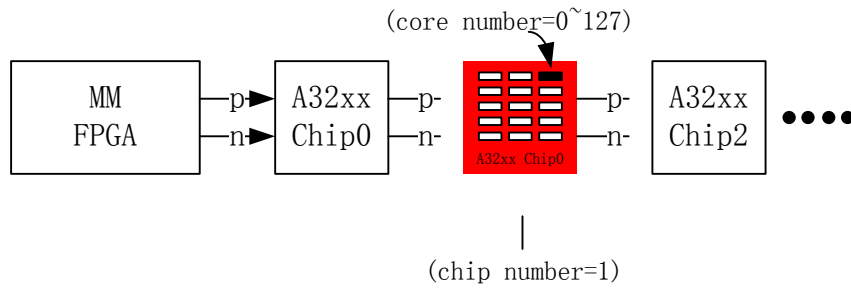
[1] A Message Block Include 4words of Tx-Information and 19words of Avalon Chip Configure Data.



[2] A Receive Block Include 5words of Information.



[3] Chip scan example & topology:



```
STATE = 0x80000000 ;//enable chip scan mode;
Msg_blk[0] = 0x1; ;//the chip number. i.e. target chip;
Msg_blk[1] = nonce; ;//the nonce for basic engine.
```

5.3 Test Case

Register Configuration:

Set Enable(0x08) Register = 0x1 ;//Only Enable PHY0

Input Data(MessageBlock):

```
msg_blk[22] = 0x220f1dbd ;
msg_blk[21] = 0xd8f8ef67 ;
msg_blk[20] = 0x12146495 ;
msg_blk[19] = 0xc44192c0 ;
msg_blk[18] = 0x7145fd6d ;
msg_blk[17] = 0x974bf4bb ;
msg_blk[16] = 0x8f41371d ;
msg_blk[15] = 0x65c90d1e ;
msg_blk[14] = 0x9cb18a17 ;
msg_blk[13] = 0xfa77fe7d ;
msg_blk[12] = 0x12cdfd7b ;
msg_blk[11] = 0x81677107 ;
msg_blk[10] = 0x62a5f25c ;
msg_blk[ 9] = 0x05b168ae ;
msg_blk[ 8] = 0x087e051a ;
msg_blk[ 7] = 0x88517050 ;
msg_blk[ 6] = 0x4ac1d001 ;
msg_blk[ 5] = 0x00000000 ;//clock cfg1
msg_blk[ 4] = 0x94E00001 ;//clock cfg0
msg_blk[ 3] = 0xFFFFFFFF ;//time out
msg_blk[ 2] = 0x19999999 ;//step
msg_blk[ 1] = 0x89abcdef ;//taskid_l or nonce
msg_blk[ 0] = 0x01234567 ;//taskid_h or chip_num
```

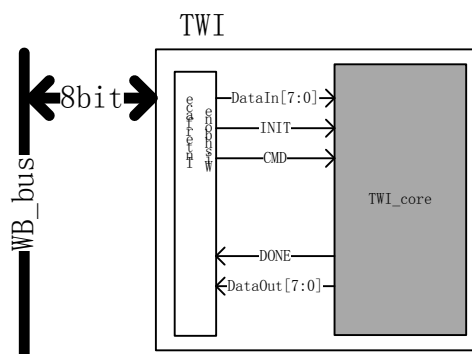
Output Data(ReceiveBlock):

```
rxdata[0] == 0x00000001; //RXID  
rxdata[1] == msg_blk[0]; //TaskID  
rxdata[2] == msg_blk[1]; //TaskID  
rxdata[3] == 0x??; //time out, Determined by the system  
rxdata[4] == 0x010f0f76; //nonce
```

- How to determine which chip&core hit, under scan mode?
chip number = rxdata[1];
core number = rxdata[4]^rxdata[2];

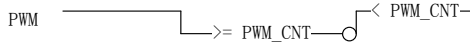
6. TwoWire & PWM(TWI&PWM) & WathDog & SHIFTER

6.1 Architecture



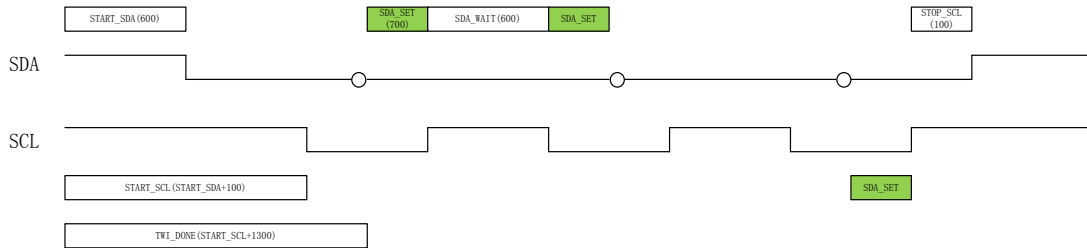
6.2 Register Description(BaseAddress:0x80000600)

Register Name	Address	Address Offset within Register Word	Description
CR	0x00	TWI Ctrl Register [0] Enable TWI. [1] TWI Transfer Start. [2] Transfer Done. [3] Reserved. [7:4] Command 0000:TWI Start; 0001:Write Data; 0010:Read Data+ACK 0011:STOP 0100:Read Data+NO ACK [31:8] Reserved.	TWI Ctrl Register
WD	0x04	TWI Write Byte [7:0] TWI Write Data. [31:8] Reserved.	TWI Write Byte
RD	0x08	TWI Read Byte [7:0] TWI Read Byte [31:8] Reserved.	TWI Read Byte
PWM	0x0C	PWM [9:0] PWM Counter(0x3ff~0x00): if PWM Counter < PWM, output 1; else output 0 ;	PWM Counter Register

		 <p>[31:10] Reserved.</p>	
WDG	0x10	<p>Watch Dog Timer</p> <p>[0] WatchDog Enable.[WR] 1: enable; 0: disable;</p> <p>[26:1] FeedDogCnt, countdown to zero will trigger a reset . [WR] 0x3~0x3ffffff;</p> <p>[31:27] Reserved.</p>	WathDog Timer
SFT	0x14	<p>Shift Controller</p> <p>[1:0] Shifter Command:[W] 00: Set master reset. 01: Shift register. 10: Storage register. 11: Output enable.</p> <p>[2] Output enable value, Only valid when Shifter Command == 2'b11(Low Active),default=1.</p> <p>[3] Done.</p> <p>[7:4] Reserved.</p> <p>[15:8] Data Register.</p> <p>[31:16] Reserved.</p>	Shift Controller
FAN0	0x18	<p>FAN0 Speed[R]</p> <p>[25:0] Fan0 speed, count of neg-edge within 1 second;[R]</p> <p>[31:26] Reserved.</p>	Fan0 Speed
FAN1	0x1c	<p>FAN1 Speed[R]</p> <p>[25:0] Fan1 speed, count of neg-edge within 1 second;[R]</p> <p>[31:26] Reserved.</p>	Fan 1 Speed
TIME	0x20	<p>Timer0 and Timer1[WR]</p> <p>[0] Timer0 load;</p> <p>[1] Timer0 INT MASK; default is 1.</p> <p>[7:2] Timer0 Seconds number.</p> <p>[8] Timer0 Done. Write to clear.</p> <p>[15:9] Reserved.</p> <p>[16] Timer1 load;</p> <p>[17] Timer1 INT MASK; Default is 1.</p> <p>[23:18] Timer1 Seconds number.</p> <p>[24] Timer1 Done. Write to clear.</p> <p>[31:25] Reserved.</p>	Timer0 and Timer1

6.3 Parameter Description for TMP102

More information please see <http://www.ti.com.cn/cn/lit/ds/symlink/tmp102.pdf>



6.4 Test Case

6.4.1 TWI Test Case

```
//TWI_CR = 0x00 ; TWI Ctrl Register
//TWI_WD = 0x04 ; TWI Write Data
//TWI_RD = 0x08 ; TWI Read Data
//SLV_ADDR[7:0]

void twi_start( void ){
    TWI_CR = 0x03 ;
    while( (TWI_CR & 0x04) != 0x04);
}

void twi_wr( char buf ){
    TWI_WD = buf ;
    TWI_CR = 0x13 ;
    while( (TWI_CR & 0x04) != 0x04);
}

char twi_rd( void ){
    TWI_CR = 0x23 ;
    while( (TWI_CR & 0x04) != 0x04);
    return TWI_RD ;
}

void twi_stop( void ){
    TWI_CR = 0x33 ;
    while( (TWI_CR & 0x04) != 0x04);
}
```

```
void twi_write_2byte( unsigned int buf )
{
    twi_start() ;
    twi_wr(SLV_ADDR) ;//slave addr
    twi_wr(0x00) ;//register addr
    twi_wr(buf) ;
    twi_wr(buf>>8) ;
    twi_stop() ;
}

unsigned int twi_read_2byte( void )
{
    unsigned int tmp ;
    twi_start();
    twi_wr(SLV_ADDR) ;//slave addr
    twi_wr(0x00) ;//register addr
    twi_start();
    twi_wr(SLV_ADDR|0x1) ;//slave addr + read
    tmp = twi_rd() ;
    tmp = (twi_rd() <<8)|tmp;
    twi_stop();
    return tmp;
}
```

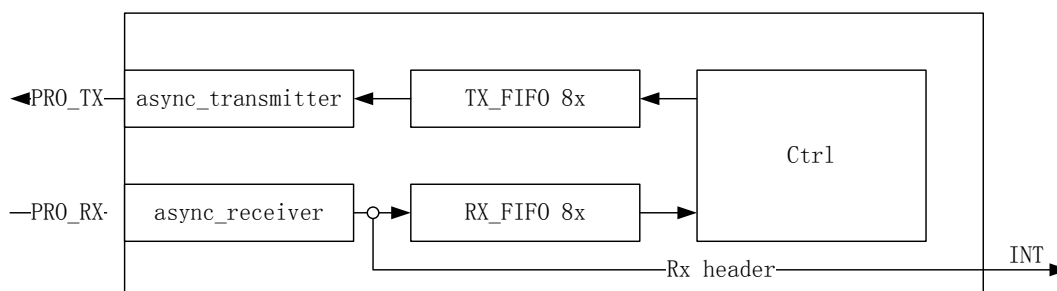
6.4.2 WDT Test Case

```
WatchDog = 0x1 | 0xff<<1; //Enable Watch Dog Timer, and feed 0xff clocks.
//wait...
WatchDog = 0x1 | 0xff<<1; //Enable Watch Dog Timer, and feed 0xff clocks again.
```

7. UART Pro

Note, define UART_PRO_EN to enable UART Pro.

7.1 Architecture



7.2 Register Description (BaseAddress: 0x80000100)

Register Name	Address	Address Offset within Register Word	Description
FIFO	0x00	TWI Ctrl Register [0] RX-FIFO Get 39Byte, Write 1 to clear.(With Int). [1] RX-FIFO INT_MASK. [2] RX-FIFO Empty. [3] Reset, Write 1 to soft reset. [13:4] RX-FIFO Count. [15:14] Reserved. [16] TX-FIFO Empty. [17] TX-FIFO Full. [29:20] TX-FIFO Count. [31:30] Reserved. Example: <pre> Interrupt_func(){ Set INT_MASK = 1 ; Read rxfifo; Clear Interrupt; Set INT_MASK = 0 ; } </pre>	FIFO Register
RXFIFO	0x04	Rx-FIFO Data POP[R] [7:0] RX-FIFO Data Output; [31:8] Reserved.	RX-FIFO POP

TXFIFO	0x08	Tx-FIFO Data PUSH[W] [7:0] TX-FIFO Data Input; [31:8] Reserved.	TX-FIFO PUSH
--------	------	--	--------------

8. Interrupt Define

No.	Function
0	GPIO
1	SPI
2	Reserved
3	Uart
4	Uart for debug
5	Reserved
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved
16	Reserved
17	Reserved
18	Reserved
19	Reserved
20	Reserved
21	Reserved
22	Reserved
23	Reserved
24	Reserved
25	Reserved
26	Reserved
27	Reserved
28	Reserved
29	Reserved
30	Reserved
31	Reserved

9. UCF Define

```

# Global controls
NET "ex_clk_i" LOC = J16;
NET "ex_clk_o" LOC = A9;

#P1<External>
#GND GND GND GND GND GND GND GND GND GND
#GND E16 E15 F16 F15 G16 H15 H16 J14 GND
#           |<-----pull-up----->|

# UART <-> FT232
NET "uartSIN"  LOC = C8; # FT232' s TX
NET "uartSOUT" LOC = A12; # FT232' s RX

NET "uartSIN_PIN"  LOC = F16; # FT232' s TX
NET "uartSOUT_PIN" LOC = F15; # FT232' s RX

# UART debug
NET "uart_debugSIN" LOC = E16;
NET "uart_debugSOUT" LOC = E15;

# aLink
NET  "RX_P[0]" LOC = N3;
NET  "RX_N[0]" LOC = P2;
NET  "TX_N[0]" LOC = R1 | SLEW = QUIETIO;
NET  "TX_P[0]" LOC = T5 | SLEW = QUIETIO;

NET  "RX_P[1]" LOC = R5;
NET  "RX_N[1]" LOC = P1;
NET  "TX_N[1]" LOC = R2 | SLEW = QUIETIO;
NET  "TX_P[1]" LOC = T4 | SLEW = QUIETIO;

NET  "RX_P[2]" LOC = H3;
NET  "RX_N[2]" LOC = K2;
NET  "TX_N[2]" LOC = H2 | SLEW = QUIETIO;
NET  "TX_P[2]" LOC = H1 | SLEW = QUIETIO;

NET  "RX_P[3]" LOC = F2;
NET  "RX_N[3]" LOC = G1;
NET  "TX_N[3]" LOC = L3 | SLEW = QUIETIO;
NET  "TX_P[3]" LOC = M2 | SLEW = QUIETIO;

```

```

NET "RX_P[4]" LOC = E2;
NET "RX_N[4]" LOC = E1;
NET "TX_N[4]" LOC = C2 | SLEW = QUIETIO;
NET "TX_P[4]" LOC = D1 | SLEW = QUIETIO;

NET "RX_P[5]" LOC = B2;
NET "RX_N[5]" LOC = C1;
NET "TX_N[5]" LOC = E3 | SLEW = QUIETIO;
NET "TX_P[5]" LOC = F1 | SLEW = QUIETIO;

NET "RX_P[6]" LOC = N4;
NET "RX_N[6]" LOC = M1;
NET "TX_N[6]" LOC = A2 | SLEW = QUIETIO;
NET "TX_P[6]" LOC = B1 | SLEW = QUIETIO;

NET "RX_P[7]" LOC = A4;
NET "RX_N[7]" LOC = A3;
NET "TX_N[7]" LOC = P4 | SLEW = QUIETIO;
NET "TX_P[7]" LOC = N1 | SLEW = QUIETIO;

NET "RX_P[8]" LOC = J4;
NET "RX_N[8]" LOC = K1;
NET "TX_N[8]" LOC = H4 | SLEW = QUIETIO;
NET "TX_P[8]" LOC = J1 | SLEW = QUIETIO;

NET "RX_P[9]" LOC = F4;
NET "RX_N[9]" LOC = G3;
NET "TX_N[9]" LOC = M4 | SLEW = QUIETIO;
NET "TX_P[9]" LOC = L1 | SLEW = QUIETIO;

NET "SFT_SHCP" LOC = K16 ;
NET "SFT_DS" LOC = M15 ;
NET "SFT_STCP" LOC = K15 ;
NET "SFT_MR_N" LOC = K14 ;
NET "SFT_OE_N" LOC = L16 ;

NET "FAN_IN0" LOC = A6 ;
NET "FAN_IN1" LOC = A7 ;

# LED gpioPIO_BOTH_OUT
NET "gpioPIO_BOTH_OUT[0]" LOC = B12 | SLEW = QUIETIO;
NET "gpioPIO_BOTH_OUT[1]" LOC = A11 | SLEW = QUIETIO;
NET "gpioPIO_BOTH_OUT[2]" LOC = B10 | SLEW = QUIETIO;
NET "gpioPIO_BOTH_OUT[3]" LOC = A10 | SLEW = QUIETIO;

```

```
NET "gpioPIO_BOTH_OUT[4]" LOC = B14 | SLEW = QUIETIO;
#ID
NET "gpioPIO_BOTH_OUT[5]" LOC = A14 | SLEW = QUIETIO;
NET "gpioPIO_BOTH_OUT[6]" LOC = C9 | SLEW = QUIETIO;

NET "INT" LOC = B16 | SLEW = QUIETIO;
NET "uartSIN_led" LOC = C16 | SLEW = QUIETIO;
NET "uartSOUT_led" LOC = D16 | SLEW = QUIETIO;

NET "TWI_SCL" LOC = D3 ;
NET "TWI_SDA" LOC = E4 ;

# SPI Flash: W25Q80BV
NET "POWER_ON" LOC = T3 | SLEW = QUIETIO;
NET "PWM" LOC = A5 | SLEW = QUIETIO;

NET "*" IOSTANDARD = LVCMOS33;

NET "ex_clk_i" TNM_NET = "ex_clk_i";
TIMESPEC TS_clk = PERIOD "ex_clk_i" 40 ns HIGH 50 %;

NET "clk_i" TNM_NET = "clk_i";
TIMESPEC TS_clk_i = PERIOD "clk_i" 10 ns HIGH 50 %;
```