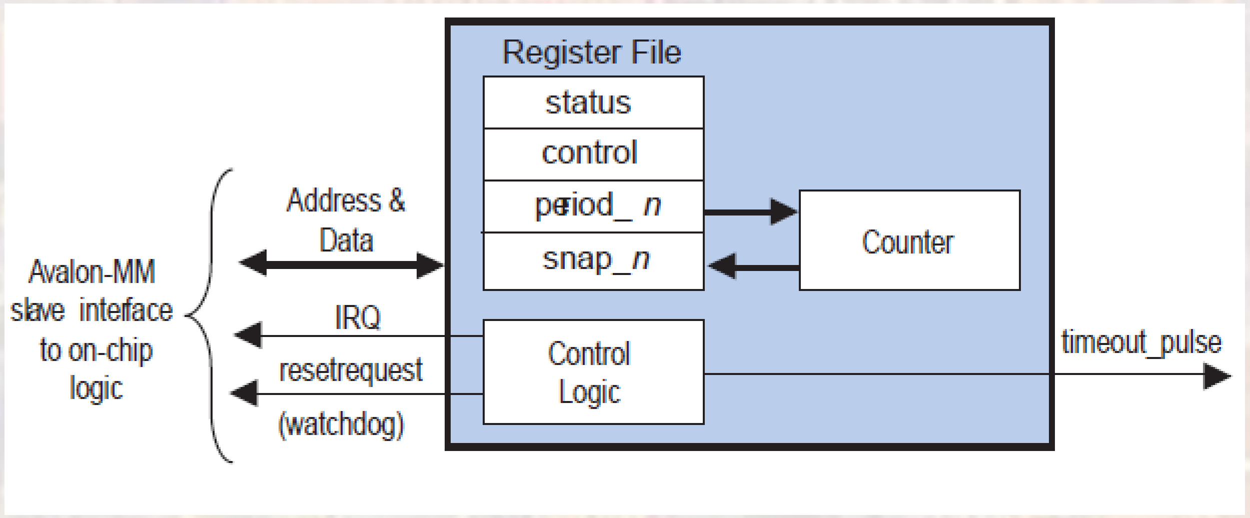
NIOS Peripherals
Interval Timer

Last update 7/20/23

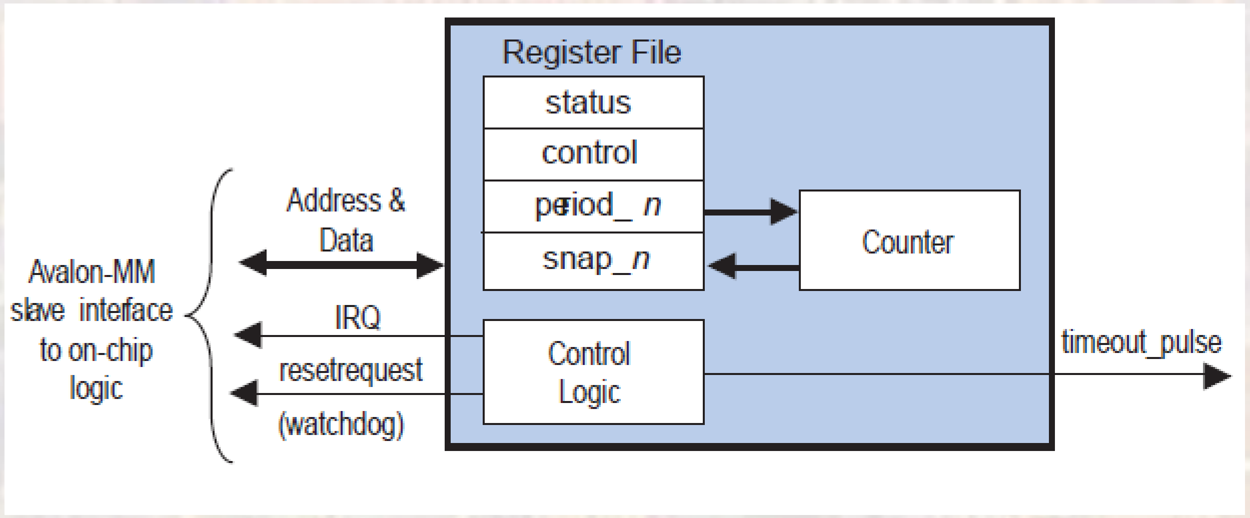
NIOS Peripherals – Interval Time

- Interval Timer
  - Count down timer
    - count down once
    - count down continuously
  - Can create an interrupt
  - Can create a watchdog timer reset
  - Can create a pulse



NIOS Peripherals – Interval Timer

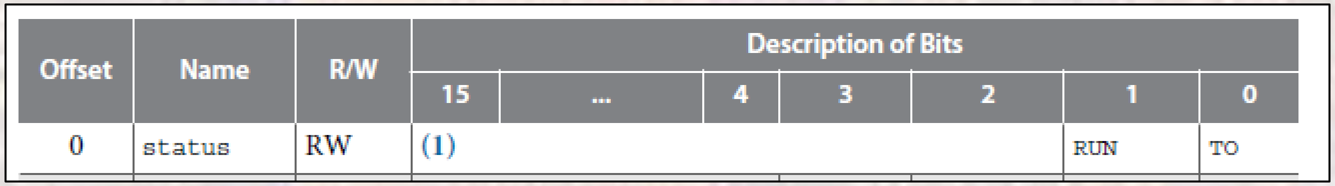
- Interval Timer
  - 32 and 64 bit timer counters
  - R/W is through 2 or 4 16 bit registers
  - Control is through a control register



NIOS Peripherals – Interval Timer

- Interval Timer
  - Status Register

  - Provides current status



Offset	Name	R/W	Description of Bits						
			15	...	4	3	2	1	0
0	status	RW	(1)					RUN	TO

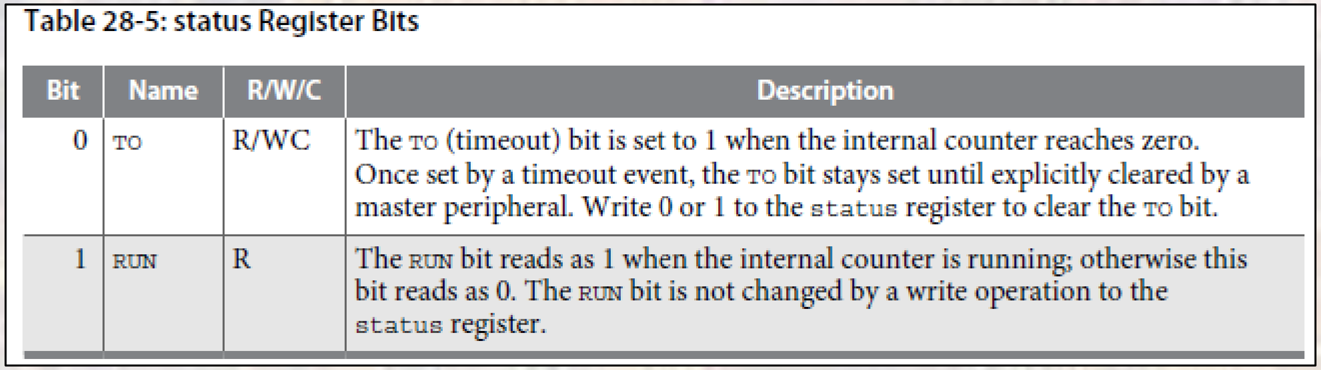
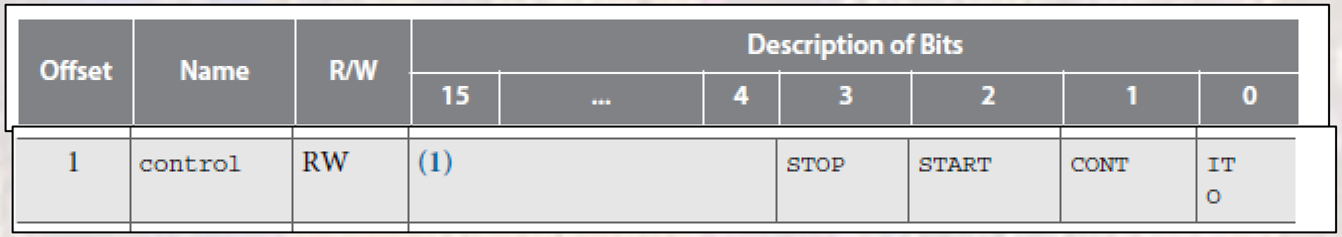


Table 28-5: status Register Bits

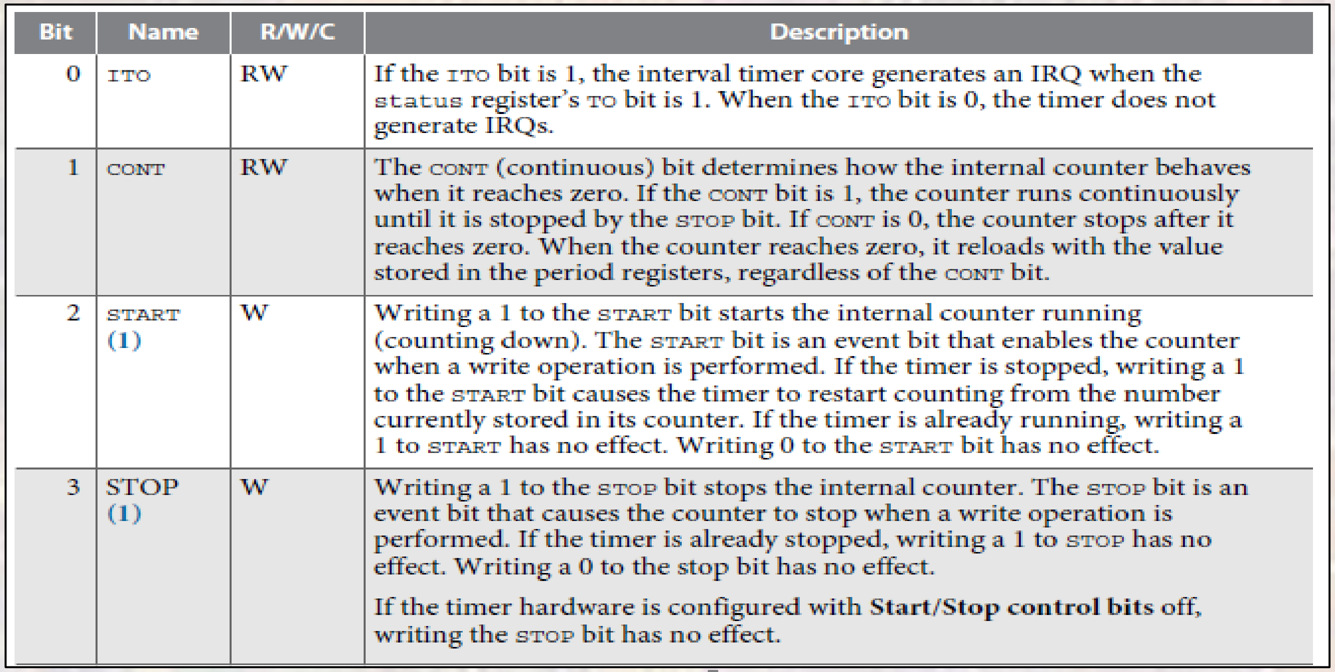
Bit	Name	R/W/C	Description
0	TO	R/WC	The TO (timeout) bit is set to 1 when the internal counter reaches zero. Once set by a timeout event, the TO bit stays set until explicitly cleared by a master peripheral. Write 0 or 1 to the status register to clear the TO bit.
1	RUN	R	The RUN bit reads as 1 when the internal counter is running; otherwise this bit reads as 0. The RUN bit is not changed by a write operation to the status register.

NIOS Peripherals – Interval Timer

- Interval Timer
  - Control Register
  - Controls Timer operation



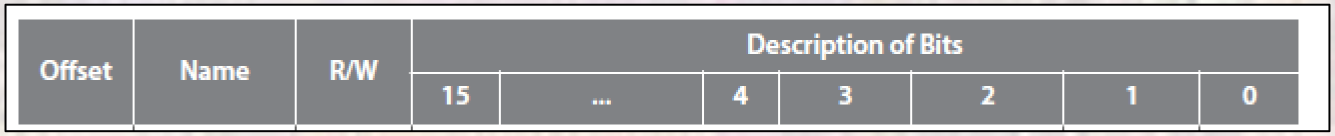
Offset	Name	R/W	Description of Bits						
			15	...	4	3	2	1	0
1	control	RW	(1)			STOP	START	CONT	ITO

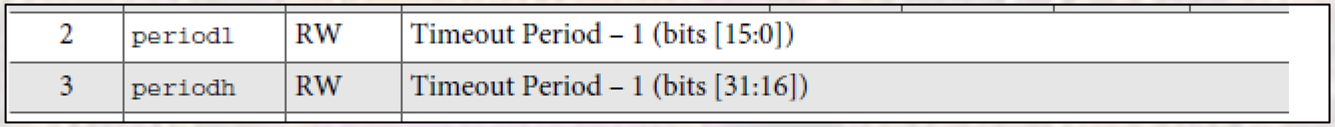


Bit	Name	R/W/C	Description
0	ITO	RW	If the ITO bit is 1, the interval timer core generates an IRQ when the status register's TO bit is 1. When the ITO bit is 0, the timer does not generate IRQs.
1	CONT	RW	The CONT (continuous) bit determines how the internal counter behaves when it reaches zero. If the CONT bit is 1, the counter runs continuously until it is stopped by the STOP bit. If CONT is 0, the counter stops after it reaches zero. When the counter reaches zero, it reloads with the value stored in the period registers, regardless of the CONT bit.
2	START (1)	W	Writing a 1 to the START bit starts the internal counter running (counting down). The START bit is an event bit that enables the counter when a write operation is performed. If the timer is stopped, writing a 1 to the START bit causes the timer to restart counting from the number currently stored in its counter. If the timer is already running, writing a 1 to START has no effect. Writing 0 to the START bit has no effect.
3	STOP (1)	W	Writing a 1 to the STOP bit stops the internal counter. The STOP bit is an event bit that causes the counter to stop when a write operation is performed. If the timer is already stopped, writing a 1 to STOP has no effect. Writing a 0 to the stop bit has no effect. If the timer hardware is configured with **Start/Stop control bits** off, writing the STOP bit has no effect.

NIOS Peripherals – Interval Timer

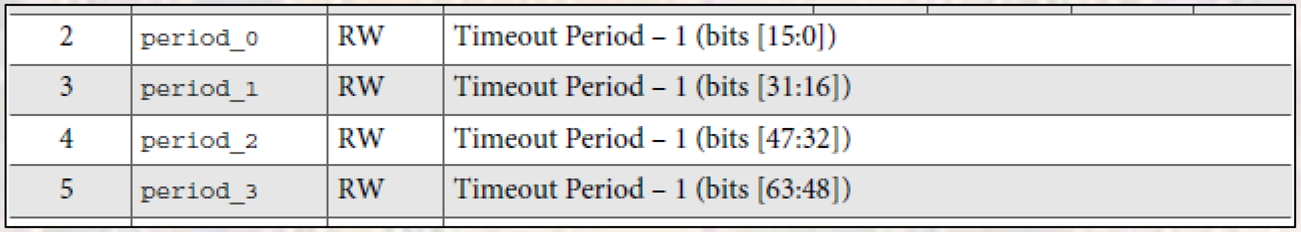
- Interval Timer
  - Period Registers
  - Hold the counter initialization value





Offset	Name	R/W	Description of Bits						
			15	...	4	3	2	1	0
2	periodl	RW	Timeout Period – 1 (bits [15:0])						
3	periodh	RW	Timeout Period – 1 (bits [31:16])						

32 bit

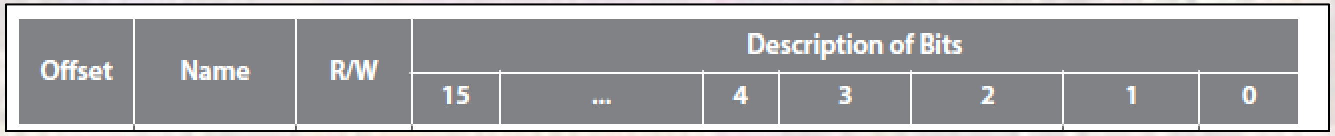


2	period_0	RW	Timeout Period – 1 (bits [15:0])
3	period_1	RW	Timeout Period – 1 (bits [31:16])
4	period_2	RW	Timeout Period – 1 (bits [47:32])
5	period_3	RW	Timeout Period – 1 (bits [63:48])

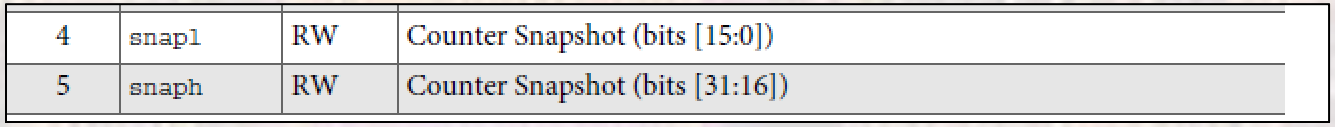
64 bit

NIOS Peripherals – Interval Timer

- Interval Timer
  - Snap Registers
  - Snapshot of the current counter value
    - Write to snap register to request current count
    - Read snap register to get the value

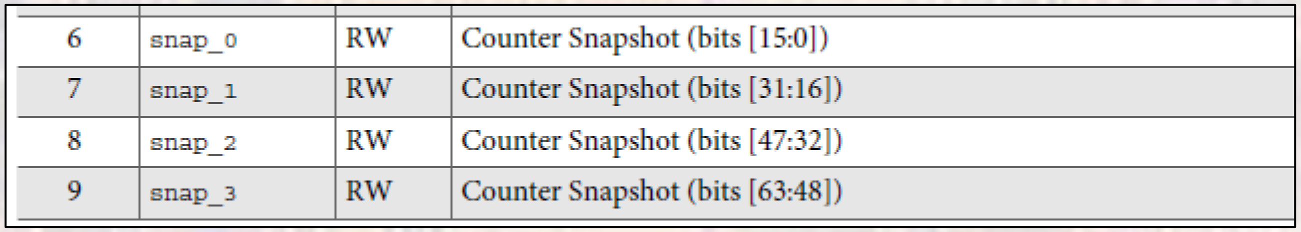


Offset	Name	R/W	Description of Bits						
			15	...	4	3	2	1	0



4	snapl	RW	Counter Snapshot (bits [15:0])
5	snaph	RW	Counter Snapshot (bits [31:16])

32 bit



6	snap_0	RW	Counter Snapshot (bits [15:0])
7	snap_1	RW	Counter Snapshot (bits [31:16])
8	snap_2	RW	Counter Snapshot (bits [47:32])
9	snap_3	RW	Counter Snapshot (bits [63:48])

64 bit

NIOS Peripherals – Interval Timer

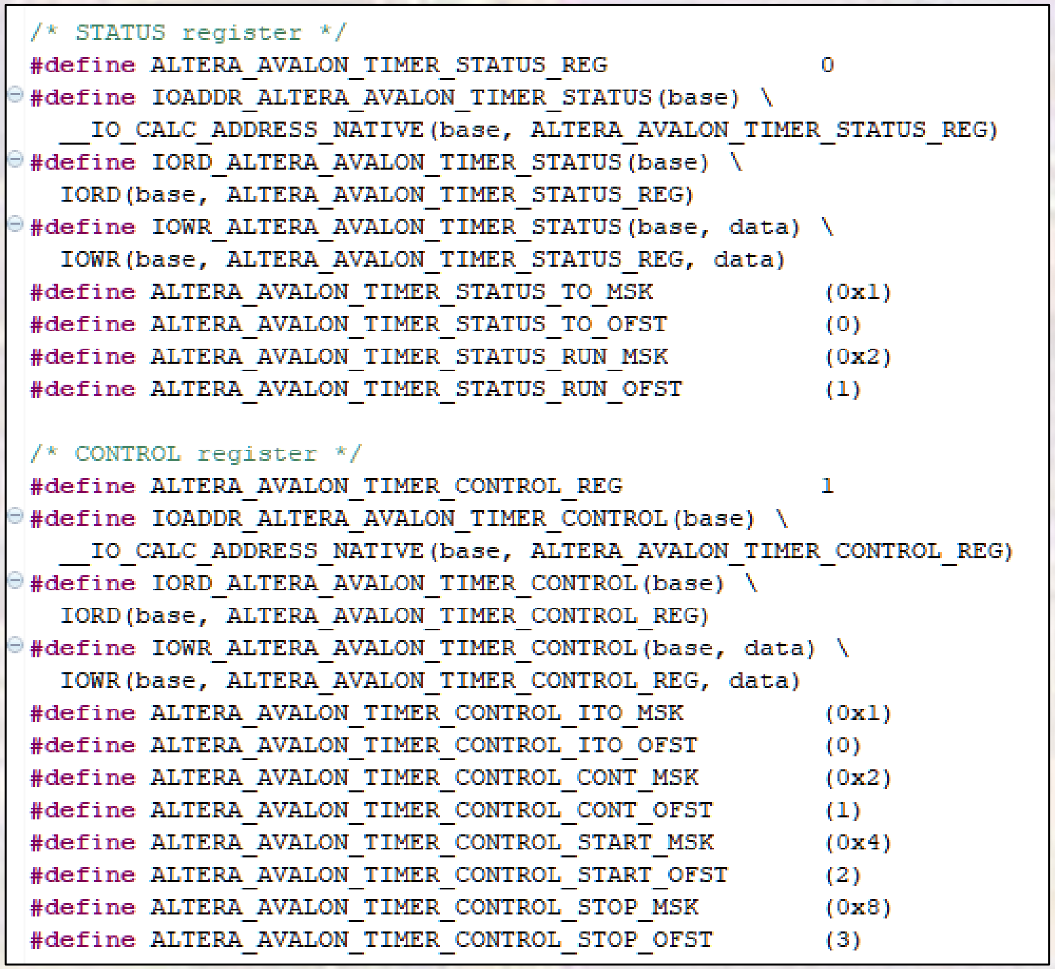
- Interval Timer
  - Interrupts
  - If enabled
    - Creates interrupts on timeout
    - Clear interrupt by writing to the TO bit in the status register
    - Read snap register to get the value

NIOS Peripherals – Interval Timer

- Interval Timer
  - Watchdog
  - In Qsys
    - Set the Timeout Period to the desired "watchdog" period.
    - Turn off Writeable period.
    - Turn off Readable snapshot.
    - Turn off Start/Stop control bits.
    - Turn off Timeout pulse.
    - Turn on System reset on timeout (watchdog).

NIOS Peripherals – Interval Timer

- Interval Timer
  - altera_avalon_timer_regs.h

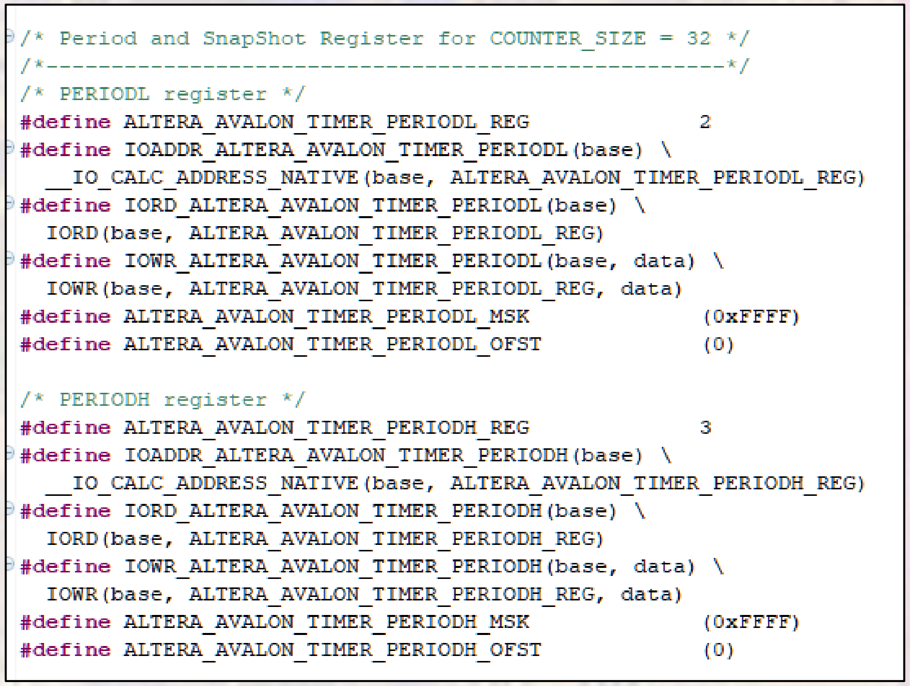


```
/* STATUS register */
#define ALTERA_AVALON_TIMER_STATUS_REG              0
#define IOADDR_ALTERA_AVALON_TIMER_STATUS(base) \
  __IO_CALC_ADDRESS_NATIVE(base, ALTERA_AVALON_TIMER_STATUS_REG)
#define IORD_ALTERA_AVALON_TIMER_STATUS(base) \
  IORD(base, ALTERA_AVALON_TIMER_STATUS_REG)
#define IOWR_ALTERA_AVALON_TIMER_STATUS(base, data) \
  IOWR(base, ALTERA_AVALON_TIMER_STATUS_REG, data)
#define ALTERA_AVALON_TIMER_STATUS_TO_MSK           (0x1)
#define ALTERA_AVALON_TIMER_STATUS_TO_OFST          (0)
#define ALTERA_AVALON_TIMER_STATUS_RUN_MSK          (0x2)
#define ALTERA_AVALON_TIMER_STATUS_RUN_OFST         (1)

/* CONTROL register */
#define ALTERA_AVALON_TIMER_CONTROL_REG             1
#define IOADDR_ALTERA_AVALON_TIMER_CONTROL(base) \
  __IO_CALC_ADDRESS_NATIVE(base, ALTERA_AVALON_TIMER_CONTROL_REG)
#define IORD_ALTERA_AVALON_TIMER_CONTROL(base) \
  IORD(base, ALTERA_AVALON_TIMER_CONTROL_REG)
#define IOWR_ALTERA_AVALON_TIMER_CONTROL(base, data) \
  IOWR(base, ALTERA_AVALON_TIMER_CONTROL_REG, data)
#define ALTERA_AVALON_TIMER_CONTROL_ITO_MSK         (0x1)
#define ALTERA_AVALON_TIMER_CONTROL_ITO_OFST        (0)
#define ALTERA_AVALON_TIMER_CONTROL_CONT_MSK        (0x2)
#define ALTERA_AVALON_TIMER_CONTROL_CONT_OFST       (1)
#define ALTERA_AVALON_TIMER_CONTROL_START_MSK       (0x4)
#define ALTERA_AVALON_TIMER_CONTROL_START_OFST      (2)
#define ALTERA_AVALON_TIMER_CONTROL_STOP_MSK        (0x8)
#define ALTERA_AVALON_TIMER_CONTROL_STOP_OFST       (3)
```

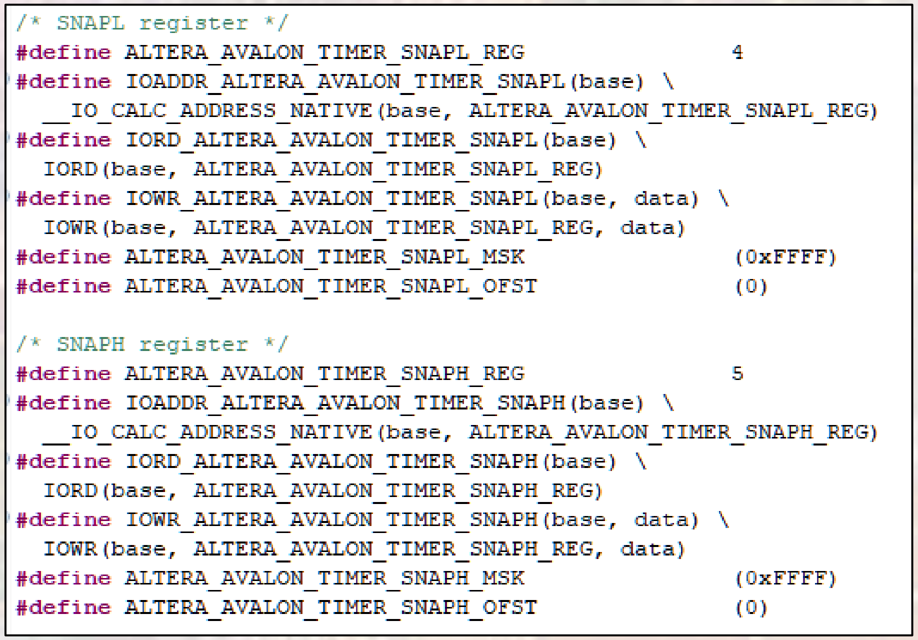
NIOS Peripherals – Interval Timer

- Interval Timer
  - altera_avalon_timer_regs.h



```
/* Period and SnapShot Register for COUNTER_SIZE = 32 */
/*----------------------------------------------------*/
/* PERIODL register */
#define ALTERA_AVALON_TIMER_PERIODL_REG             2
#define IOADDR_ALTERA_AVALON_TIMER_PERIODL(base) \
    __IO_CALC_ADDRESS_NATIVE(base, ALTERA_AVALON_TIMER_PERIODL_REG)
#define IORD_ALTERA_AVALON_TIMER_PERIODL(base) \
  IORD(base, ALTERA_AVALON_TIMER_PERIODL_REG)
#define IOWR_ALTERA_AVALON_TIMER_PERIODL(base, data) \
  IOWR(base, ALTERA_AVALON_TIMER_PERIODL_REG, data)
#define ALTERA_AVALON_TIMER_PERIODL_MSK             (0xFFFF)
#define ALTERA_AVALON_TIMER_PERIODL_OFST            (0)

/* PERIODH register */
#define ALTERA_AVALON_TIMER_PERIODH_REG             3
#define IOADDR_ALTERA_AVALON_TIMER_PERIODH(base) \
    __IO_CALC_ADDRESS_NATIVE(base, ALTERA_AVALON_TIMER_PERIODH_REG)
#define IORD_ALTERA_AVALON_TIMER_PERIODH(base) \
  IORD(base, ALTERA_AVALON_TIMER_PERIODH_REG)
#define IOWR_ALTERA_AVALON_TIMER_PERIODH(base, data) \
  IOWR(base, ALTERA_AVALON_TIMER_PERIODH_REG, data)
#define ALTERA_AVALON_TIMER_PERIODH_MSK             (0xFFFF)
#define ALTERA_AVALON_TIMER_PERIODH_OFST            (0)
```



```
/* SNAPL register */
#define ALTERA_AVALON_TIMER_SNAPL_REG               4
#define IOADDR_ALTERA_AVALON_TIMER_SNAPL(base) \
    __IO_CALC_ADDRESS_NATIVE(base, ALTERA_AVALON_TIMER_SNAPL_REG)
#define IORD_ALTERA_AVALON_TIMER_SNAPL(base) \
  IORD(base, ALTERA_AVALON_TIMER_SNAPL_REG)
#define IOWR_ALTERA_AVALON_TIMER_SNAPL(base, data) \
  IOWR(base, ALTERA_AVALON_TIMER_SNAPL_REG, data)
#define ALTERA_AVALON_TIMER_SNAPL_MSK               (0xFFFF)
#define ALTERA_AVALON_TIMER_SNAPL_OFST              (0)

/* SNAPH register */
#define ALTERA_AVALON_TIMER_SNAPH_REG               5
#define IOADDR_ALTERA_AVALON_TIMER_SNAPH(base) \
    __IO_CALC_ADDRESS_NATIVE(base, ALTERA_AVALON_TIMER_SNAPH_REG)
#define IORD_ALTERA_AVALON_TIMER_SNAPH(base) \
  IORD(base, ALTERA_AVALON_TIMER_SNAPH_REG)
#define IOWR_ALTERA_AVALON_TIMER_SNAPH(base, data) \
  IOWR(base, ALTERA_AVALON_TIMER_SNAPH_REG, data)
#define ALTERA_AVALON_TIMER_SNAPH_MSK               (0xFFFF)
#define ALTERA_AVALON_TIMER_SNAPH_OFST              (0)
```