



- **Interrupt control**

---

- 

---

- 

---

- 

---

- Application Note

---

## Index of contents

|          |   |           |
|----------|---|-----------|
| <b>0</b> | <b>INTRODUCTION.....</b>                            | <b>5</b>  |
| 0.1      | UART'S .....  | 5         |
| 0.2      | ASIC INTERRUPTS .....                               | 5         |
| 0.3      | TIMER INTERRUPT .....                               | 5         |
| 0.4      | UART INTERRUPT .....                                | 6         |
| 0.5      | GPIO LINES .....                                    | 6         |
| 0.5.1    | The Real Time Clock.....                            | 9         |
| 0.5.2    | Interrupts .....                                    | 9         |
| 0.5.3    | BUS Interface.....                                  | 9         |
| 0.6      | CHIP SELECT WAIT STATES .....                       | 9         |
| 0.7      | DIGITAL CHIP ADDRESS SPACE.....                     | 9         |
| 0.8      | INTERRUPT CONTROL MODULE.....                       | 11        |
| 0.8.1    | Interrupt Controller Functional Description .....   | 12        |
| 0.9      | BIU CHIP SELECT REGISTERS .....                     | 12        |
| 0.10     | PAUSE CONTROL FUNCTION.....                         | 12        |
| 0.11     | INTERRUPT CONTROLLER REGISTER MAPPING .....         | 13        |
| 0.12     | PULSE INTERRUPT MANAGEMENT .....                    | 14        |
| 0.13     | DATA_READY INTERRUPT .....                          | 14        |
| 0.14     | UART INTERRUPT .....                                | 15        |
| 0.15     | TIMER INTERRUPTS.....                               | 16        |
| 0.16     | TIMER INTERRUPT ACKNOWLEDGE .....                   | 16        |
| 0.16.1   | Beacon Interrupt .....                              | 16        |
| 0.16.2   | External Interrupt Active Level.....                | 17        |
| 0.16.3   | UARTModule .....                                    | 17        |
| <b>1</b> | <b>THREE- WIRE INTERFACE .....</b>                  | <b>21</b> |
| 1.1      | THREE-WIRE MASTER MODE .....                        | 22        |
| 1.1.1    | Three-Wire Slave Mode .....                         | 22        |
| 1.2      | GPIO .....  | 24        |
| 1.2.1    | GPIO Sel .....                                      | 25        |
| 1.2.2    | GPIO State.....                                     | 26        |
| 1.2.3    | GPIO PortVal .....                                  | 27        |
| 1.2.4    | GPIO PortDir.....                                   | 28        |
| 1.3      | INPUTS/OUTPUTS .....                                | 28        |
| 1.4      | SCHEMATICS OF VARIOUS GPIO PIN CONFIGURATIONS ..... | 29        |
| 1.5      | EVAL-BOARD LED ACTIVATION.....                      | 31        |
| 1.6      | USED ABBREVIATIONS .....                            | 32        |

## Version history:

| Version number | Author       | Changes   |
|----------------|--------------|---|
| 1.00           | Fadil Beqiri | - Initial version                                   |
| 1.01           | Fadil Beqiri | - Table 1 in chapter 0.5 updated.                   |
| 1.02           | Fadil Beqiri | - Chapter 1.3 added (functions for inputs/outputs). |

## Cautions

Information furnished herein by FALCOM are accurate and reliable. However, no responsibility is assumed for its use.

Please, read carefully the safety precautions.

If you have any technical questions regarding this document or the product described in it, please contact your vendor.

General information about FALCOM and its range of products are available at the following Internet address: <http://www.falcom.de/>

## Trademarks

Some mentioned products are registered trademarks of their respective companies.

## Copyright

This description is copyrighted by FALCOM GmbH with all rights reserved. No part of this user's guide may be produced in any form without the prior written permission of FALCOM GmbH.

## FALCOM GmbH.

No patent liability is assumed with respect to the use of the information contained herein.

## 0 INTRODUCTION

The ARM7TDMI Microprocessor integrated to the Falcom GPS receiver is a member of the Advanced RISC Machines (ARM) family of general purpose, 32-bit microprocessors, which offer high performance for very low power consumption and price. The ARM7TDMI runs at a maximum speed of 50 MHz to deliver about 40 million instructions per second.

The Module Interface (MI) module is available as object code and provides an interface mechanism between the GPS Core and the User Interface (UI) Module controlling the I/O protocol for communicating with external devices. In general, the UI is driven by events that are signaled by the GPS Core. When these events are received, the generic UI manager code determines the current protocol and calls an appropriate function in that protocol to handle the event. Inside the current I/O protocol, the event may signal that system information is supposed to be output.

### 0.1 UART'S

The embedded internal GPS receiver into the XF55-AVL and STEPP II devices provides two full duplex serial ports that are typically used for the following purposes:

- ❖ Port 1: GPS serial data output and receiver control
- ❖ Port 2: Reception of differential corrections (the STEPP II terminal uses this port to communicate with the GSM serial port)

Both UARTs are full duplex with both transmit and receive sides containing a 16-byte deep FIFO. The Baud rates on both channels are independently selectable from 1200 bps to 115.2 kbps. The parity, data bits and stop bits are also configurable. Each receive register provides error signals for frame error, parity error, and break interrupt.

### 0.2 ASIC Interrupts

The satellite signal tracking loop processing is handled by the satellite signal tracking engine (SSTE) implemented in hardware. The fastest periodic interrupt generated by the embedded receiver is at a 100 ms rate. This rate is greatly reduced compared to the previous generations of GPS ASICs requiring 1 ms interrupt servicing. As a result of this, the software scheduler currently runs at a maximum rate of 10 Hz (100 ms). The digital part of embedded receiver is still capable of generating both 1 ms and 20 ms interrupts. The embedded receiver has two interrupt levels, IRQ and FIQ, with FIQ being the highest. All ASIC interrupts are generated on the same ARM IRQ interrupt level.

### 0.3 Timer Interrupt

Timer interrupts are generated periodically based on the system clock (different from the processor clock). The timing of this interrupt is controlled by software. Currently it is set at 100 ms which is the minimum rate fully supported by the Satellite Signal Tracking Engine, or Tracker (SSTE). The GPS software tasks are scheduled from this interrupt handler, including the 100 ms task, the 1 Hz execution task, and the background task (scheduled at 1 Hz). The 100 ms task performs processing of measurement data of channels that are tracking, or processing of reacquisition data if the channels are in search. The 1 Hz execution task controls the receiver manager,

50 bps data processing, navigation, and user interface through the serial communication port.

## 0.4 UART Interrupt

UART interrupts are generated by the UART on the ASIC whenever there is data to be transmitted or data is received. The processing of these interrupts is done by the UART functions. These functions call the appropriate transmit or receive processing for UART A and/or UART B. The data is buffered and passed to the application layer (user interface) after the GPS core has signaled a `MI_EV_NAV_COMPLETE` or `MI_EV_WAIT_INITIAL_ACQ` event.

## 0.5 GPIO Lines

The XF55-AVL provides 16 GPIO lines, whereas the STEPP II terminal provides 4 GPIO lines. Most GPIO pins are multi-functional and are often used for other purposes. Examples include chip selects, debugging support pins, microwire interface and external interrupts. Some of the GPIO lines are also used for power control during Low Power operation and may not be available. To enable the alternate function(s) of a GPIO, the `ASIC_GPIO_SEL( 0x80010100)` register must be modified. For information on how the Eval-Board LEDs are controlled using GPIO lines see "Eval-Board LED Activation" chapter 1.5.

- One of the first actions is to initialize the `GPIO_SEL` register (0x80010100). `GPIO_SEL` is also referred to as `ASIC_GPIO_SEL`. The GPIOs select register is declared in the `PORTS.H` extended file, and is mapped from `ASIC_BASE_ADDR (0x80000000) + 0x10100`.

Tables 1 and 2 below list the GPIO lines and their alternate functions for the embedded GPS receiver. The Alternate Enable Bit is the bit in the `ASIC_GPIO_SEL` that must be set high to enable the alternate function. When alternate input is not selected, the default input is specified in the Default Alternate Input Value column. On most GPIO lines, either a pull-up or pull-down resistor is built into the chip of receiver.

| PIN Name | Default alternative input Value | Internal Pull-up/ Pull-down | NOTE  |          |
|----------|---------------------------------|-----------------------------|---|----------|
|          |                                 |                             | STEPPII   | XF55-AVL |
| GPIO0    | 0                               | Pull down                   | This pin (as an input one) is internally wired to the microcontroller. Nitron microcontroller response for the shutdown of GPS processor.   | PIN 32   |
| GPIO1    | 0                               | Pull down                   | A watchdog logic implemented in the microcontroller periodically receives low pulses from the this GPIO1-pin, which is internally wired to the microcontroller. The low pulses have to be sent to the GPIO1-pin. These alternate pulses allow the STEPP II device to maintain itself in the normal operation mode. If these alternate pulses fail to arrive the watchdog periodically (at least a 100 ms pulse within 120 seconds time interval, recommended a 100ms pulse every second), the STEPP II terminal forces internally a reset. The GPIO1 pin has to be programmed as an output pin. | PIN 31   |
| GPIO2    | 0                               | Pull down                   | -   | -        |

|        |   |           |   |  |
|--------|---|-----------|---|--|
| GPIO3  | - | Pull down | The GPIO3-pin is internally linked to the GSM_ON pin of the GSM part. By powering up the module the direction of GPIO3 has to be programmed as an output and set to high level (it enables the GSM engine to be switched on). To set the GSM_ON pin to low level, the value of GPIO3 signal has to be set to low for at least 20 seconds. | The GPIO3-pin is internally linked to the GSM_ON pin (PIN 66) of the GSM part. |
| GPIO4  | - | None      | The GPIO4 line is controlled by the internal GPS software and should not be programmed (steered) by the user; otherwise the GPS receiver will be not functional. This pin steers the RF-Front (analogue part) of GPS receiver and is defaulted to logic HIGH at power up.   |  |
| GPIO5  | 0 | Pull down | D/A converter IN  | PIN 30   |
| GPIO6  | - | Pull down | D/A converter OUT   | PIN 29   |
| GPIO7  | 0 | Pull down | D/A converter - CLK   | PIN 28   |
| GPIO8  | - | None      | The GPIO8-line is controlled by the internal GPS software, and it should not be programmed (steered) by the user; otherwise the GPS receiver will be not functional. This pin is connected to the RF regulator (analogue part) of GPS receiver and is defaulted to logic HIGH at power up.  |  |
| GPIO9  | - | Pull down | The GPIO9 line is predefined on the software as output (known as T_MARK pin on the GPS receiver.) This pin is active HIGH each second (povides 1 pulse per second), which is synchronized to within 1 microsecond of GPS time.  |  |
| GPIO10 | 1 | Pull up   | CS EPROM  | PIN 27   |
| GPIO11 | 1 | Pull up   | -   | -  |
| GPIO12 | 1 | Pull up   | -   | -  |
| GPIO13 | - | Pull up   | -   | -  |
| GPIO14 | - | Pull up   | -   | -  |
| GPIO15 | - | Pull up   | CS D/A converter  | PIN 26   |

Table 1: Lists of the GPIO lines

Please note that, in the table 2 the "Set Bit X to Y" means the X-th bit counting begins from 0, and from right to left. For more details, see chapter 1.2.

| PIN Name | Set Alternate/GPIO function   | Set Direction (Input/Output)  | Set Value (HIGH/LOW)<br>Bit(x) = 1;(x)=active HIGH  |
|----------|---|---|---|
| GPIO0    |   | Register<br>ASIC_GPIO_PORTDIR0= 80010134<br>As Input:<br>Set Bit 7 to 0<br>As Output:<br>Set Bit 7 to 1 | Register<br>ASIC_GPIO_PORTVAL0= 80010120<br>ON:<br>Set Bit 7 to 1<br>OFF:<br>Set Bit 7 to 0 |
| GPIO1    | Register<br>ASIC_GPIO_SEL=80010100<br>As Alternate:<br>Set Bit 0 to 1<br>As GPIO:<br>Set Bit 0 to 0 | Register<br>ASIC_GPIO_PORTDIR0= 80010134<br>As Input:<br>Set Bit 6 to 0<br>As Output:<br>Set Bit 6 to 1 | Register<br>ASIC_GPIO_PORTVAL0= 80010120<br>ON:<br>Set Bit 6 to 1<br>OFF:<br>Set Bit 6 to 0 |
| GPIO2    |   | Register<br>ASIC_GPIO_PORTDIR0= 80010134<br>As Input:<br>Set Bit 5 to 0<br>As Output:<br>Set Bit 5 to 1 | Register<br>ASIC_GPIO_PORTVAL0= 80010120<br>ON:<br>Set Bit 5 to 1<br>OFF:<br>Set Bit 5 to 0 |
| GPIO3    | Register<br>ASIC_GPIO_SEL=80010100<br>As Alternate:<br>Set Bit 1 to 1<br>As GPIO:                   | Register<br>ASIC_GPIO_STATE1= 80010108<br>As Input:<br>Set Bit 14 to 0<br>As Output:<br>Set Bit 14 to 1 | Register<br>ASIC_GPIO_STATE1= 80010108<br>ON:<br>Set Bit 15 to 1<br>OFF:<br>Set Bit 15 to 0 |

|        |   |   |   |
|--------|---|---|---|
| GPIO4  | Set Bit 1 to 0  | Register<br>ASIC_GPIO_STATE2= 8001010C<br>As Input:<br>Set Bit 14 to 0<br>As Output:<br>Set Bit 14 to 1 | Register<br>ASIC_GPIO_STATE2= 8001010C<br>ON:<br>Set Bit 15 to 1<br>OFF:<br>Set Bit 15 to 0 |
| GPIO5  |   | Register<br>ASIC_GPIO_PORTDIR0= 80010134<br>As Input:<br>Set Bit 2 to 0<br>As Output:<br>Set Bit 2 to 1 | Register<br>ASIC_GPIO_PORTVAL0= 80010120<br>ON:<br>Set Bit 2 to 1<br>OFF:<br>Set Bit 2 to 0 |
| GPIO6  | <u>Register</u><br>ASIC_GPIO_SE<br>L=80010100<br><u>As Alternate:</u><br>Set Bit 2 to 1<br>As GPIO:<br>Set Bit 2 to 0 | Register<br>ASIC_GPIO_PORTDIR0= 80010134<br>As Input:<br>Set Bit 1 to 0<br>As Output:<br>Set Bit 1 to 1 | Register<br>ASIC_GPIO_PORTVAL0= 80010120<br>ON:<br>Set Bit 1 to 1<br>OFF:<br>Set Bit 1 to 0 |
| GPIO7  |   | Register<br>ASIC_GPIO_PORTDIR0= 80010134<br>As Input:<br>Set Bit 0 to 0<br>As Output:<br>Set Bit 0 to 1 | Register<br>ASIC_GPIO_PORTVAL0= 80010120<br>ON:<br>Set Bit 0 to 1<br>OFF:<br>Set Bit 0 to 0 |
| GPIO8  | <u>Register</u><br>ASIC_GPIO_SE<br>L=80010100<br>X= (GPIO<br>number -5)<br><u>As Alternate:</u><br>Set Bit X to 1     | Register<br>ASIC_GPIO_STATE3= 80010110<br>As Input:<br>Set Bit 14 to 0<br>As Output:<br>Set Bit 14 to 1 | Register<br>ASIC_GPIO_STATE3= 80010110<br>ON:<br>Set Bit 15 to 1<br>OFF:<br>Set Bit 15 to 0 |
| GPIO9  | As GPIO:<br>Set Bit X to 0  | Register<br>ASIC_GPIO_STATE4= 80010114<br>As Input:<br>Set Bit 14 to 0<br>As Output:<br>Set Bit 14 to 1 | Register<br>ASIC_GPIO_STATE4= 80010114<br>ON:<br>Set Bit 15 to 1<br>OFF:<br>Set Bit 15 to 0 |
| GPIO10 |   | Register<br>ASIC_GPIO_STATE5= 80010118<br>As Input:<br>Set Bit 14 to 0<br>As Output:<br>Set Bit 14 to 1 | Register<br>ASIC_GPIO_STATE5= 80010118<br>ON:<br>Set Bit 15 to 1<br>OFF:<br>Set Bit 15 to 0 |
| GPIO11 |   | Register<br>ASIC_GPIO_STATE6= 8001011C<br>As Input:<br>Set Bit 14 to 0<br>As Output:<br>Set Bit 14 to 1 | Register<br>ASIC_GPIO_STATE6= 8001011C<br>ON:<br>Set Bit 15 to 1<br>OFF:<br>Set Bit 15 to 0 |
| GPIO12 |   | Register<br>ASIC_GPIO_PORTDIR1= 80010138<br>As Input:<br>Set Bit 7 to 0<br>As Output:<br>Set Bit 7 to 1 | Register<br>ASIC_GPIO_PORTVAL1= 80010124<br>ON:<br>Set Bit 7 to 1<br>OFF:<br>Set Bit 7 to 0 |
| GPIO13 |   | Register<br>ASIC_GPIO_STATE0= 80010104<br>As Input:<br>Set Bit 14 to 0<br>As Output:<br>Set Bit 14 to 1 | Register<br>ASIC_GPIO_STATE0= 80010104<br>ON:<br>Set Bit 15 to 1<br>OFF:<br>Set Bit 15 to 0 |



|        |   |   |
|--------|---|---|
| GPIO14 | Register<br>ASIC_GPIO_PORTDIR0= 80010134<br>As Input:<br>Set Bit 4 to 0<br>As Output:<br>Set Bit 4 to 1 | Register<br>ASIC_GPIO_PORTVAL0= 80010120<br>ON:<br>Set Bit 4 to 1<br>OFF:<br>Set Bit 4 to 0 |
| GPIO15 | Register<br>ASIC_GPIO_PORTDIR0= 80010134<br>As Input:<br>Set Bit 3 to 0<br>As Output:<br>Set Bit 3 to 1 | Register<br>ASIC_GPIO_PORTVAL0= 80010120<br>ON:<br>Set Bit 3 to 1<br>OFF:<br>Set Bit 3 to 0 |

Table 2: Lists the GPIO lines and their alternate functions

### 0.5.1 The Real Time Clock

The high precision Real-Time Clock (RTC) is used for GPS tracking and navigation functions. This module also supports the system power management by providing a wake-up interrupt to the CPU during power down.

### 0.5.2 Interrupts

Timer and interrupt functions are included in the digital part of embedded GPS receiver chip. The possible interrupt sources are data-ready interrupts, timer interrupts, dual-channel UART interrupts, beacon receiver interrupts and external interrupts. The data-ready interrupt is generated for each channel separately.

### 0.5.3 BUS Interface

The Microwire bus interface protocol typically connects the serial EEPROM and external A/D to the GPS receiver's digital chip. It is a multi-master bus (more than one device capable of controlling the bus can be connected to it). The bus uses three wires; serial data in (GPIO5), serial data out (GPIO6), and serial clock (GPIO7) to carry information between the devices. To support the System-on-a-Chip architecture, testability and the internal peripheral bus are implemented.

- ☞ For getting access to the Microwire bus interface, is necessary to set GPIO5-7 to "alternate function".

## 0.6 Chip Select Wait States

The wait states are set in the Chip select registers. Some of these registers (CSN0, CSN1 and CSN2) are initialized in `PORTS.H`.

## 0.7 Digital chip Address Space

The following table summarizes the user-accessible digital chip address space. It is broken up into four modules:

- Interrupt Control Module
- Universal Asynchronous Receiver/Transmitter (UART) Module
- Three-Wire Serial Synchronous Interface Module

- Miscellaneous Module

| Register Name   | # of Half Words | R/W | CPU Address Bus BA[31:0] | Description                       | Reset Value |
|---|-----------------|-----|--------------------------|-----------------------------------|-------------|
| <b>BUI MODULE</b>                                     |                 |     |                          |                                   |             |
| CSN2_REG  | 1               | R/W | 0x80090008               | CSN2 configuration                | 0xF207      |
| <b>INTERRUPT CONTROL MODULE</b>                       |                 |     |                          |                                   |             |
| INTREG  | 1               | R   | 0x800A0000               | Interrupt source status           | 0x0000      |
| INTENA  | 1               | R/W | 0x800A0004               | Interrupt enable                  | 0x0000      |
| INTSEL  | 1               | R/W | 0x800A0008               | Interrupt selection               | 0x0000      |
| INTFIQ  | 1               | R   | 0x800A000C               | FIQ status                        | 0x0000      |
| INTIRQ  | 1               | R   | 0x800 A0010              | IRQ status                        | 0x0000      |
| ITPAUSE   | 1               | W   | 0x800A0040               | Wait interrupt configuration      | N/A         |
| DATAINTENA  | 1               | R/W | 0x800A0020               | Data ready interrupts enable      | 0x0000      |
| DASTATUS  | 1               | R   | 0x800A0024               | Data ready interrupts status      | 0x0000      |
| DATACKN   | 1               | W   | 0x800A0028               | Data ready interrupts acknowledge | N/A         |
| UARTINTENA  | 1               | R/W | 0x800A002C               | UART interrupts enable            | 0x0000      |
| UARTSTATUS  | 1               | R   | 0x800A0030               | UART interrupts status            | 0x0000      |
| UARTACKN  | 1               | W   | 0x800A0034               | UART interrupts acknowledge       | N/A         |
| TIMERACKN   | 1               | w   | 0x800A0038               | Timer interrupts acknowledge      | N/A         |
| BEACKN  | 1               | w   | 0x800A003C               | Beacon interrupts acknowledge     | N/A         |
| EIT_LEV   | 1               | R/W | 0x800A0034               | External interrupts active level  | 0x0000      |
| <b>UART MODULE</b>                                    |                 |     |                          |                                   |             |
| URT_A_CTL   | 1               | R/W | 0x80030000               | Control                           | 0x0000      |
| URT_A_STAT  | 1               | R   | 0x80030002               | Status                            | 0x0000      |
| URT_A_TX  | 1               | W   | 0x80030004               | Transmit data                     | N/A         |
| URT_A_RX  | 1               | R   | 0x80030006               | Receive data                      | 0x0000      |
| URT_A_BAUD  | 1               | R/W | 0x80030008               | Baud rate                         | 0x0A00      |
| URT_B_CTL   | 1               | R/W | 0x80030010               | Control                           | 0x0000      |
| URT_B_STAT  | 1               | R   | 0x80030012               | Status                            | 0x0000      |
| URT_B_TX  | 1               | W   | 0x80030014               | Transmit data                     | N/A         |
| URT_B_RX  | 1               | R   | 0x80030016               | Receive data                      | 0x0000      |
| URT_B_BAUD  | 1               | R/W | 0x80030018               | Baud rate                         | 0x0A00      |
| <b>THREE-WIRE SERIAL SYNCHRONOUS INTERFACE MODULE</b> |                 |     |                          |                                   |             |
| THREE WIRE DATAIN                                     | 1               | R/W | 0x8002000C               |                                   |             |

|                             |        |  |     |            |  |        |
|-----------------------------|--------|--|-----|------------|--|--------|
| THREE<br>DATAOUT            | WIRE 1 |  | R/W | 0x8002000E |  |        |
| <b>MISCELLANEOUS MODULE</b> |        |  |     |            |  |        |
| GPIOSEL                     | 1      |  | R/W | 0x80010100 |  | 0x0000 |
| GPIOSTATE0                  | 1      |  | R/W | 0x80010104 |  | 0x0000 |
| GPIOSTATE1                  | 1      |  | R/W | 0x80010108 |  | 0x0000 |
| GPIOSTATE2                  | 1      |  | R/W | 0x8001010C |  | 0xE000 |
| GPIOSTATE3                  | 1      |  | R/W | 0x80010110 |  | 0xC000 |
| GPIOSTATE4                  | 1      |  | R/W | 0x80010114 |  | 0x0000 |
| GPIOSTATE5                  | 1      |  | R/W | 0x80010118 |  | 0x0000 |
| GPIOSTATE6                  | 1      |  | R/W | 0x8001011C |  | 0x0000 |
| GPIOPORTVAL0                | 1      |  | R/W | 0x80010120 |  | 0x0000 |
| GPIOPORTVAL1                | 1      |  | R/W | 0x80010124 |  | 0x0000 |
| GPIOPORTVAL2                | 1      |  | R/W | 0x80010128 |  | 0x0000 |
| GPIOPORTVAL3                | 1      |  | R/W | 0x8001012C |  | 0x0000 |
| GPIOPORTVAL4                | 1      |  | R/W | 0x80010130 |  | 0x0000 |
| GPIOPORTDIR0                | 1      |  | R/W | 0x80010134 |  | 0x0000 |
| GPIOPORTDIR1                | 1      |  | R/W | 0x80010138 |  | 0x0000 |
| GPIOPORTDIR2                | 1      |  | R/W | 0x8001013C |  | 0x0000 |
| GPIOPORTDIR3                | 1      |  | R/W | 0x80010140 |  | 0x0000 |
| GPIOPORTDIR4                | 1      |  | R/W | 0x80010144 |  | 0x0000 |

Table 3: Digital chip address space

## 0.8 Interrupt Control Module

The interrupt controller provides a simple software interface that has two interrupt levels:

- ✓ Fast Interrupt Request (FIQ) for fast and low latency interrupt handling.
- ✓ Interrupt Request (IRQ) for general interrupts.

All interrupt inputs are active-HIGH and level sensitive. Except for the IRQ/ FIQ selection, no hardware priority scheme or interrupt vectoring is provided. Therefore, these functions must be implemented in the software.

### 0.8.1 Interrupt Controller Functional Description

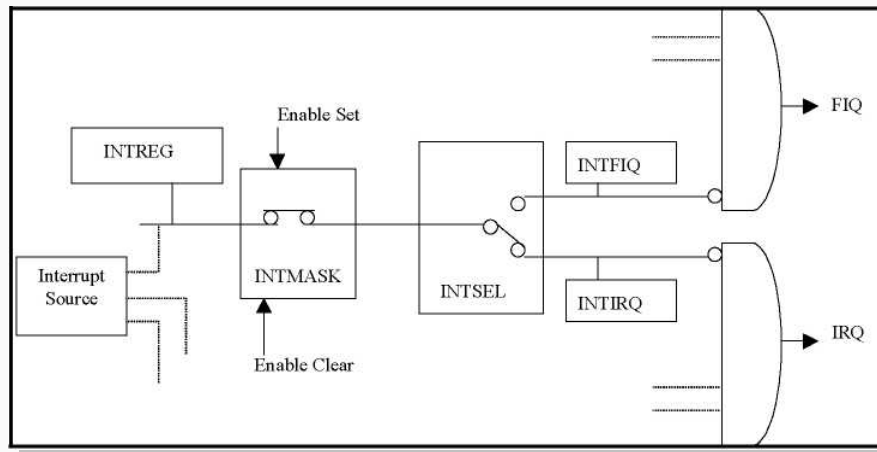


Figure 1 Interrupt Controller Block Diagram

At the top level, the interrupt controller provides an interrupt status register (INTREG), an interrupt enable (mask) register (INTENA), an interrupt select register (INTSEL) that steers requests to FIQ or IRQ, and two status registers for FIQ and IRQ status (INTFIQ and INTIRQ).

INTREG displays the interrupt request status prior to masking. The interrupt source is active High and so a logic HIGH in INTREG indicates that the interrupt request is active. Interrupt requests are not latched inside the interrupt controller; they are assumed to be latched at the interrupt source. Thus, an interrupt requests from the GPS receiver must be cleared by writing to the interrupt acknowledge register allocated to that interrupt. External interrupts must be cleared at their respective sources by the ISR.

When set high, a bit in INTENA passes the respective interrupt request on to the FIQ/IRQ steering logic. A bit set to a one in the FIQ/IRQ selection register INTSEL steers its respective interrupt to the FIQ exception, and conversely, a zero steers the request to the IRQ exception. The INTFIQ and INTIRQ registers display active FIQ and IRQ requests, respectively, as ones.

## 0.9 BIU Chip Select Registers

## 0.10 Pause Control Function

The interrupt controller also manages the wait for interrupt or PAUSE function. Any access to the ITPAUSE register turns off BCLK, (i.e., the CPU clock, forcing the ARM CPU to enter a wait for interrupt state). The CPU exits the PAUSE state at either of two events:

1. When either IRQ or FIQ becomes active.
2. When RESET is activated, resetting the Interrupt Controller.

### ITPAUSE Register

**(WRITE ONLY); Address [0x800A0040]; Reset value = Unknown;**

|        |    |    |    |    |    |    |   |   |
|--------|----|----|----|----|----|----|---|---|
| Bit    | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Source | X  | X  | X  | X  | X  | X  | X | X |
| Bit    | 7  | 6  | 5  | 4  | 3  | 2  | 1 | 0 |

|        |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|
| Source | X | X | X | X | X | X | X | X |
|--------|---|---|---|---|---|---|---|---|

## 0.11 Interrupt Controller Register Mapping

### Interrupt Source Status

**INTREG (READ ONLY); Address [0x800A0000]; Reset value = 0x0000;**

| Bit <sup>1</sup> | 15         | 14         | 13          | 12        | 11        | 10        | 9       | 8         |
|------------------|------------|------------|-------------|-----------|-----------|-----------|---------|-----------|
| Source           | Data ready | Beacon int | wakeup int  | Uwire int | UartA_int | UartB_int | GPIO_IT | Watch dog |
| Bit              | 7          | 6          | 5           | 4         | 3         | 2         | 1       | 0         |
| Source           | Timer 20   | Timer_100  | Timer sleep | Reserved  | Reserved  | Ext2      | Ext1    | Ext0      |

<sup>1</sup> If Bit(i) = 1, interrupt request(i) active.

### Interrupt Enable

**INTENA enable (WRITE/READ); Address[0x800A0004]; Reset value = 0x0000;**

| Bit <sup>1</sup> | 15         | 14         | 13          | 12        | 11        | 10        | 9           | 8            |
|------------------|------------|------------|-------------|-----------|-----------|-----------|-------------|--------------|
| Source           | Data_ready | Beacon_int | wakeup_int  | Uwire_int | UartA_int | UartB_int | GPIO_I<br>T | Watchd<br>og |
| Bit              | 7          | 6          | 5           | 4         | 3         | 2         | 1           | 0            |
| Source           | Timer_20   | Timer J 00 | Timer_sleep | Reserved  | Reserved  | Ext2      | Ext1        | Ext0         |

<sup>1</sup> If Bit(i) = 1, enable interrupt(i), else disable interrupt(i).

### Interrupt Selection

After reset, all interrupts are mapped by default to IRQ.

**INTSEL (READ/WRITE); Address[0x800A0008]; Reset value = 0x0000;**

| Bit <sup>1</sup> | 15         | 14         | 13         | 12        | 11        | 10        | 9           | 8            |
|------------------|------------|------------|------------|-----------|-----------|-----------|-------------|--------------|
| Source           | Data_ready | Beacon_int | wakeup_int | Uwire_int | UartA_int | UartB_int | GPIO_I<br>T | Watchd<br>og |
| Bit              | 7          | 6          | 5          | 4         | 3         | 2         | 1           | 0            |
| Source           | Timer 20   | Timer J 00 | Timersleep | Reserved  | Reserved  | Ext2      | Ext1        | Ext0         |

<sup>1</sup> If Bit(i)=1, enable FIQ, else IRQ.

### FIQ Status Register

**INTFIQ (READ ONLY); Address[0x800A000C]; Reset value = 0x0000;**

| Bit <sup>1</sup> | 15         | 14         | 13         | 12        | 11        | 10                | 9                 | 8            |
|------------------|------------|------------|------------|-----------|-----------|-------------------|-------------------|--------------|
| Source           | Data_ready | Beacon_int | wakeup_int | Uwire_int | UartA_int | UartB_int         | GPIO_I<br>T       | Watchd<br>og |
| Bit              | 7          | 6          | 5          | 4         | 3         | 2                 | 1                 | 0            |
| Source           | Timer 20   | Timer J 00 | Timersleep | Reserved  | Reserved  | Ext2 <sup>2</sup> | Ext1 <sup>3</sup> | Ext0         |

<sup>1</sup> If Bit(i) = 1, FIQ request(i) active.

### IRQ Status Register

**INTIRQ (READ ONLY); Address[0x800A0010]; Reset value = 0x0000;**

| Bit <sup>1</sup> | 15         | 14         | 13          | 12        | 11        | 10        | 9           | 8            |
|------------------|------------|------------|-------------|-----------|-----------|-----------|-------------|--------------|
| Source           | Data_ready | Beacon_int | wakeup_int  | Uwire_int | UartA_int | UartB_int | GPIO_I<br>T | Watchdo<br>g |
| Bit              | 7          | 6          | 5           | 4         | 3         | 2         | 1           | 0            |
| Source           | Timer_20   | Timer_100  | Timer_sleep | Reserved  | Reserved  | Ext2      | Ext1        | Ext0         |

<sup>1</sup> If Bit(i) = 1, FIQ request(i) active.

## 0.12 Pulse Interrupt Management

Some modules generate only pulse interrupts, so an interface is required to convert them to level sensitive interrupts. Because power is managed by shutting down the various clocks within the chip, each pulse interrupt is converted using its native module clock (e.g., Data\_Ready uses GSP2CLK, UART interrupt uses UARTCLK, etc.). The interrupt is cleared by writing a one to the respective interrupt acknowledge bit.

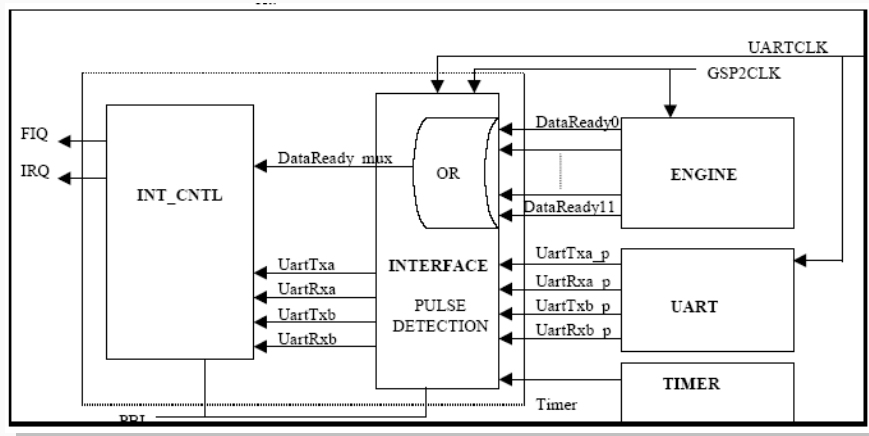


Figure 2 Pulse Interrupt Management Interface

## 0.13 Data\_Ready Interrupt

The 12 Data\_Ready interrupts (one per channel) are managed by three registers. Each channel's Data\_ready interrupt is enabled or disabled by writing a 1 or 0, respectively, to its respective bit in DATAINTENA. DATASTATUS displays active Data\_Ready interrupt requests as ones. Writing a one to a bit in DATAACKN clears the respective Data\_Ready interrupt. The 12 interrupt requests are ORed into one Data\_Ready interrupt that is managed as a single interrupt by the top level control registers INTREG, INTENA, and INTSEL described earlier.

### Data\_Ready Interrupt Enable

**DATAINTENA Enable (WRITE/READ); Address[0x800A0020]; Reset value = 0x0000;**

| Bit <sup>1</sup> | 15          | 14          | 13           | 12          | 11           | 10           | 9            | 8           |
|------------------|-------------|-------------|--------------|-------------|--------------|--------------|--------------|-------------|
| Source           | unused      | unused      | unused       | unused      | Data_Ready11 | Data_Ready10 | Data_Ready 9 | Data_Ready8 |
| Bit              | 7           | 6           | 5            | 4           | 3            | 2            | 1            | 0           |
| Source           | Data_Ready7 | Data_Ready6 | Data_Ready 5 | Data_Ready4 | Data_Ready3  | Data_Ready2  | Data_Ready 1 | Data_Ready0 |

<sup>1</sup> Data\_Ready(i) = 1: Enable interrupt.

### Data\_Ready Interrupt Sources Status

**DATASTATUS (READ ONLY); Address[0x800A0024]; Reset value = 0x0000;**

| Bit <sup>1</sup> | 15          | 14          | 13          | 12          | 11           | 10           | 9            | 8           |
|------------------|-------------|-------------|-------------|-------------|--------------|--------------|--------------|-------------|
| Source           | unused      | unused      | unused      | unused      | Data_Ready11 | Data_Ready10 | Data_Ready 9 | Data_Ready8 |
| Bit              | 7           | 6           | 5           | 4           | 3            | 2            | 1            | 0           |
| Source           | Data_Ready7 | Data_Ready6 | Data_Ready5 | Data_Ready4 | Data_Ready3  | Data_Ready2  | Data_Ready 1 | Data_Ready0 |

**Data\_Ready Interrupt Acknowledge****DATAACKN (WRITE ONLY); Address[0x800A0028];**

| Bit <sup>1</sup> | 15          | 14          | 13          | 12          | 11           | 10           | 9            | 8           |
|------------------|-------------|-------------|-------------|-------------|--------------|--------------|--------------|-------------|
| Source           | unused      | unused      | unused      | unused      | Data_Ready11 | Data_Ready10 | Data_Ready 9 | Data_Ready8 |
| Bit              | 7           | 6           | 5           | 4           | 3            | 2            | 1            | 0           |
| Source           | Data_Ready7 | Data_Ready6 | Data_Ready5 | Data_Ready4 | Data_Ready3  | Data_Ready2  | Data_Ready 1 | Data_Ready0 |

<sup>1</sup> Data\_Ready(i) = 1: Acknowledge (clear) channel(i) interrupt.**0.14 UART Interrupt**

There are four UART interrupt sources: the TX and RX halves of Channels A and B. The four interrupts are ORed together to form the single UART interrupt request into INTREG and INTENA.

**UART Interrupt Enable****UARTINTENA (WRITE/READ); Address[0x800A002C]; Reset value = 0x0000;**

| Bit <sup>1</sup> | 15     | 14     | 13     | 12     | 11       | 10       | 9        | 8        |
|------------------|--------|--------|--------|--------|----------|----------|----------|----------|
| Source           | Unused | Unused | Unused | Unused | Unused   | Unused   | Unused   | Unused   |
| Bit              | 7      | 6      | 5      | 4      | 3        | 2        | 1        | 0        |
| Source           | Unused | Unused | Unused | Unused | Uart_Txa | Uart_Rxa | Uart_Txb | Uart_Rxb |

<sup>1</sup> Bit(i) = 1: Enable interrupt(i).**UART Interrupt Source Status****UARTSTATUS (READ ONLY); Address[0x800A0030]; Reset value = 0x0000;**

| Bit <sup>1</sup> | 15     | 14     | 13     | 12     | 11       | 10       | 9        | 8        |
|------------------|--------|--------|--------|--------|----------|----------|----------|----------|
| Source           | Unused | Unused | Unused | Unused | Unused   | Unused   | Unused   | Unused   |
| Bit              | 7      | 6      | 5      | 4      | 3        | 2        | 1        | 0        |
| Source           | Unused | Unused | Unused | Unused | Uart_Txa | Uart_Rxa | Uart_Txb | Uart_Rxb |

<sup>1</sup> If Bit(i) = 1: Interruptrequest(i) active.**UART Interrupt Acknowledge****UARTACKN (WRITE ONLY); Address[0x800A0034];**

| Bit <sup>1</sup> | 15     | 14     | 13     | 12     | 11       | 10       | 9        | 8        |
|------------------|--------|--------|--------|--------|----------|----------|----------|----------|
| Source           | Unused | Unused | Unused | Unused | Unused   | Unused   | Unused   | Unused   |
| Bit              | 7      | 6      | 5      | 4      | 3        | 2        | 1        | 0        |
| Source           | Unused | Unused | Unused | Unused | Uart_Txa | Uart_Rxa | Uart_Txb | Uart_Rxb |

<sup>1</sup> Bit(i) = 1: Acknowledge (clear) interrupt

## 0.15 Timer Interrupts

There are three interrupts generated from the GSP2e user time counter:

- ✓ ms or Watchdog interrupt
- ✓ 20 ms Timer\_20 interrupt
- ✓ 100 ms Timer\_100 interrupt

When enabled in INTENA, these interrupts are set high (active) coincident with the 1, 20, and 100 ms user time epochs generated from GPSCLK.

The Timer\_sleep interrupt is either a 1 ms (default mode) or programmable quasi-100 ms interrupt. In its default mode, the Timer\_sleep interrupt goes high every millisecond immediately after the Hardwired Tracking Loop has completed the tasks and gone to sleep (i.e., no later than 200  $\mu$ s after the 1 ms user time epoch from the time of its occurrence until the next 1 ms epoch) software can safely read or write to any RAM4 memory location allocated to the Hardwired Tracking Loop.

In the programmable-delay mode, the Timer\_sleep interrupt is set only when the Hardwired Tracking Loop has gone to sleep *and* the 100 ms user-time counter state equals the value programmed in GO TO SLEEP INTERRUPT MODULO 100MUT register (TRKMOD address = 0x44). This mode facilitates the gathering of measurement data at times other than the fixed 100 ms user time epoch.

The relevant interrupts are managed by the INTREG, INTENA, and INTSEL registers. They are acknowledged (cleared) by writing to the TIMERACKN register.

## 0.16 Timer Interrupt Acknowledge

### TIMERACKN (WRITE ONLY); Address[0x800A0038]:

| Bit <sup>1</sup> | 15     | 14     | 13     | 12     | 11                | 10      | 9        | 8         |
|------------------|--------|--------|--------|--------|-------------------|---------|----------|-----------|
| Source           | Unused | Unused | Unused | Unused | Unused            | Unused  | Unused   | Unused    |
| Bit              | 7      | 6      | 5      | 4      | 3                 | 2       | 1        | 0         |
| Source           | Unused | Unused | Unused | Unused | Watchdog (1 msec) | GPIO_IT | Timer_20 | Timer_100 |

<sup>1</sup> Timer\_x = 1: Acknowledge (clear) Timer\_x interrupt.

### 0.16.1 Beacon Interrupt

The INTREG, INTENA, and INTSEL registers manage the Beacon and Three Wire interrupts. BEACKN is used to acknowledge (clear) the interrupts.

#### BEACON Interrupt Acknowledge

### BEACKN (WRITE ONLY); Address[0x800A003C]:

| Bit <sup>1</sup> | 15     | 14     | 13     | 12     | 11     | 10     | 9         | 8          |
|------------------|--------|--------|--------|--------|--------|--------|-----------|------------|
| Source           | Unused | Unused | Unused | Unused | Unused | Unused | Unused    | Unused     |
| Bit              | 7      | 6      | 5      | 4      | 3      | 2      | 1         | 0          |
| Source           | Unused | Unused | Unused | Unused | Unused | Unused | Uwire_int | Beacon_int |

<sup>1</sup> If Uwire\_int or Beacon\_int = 1, interrupt is acknowledged (cleared).



### 0.16.2 External Interrupt Active Level

The active level of the three external interrupts can be controlled. By default they are active low. Writing a one to any on bits [2:0] turns the respective interrupt active high. This is done internally by inverting the signal before it reaches the INTREG (i.e., the INTREG still sees the interrupt active as a one).

#### EIT Level Reg

**EIT\_LEV (READ/WRITE); Address [0x800 A004C]; Reset value = 0x0000;**

| Bit <sup>1</sup> | 15     | 14     | 13     | 12     | 11     | 10     | 9      | 8      |
|------------------|--------|--------|--------|--------|--------|--------|--------|--------|
| Source           | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused |
| Bit              | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
| Source           | Unused | Unused | Unused | Unused | Unused | EIT 2  | EIT 1  | EIT0   |

<sup>1</sup> Bit(i) = 0: Interrupt is active low, Bit(i) = 1: Interrupt is active high

### 0.16.3 UARTModule

The UART module consists of two independent channels for serial interface (UARTA and UARTB). The two UARTs are identical, each consisting of a transmitter block and a receiver block.

The transmitter block of each UART is comprised of a 16-deep FIFO (8 bits wide), an 8-bit parallel-in/serial-out shift register, and a parity generator.

The receiver block of each UART is comprised of a 16-deep FIFO by 11 bits wide (8 data bits + 3 error bits), an 8-bit serial in/parallel out shift register, and error detectors. The receiver accepts the serial input data and assembles it into parallel data that is entered into the receiver FIFO along with the error signals (frame error, parity error, and break interrupt). The frame, parity, and break bits to the interrupt controller are updated every time a byte is read out of the FIFO. The dataRdy and overrun bits are updated every time a new frame is received into the FIFO.

The UART module outputs four interrupts to the Interrupt controller: UART\_TXA, UART\_RXA, UART\_TXB, and UART\_RXB. These interrupts are the OR of the respective enabled transmit and receive interrupts and are pulse detected by the interrupt controller to save the interrupts until acknowledged by the CPU.

#### UART A CONTROL: (Address=0x80030000)

| RW | RW | RW | RW | RW | RW | RW | RW |
|----|----|----|----|----|----|----|----|
| 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  |
| RW | RW | RW | RW | RW | RW | RW | RW |
| 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |

(Reset to x0000)

- D[7:0] Status Enable bits:
- D0: Break Interrupt Enable
- D1: Parity Error Enable
- D2: Frame Error Enable
- D3: 2-bytes-to-Rxa-FIFO-full Enable
- D4: Rxa Data Ready Enable
- D5: Rxa Full Enable
- D6: Txa Full Enable
- D7: Txa Empty Enable
- D[15:8] Frame Format bits:

|      |                       |  |
|------|-----------------------|--|
| D8:  | Data7                 | (0: 8-bits, 1: 7-bits)   |
| D9:  | Stop2                 | (0: ONE STOP bit      1: TWO STOP bits)                            |
| D10: | OddPar                | (0: Even Parity      1: Odd Parity)                                |
| D11: | ParEn                 | (0: Disable Parity      1: Enable Parity)                          |
| D12: | LoopBack              | (0: Disabled      1: Enable LoopBack Mode)                         |
| D13: | ResetStatus           | (If high, resets the overrun status bits)                          |
| D14: | Set break on Txa line | (forces data out low at end of current data)                       |
| D15: | ResetChannel          | (synced rising edge works as a software reset to both Txa and Rxa) |

**UART A STATUS (Address=0x80030002)**

|  |  |  |  |  |  |  |  |   |   |   |   |   |   |   |   |   |
|--|--|--|--|--|--|--|--|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  | R | R | R | R | R | R | R | R | R |
|  |  |  |  |  |  |  |  | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**UART A TRANSMIT DATA (Address=0x80030004)**

|  |  |  |  |  |  |  |  |  |  |   |   |   |   |   |   |   |   |
|--|--|--|--|--|--|--|--|--|--|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |  |  | W | W | W | W | W | W | W | W |
|  |  |  |  |  |  |  |  |  |  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

7:0 UART A transmit data

**UART A RECEIVE DATA (Address=0x80030006)**

|  |  |  |  |  |  |  |  |  |  |   |   |   |   |   |   |   |   |
|--|--|--|--|--|--|--|--|--|--|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |  |  | R | R | R | R | R | R | R | R |
|  |  |  |  |  |  |  |  |  |  | 7 | 6 | 5 | 4 | 3 |   | 1 | 0 |

7:0 UART A receive data.

**UART A Baud Rate (Address=0x80030008)**

|    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |
|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|
| RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | R | R | R | R |
| 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  |   |   | 1 | 0 |

(reset to xA000)

 Note: bits [3:0] are forced internally to zero

Transmit and Receive operate at the same Baud rate. Baud rate is derived from the GPS clock or the external clock (ECLK), as selected by the CLK\_SELECT register (address 0x80010018 in the MISMOD). The selected clock is divided by a number that gives the desired Baud rate. The number by which the clock should be divided is programmed into this register. The following table lists the value to be programmed in this register for the various Baud rates if the GPS clock is selected as the source.

| bits 15:0 | Baud Rate (if GPS clock is selected) |
|-----------|--------------------------------------|
| 0xA000    | 1200(default)                        |
| 0x5000    | 2400                                 |
| 0x2800    | 4800                                 |
| 0x1400    | 9600                                 |
| 0x0A00    | 19200                                |
| 0x0500    | 38400                                |
| 0x0350    | 57600                                |
| 0x01B0    | 115200                               |

**UART\_B\_CONTROL: (Address=0x80030010)**

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

|         | 15  | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7                       | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------|---|----|----|----|----|----|---|---|-------------------------|---|---|---|---|---|---|---|
| D[7:0]  | Status Enable bits:   |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D0:     | Break Interrupt Enable  |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D1:     | Parity Error Enable   |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D2:     | Frame Error Enable  |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D3:     | 2-bytes-to-Rxa-FIFO-full Enable   |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D4:     | Rxa Data Ready Enable   |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D5:     | Rxa Full Enable   |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D6:     | Txa Full Enable   |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D7:     | Txa Empty Enable  |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D[15:8] | Frame Format bits:  |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D8:     | Data7 (0: 8-bits,   |    |    |    |    |    |   |   | 1:7-bits)               |   |   |   |   |   |   |   |
| D9:     | Stop2 (0: ONE STOP bit  |    |    |    |    |    |   |   | 1:TWO STOP bits)        |   |   |   |   |   |   |   |
| D10:    | OddPar (0: Even Parity  |    |    |    |    |    |   |   | 1:Odd Parity)           |   |   |   |   |   |   |   |
| D11:    | ParEn (0: Disable Parity  |    |    |    |    |    |   |   | 1:Enable Parity)        |   |   |   |   |   |   |   |
| D12:    | LoopBack (0: Disabled   |    |    |    |    |    |   |   | 1:Enable LoopBack Mode) |   |   |   |   |   |   |   |
| D13:    | ResetStatus (If high, resets the overrun status bits)                           |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D14:    | Set break on Txb line (Forces data out low at end of current data)              |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |
| D15:    | ResetChannel (Synced rising edge works as a software reset to both Txb and Rxb) |    |    |    |    |    |   |   |                         |   |   |   |   |   |   |   |

**UART\_B\_STATUS (Address=0x80030012):**

|  |  |  |  |  |  |  |  |   |   |   |   |   |   |   |   |   |
|--|--|--|--|--|--|--|--|---|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  | R | R | R | R | R | R | R | R | R |
|  |  |  |  |  |  |  |  |   |   |   |   |   |   |   |   |   |

|     |                          |
|-----|--------------------------|
| D0: | Break Interrupt          |
| D1: | Parity Error             |
| D2: | Frame Error              |
| D3: | OverRun Error            |
| D4: | Rxa Data Ready           |
| D5: | Rxa Full                 |
| D6: | Txa Full                 |
| D7: | Txa Empty                |
| D8: | 2 bytes to Rxa FIFO full |

**UART B TRANSMIT DATA (Address=0x80030014):**

|  |  |  |  |  |  |  |  |  |  |   |   |   |   |   |   |   |   |
|--|--|--|--|--|--|--|--|--|--|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |  |  | W | W | W | W | W | W | W | W |
|  |  |  |  |  |  |  |  |  |  |   |   |   |   |   |   |   |   |

7:0 UART A transmit data.

**UART\_B\_RECEIVE DATA (Address=0x80030016):**

|  |  |  |  |  |  |  |  |  |  |   |   |   |   |   |   |   |   |
|--|--|--|--|--|--|--|--|--|--|---|---|---|---|---|---|---|---|
|  |  |  |  |  |  |  |  |  |  | R | R | R | R | R | R | R | R |
|  |  |  |  |  |  |  |  |  |  |   |   |   |   |   |   |   |   |

7:0 UART A receive data.

**UART B Baud Rate (Address=0x80030018):**

|    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |
|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|
| RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | R | R | R | R |
|    |    |    |    |    |    |    |    |    |    |    |    |    |   |   |   |   |

Note: bits [3:0] are forced internally to zero.

Transmit and Receive operate at the same Baud rate. Baud rate is derived from the GPS clock or the external clock (ECLK), as selected by the CLK\_SELECT register (address 0x80010018 in the MISMOD). The selected clock is divided by a number that gives the desired Baud rate. The number by which the clock should be divided is programmed into this register. The table following table lists the value to be programmed in this register for the various Baud rates if the GPS clock is selected as the source.

| bits 15:0 | Baud Rate (if GPS clock is selected) |
|-----------|--------------------------------------|
| 0xA000    | 1200(default)                        |
| 0x5000    | 2400                                 |
| 0x2800    | 4800                                 |
| 0x1400    | 9600                                 |
| 0x0A00    | 19200                                |
| 0x0500    | 38400                                |
| 0x0350    | 57600                                |
| 0x01B0    | 115200                               |

# 1 THREE- WIRE INTERFACE

This section describes an interface that can be programmed to be Synchronous Serial Three Wire compatible or used to control an external ADC. The ADC mode is intended to support the internal Differential Beacon receiver. Three IO wires are utilized: SI (serial data in), SO (serial wire out), and SK (serial data clock). In Synchronous Serial mode, data is clocked out on SO on the falling edge of SK and clocked in from SI on the rising edge of SK. It can be programmed to operate in master or in slave mode. For ADC mode, the active edges are controllable with bit 12 of the Data Out Register.

Three Wire Data Input Register (Address=0x8002000c)

This register controls the operation of the Three Wire Interface.

| Bit   | CPU | TRACK | Reset | Description  |
|-------|-----|-------|-------|--|
| 7:0   | R   |       | 3     | 8 bits of data received from the last master/slave command.            |
| 10:8  | RW  |       | 3     | Clock divider program bits. ClkDivProg[2:0].                           |
| 11    | RW  |       | 3     | ADC wait state enable bit.   |
| 14:12 | RW  |       | 3     | ADC output bits select.  |
| 15    | RW  |       | 3     | 8-bit. Set to 1 interface with 8-bit ADC, 0 interface with 10-bit ADC. |

**Note** - The SK frequency is a divide down of the GSPCLK.

| ClockDividerProg[2:0] | SK Frequency |
|-----------------------|--------------|
| 000                   | 3.507 MHz    |
| 001                   | 1.754 MHz    |
| 010                   | 877 KHz      |
| 011                   | 438 KHz      |
| 100                   | 219 KHz      |
| 101                   | 110 KHz      |
| 110                   | 55 KHz       |
| 111                   | 27 KHz       |

Three-Wire Data Output Register (Address=0x8002000e)

This register controls the operation of the Three Wire Interface. Bits 8, 9, and 10 must be set exclusively of each other (i.e., at most only one of them should contain a 1).

| Bit | CPU | TRACK | Reset | Description   |
|-----|-----|-------|-------|---|
| 7:0 | RW  |       | 3     | 8 bits of data/command to transmit.   |
| 8   | RW  |       | 3     | Xchs bit. Write a 1 to put in slave mode and issue a slave command.   |
| 9   | RW  |       | 3     | Xchm bit. Write a 1 to put in master mode and issue a master command.   |
| 10  | RW  |       | 3     | ADCMode bit. Write a 1 to enable ADC mode.  |
| 11  | RW  |       | 1     | SO output enable bit. Write a 0 to enable the driver, 1 to tristate.  |
| 12  | RW  |       | 3     | ClkEdge bit. When in ADC mode, 1 using positive clock edge to clock out data, negative clock edge to clock in data. 0 means the opposite. |

|    |    |  |   |  |
|----|----|--|---|--|
| 13 | RW |  | 3 | ADC DO Idle bit. When in ADC mode, this determines the logic level of DO when no command word is sent out. |
|----|----|--|---|--|

## 1.1 Three-Wire Master Mode

Writing a 1 to the Xchm bit puts the interface into Three Wire Master mode and issues a master command. In this mode, the interface drives the SO and SK pin. Whenever a master command happens, 8 bits of data are clocked out on the falling clock edge of SK and 8 bits of data are clocked in on the rising clock edge of SK. An interrupt is pending at the end of the data transmission. CPU can clear the interrupt by writing to the data output register or writing/reading the data input register.

### 1.1.1 Three-Wire Slave Mode

Writing a 1 to the Xchs bit puts the interface into Three-Wire Slave mode and issues a slave command. In this mode, the interface receives the SK and must only drive the SO pin when the master accesses the ASIC. Whenever a slave command happens, 8 bits of data are clocked out on the falling clock edge of SK and 8 bits of data are clocked in on the rising clock edge of SK. An interrupt is pending at the end of the data transmission. CPU can clear the interrupt by writing to the data output register or writing/reading the data input register.

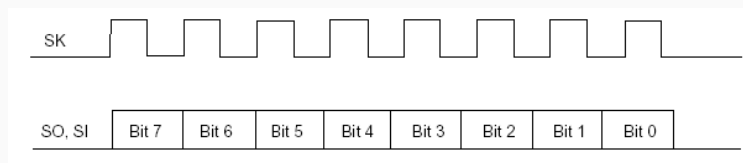


Figure 3 Timing Diagram for Three-Wire Slave Mode

#### **RTCMINSEC (REAL TIME MIN\_SEC): (address = 0x80010030):**

Shows time of the day in minutes and seconds. This can change in the middle of a read cycle. A warning that this value is about to change can be obtained from the status/control register. The counters are updated with the values written into this register only when the REAL\_TIME\_DAYS\_HOURS register is written into. The counters can take up to about (30 \* 4=) 120 microseconds to be updated after writing to the register. Certain bits in the status/control register indicate when this register must not be written to. (Although the counters are read back in normal mode, the latches that temporarily store the value written in can be read from this location by setting the appropriate bit in the TEST\_CTL register. This may be useful if a 16-bit battery backed storage is required.)

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW | RW |
| 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |

5:0 Real\_time\_seconds(5:0)

Bat\_reset: Not effected by battery reset.

13:8 Real\_time\_minutes(5:0)

Bat\_reset: Not effected by battery reset.

Note - RTCMINSEC should be cleared by software after a battery reset.

#### **RTCDAYSHRS(REAL TIME DAYS\_HOURS): (address = 0x80010034)**

Shows time of the day in days and hours. Maximum of 255 days. This can change in the middle of a read cycle. A warning that this value is about to change can be obtained from the status/control register. The counters can take up to about (30 \* 4=) 120 microseconds to be updated after writing to this register. Certain bits in the status/control register indicate when this register should not be written to.

|    |    |    |    |    |    |    |    |  |  |  |  |    |    |    |    |    |
|----|----|----|----|----|----|----|----|--|--|--|--|----|----|----|----|----|
| RW | RW | RW | RW | RW | RW | RW | RW |  |  |  |  | RW | RW | RW | RW | RW |
|    |    |    |    |    |    |    |    |  |  |  |  |    |    |    |    |    |

4:0 Real\_time\_hours(4:0)  
 Bat\_reset: 0 Not effected by battery reset.  
 5:8 Real\_time\_days(7:0)  
 Bat reset: 0 Not effected by battery reset.  
 Note - RTCDAYSHRS should be cleared by software after a battery reset.

## 1.2 GPIO

The embedded GPS receiver provides 16 general-purpose input/output (GPIO) pins, respectively. Each GPIO pin has a value (val) and a direction (dir) control bit, and those that are multiplexed with other functions have alternate function select control bits as well. These alternate function select bits are located in the `GPIO_Sel` register. The `val` and `dir` GPIO control bits are distributed in seven `GPIO_State` registers, five `GPIO_PortVal` registers and five `GPIO_PortDir` registers.

The reset value for the `dir` bits is INPUT for all but two GPIOs (GPIO[4] and GPIO[8]) that are reset to OUTPUT mode. The output drivers for the GPIO pins that reset to the input mode are tri-stated and the other two are configured to drive the pins high. Furthermore, the latter two GPIO cannot be programmed as inputs.

- GPIO\_Sel:** This register applies to GPIO pins that are multiplexed with alternate functions. Programming a one into a register bit selects the respective alternate function.
- GPIO\_State:** The seven state registers provide the interface to seven GPIO pins. Each register selects the direction and output value if direction is output; if direction is set to input mode, the input pin value is read from the register.
- GPIO\_PortVal:** There are five 8-bit registers for this function. Together they can be used to read from or write to the remaining 16 GPIO pins.
- GPIO\_PortDir:** There are also five of these 8-bit registers, one for each PortVal register. These bits program the 16 GPIO drivers as either tri-state or active.

On the delivered CD, you will find some source code to see how the provided GPIO's can be programmed (steered). These source codes are free source and can be used, modified and adapted to the user requirements. The source codes are written in C++ programming language, see also chapter 1.5.

| Pin Name         | Bit#  | Register(s)                     |
|------------------|-------|---------------------------------|
| GPIO[13]         | 14/15 | GPIO_State0                     |
| GPIO[3]          | 14/15 | GPIO_State1                     |
| GPIO[4]          | 14/15 | GPIO_State2                     |
| GPIO[8]          | 14/15 | GPIO_State3                     |
| TIMEMARK/GPIO[9] | 14/15 | GPIO_State4                     |
| GPIO[10]         | 14/15 | GPIO_State5                     |
| GPIO[11]         | 14/15 | GPIO_State6                     |
| GPIO[0]          | 7     | GPIO_PortDir0 and GPIO_PortVal0 |
| GPIO[ 1 ]        | 6     |                                 |
| GPIO[2]          | 5     |                                 |
| GPIO[14]         | 4     |                                 |
| GPIO[15]         | 3     |                                 |
| GPIO[5]          | 2     |                                 |
| GPIO[6]          | 1     |                                 |



|          |   |                                 |
|----------|---|---------------------------------|
| GPIO[7]  | 0 |                                 |
| GPIO[12] | 7 | GPIO_PortDir1 and GPIO_PortVal1 |

Table 4: GPIO Pin and Associated Control/Data Register(s)

### 1.2.1 GPIO Sel

#### GPIO\_Sel: (address 0x80010100)

This register configures 20 pins as either GPIO or alternate functions. Each alternate function is allocated one bit.

| GPIO Sel           |          |          |          |                                  |         | 10                                 | 9         | 8         |
|--------------------|----------|----------|----------|----------------------------------|---------|------------------------------------|-----------|-----------|
|                    |          |          |          |                                  |         | RW                                 | RW        | RW        |
| Reset value        |          |          |          |                                  |         | 0                                  | 0         | 0         |
| Alternate function |          |          |          |                                  |         | GPIO[15]                           | GPIO[14]  | GPIO[13]  |
| GPIO Sel           | 7        | 6        | 5        | 4                                | 3       | 2                                  | 1         | 0         |
|                    | RW       | RW       | RW       | RW                               | RW      | RW                                 | RW        | RW        |
| Reset value        | 0        | 0        | 0        | 0                                | 0       | 0                                  | 0         | 0         |
| Alternate function | GPIO[12] | GPIO[11] | GPIO[10] | 1 PPS<br>Timemark or<br>GPIO [9] | GPIO[8] | UWire<br>Interface or<br>GPIO[7:5] | GPIO[4:3] | GPIO[2:0] |

Table 5: GPIO Pin or Alternate Function

Description: GPIO\_Sel[i] = 0: enable GPIO port function.  
GPIO\_Sel[i] = 1: enable alternate function.

| Bit | Controlled Pins       |
|-----|-----------------------|
| 0   | DBGQR, DBGEN, BREAKPT |
| 1   | AGCCLOCK, AGCSTRB     |
| 2   | SI, SO, SK            |
| 3   |                       |
| 4   | TIMEMARK              |
| 5   | EIT[0]                |
| 6   | EIT[1]                |
| 7   | EIT[2]                |
| 8   | CS_N[1]               |
| 9   | CS_N[2]               |
| 10  | CS_N[3]               |
| 11  | CS_N[4]               |
| 12  | CS_N[5]               |
| 13  | CS_N[6]               |
| 14  | CS_N[7]               |

Table 6: Alternate Function GPIO Controlled by GPIO\_SEL Register

## 1.2.2 GPIO State

One read/write address (16 bits wide) is allocated to each of 7 GPIO pins. The MSB bit (bit 15) value corresponds to GPIO pin value. Bit 14 controls the direction. The seven state port registers, `GPIO_State6` to `GPIO_State0` addresses, are mapped from MISBASE (0x10000) + 0x104 to 0x11C (step by four). Pin names are given in the table below.

**GPIO\_State0: Address: 0x80010104 and**

**GPIO\_State1: Address: 0x80010108**

| GPIO_State | 15  | 14  | [13:0] |
|------------|-----|-----|--------|
|            | RW  | RW  | R      |
| Reset val  | 0   | 0   | 0000   |
|            | val | dir | unused |

Table 7: *GPIO State 1*

Description: GPIO\_Dir[i] = 0; GPIO pin is an input  
GPIO\_Dir[i] = 1; GPIO pin is an output

**GPIO\_State2: Address: 0x8001010C**

This register controls GPIO[4]. It is used to control the power up on RF-Front (analogue part) of GPS receiver.

| GPIO_State | 15  | 14  | 13   | [12:0] |
|------------|-----|-----|------|--------|
|            | RW  | RW  | RW   | R      |
| Reset val  | 1   | 1   | 1    | 0000   |
|            | val | dir | mask | unused |

Table 8: *GPIO State2*

**GPIO\_State3: Address: 0x80010110**

This register controls GPIO[8]. This pin is used as output and steers the RF-Front (analogue part) of GPS receiver.

| GPIO_State | 15  | 14  | [13:0] |
|------------|-----|-----|--------|
|            | RW  | RW  | R      |
| Reset val  | 1   | 1   | 0000   |
|            | val | dir | unused |

Table 9: *GPIO State3*

Description: GPIO\_Dir[8] = 0; GPIO pin is tri-stated  
GPIO\_Dir[8] = 1; GPIO pin is an output

**GPIO\_State4, GPIO\_State5, GPIO\_State6:**

**Addresses: 0x80010114, 0x80010118 & 0x8001011C**

| GPIO_State | 15  | 14  | [13:0] |
|------------|-----|-----|--------|
|            | RW  | RW  | R      |
| Reset val  | 0   | 0   | 0000   |
|            | val | dir | unused |

Table 10: *GPIO\_States4 through 6*

Description: GPIO\_Dir[i] = 0; GPIO pin is an input  
GPIO\_Dir[i] = 1; GPIO pin is an output

### 1.2.3 GPIO PortVal

There are five of these registers; each is paired with respective GPIO\_PortDir registers. Except for GPIO\_PortVal2, in which only the seven LSBs are applicable, only the eight LSBs of each 16-bit register are mapped to GPIO pins. The bit values correspond to GPIO pin values. The five portVal registers, GPIO\_PortVal0 to GPIO\_PortVal4 addresses are mapped from 0x80010120 to 0x80010130 (step by four), respectively.

|             |           |           |           |           |           |           |          |          |
|-------------|-----------|-----------|-----------|-----------|-----------|-----------|----------|----------|
| <b>Bit</b>  | <b>15</b> | <b>14</b> | <b>13</b> | <b>12</b> | <b>11</b> | <b>10</b> | <b>9</b> | <b>8</b> |
|             | R         | R         | R         | R         | R         | R         | R        | R        |
| Reset value | 0         | 0         | 0         | 0         | 0         | 0         | 0        | 0        |
|             | Unused    | Unused    | Unused    | Unused    | Unused    | Unused    | Unused   | Unused   |
| <b>Bit</b>  | <b>7</b>  | <b>6</b>  | <b>5</b>  | <b>4</b>  | <b>3</b>  | <b>2</b>  | <b>1</b> | <b>0</b> |
|             | RW        | RW        | RW        | RW        | RW        | RW        | RW       | RW       |
| Reset value | 0         | 0"        | 0         | 0         | 0         | 0         | 0        | 0        |
|             | Val       | Val       | Val       | Val       | Val       | Val       | Val      | Val      |

Table 11: *GPIO PortVal 0 to 4*

Description: Read value is level on pin.  
Write value drives output pin.

### 1.2.4 GPIO PortDir

These five registers are paired with respective GPIO\_PortVal registers. Except for GPIO\_PortDir2, in which only the seven LSBs are applicable, only the eight LSBs of each 16-bit register are mapped to GPIO pins. The bit values control the direction of the GPIO pins. The five PortDir registers, GPIO\_PortDir0 to GPIO\_PortDir4 addresses are mapped from 0x80010134 to 0x80010144 (step by four), respectively.

| GPIO_PortDir | 15     | 14     | 13     | 12     | 11     | 10     | 9      | 8      |
|--------------|--------|--------|--------|--------|--------|--------|--------|--------|
|              | R      | R      | R      | R      | R      | R      | R      | R      |
| Reset value  | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
|              | Unused | Unused | Unused | Unused | Unused | Unused | Unused | Unused |
| GPIO_PortDir | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
|              | RW     | RW     | RW     | RW     | RW     | RW     | RW     | RW     |
| Reset value  | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |
|              | Dir    | Dir    | Dir    | Dir    | Dir    | Dir    | Dir    | Dir    |

Table 12: GPIO PortDir 0 to 4

Description: GPIO\_Dir[i] = 0; GPIO pin is an input  
GPIO\_Dir[i] = 1; GPIO pin is an output

## 1.3 Inputs/outputs

```
void SetIO(int value)

    {//set outputs OUT0- OUT7
    ASIC_GPIO_SEL |= GS_CS2;           //CS2 to enable the IO-Latches
    ASIC_BIU_CSN2 = 0xF251;           //GPIO_RWE; // rw access
    ASIC_GPIO_USER = value;           //usually writing this value once is enough.
                                        depending on hardware timing, several writes
                                        and/or wait states are necessary

    //the following 4 lines assure that writing outputs works in any case
    ASIC_GPIO_USER = value;
    RTCWaitMs(1);                       //wait 1 ms
    ASIC_GPIO_USER = value;
    ASIC_GPIO_USER = value;
    ASIC_GPIO_SEL &= ~GS_CS2;           //disable the IO-Latches
    }

int GetIO(void)
    {//read inputs IO0-IO7 (returns a 8 bit value which contains the current state of all inputs)
    int input;
    ASIC_GPIO_SEL |= GS_CS2;           //CS2 to enable the IO-Latches
    ASIC_BIU_CSN2 = 0xF251;           //GPIO_RWE; // rw access
    // input = ((~ASIC_GPIO_USER) >> 8) & 0xFF;
    RTCWaitMs(WAITSTATE_IO);           //wait 1 ms
    input = ((ASIC_GPIO_USER) >> 8) & 0xFF;
    ASIC_GPIO_SEL &= ~GS_CS2;           //disable the IO-Latches
    return input;
    }
```

## 1.4 Schematics of Various GPIO Pin Configurations

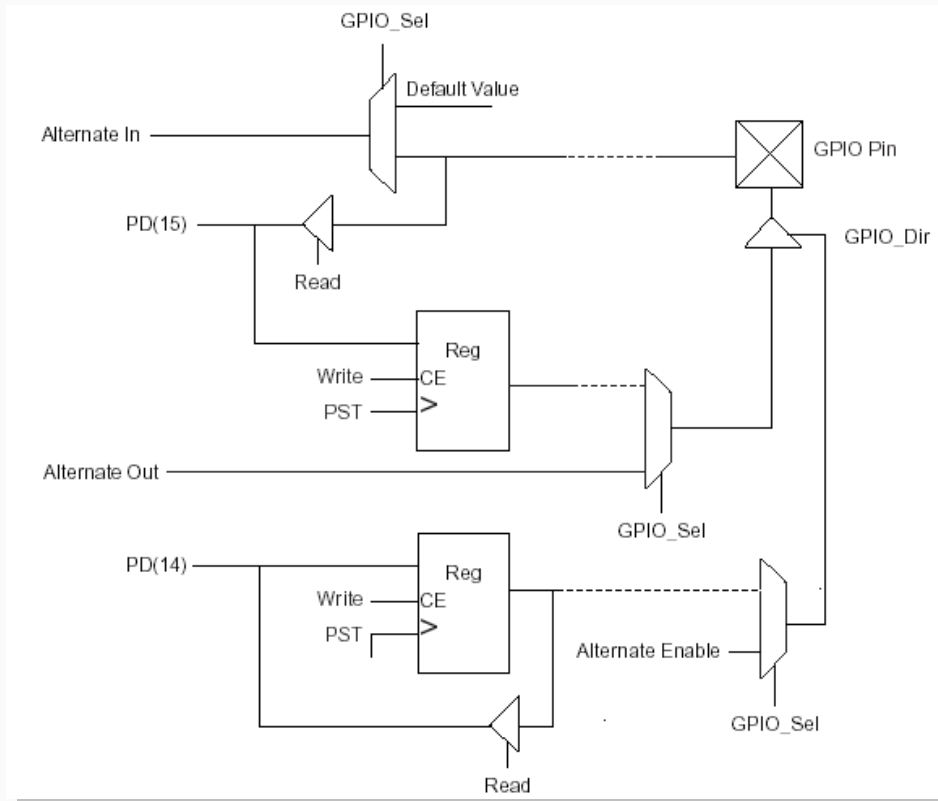


Figure 4: GPIO Pin Multiplexed with an Output Alternate Function

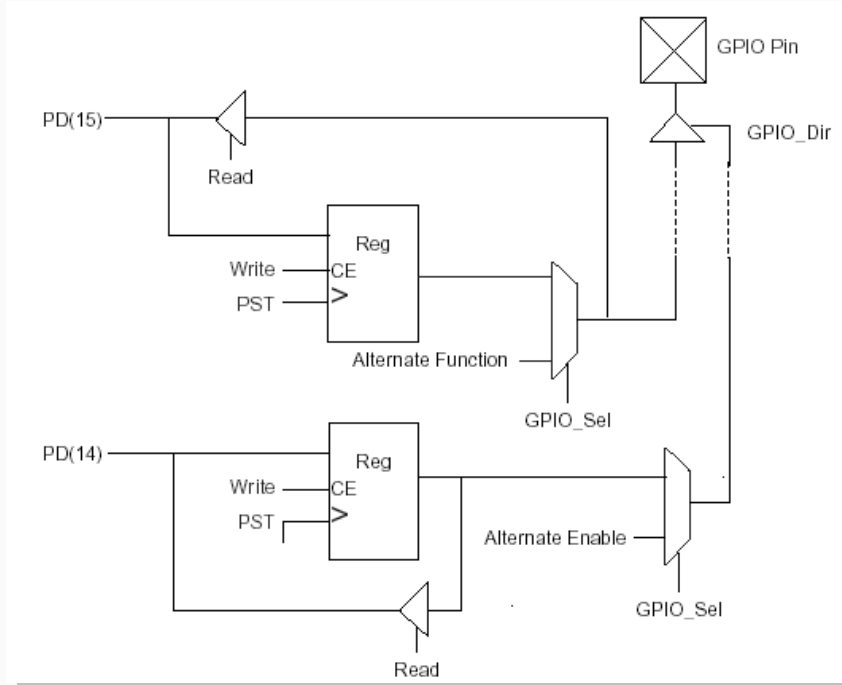


Figure 5: GPIO Pin without an Alternate Function

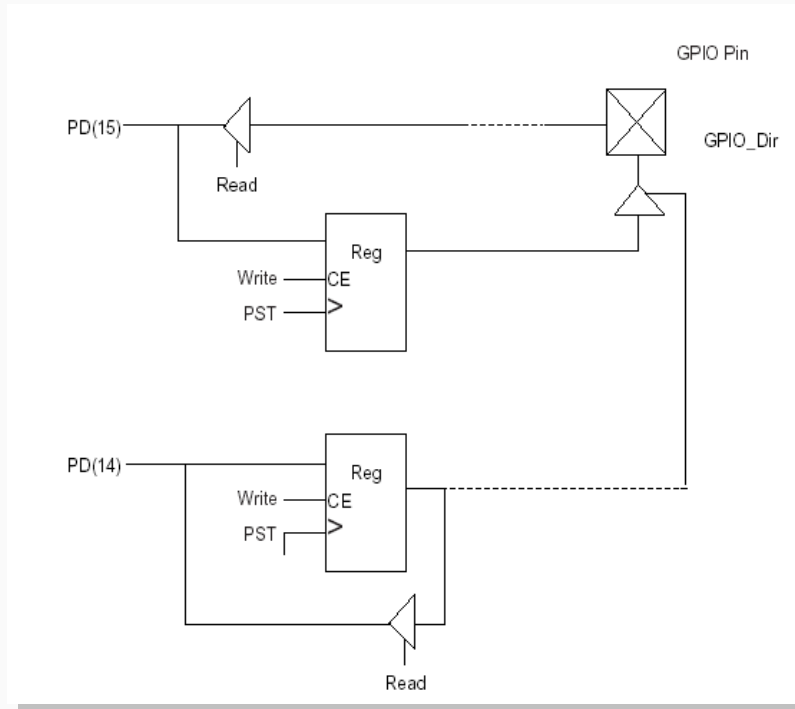


Figure 6: GPIO Pin Multiplex with a Bidirectional Alternate Function (e.g., GPIO[7] ( $\infty$ Wire clock))

## 1.5 Eval-Board LED Activation

The XF55-AVL evaluation-board has eight color LEDs, which are connected to 8 GPIO lines of embedded GPS receiver. These LEDs may be used by users for status display and debug. The GPIO signals are 0, 1, 5, 6, 7, 10 and 15 shown in the following table.

| GPIO Line | LED        | Ports and Bit                         |
|-----------|------------|---------------------------------------|
| 0         | D0 Yellow  | GPIO_PortDir0 and GPIO_PortVal0 Bit 7 |
| 1         | D1 Yellow  | GPIO_PortDir0 and GPIO_PortVal0 Bit 6 |
| 5         | D5 Yellow  | GPIO_PortDir0 and GPIO_PortVal0 Bit 2 |
| 6         | D6 Yellow  | GPIO_PortDir0 and GPIO_PortVal0 Bit 1 |
| 7         | D7 Yellow  | GPIO_PortDir0 and GPIO_PortVal0 Bit 0 |
| 10        | D10 Yellow | GPIO_PortState5 Bit 14 and 15         |
| 15        | D15 Yellow | GPIO_PortDir0 and GPIO_PortVal0 Bit 3 |

To light a LED, the Direction and Value of the GPIO pin must be set to 1. To clear the LED, the Direction must be set to 1 and the value set to 0. See the GPIO register memory locations. The memory locations are included here for convenience.

GPIO\_PortDir0 is at 0x800 10134

GPIO\_PortDir1 is at 0x80010138

GPIO\_PortDir2 is at 0x8001013C

GPIO\_PortVal0 is at 0x80010120


GPIO\_PortVal1 is at 0x80010124

GPIO\_PortVal2 is at 0x80010128

Below you will find a brief description of the included three files on the delivered CD which can be used on your own application:

`ioaccess.h` → this file includes the basic functions to access various IO's ports such as configuring alternative function, configuration of GPIOs as input or output, read the GPIO value and write the GPIO value. The defined functions can be called by using the function name (identifier) including a parameter enclosed in parentheses. The enter parameter (integer value) must be valid, by entering one of following values 0, 1, 5, 6, 7, 10 and 15. These numbers correspond to the provided GPIO respectively.

`ioaccess.c` → in this file a statement was added to configure the alternate functions of some of the General Purpose I/O (GPIO) lines and IO ports. The `ASIC_GPIO_SEL` select register value for all provided GPIOs is also defined in the `ports.h` extended file. The reference to the external file is also included using the `#include "ports.h"`.

 **HINT:** The defined function `SETIO(init value)` can be called by using the function name (identifier) including a value enclosed in parentheses. The value is an integer and can be set from 0 to 255. The 8-bit user defined value indicates which IOs will be activated. The specified user value will be converted in 8-bit binary format and processed internally.

For example the user defined value is 10. The hexadecimal value (10) corresponds to "0000 1010" converted in the binary format. The binary format indicates that the output 2 and 4 are set to active HIGH.

`ports.h` → in this file are included all required GPIO select registers. The GPIO select registers must not be modified by the user.

Example:

To toggle the LED at D1 (yellow or off) execute the following lines. Do not alter any bits in the register other than the one(s) you are interested in.

```
ASIC_GPIO_SEL |= (1<<0); // configure the GPIO as alternative function
ASIC_GPIO_PORTDIR0 &= ~(1 << 7); //The direction of GPIO0 is an output
ASIC_GPIO_PORTVAL0 |= (1 << 7); //the GPIO 0 will be switched on (is set to HIGH)
```

## 1.6 Used abbreviations

| Abbreviation | Description  |
|--------------|--|
| ASIC         | Application Specific Integrated Circuit                |
| GPS          | Global Positioning System                              |
| ARM          | Advanced RISC Machines                                 |
| UART         | Universal Asynchronous Receiver/Transmitter            |
| FIFO         | First In First Out                                     |
| SSTE         | Satellite Signal Tracking Engine                       |
| GPIO         | General Propose Input Output                           |
| LED          | Light Emitting Diode                                   |
| RF           | Radio Frequency  |
| RTC          | Real Time Clock  |
| CPU          | Central Processing Unit                                |
| A/D          | Analog/Digital   |
| EEPROM       | Electrically Erasable Programmable Read Only Memory    |
| TX           | Transmit Data  |
| RX           | Receive Data   |
| SI           | serial data in   |
| SO           | serial wire out  |
| SK           | serial data clock                                      |
| ADC          | Analog to Digital Converter                            |
| CD           | Compact Disc   |
| MSB          | Most Significant Bit (within a binary word or a byte.) |
| LSB          | Least Significant Bit (of a binary word).              |
| MI           | Module Interface                                       |
| UI           | User Interface   |
| Hz           | Hertz (a unit of frequency)                            |
| MHz          | Mega Hertz (a unit of frequency)                       |
| bps          | Bits per second  |
| I/O          | Input/Output   |