**QUADCOPTER**

**DOCUMENTATION**

by Martin Garaj

<garaj.martin@gmail.com>

Table of Contents

[Radio Communication 3](#_Toc483775678)

[Quartus II project - GHRD\_quad\_v2 3](#_Toc483775679)

[nios2\_gen2 configuration 4](#_Toc483775680)

[nios2\_ram configuration 7](#_Toc483775681)

[shared\_mem configuration 8](#_Toc483775682)

[pin definition in ghrd 9](#_Toc483775683)

[NIOS II Project 10](#_Toc483775684)

[uart internal registers 10](#_Toc483775685)

[Avalon Memory Mapped - bus width and its consequences 10](#_Toc483775686)

[Eclipse Project 11](#_Toc483775687)

[Avalon Memory Mapped - bus width and its consequences 11](#_Toc483775688)

[Debugging 12](#_Toc483775689)

[auto\_signaltap\_0 12](#_Toc483775690)

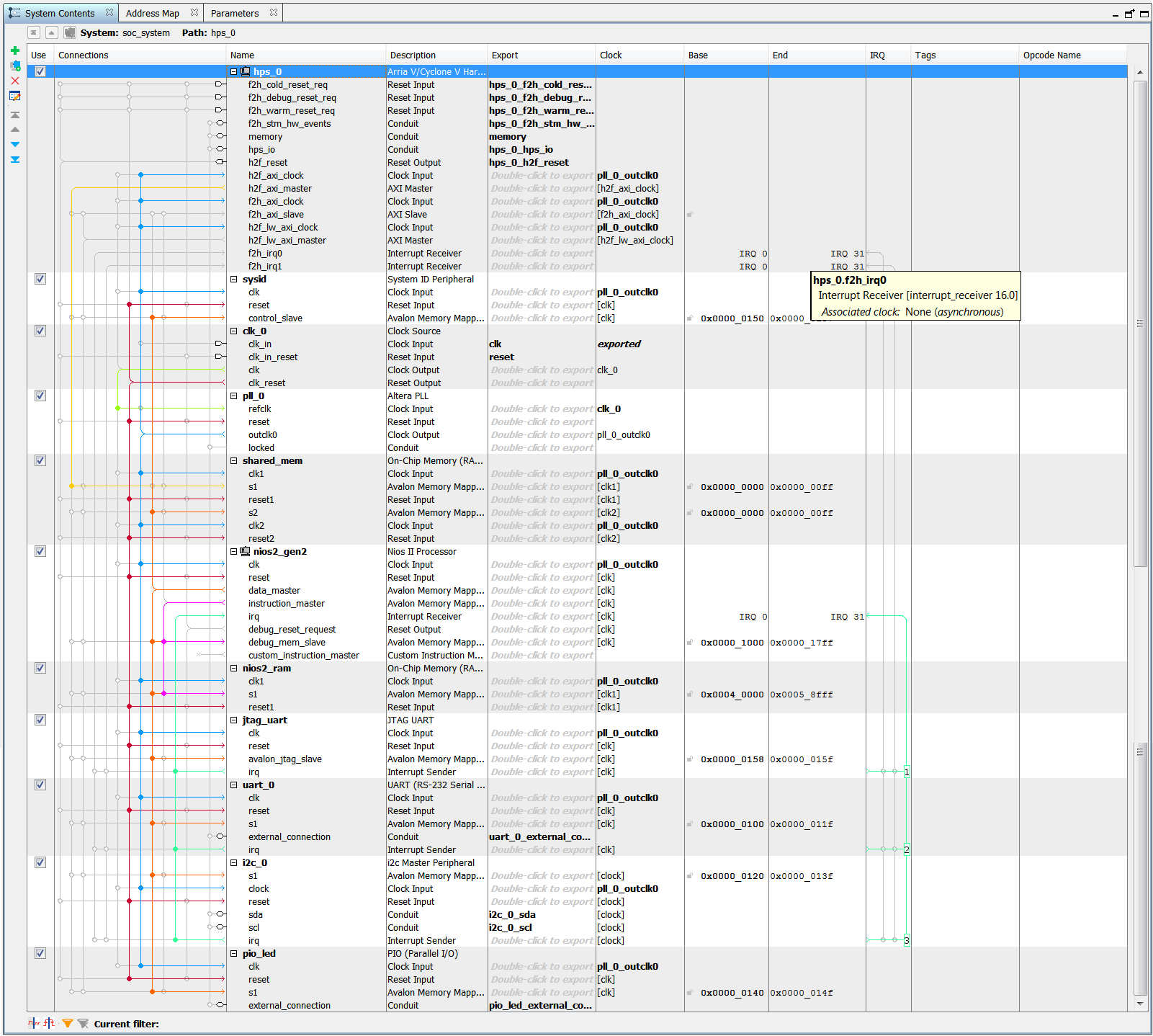
[Nios II Eclipse 12](#_Toc483775691)

[Radio Communication 14](#_Toc483775692)

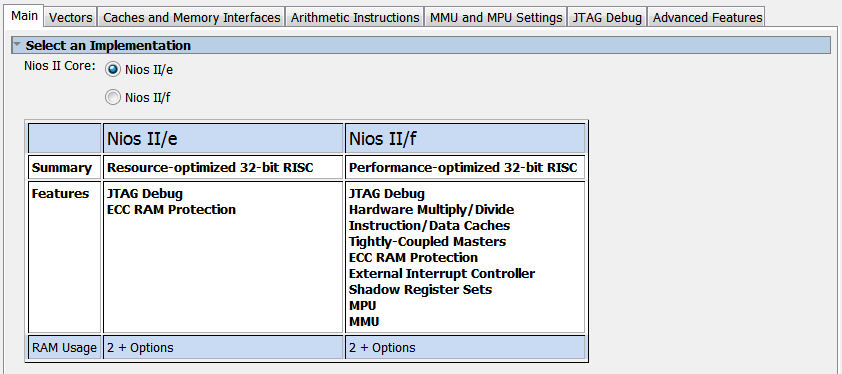
# Radio Communication projects

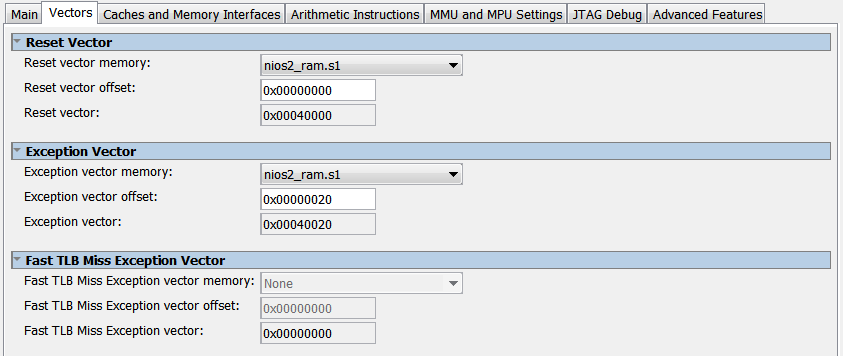
## Quartus II project - GHRD\_quad\_v2

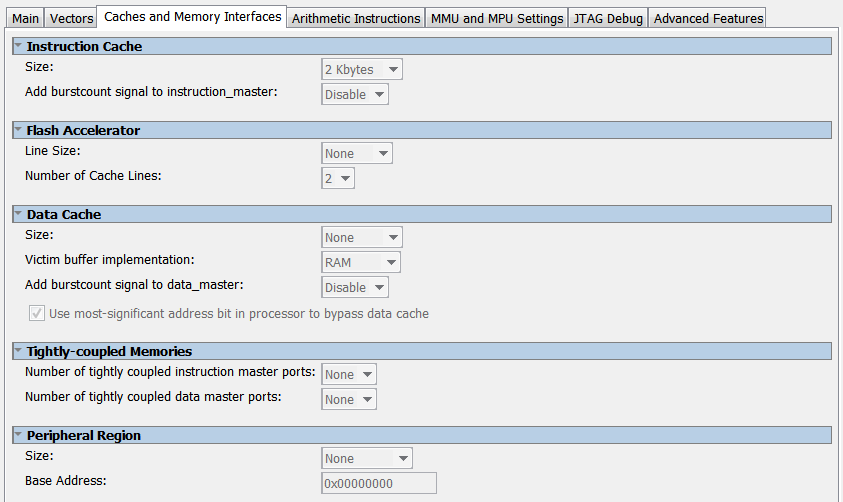
The figure presents the components inside the FPGA fabric

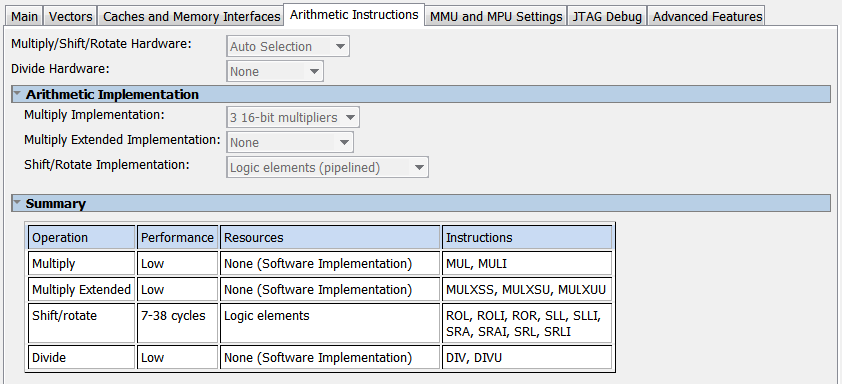


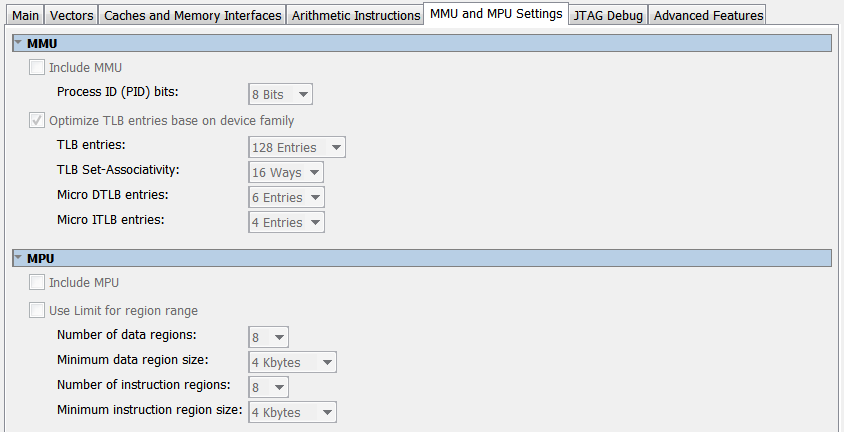
### nios2\_gen2 configuration

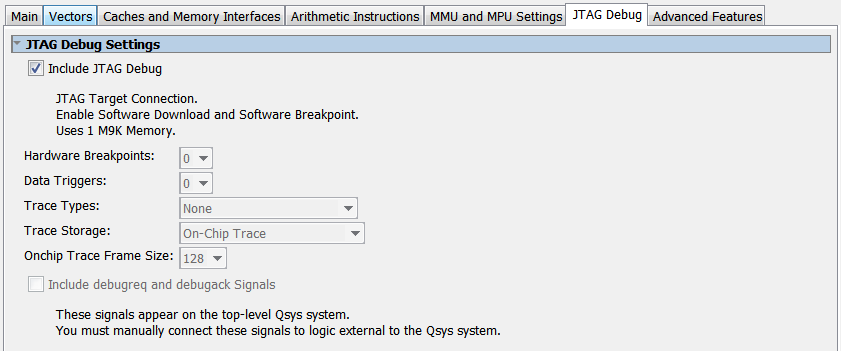


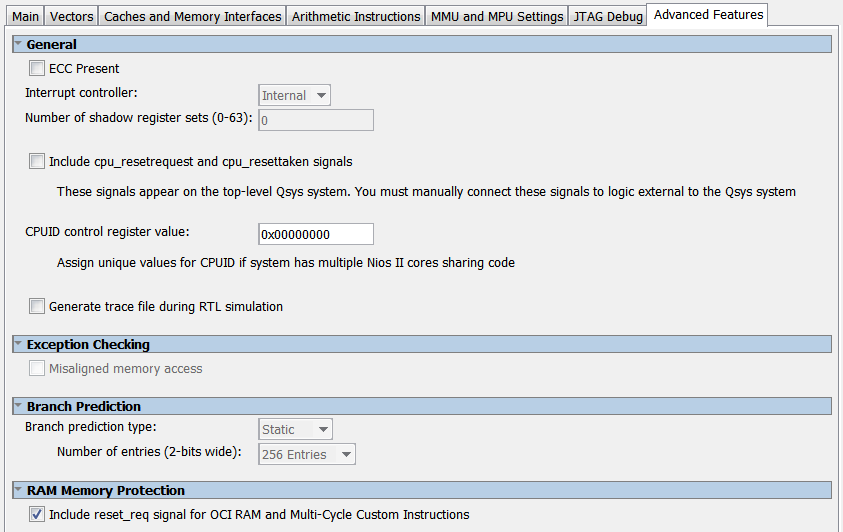




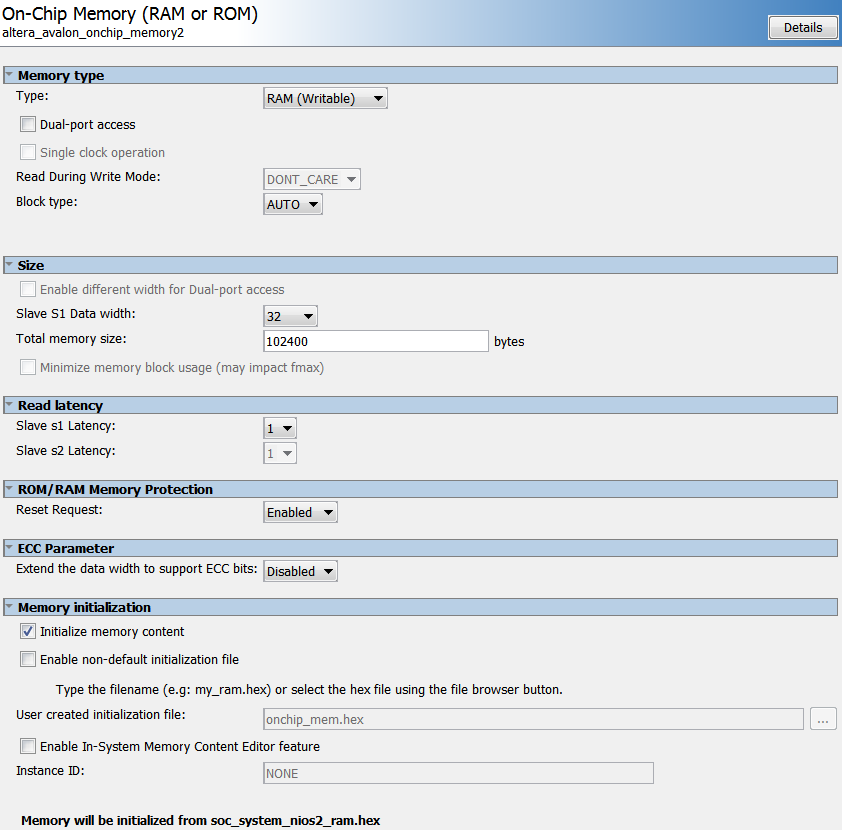








### nios2\_ram configuration



### shared\_mem configuration

## 

### pin definition in ghrd

.uart\_0\_external\_connection\_rxd (GPIO\_0[1]), // uart\_0\_external\_connection.rxd

.uart\_0\_external\_connection\_txd (GPIO\_0[0]), // .txd

.uart\_0\_external\_connection\_cts\_n (GPIO\_0[3]), // .cts\_n

.uart\_0\_external\_connection\_rts\_n (GPIO\_0[2]), // .rts\_n

.i2c\_0\_sda\_export (GPIO\_0[5]), // i2c\_0\_sda.export

.i2c\_0\_scl\_export (GPIO\_0[4]), // i2c\_0\_scl.export

// GPIO pins on De0\_nano\_SoC are as follows :

// ---------------------------------------------

// UART 0 TX | JP1 pin 1 |

// UART 0 RX | JP1 pin 2 |

// UART 0 RTS | JP1 pin 3 |

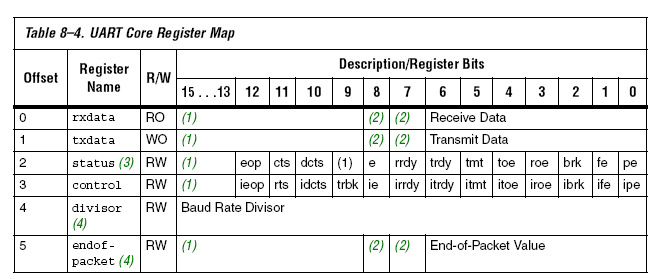
// UART 0 CTS | JP1 pin 4 |

// i2c 0 SCL | JP1 pin 5 |

// i2c 0 SDA | JP1 pin 6 |

## NIOS II Project - quad\_nios\_II\_v1

### uart internal registers



### Avalon Memory Mapped - bus width and its consequences

Nios II can use different macros to access data through Avalon MM bus. There are 2 variables, when designing communication :

1) Avalon MM bus width

2) Nios II macro

The alignment of the memory being accessed (such as in this case, the onchip\_mem) DEPENDS ON the Nios II macro used. Different type of macros :

general macros :

IORD(base, offset) WORD (4 byte) aligned   
 IOWR(base, offset, data) WORD (4 byte) aligned

size oriented macors

IORD\_8DIRECT(base, offset) BYTE alligned

IOWR\_8DIRECT(base, offset, data) BYTE alligned  
 IORD\_16DIRECT(base, offset) 2 BYTES alligned

IOWR\_16DIRECT(base, offset, data) 2 BYTES alligned  
 IORD\_32DIRECT(base, offset) WORD (4 byte) aligned  
 IOWR\_32DIRECT(base, offset, data) WORD (4 byte) aligned

NIOS can access any byte in On-Chip Memory, regardless of the Avalon MM bus width. The addressing is byte-by-byte, see the table below :

|  |  |
| --- | --- |
| BASE\_ADDRESS + 0 | 1st byte in On-Chip Memory |
| BASE\_ADDRESS + 1 | 2nd byte in On-Chip Memory |
| BASE\_ADDRESS + 2 | 3rd byte in On-Chip Memory |

This has been tested using 8bit and 32bit Avalon MM bus NIOS<->On-Chip Memory, using the macro IOWR\_8DIRECT.

## Eclipse Project - quad\_radio\_v2

### Avalon Memory Mapped - bus width and its consequences

HPS can access the On-Chip Memory through h2f\_axi\_master. Regardless of the On-Chip Memory port width (in other words connecting Avalon MM bus width between On-Chip Memory and HPS), the addressing is 1 word (4 bytes) aligned. The following table illustrates the addressing :

|  |  |
| --- | --- |
| HPS(BASE\_ADDRESS)+ 0 | 1st word (4 bytes) in On-Chip Memory |
| HPS(BASE\_ADDRESS)+ 1 | 2nd word (4 bytes) in On-Chip Memory |
| HPS(BASE\_ADDRESS)+ 2 | 3rd word (4 bytes) in On-Chip Memory |

When trying to access only the lower 2 bytes, using function such as follows :

**void** **set\_reg**(int16\_t reg\_value){

shared\_mem\_addr[0] = reg\_value;

}

and calling :

**set\_reg**(0x1234);

The On-Chip Memory contain the value 0xFFFF1234, the upper 2 bytes will be overwritten by 0xFFFF value. The correct way to access the word aligned memory is by the following function :

**void** **set\_reg**(int32\_t reg\_value){

shared\_mem\_addr[0] = reg\_value;

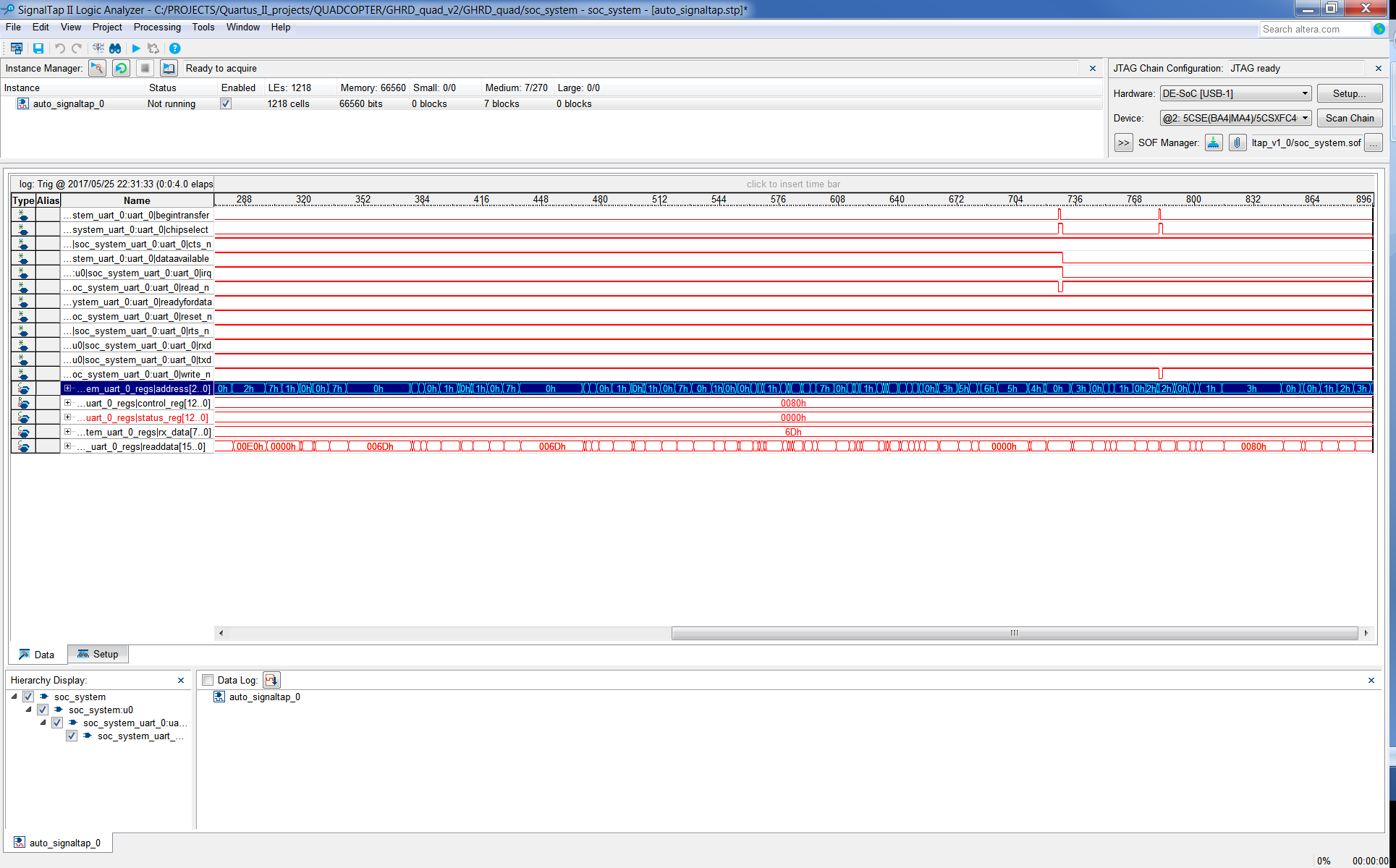
}

The above behavior has been tested using 16bit and 32bit Avalon MM bus HPS<->On-Chip Memory

## Debugging

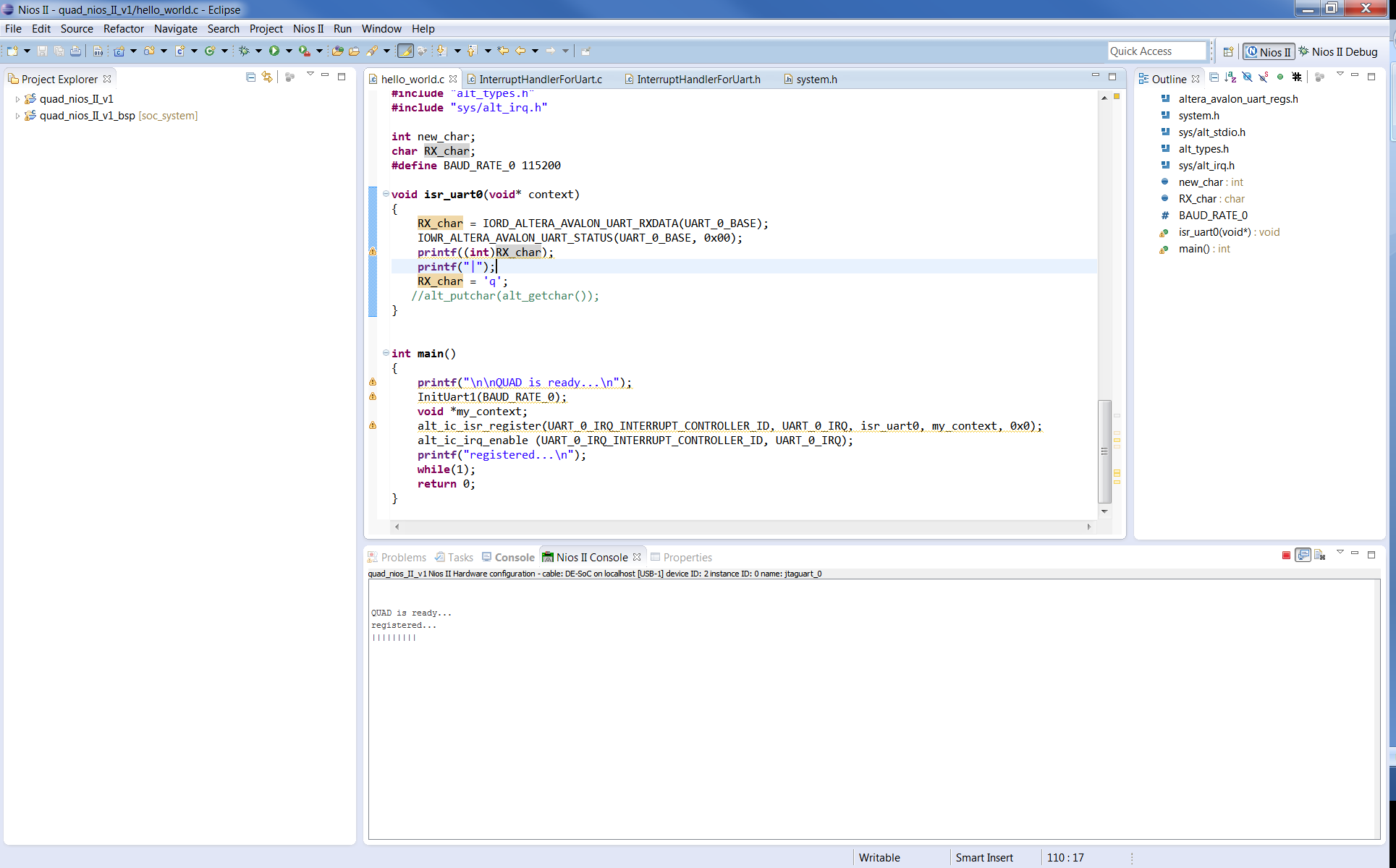
The FPGA design includes signal tap, tapping the UART module, this is regarding the IRQ and Avalon bus control, as well as Avalon bus data and address.

### auto\_signaltap\_0



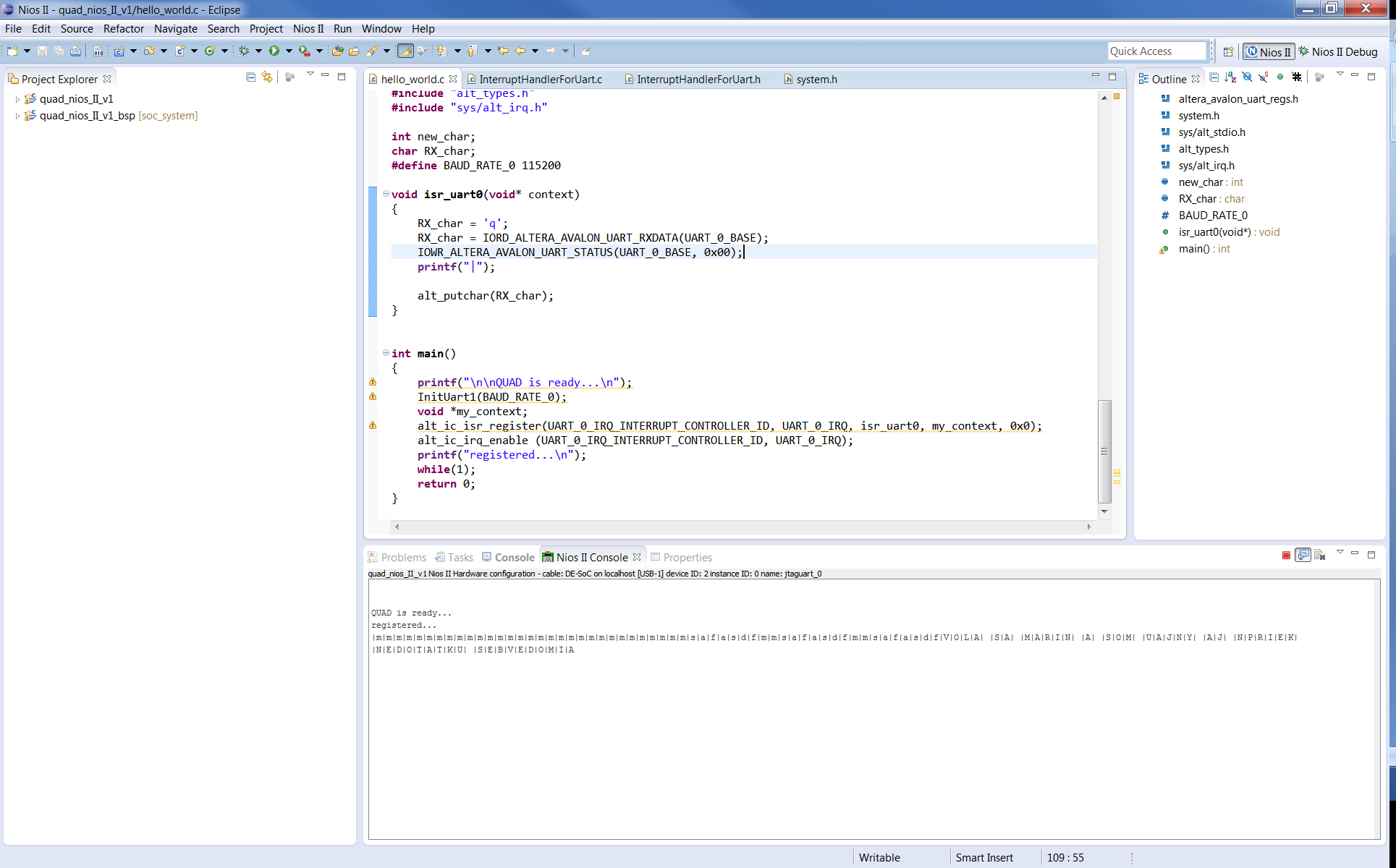
The IRQ of UART receiving a byte is recognized correctly. The above graph shows the Avalon bus communication during servicing the received IRQ. Both the RX (reading) and STATUS (writing) registers are accessed.

### Nios II Eclipse



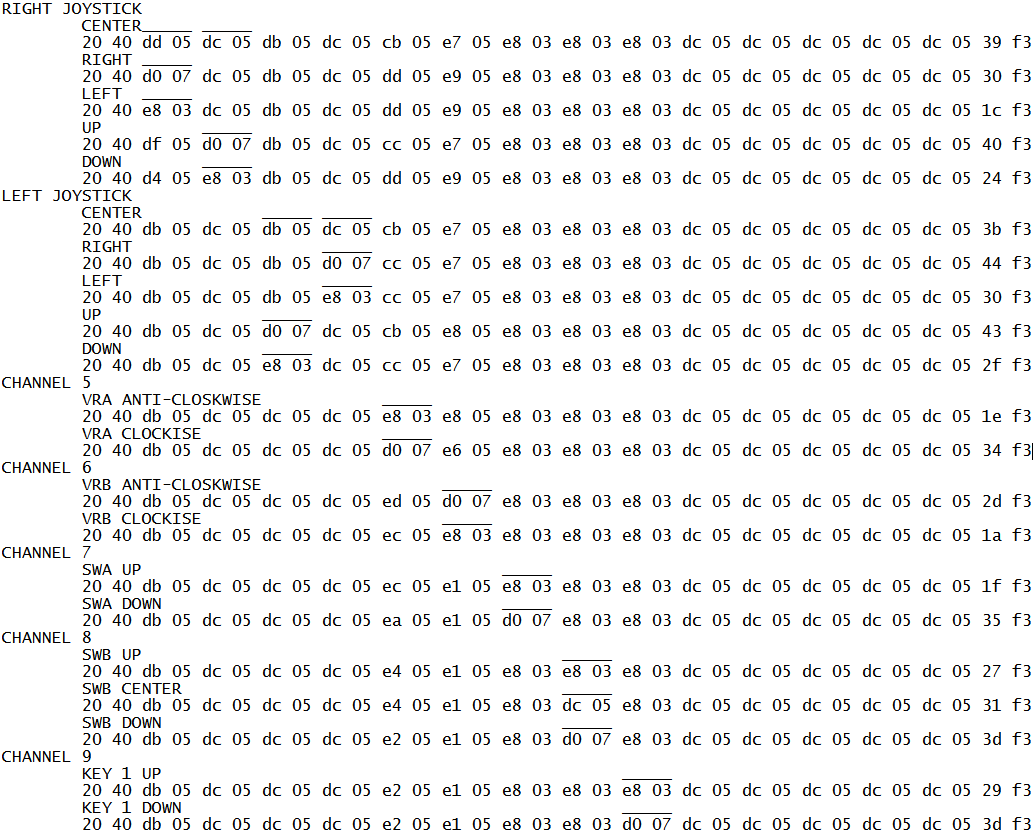
The received byte is not printed.

WORKING NOW



DONT USE printf !!!

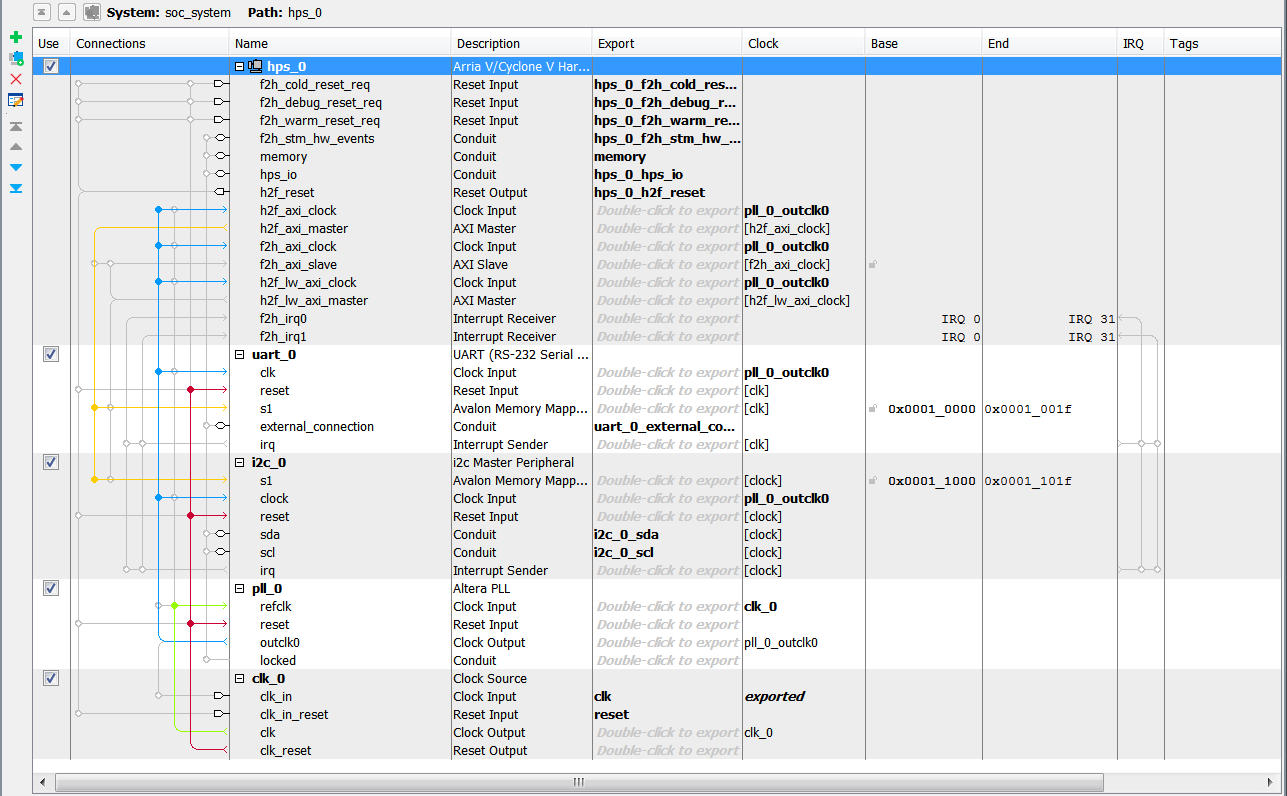
## Radio Communication



data sent by FS-iA10B according to radio transmitted signal (the channel 10 used for fly-mode change)

# Sensor communication

## Quartus II project - GHRD\_quad\_v1



### pin definition in ghrd

.uart\_0\_external\_connection\_rxd (GPIO\_0[1]), // uart\_0\_external\_connection.rxd

.uart\_0\_external\_connection\_txd (GPIO\_0[0]), // .txd

.uart\_0\_external\_connection\_cts\_n (GPIO\_0[3]), // .cts\_n

.uart\_0\_external\_connection\_rts\_n (GPIO\_0[2]), // .rts\_n

.i2c\_0\_sda\_export (GPIO\_0[5]), // i2c\_0\_sda.export

.i2c\_0\_scl\_export (GPIO\_0[4]) // i2c\_0\_scl.export

// GPIO pins on De0\_nano\_SoC are as follows :

// ---------------------------------------------

// UART 0 TX | JP1 pin 1 |

// UART 0 RX | JP1 pin 2 |

// UART 0 RTS | JP1 pin 3 |

// UART 0 CTS | JP1 pin 4 |

// i2c 0 SCL | JP1 pin 5 |

// i2c 0 SDA | JP1 pin 6 |

Eclipse project - quad\_MPU6050\_DMP\_v3