# Everest-RISCV-Demo

## Getting Started

# Contents

# Figures

# Tables

# 1. Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

## 1.1 Revision 1.2

The document was updated for Libero SoC v12.0.

## 1.2 Revision 1.1

The document was updated for Libero SoC PolarFire v2.2.

## 1.3 Revision 1.0

Revision 1.0 is the first publication of this document.

# 2. Getting Started

This demo design implements a RISCV soft processor subsystem with GPIO's and UART functionality and is based on the Microsemi PolarFire™ RISCV tutorial (TU0775_V1). Internal SRAM blocks are used for both, program and data memory. Two UART terminals are operated through USB connector J9. For further information relating on design flow and configuration of the IP cores please refer on:
"Microsemi_PolarFire_FPGA_Building_RISCV_Processor_Subsystem_Tutorial_TU0775_V1.pfd"

The application prints "Hello World" on two terminals. Characters entered on UART 0 are printed on UART 1 and vice versa. Four LEDs are toggled by a software counter in the main loop, that could be reseted by pressing one of the four push buttons.

## 2.1 Prerequisites

For the Everest Cortex M1 Demo the following is needed:

| **Item** | **Quantity** |
|---|---|
| Everest DEV Board | 1 |
| 12 V / 5 A wall-mounted power adapter | 1 |
| USB 2.0 A male to mini-USB B cable for UART / Programming interface to PC | 1 |
| Free one-year Libero Silver software license | 1 |

**Note 1:** The Everest DEV Board offers an on-board FlashPro5 programmer, which can be used to program and debug with Identify, SmartDebug and embedded application software using SoftConsole.

**Note 2:** The descripted design is suitable for Everest Dev Board Rev PROTO, A and B.

## 2.2 Handling the Board

Pay attention to the following points while handling or operating the board:

Handle the board with electrostatic discharge (ESD) precautions to avoid damage.

For information about ESD precautions see

https://www.microsemi.com/documentportal/doc_view/126483-esd-appnote.

## 2.3 Board-Setup Revision PROTO

### 2.3.1 Toggle-Switch S1 – PCIe

Warning: S1-1 and S1-2 must not be at position on at the same time!

| SWITCH ON | PCIe LANES |
|---|---|
| S1-1 | x1 |
| S1-2 | x4 |

### 2.3.2 Toggle -Switch S5 – SC SPI-Flash enable

Warning: S5-1 and S5-2 must not be at position on at the same time!

| SWITCH ON | SC SPI-FLASH |
|---|---|
| S5-1 | ENABLE |
| S5-2 | DISABLE |

### 2.3.3 DIP-Switch S8 – FMC Voltage Selector

Warning: S8-1 to S8-4 must not be at position on at the same time!

| SWITCH ON | FMC VOLTAGE |
|---|---|
| S8-1 | 3.3 V |
| S8-2 | 2.5 V |
| S8-3 | 1.8 V |
| S8-4 | undefined (not connected) |

### 2.3.4 Toggle -Switch S9 – VDDAUX2 & VDDAUX5 Voltage

Warning: S9-1 and S9-2 must not be at position on at the same time!

| SWITCH ON | VDDAUX2 & VDDAUX5 |
|---|---|
| S9-1 | 2.5 V |
| S9-2 | FMC voltage |

## 2.4 Board-Setup Revision A and B

### 2.4.1 Toggle-Switch S1 – PCIe

| **SWITCH** | **PCIe LANES** |
|---|---|
| S1-1 (marking) | x4 |
| S1-2 | x1 |

### 2.4.2 Toggle -Switch S5 – SC SPI-Flash enable

| **SWITCH** | **SC SPI-FLASH** |
|---|---|
| S5-1 (marking) | DISABLE |
| S5-2 | ENABLE |

### 2.4.3 DIP-Switch S8 – FMC Voltage Selector

| **SWITCH** | **FMC VOLTAGE** |
|---|---|
| S8-1 off, S8-2 off | 1.8 V |
| S8-1 on, S8-2 off | 2.5 V |
| S8-1 off, S8-2 on | undefined (not recommended) |
| S8-1 on, S8-2 on | 3.3 V |

### 2.4.4 Toggle -Switch S9 – VDDAUX2 & VDDAUX5 Voltage

| **SWITCH** | **VDDAUX2 & VDDAUX5** |
|---|---|
| S9-1 (marking) | 2.5 V |
| S9-2 | FMC voltage |

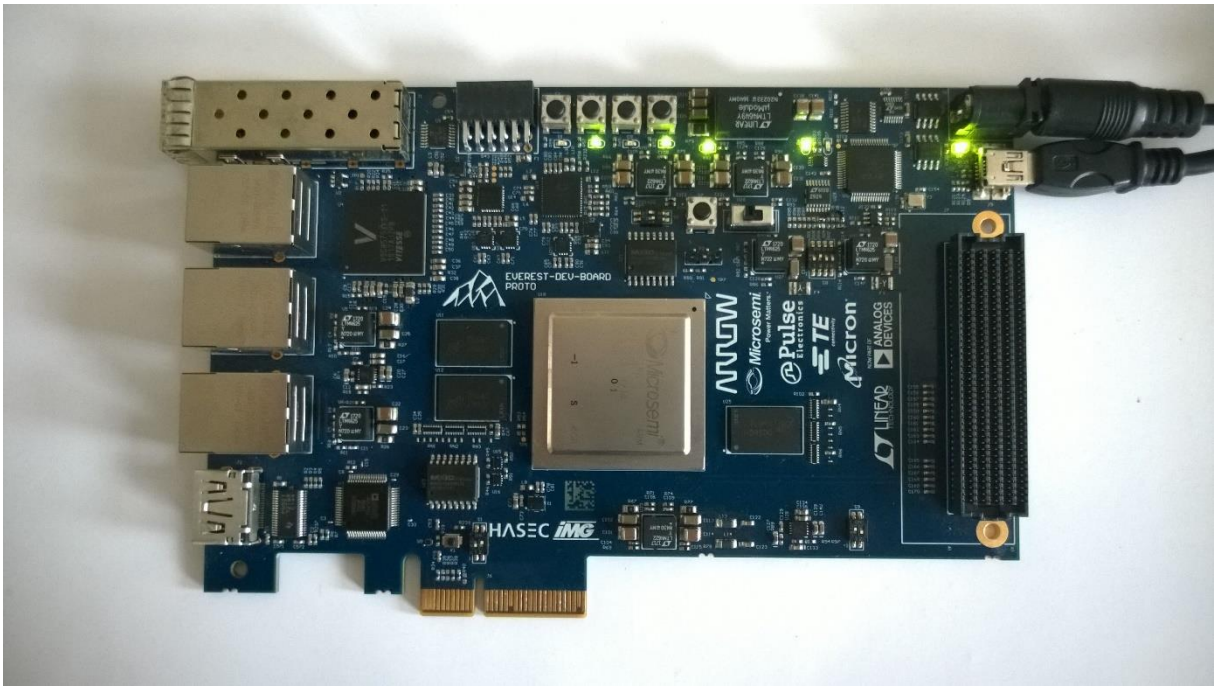


**Figure 1: Everest Board**

## 2.5 Powering up the Board

The Everest DEV Board is powered up using either the 12 V DC jack or the PCIe connector. For programming connect it although with your computer using USB mini B connector J9.

# 3. Demo Design

## 3.1 Prerequisites

**Table 1: Software / IP Requirements**

| **Software** | **Version** |
|---|---|
| Libero SoC PolarFire | V12.0 |
| Synplify Pro | L2017.09M-SP1-1 |
| FlashPro PolarFire | V2.0 |
| **IP** | |
| CORERISCV_AXI4 | 2.0.102 |
| AXI_GLUE_LOGIC | 1.0.7 |
| COREJTAGDEBUG | 3.0.100 |
| PF_SRAM_ABHL_AXI | 1.1.127 |
| PF_INIT_MONITOR | 2.0.103 |
| CoreAHBLite | 5.3.101 |
| COREAXITOAHBL | 3.3.102 |
| CoreAHBTOAPB3 | 3.1.100 |
| CoreAPB3 | 4.1.100 |
| PF_OSC | 1.0.102 |
| PF_CCC | 1.0.115 |
| CoreUARTapb | 5.6.102 |
| CoreGPIO | 3.2.102 |

Before you start you have to make sure, that all cores are downloaded to your local vault.

## 3.2 Design Implementation

The following table lists the clock frequencies used in the design.

**Table 2: Hardware Design Clock Frequencies**

| **Clock** | **Frequency (MHz)** |
|---|---|
| PF_OSC | 160 |
| PF_CCC OUT0_FABCLK | 80 |
| HCLK / PCLK | 80 |

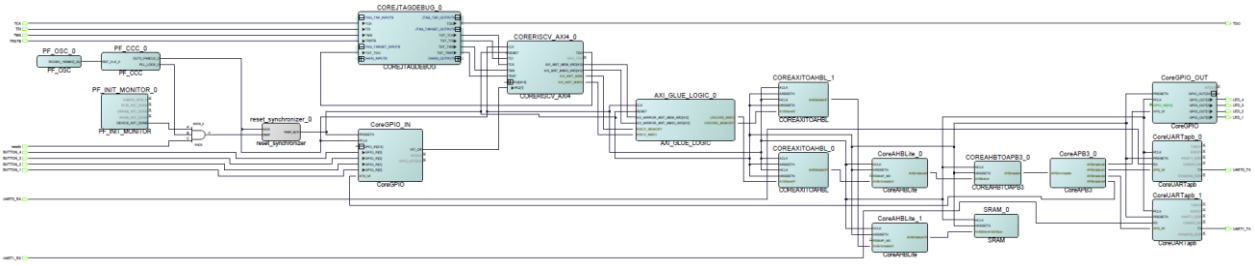

**Figure 2: Design Implementation**

The design is already fully implemented and ready to be programmed on the Everest Board. The board has to be connected with the power supply and to the PC with the USB cable. All drivers have to be installed (which should happen automatically when plugged in the first time) To program the design, there are two possibilities:

- Programming via Libero PolarFire SoC: Programming is started with the “Run PROGRAM Action” Button in the Design Flow Pane
- Programming via FlashPro Software: There is a STAPL-File (“<Design Directory>\designer\PROC_SUBSYSTEM\export\PROC_SUBSYSTEM.stp”) which can be programmed with the FlashPro Software. A new FlashPro project has to be generated and the programming file loaded into.

## 3.3 Running the Design

In Order to run the design, the RISCV-Processor has to be loaded with the firmware. To do so, load the provided SoftConsole Workspace.

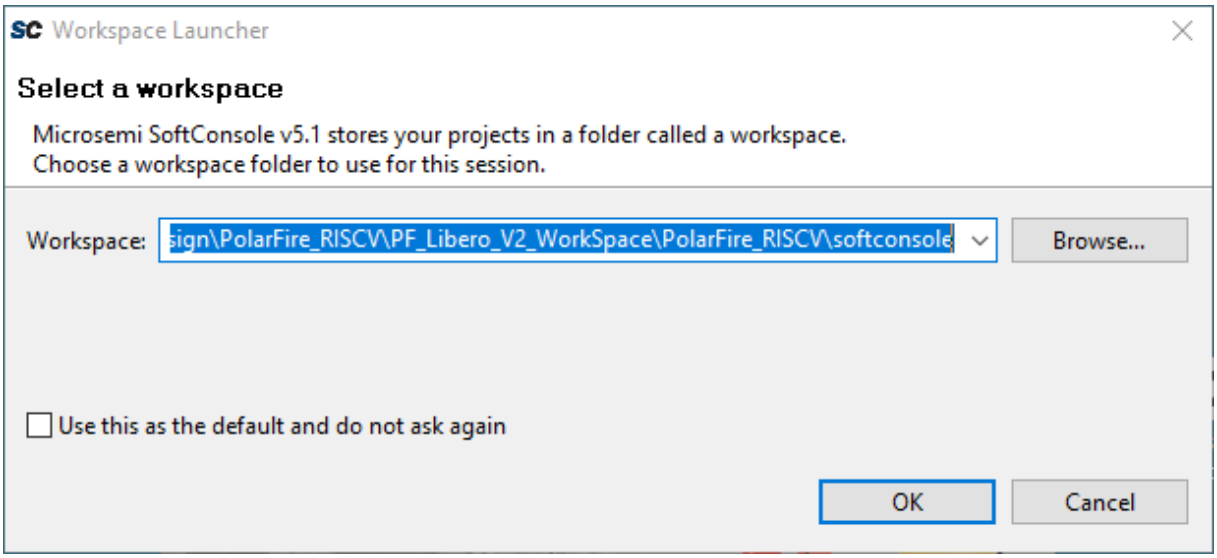


**Figure 3: SoftConsole v5.1 workspace launcher**

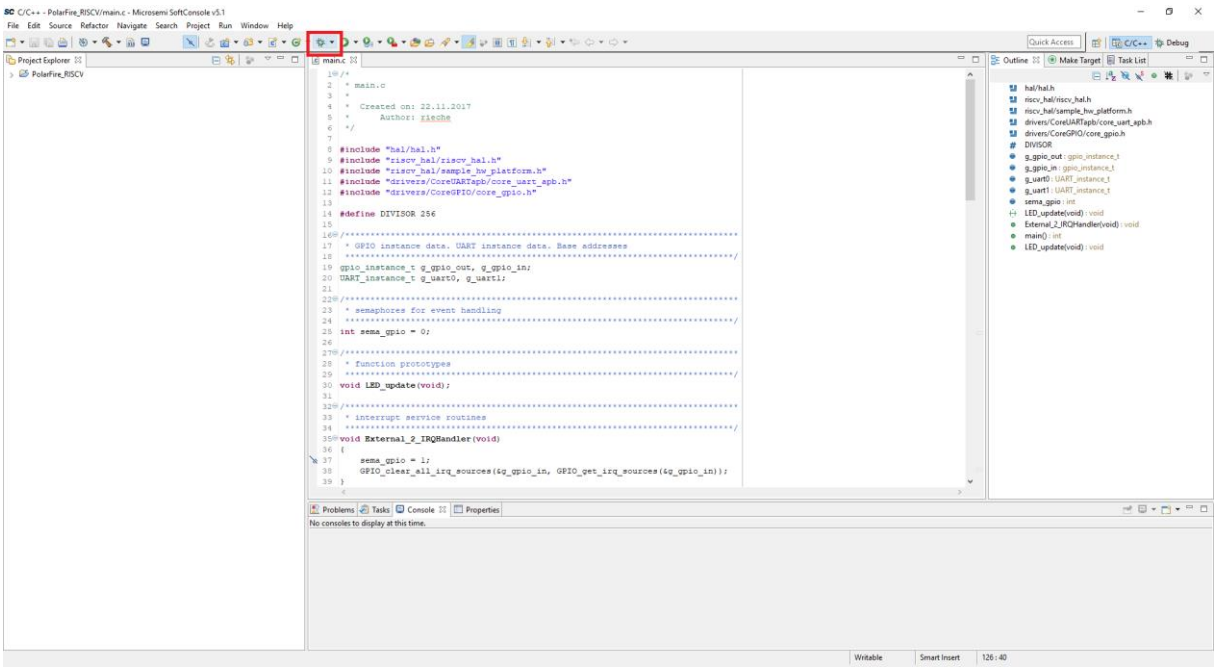

**Figure 4: SoftConsole v5.1 - starting the debug session**

A debug configuration is provided to download the firmware to the RISCV processor and start the application.

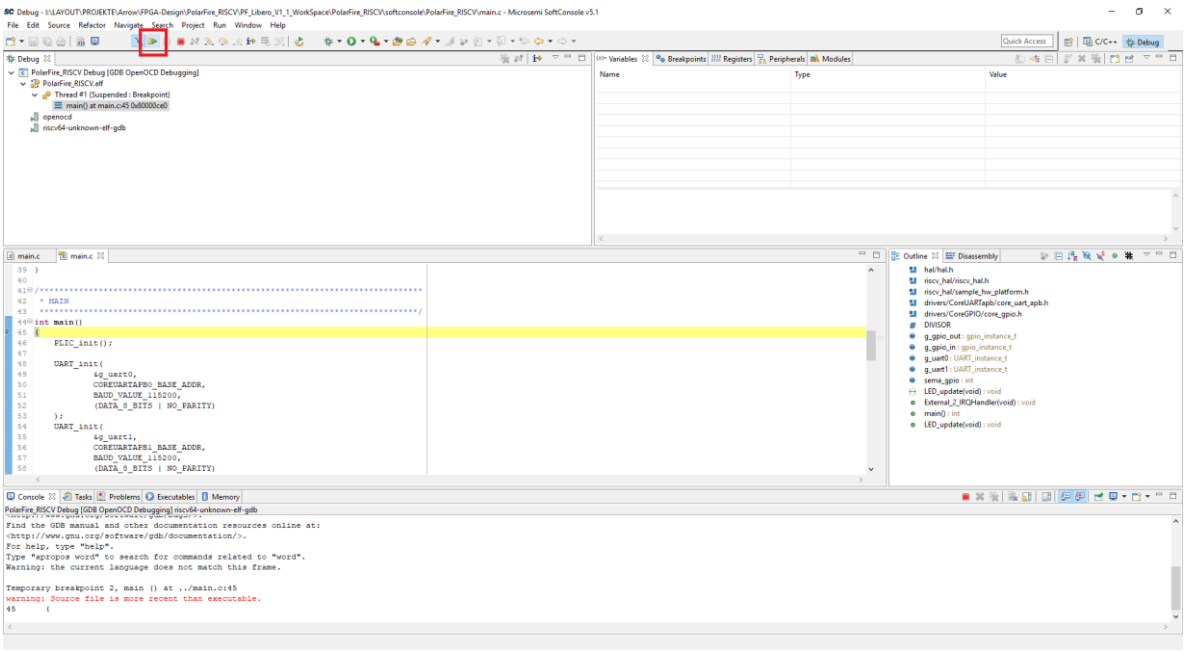


**Figure 5: SoftConsole v5.1 - running the design**

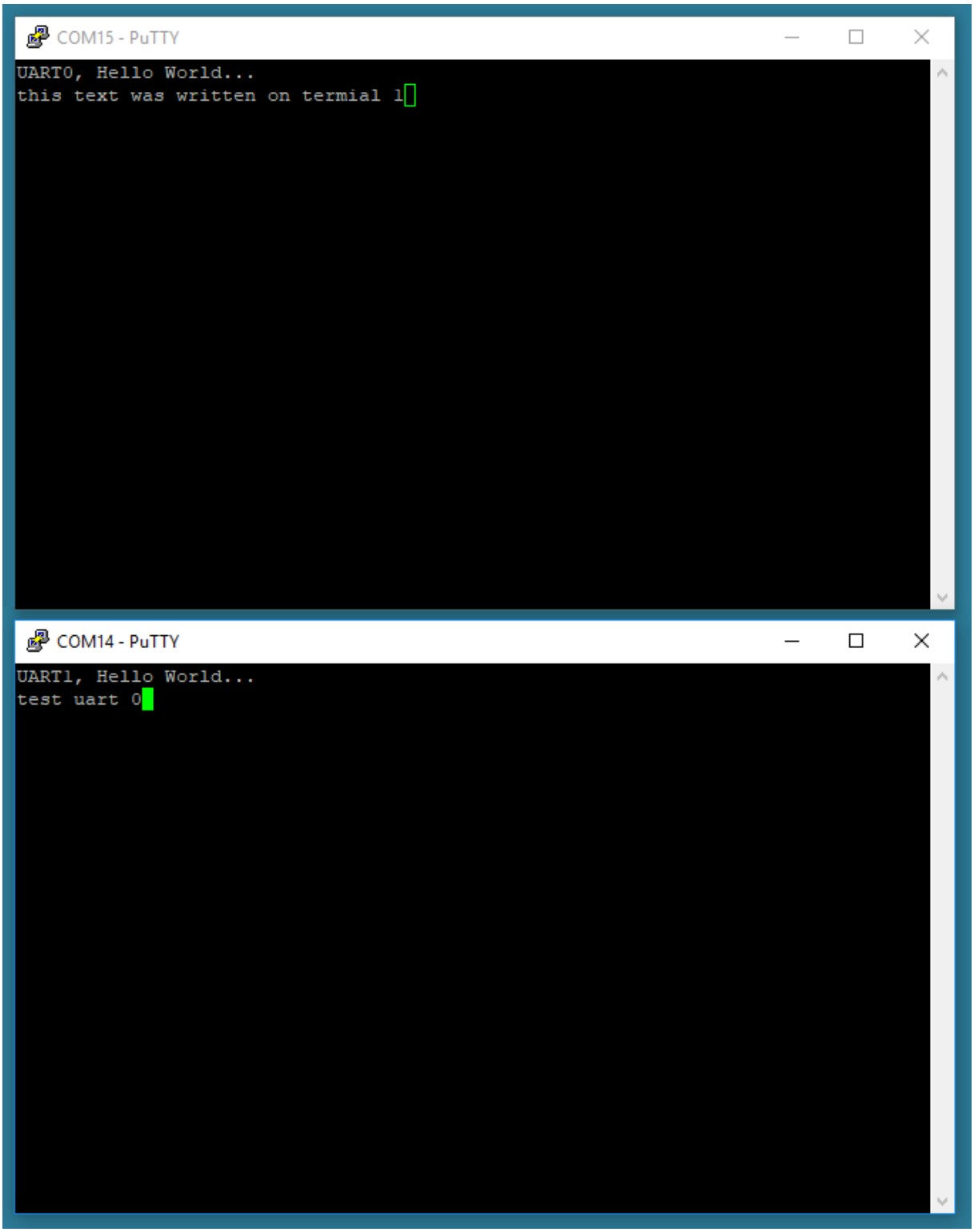


**Figure 6: terminal output**