

Everest-CortexM1-SFP+Loop-Demo

Getting Started

Project: Everest-CortexM1-SFP+Loop-Demo Getting Started	created:	S. Rieche	Date	2018-02-19
	edited:	S. Rieche	Date:	2019-03-19
	approved:		Date:	
Filename:	Everest-CortexM1-SFP+Loop-Demo--Getting_Started_1p2.docx			
Arrow Central Europe GmbH	Version:	1.2	Page 1 of 19	

Contents

1. Revision History	5
1.1 Revision 1.2.....	5
1.2 Revision 1.1.....	5
1.3 Revision 1.0.....	5
2. Getting Started	6
2.1 Prerequisites.....	6
2.2 Handling the Board.....	7
2.3 Board-Setup Revision PROTO.....	7
2.3.1 Toggle-Switch S1 – PCIe	7
2.3.2 Toggle -Switch S5 – SC SPI-Flash enable	7
2.3.3 DIP-Switch S8 – FMC Voltage Selector.....	7
2.3.4 Toggle -Switch S9 – VDDAUX2 & VDDAUX5 Voltage.....	7
2.4 Board-Setup Revision A and B	8
2.4.1 Toggle-Switch S1 – PCIe	8
2.4.2 Toggle -Switch S5 – SC SPI-Flash enable	8
2.4.3 DIP-Switch S8 – FMC Voltage Selector.....	8
2.4.4 Toggle -Switch S9 – VDDAUX2 & VDDAUX5 Voltage.....	8
2.5 Powering up the Board.....	9
3. Demo Design	10
3.1 SFP+ related differences between Everest DEV Board PROTO and Revision A and B	10
3.2 Prerequisites.....	12
3.3 Design Implementation	13
3.4 Running the Design.....	16

Figures

Figure 1: Everest Board.....	9
Figure 2: SFP+ hardware Everest DEV Board PROTO	10
Figure 3: SFP+ hardware Everest DEV Board Rev. A and B.....	11
Figure 4: Design Implementation – Top Level Everest DEV Board PROTO	13
Figure 5: Design Implementation – Top Level Everest DEV Board Rev. A and B.....	14
Figure 6: Design Implementation – Modul PF_XCVR_8b10b.....	14
Figure 7: SoftConsole v5.1 workspace launcher	16
Figure 8: SoftConsole v5.1 - starting the debug session	16
Figure 9: SoftConsole v5.1 - running the design	17
Figure 10: terminal output after startup	18
Figure 11: Terminal Output - sending some characters via SFP+ Loop	19

Tables

Table 1: Software / IP Requirements 12

Table 2: Hardware Design Clock Frequencies 13

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 SFP+ loop based on the Cortex M1 Demo design, offering a soft processor subsystem with GPIO's, UART and I2C functionality. Internal SRAM blocks are used for both, program and data memory. Two UART terminals are operated through USB connector J9.

The application prints "Hello World" on both terminals. Characters entered on UART 0 are printed on UART 1 and vice versa. LEDs 1 and 2 are toggled by a software counter in the main loop, that could be reseted by pressing one of the four push buttons. LED 4 indicates that the receiver of XCVR lane 0 is ready and LED 3 is on between the transmission and reception of a frame. Every frame consists of the comma character "BC", the actual frame counter value and the hexadecimal representation of the ASCII character and is printed out on UART 0.

2.1 Prerequisites

For the Everest Cortex M1 SFP+ Loop 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
Intel X520-DA2 10GbE Adapter PCIE	1
Finisar FTLX8574D3BCV 10G Optical Transceiver	2
Tripp Lite N820-03M Optical Duplex LC Cable	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: There are differences between Everest Dev Board PROTO and Rev. A and B concerning the configuration of the SFP+ interface that are described in chapter 3.1.

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

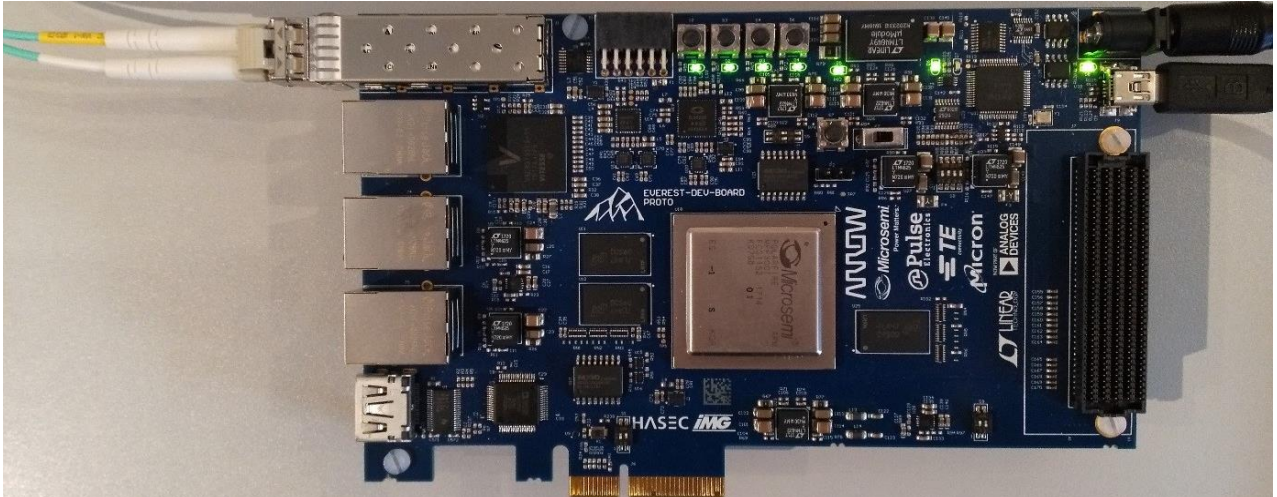


Figure 1: Everest Board

2.5 Powering up the Board

Insert the Finisar Optical Transceiver into the J1 connector (SFP+ module cage). A loop has to be built by connecting the transmitter with the receiver with an optical fiber cable. The Everest DEV Board is powered up using the 12 V DC jack. For programming connect it although with your computer using USB mini B connector J9.

3. Demo Design

3.1 SFP+ related differences between Everest DEV Board PROTO and Revision A and B

On Everest DEV Board PROTO the SFP+ signals *RX_LOS*, *RS0*, *RS1*, *TX_DIS* and *TX_FAULT* could only be accessed via the I2C I/O expander PCA9538 that is connected to the same I2C bus as the SFP+ interface itself.

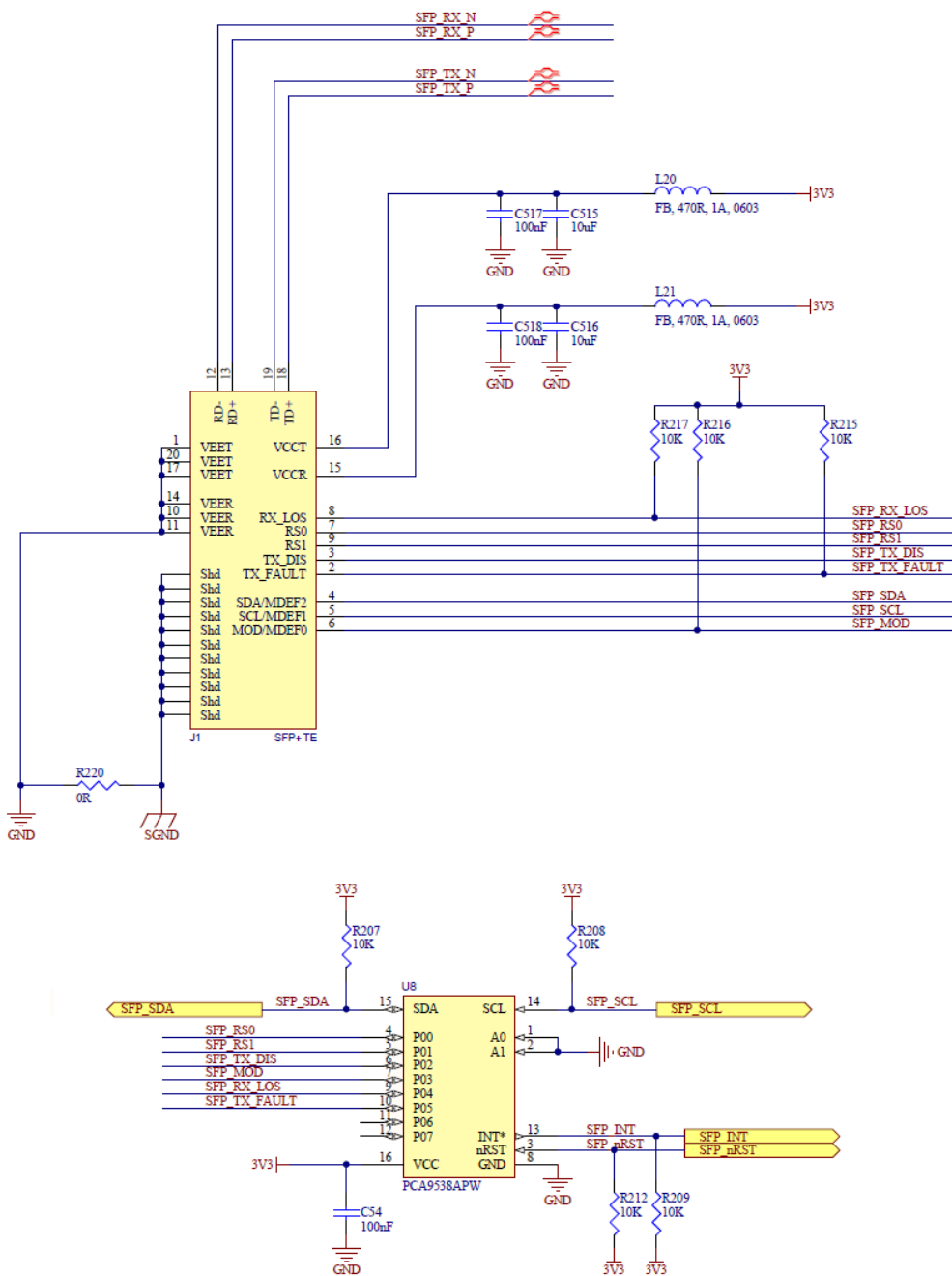


Figure 2: SFP+ hardware Everest DEV Board PROTO

On Everest DEV Board Rev. A and B these signals are routed directly to GPIOs.

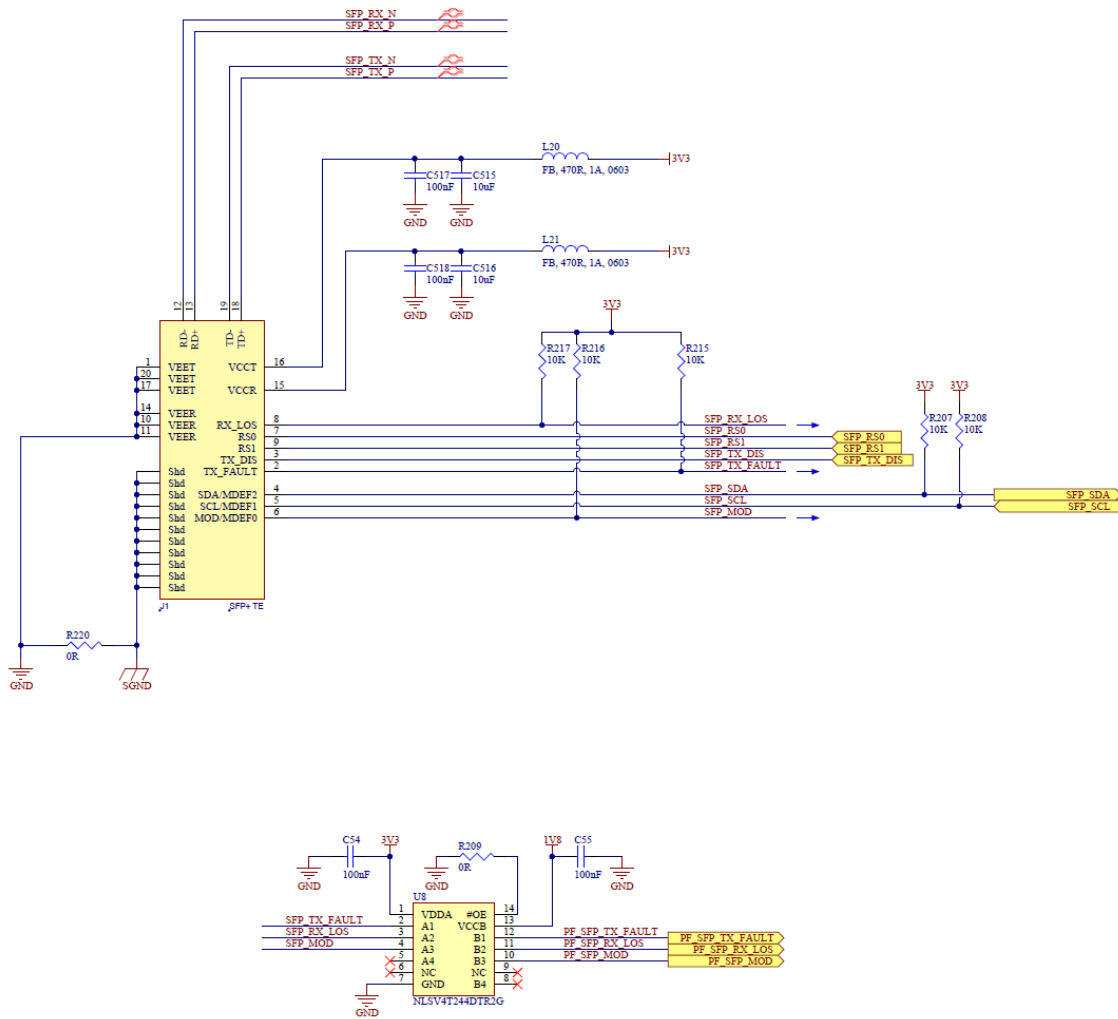


Figure 3: SFP+ hardware Everest DEV Board Rev. A and B

3.2 Prerequisites

Software and IP core requirements are the same for Everest DEV Board PROTO and Rev. A and B.

Table 1: Software / IP Requirements

Software	Version
Libero SoC	V12.0
Synplify Pro	N-2018.03M-SP1-1
FlashPro PolarFire	V2.0
IP	
CortexM1	3.0.100
PF_SRAM_ABHL_AXI	1.1.127
PF_INIT_MONITOR	2.0.103
CoreAHBLite	5.3.101
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
COREI2C	7.2.101
PF_XCVR_REF_CLK	1.0.103
PF_TX_PLL	2.0.002
PF_XCVR_ERM_C0	2.0.201

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

3.3 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	27.5
HCLK / PCLK	27.5
PF_TX_PLL	156.25
DIV_CLK	125

The top-level design implementation for Everest DEV Board PROTO is shown in Figure 4.

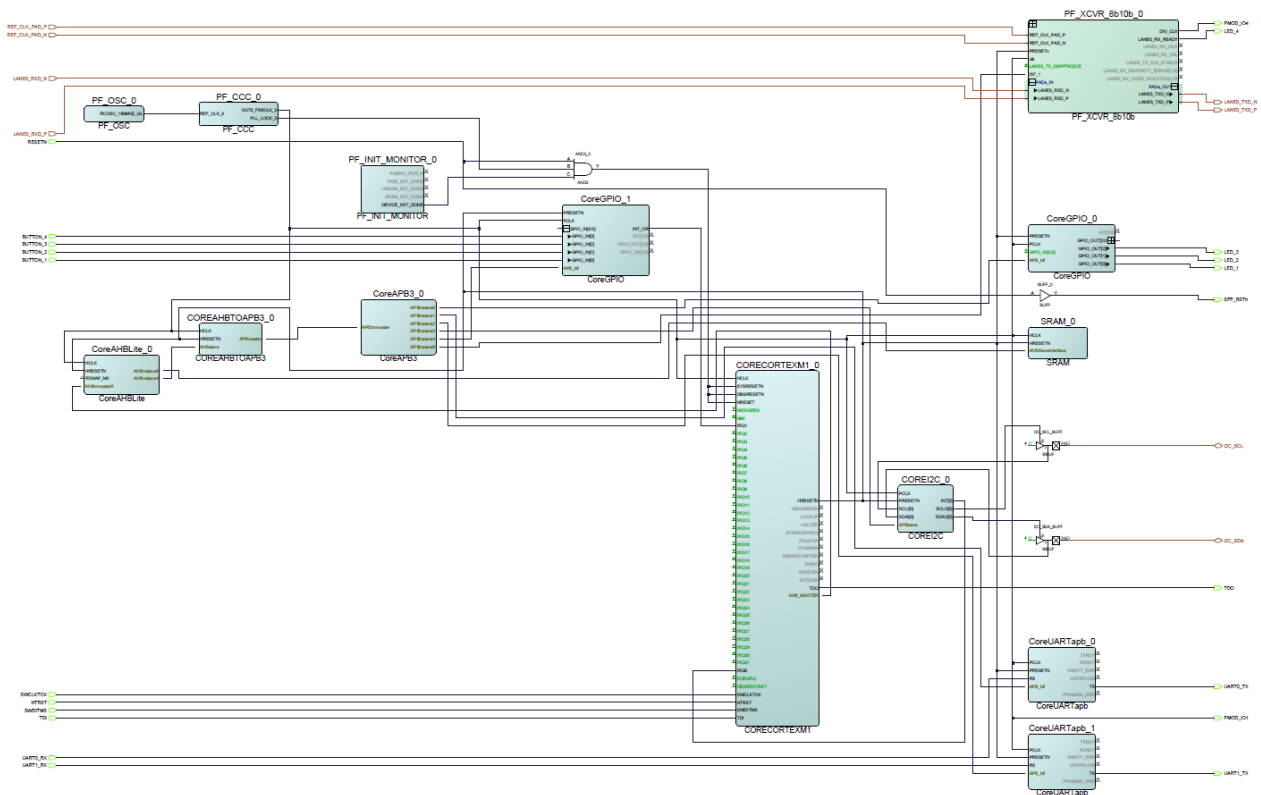


Figure 4: Design Implementation – Top Level Everest DEV Board PROTO

The top-level design implementation for Everest DEV Board Rev. A and B has an extra CoreGPIO called *SFP_CTRL* that receives the signal *SFP_MOD*, *SFP_TX_FAULT* and *SFP_RX_LOS*, including interrupt generation for those signal, and drives the signals *SFP_TX_DIS*, *SFP_RS0* and *SFP_RS1*.

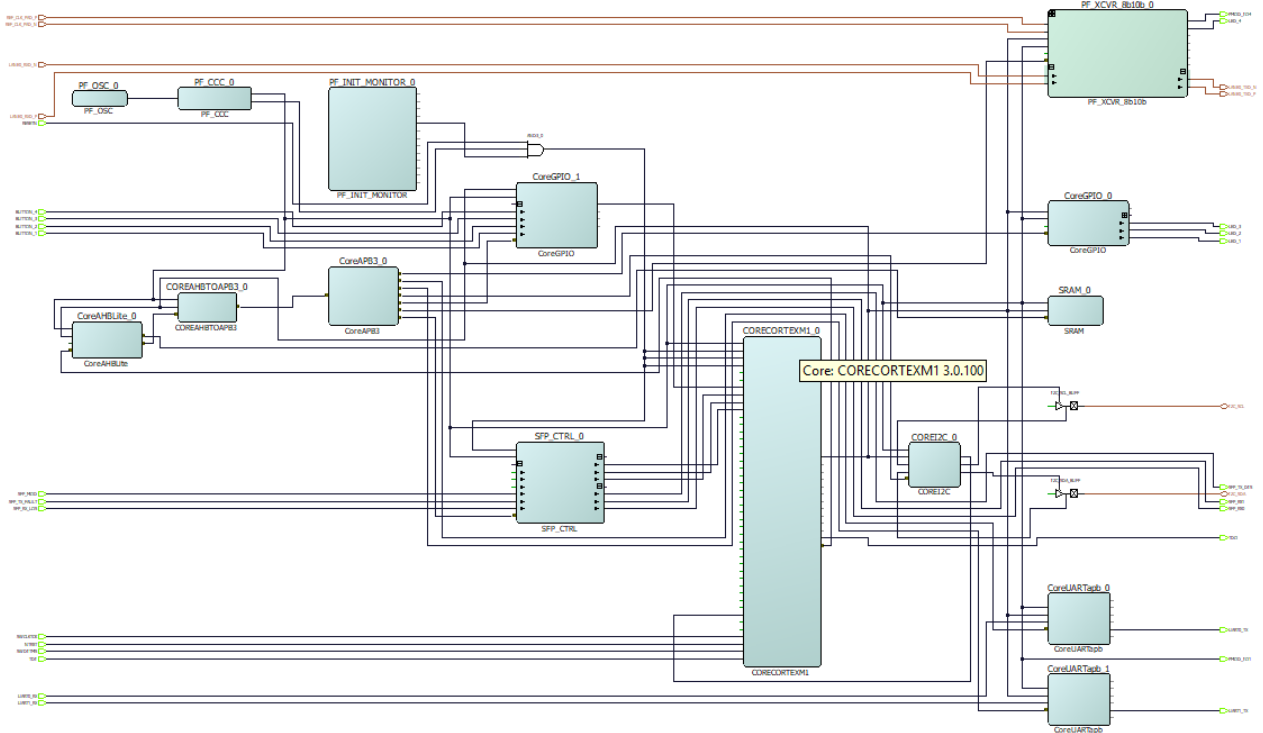


Figure 5: Design Implementation – Top Level Everest DEV Board Rev. A and B

The implementation of submodule *PF_XCVR_8b10b* is the same for all revisions.

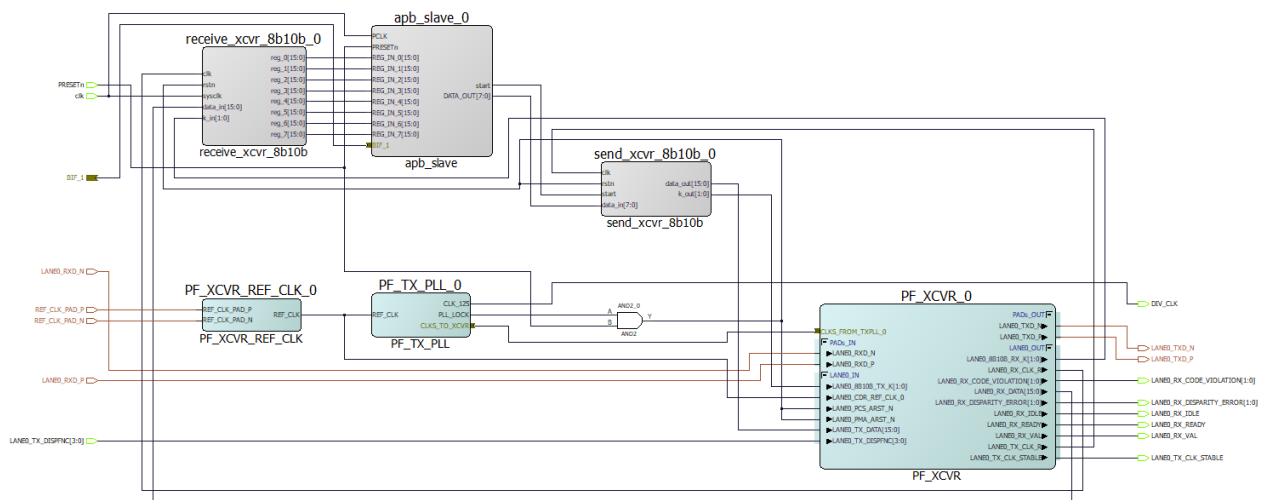


Figure 6: Design Implementation – Modul PF_XCVR_8b10b

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: For preproduction and production devices use the STAPL-file in the “Bitstream” folder. The STAPL-file for engineering samples is located in the folder “Bitstream_ES”. A new FlashPro project has to be generated and the programming file loaded into.

3.4 Running the Design

In Order to run the design, the CortexM1-Processor has to be loaded with the firmware. To do so, load the provided SoftConsole Workspace.

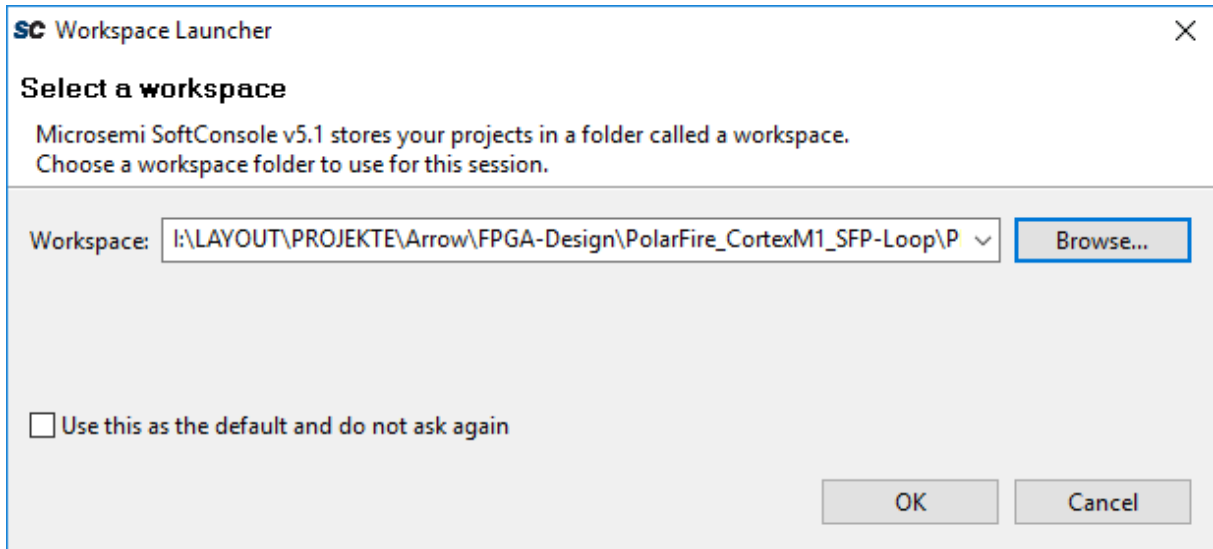


Figure 7: SoftConsole v5.1 workspace launcher

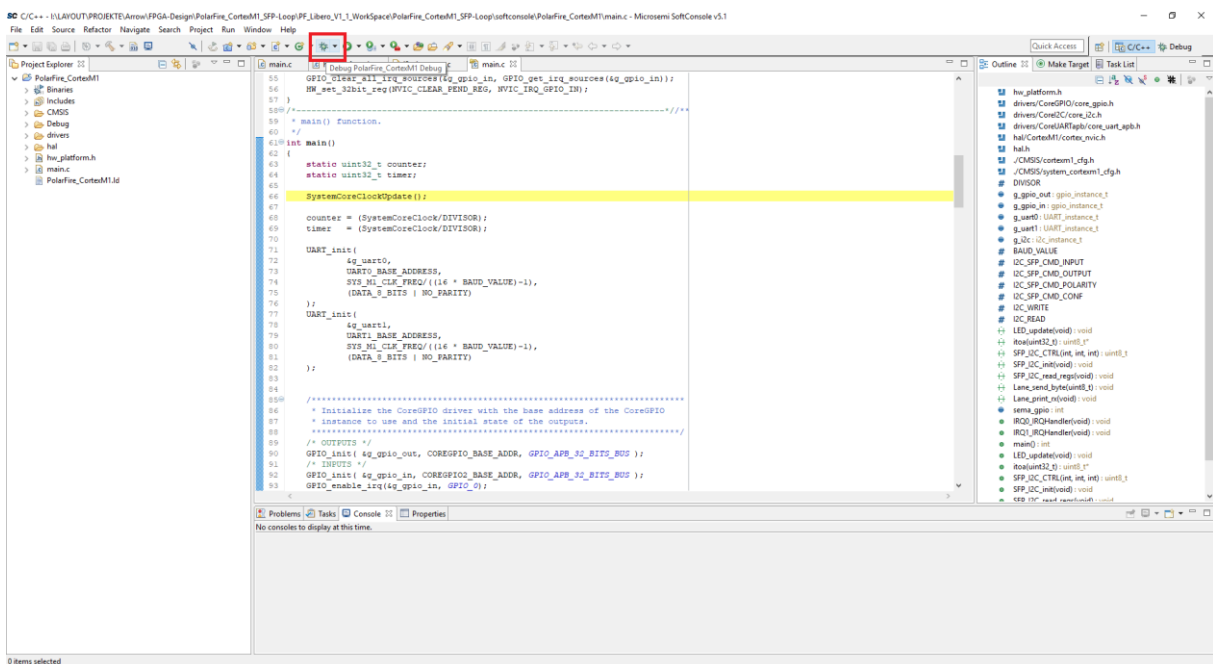


Figure 8: SoftConsole v5.1 - starting the debug session

A debug configuration is provided to download the firmware to the CortexM1 processor and start the application.

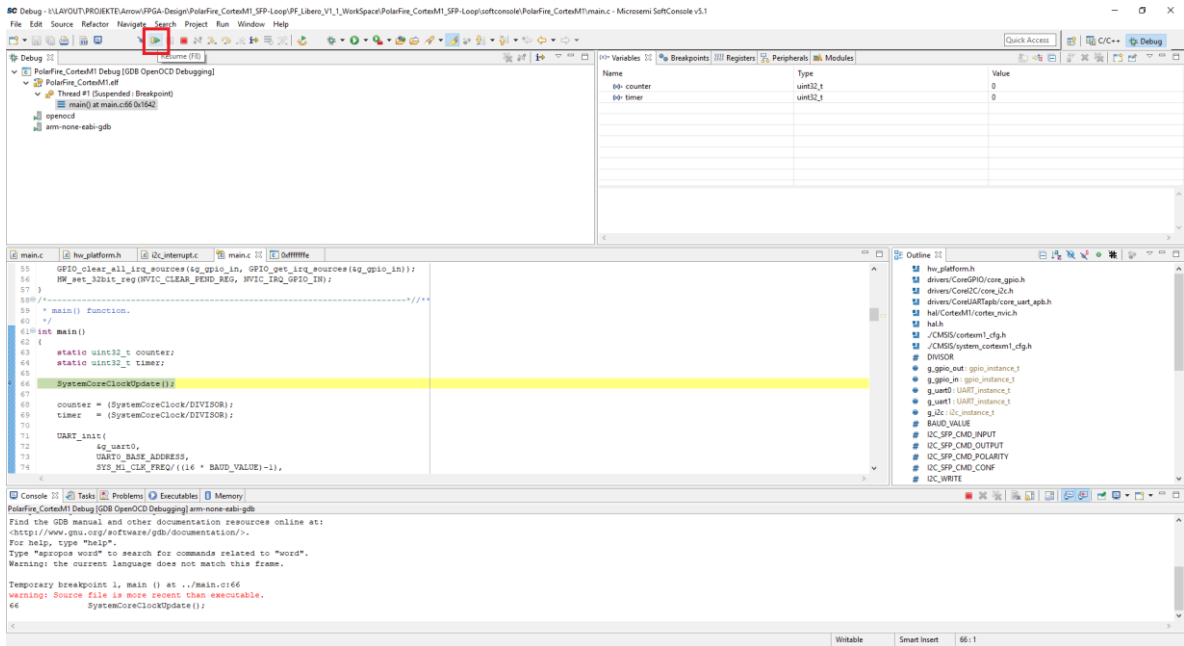
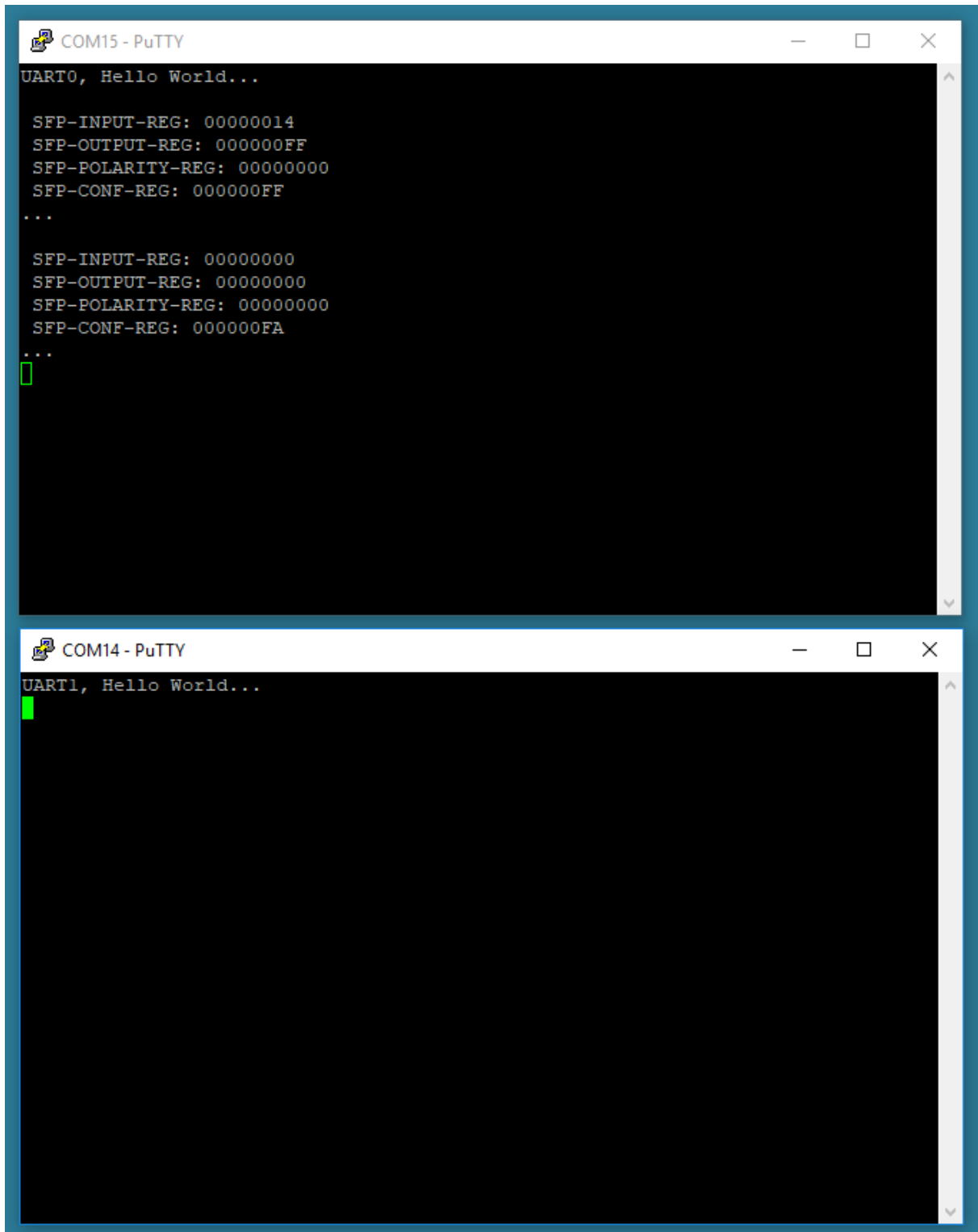


Figure 9: SoftConsole v5.1 - running the design



```
COM15 - PuTTY
UART0, Hello World...

SFP-INPUT-REG: 00000014
SFP-OUTPUT-REG: 000000FF
SFP-POLARITY-REG: 00000000
SFP-CONF-REG: 000000FF
...

SFP-INPUT-REG: 00000000
SFP-OUTPUT-REG: 00000000
SFP-POLARITY-REG: 00000000
SFP-CONF-REG: 000000FA
...
█

COM14 - PuTTY
UART1, Hello World...
█
```

Figure 10: terminal output after startup

Any character entered in terminal 1 will be looped through the fiber of the SFP+. The output of terminal 0 shows the content of the receiver module. Every line starts with the comma character “BC” followed by the actual frame counter value. The end represents the hexadecimal value of the ASCII character (0x78 = ‘x’, 0x20 = ‘space’, etc.). Figure 11 shows the terminal output after sending some characters.

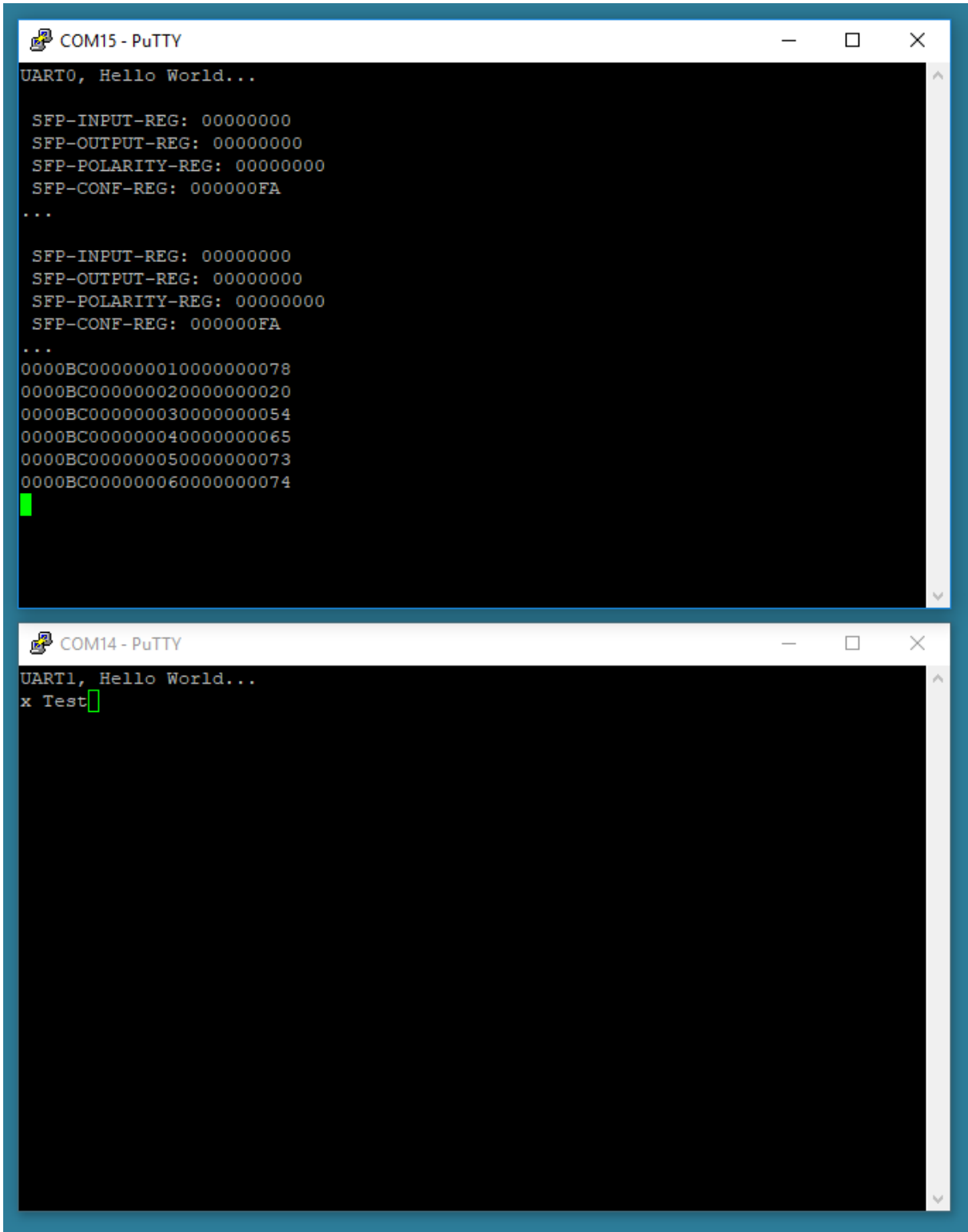


Figure 11: Terminal Output - sending some characters via SFP+ Loop