

Configuring the peripherals of a PolarFire SoC

Created by: David Rubio G.

Blog post: <https://soceame.wordpress.com/2025/03/11/configuring-the-peripherals-of-a-polarfire-soc/>

Blog: <https://soceame.wordpress.com/>

GitHub: <https://github.com/DRubioG>

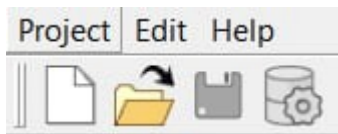
Last modification date: 11/03/25

To activate the internal peripherals of a PolarFire SoC, you need to open the **PFSoc MSS Configurator** program. This is the program that configures the SoC that then has to be imported into Libero as an MSS SmartDesign.



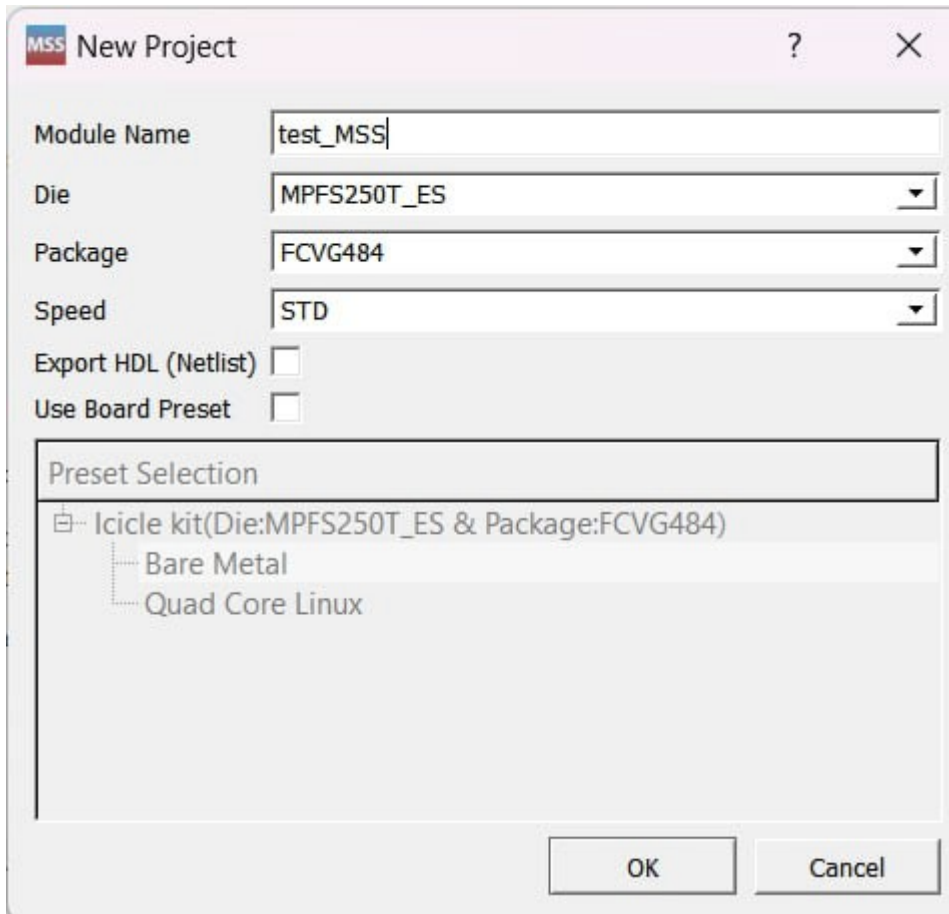
PFSoc MSS Configurator

When you open the program, you are given two options: create a new configuration or open an existing one.

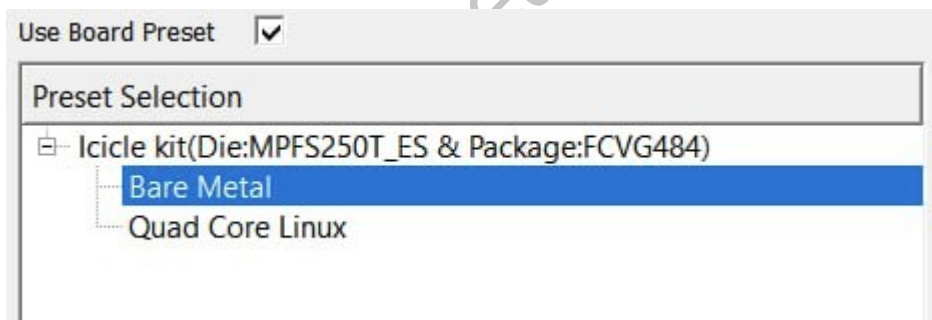


- **Create a new configuration**

The first thing it does is open a tab where it will ask you what type of PolarFire you are going to configure.



It also has an option that allows you to use a board that is already preconfigured in the program. At the moment, only the Icicle Kit is configured.



- **Open an existing one**

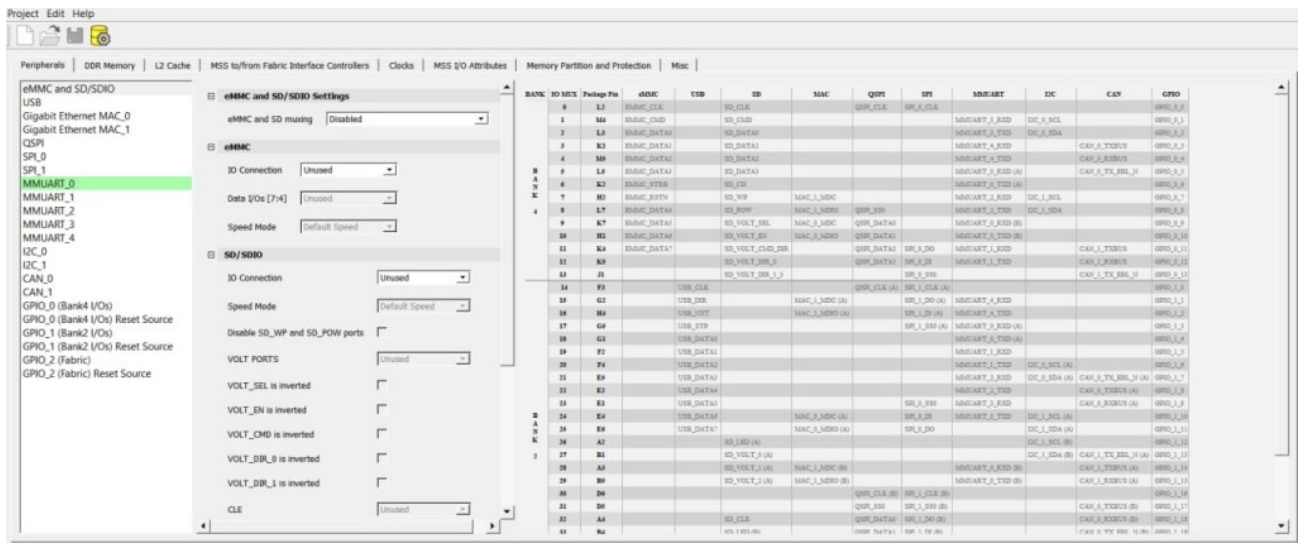
To open an existing one, it will ask you to provide the .cfg file with the configuration. From this configuration file, it will open the previously saved configuration.

Here you have a base project for the PolarFire SoC Discovery Kit:

https://github.com/DRubioG/Polarfire_basic_project/tree/main/PolarFire_config

NOTE: it is recommended to work on an existing configuration because one of the characteristics that it asks for to configure the SoC is the RAM memory that the SoC has.

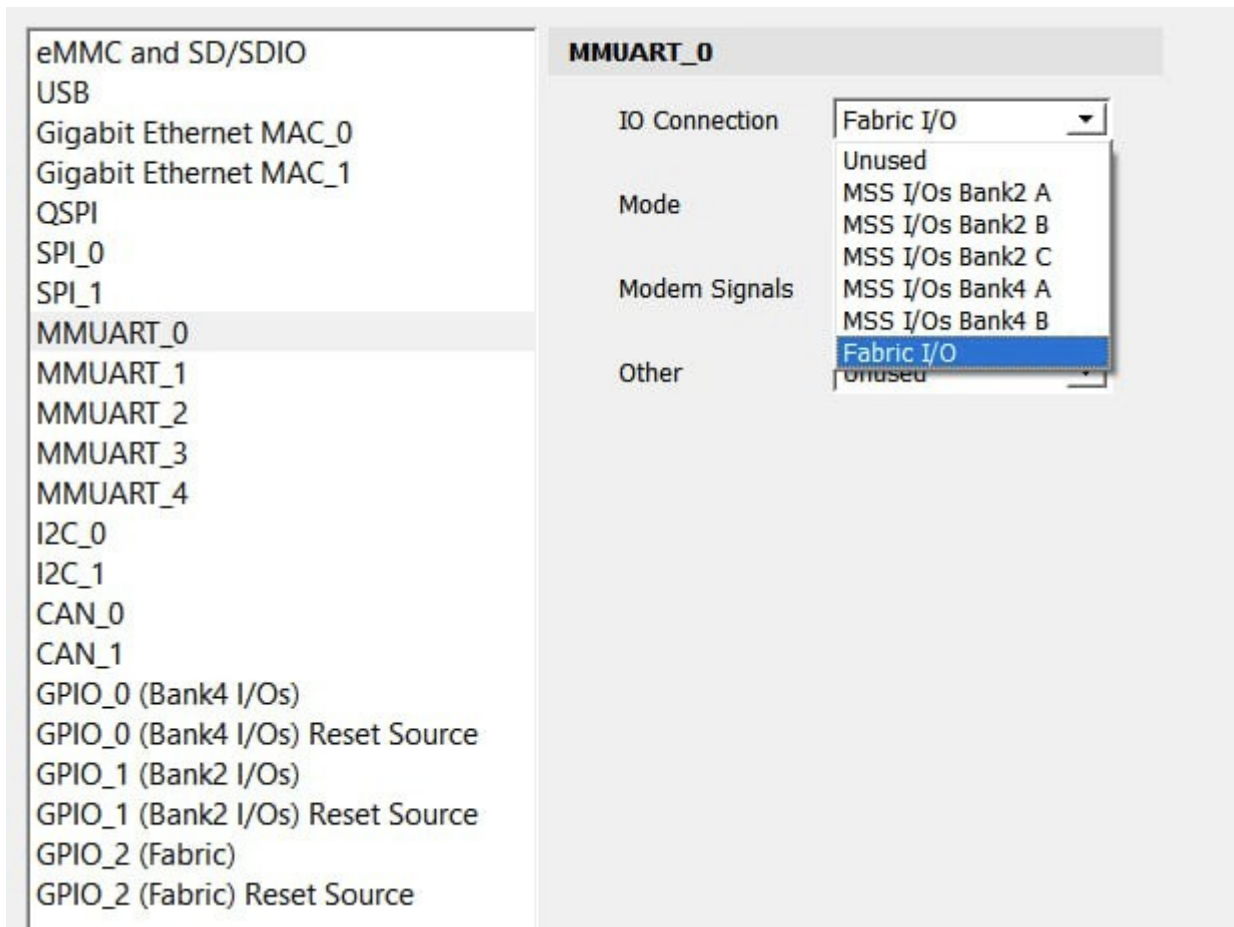
Once the program is opened you can see a tab like the following one, in this tab you can see all the possible configurations of the PolarFire SoC.



This program allows you to configure the interfaces for different peripherals (Peripherals). For example, it allows up to 5 UARTs, 2 I2Cs, 2 SPIs, GPIOs for the existing banks.



The specific pins of the SoC are the pins called «**MSS I/Os Bank ...**», these pins are multiplexed, so it is not possible to use all the interfaces of the SoC at the same time through these pins. To solve this, the so-called **Fabric I/O** pins appear, which are pins that do not belong to the SoC, but to the programmable logic of the FPGA. So, it is the user who chooses the output pins of the interface after synthesizing.

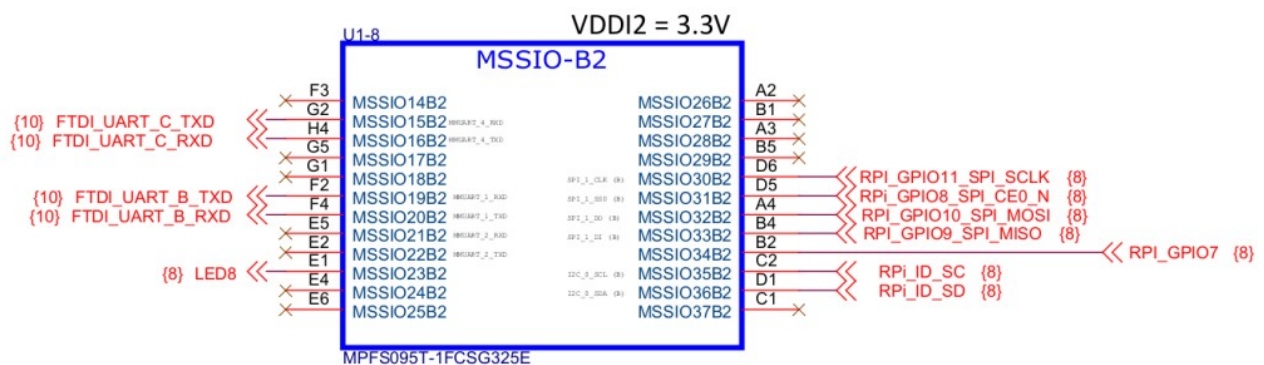


The multiplexed pin table is the following, where you can see that pins 17 and 18 are selected. These pins are in bank 2 of the SoC.

NOTE: in this pin table the specific name of the peripheral appears on the pins, so it becomes easier to know which pins to configure.

BANK	IO MUX	Package Pin	eMMC	USB	SD	MAC	QSPI	SPI	MMUART	I2C	CAN	GPIO
BANK 4	0	L2	EMMC_CLK		SD_CLK		QSPI_CLK	SPI_0_CLK				GPIO_0_0
	1	M4	EMMC_CMD		SD_CMD				MMUART_3_RXD	I2C_0_SCL		GPIO_0_1
	2	L3	EMMC_DATA0		SD_DATA0				MMUART_3_TXD	I2C_0_SDA		GPIO_0_2
	3	K1	EMMC_DATA1		SD_DATA1				MMUART_4_RXD		CAN_0_TXBUS	GPIO_0_3
	4	M5	EMMC_DATA2		SD_DATA2				MMUART_4_TXD		CAN_0_RXBUS	GPIO_0_4
	5	L5	EMMC_DATA3		SD_DATA3				MMUART_0_RXD (A)		CAN_0_TX_EBL_N	GPIO_0_5
	6	K2	EMMC_STRB		SD_CD				MMUART_0_TXD (A)			GPIO_0_6
	7	H2	EMMC_RSTN		SD_WP	MAC_1_MDC			MMUART_2_RXD	I2C_1_SCL		GPIO_0_7
	8	L7	EMMC_DATA4		SD_POW	MAC_1_MDIO	QSPI_SS0		MMUART_2_TXD	I2C_1_SDA		GPIO_0_8
	9	K7	EMMC_DATA5		SD_VOLT_SEL	MAC_0_MDC	QSPI_DATA0		MMUART_0_RXD (B)			GPIO_0_9
	10	H1	EMMC_DATA6		SD_VOLT_EN	MAC_0_MDIO	QSPI_DATA1		MMUART_0_TXD (B)			GPIO_0_10
	11	K4	EMMC_DATA7		SD_VOLT_CMD_DIR		QSPI_DATA2	SPI_0_DO	MMUART_1_RXD		CAN_1_TXBUS	GPIO_0_11
	12	K5			SD_VOLT_DIR_0		QSPI_DATA3	SPI_0_DI	MMUART_1_TXD		CAN_1_RXBUS	GPIO_0_12
BANK 2	13	J1			SD_VOLT_DIR_1_3			SPI_0_SS0			CAN_1_TX_EBL_N	GPIO_0_13
	14	F3		USB_CLK			QSPI_CLK (A)	SPI_1_CLK (A)				GPIO_1_0
	15	G2		USB_DIR		MAC_1_MDC (A)		SPI_1_DO (A)	MMUART_4_RXD			GPIO_1_1
	16	H4		USB_NXT		MAC_1_MDIO (A)		SPI_1_DI (A)	MMUART_4_TXD			GPIO_1_2
	17	G5		USB_STP				SPI_1_SS0 (A)	MMUART_0_RXD (A)			GPIO_1_3
	18	G1		USB_DATA0					MMUART_0_TXD (A)			GPIO_1_4
	19	F2		USB_DATA1					MMUART_1_RXD			GPIO_1_5
	20	F4		USB_DATA2					MMUART_1_TXD	I2C_0_SCL (A)		GPIO_1_6
	21	E5		USB_DATA3					MMUART_2_RXD	I2C_0_SDA (A)	CAN_0_TX_EBL_N (A)	GPIO_1_7
	22	E2		USB_DATA4					MMUART_2_TXD		CAN_0_TXBUS (A)	GPIO_1_8
	23	E1		USB_DATA5				SPI_0_SS0	MMUART_3_RXD		CAN_0_RXBUS (A)	GPIO_1_9
	24	E4		USB_DATA6		MAC_0_MDC (A)		SPI_0_DI	MMUART_3_TXD	I2C_1_SCL (A)		GPIO_1_10
	25	E6		USB_DATA7		MAC_0_MDIO (A)		SPI_0_DO		I2C_1_SDA (A)		GPIO_1_11
	26	A2			SD_LED (A)					I2C_1_SCL (B)		GPIO_1_12
	27	B1			SD_VOLT_0 (A)					I2C_1_SDA (B)	CAN_1_TX_EBL_N (A)	GPIO_1_13
	28	A3			SD_VOLT_1 (A)	MAC_1_MDC (B)			MMUART_0_RXD (B)		CAN_1_TXBUS (A)	GPIO_1_14
	29	B5			SD_VOLT_2 (A)	MAC_1_MDIO (B)			MMUART_0_TXD (B)		CAN_1_RXBUS (A)	GPIO_1_15
	30	D6					QSPI_CLK (B)	SPI_1_CLK (B)				GPIO_1_16
	31	D5					QSPI_SS0	SPI_1_SS0 (B)			CAN_0_TXBUS (B)	GPIO_1_17
	32	A4			SD_CLE		QSPI_DATA0	SPI_1_DO (B)			CAN_0_RXBUS (B)	GPIO_1_18
	33	B4			SD_LED (B)		QSPI_DATA1	SPI_1_DI (B)			CAN_0_TX_EBL_N (B)	GPIO_1_19
	34	B2			SD_VOLT_0 (B)		QSPI_DATA2				CAN_1_TXBUS (B)	GPIO_1_20
	35	C2			SD_VOLT_1 (B)	MAC_0_MDC (B)	QSPI_DATA3		MMUART_0_RXD (C)	I2C_0_SCL (B)	CAN_1_RXBUS (B)	GPIO_1_21
	36	D1			SD_VOLT_2 (B)	MAC_0_MDIO (B)			MMUART_0_TXD (C)	I2C_0_SDA (B)	CAN_1_TX_EBL_N (B)	GPIO_1_22
	37	C1					QSPI_CLK (C)	SPI_0_CLK				GPIO_1_23

These pins correspond to the connections established in the schematic of the board for bank 2 of the SoC.



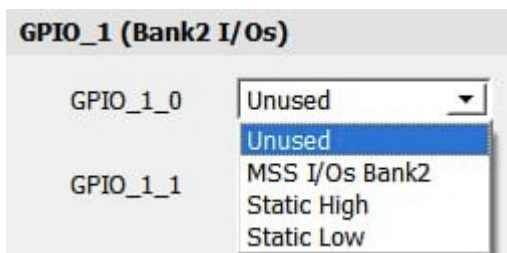
As you can see, the I2C also has the option to select the type of pin through which the connection can be established.



I2C_0

IO Connection	Unused
Speed Mode	Unused
Baud Rate Clock Source	MSS I/Os Bank2 A
	MSS I/Os Bank2 B
	MSS I/Os Bank4
	Fabric I/O
System Management Bus Signals	Unused

The same with the GPIOs.



GPIO_1 (Bank2 I/Os)

GPIO_1_0	Unused
GPIO_1_1	Unused
	MSS I/Os Bank2
	Static High
	Static Low

Now if you look at the GPIOs, two types of GPIOs appear, those that go through the SoC pins and those that can only go through the programmable logic (*Fabric*).

The screenshot shows a configuration window for a PolarFire SoC. On the left is a sidebar with a list of peripherals. The 'GPIO_1 (Bank2 I/Os)' option is selected and highlighted in blue. The main area of the window is titled 'GPIO_1 (Bank2 I/Os)' and contains a table of 12 pins. Each pin has a dropdown menu set to 'Unused' and a 'Direction' dropdown set to 'Input'.

Pin	Function	Direction
GPIO_1_0	Unused	Input
GPIO_1_1	Unused	Input
GPIO_1_2	Unused	Input
GPIO_1_3	Unused	Input
GPIO_1_4	Unused	Input
GPIO_1_5	Unused	Input
GPIO_1_6	Unused	Input
GPIO_1_7	Unused	Input
GPIO_1_8	Unused	Input
GPIO_1_9	Unused	Input
GPIO_1_10	Unused	Input
GPIO_1_11	Unused	Input

The next tab allows us to configure the RAM memory.

Peripherals | **DDR Memory** | L2 Cache | MSS to/from Fabric Interface Controllers | Clocks | MSS I/O Attributes | Memory Partition and Protection | Misc

DDR Memory Type: **DDR4** | Lock Down DDR I/Os: ☐

JEDEC
MSS_DDR4_default_configuration

Apply | New preset...

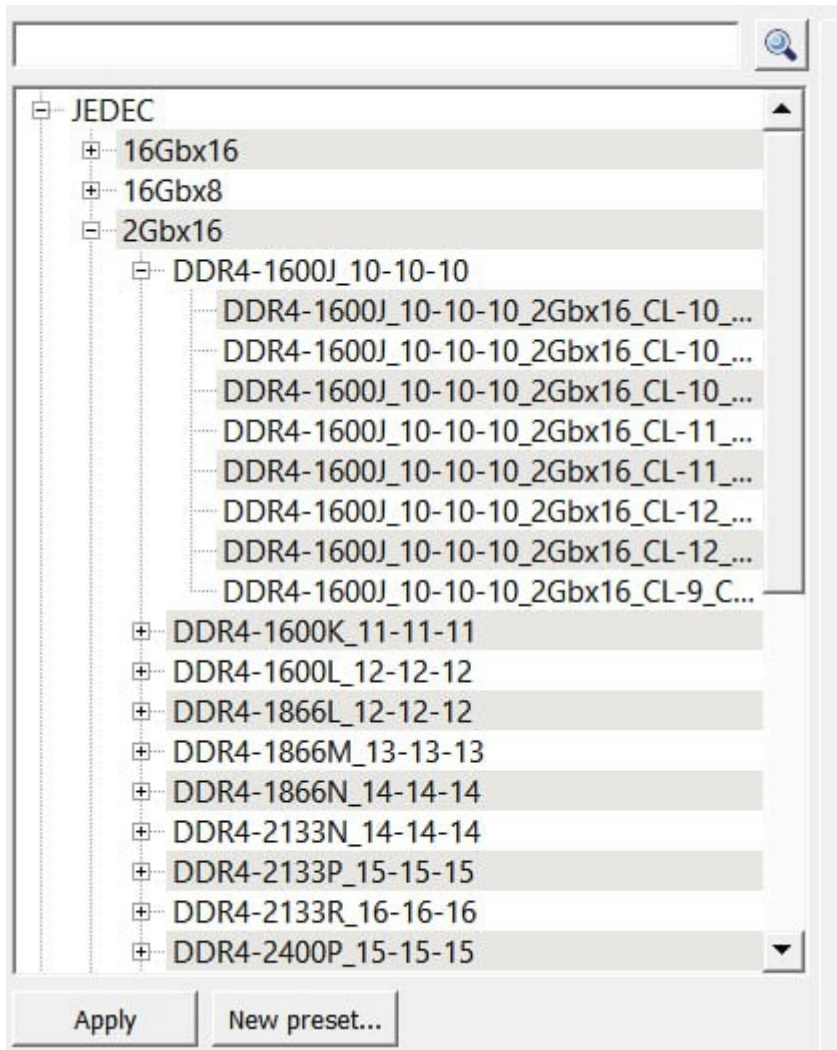
DDR Topology

DQ Width: 16
SDRAM Number of Ranks: 1
Enable DM: DM
Enable Parity/Alert: ☐
Enable ECC: ☐
Bank Group Address Width: 1
Number of Clock Outputs: 1
Memory Format: COMPONENT
Enable address mirroring on odd ranks: ☐
Row Address Width: 16
Column Address Width: 10
Bank Address Width: 2

Clock

Memory Clock Frequency (MHz): 800

It has a lot of RAM memories inside, so unless you know specifically what memory you have in the SoC it is better to have a preconfigured profile.



The next tab allows you to configure the L2 Cache.

L2-LIM WAYS (128 KB for each WAY)

Scratchpad WAYS (128 KB for each WAY)

Initiators	L2 Cache Size	WAY0	WAY1	WAY2	WAY3	WAY4	WAY5	WAY6	WAY7	WAY8	WAY9	WAY10	WAY11	WAY12	WAY13
DMA	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓	Scratchpad 512 KB					
FPGA Port0	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
FPGA Port1	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
FPGA Port2	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
FPGA Port3	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
E51 D\$	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
E51 I\$	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
U54_1 D\$	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
U54_1 I\$	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
U54_2 D\$	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
U54_2 I\$	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
U54_3 D\$	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
U54_3 I\$	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
U54_4 D\$	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						
U54_4 I\$	1024 KB	✓	✓	✓	✓	✓	✓	✓	✓						

Note1: 'L2 Cache Size' in each row is the size accessible by that particular Initiator out of 'Total Shared L2 Cache Size' (1024 KB)

Note2: Initiator cannot evict from cache when WAY is unselected

The next tab is important, because it is the one that configures the communication interface with the different IP blocks that are used in the SoC. You can see that there are up to 4 profiles, 3 for AXI and one for APB.

Peripherals	DDR Memory	L2 Cache	MSS to/from Fabric Interface Controllers	Clocks	MSS I/O
FIC_0 (AXI4)					
Use Initiator Interface	<input type="checkbox"/>	Use Target Interface	<input type="checkbox"/>		
Use Embedded DLL	<input type="checkbox"/>	Embedded DLL Jitter Range	<div>Low</div>		
FIC_1 (AXI4)					
Use Initiator Interface	<input type="checkbox"/>	Use Target Interface	<input type="checkbox"/>		
Use Embedded DLL	<input type="checkbox"/>	Embedded DLL Jitter Range	<div>Low</div>		
FIC_2 (AXI4)					
Use Target Interface	<input type="checkbox"/>				
Use Embedded DLL	<input type="checkbox"/>	Embedded DLL Jitter Range	<div>Low</div>		
FIC_3 (APB)					
Use Initiator Interface	<input checked="" type="checkbox"/>				
Use Embedded DLL	<input type="checkbox"/>	Embedded DLL Jitter Range	<div>Low</div>		

The next one is for the configuration of the output clocks of the SoC, not the input that makes the SoC work.

Peripherals

DDR Memory

L2 Cache

MSS to/from Fabric Interface Controllers

Clocks

MSS I/O Attributes

Memory Partition and Protection

Misc

MSS PLL and dividers

MSS PLL reference clock source

Dedicated I/O from Bank5 (REFCLK)

MSS PLL required clock frequency (MHz)

600

Actual PLL clock

MSS CPU cores clock frequency divider

/1

Actual CPU cores

MSS AXI clock frequency divider

/2

Actual AXI cores

MSS AHB/APB clock frequency divider

/4

Actual AHB/APB

DDR

DDR reference clock source

Dedicated I/O from Bank5 (REFCLK)

Real Time Clock (RTC) / Gigabit Ethernet MAC

RTC / MAC SGMI reference clock source

Dedicated I/O from Bank5 (REFCLK)

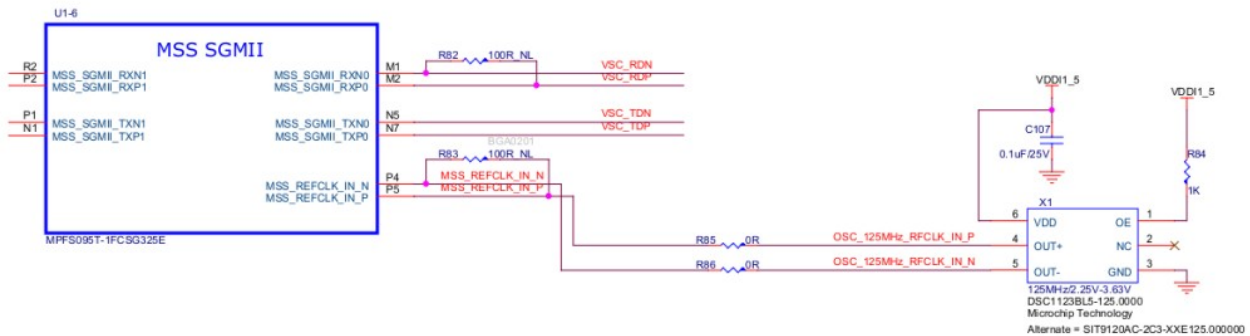
Clock Sources Frequency

Dedicated I/O from Bank5 (REFCLK) frequency (MHz)

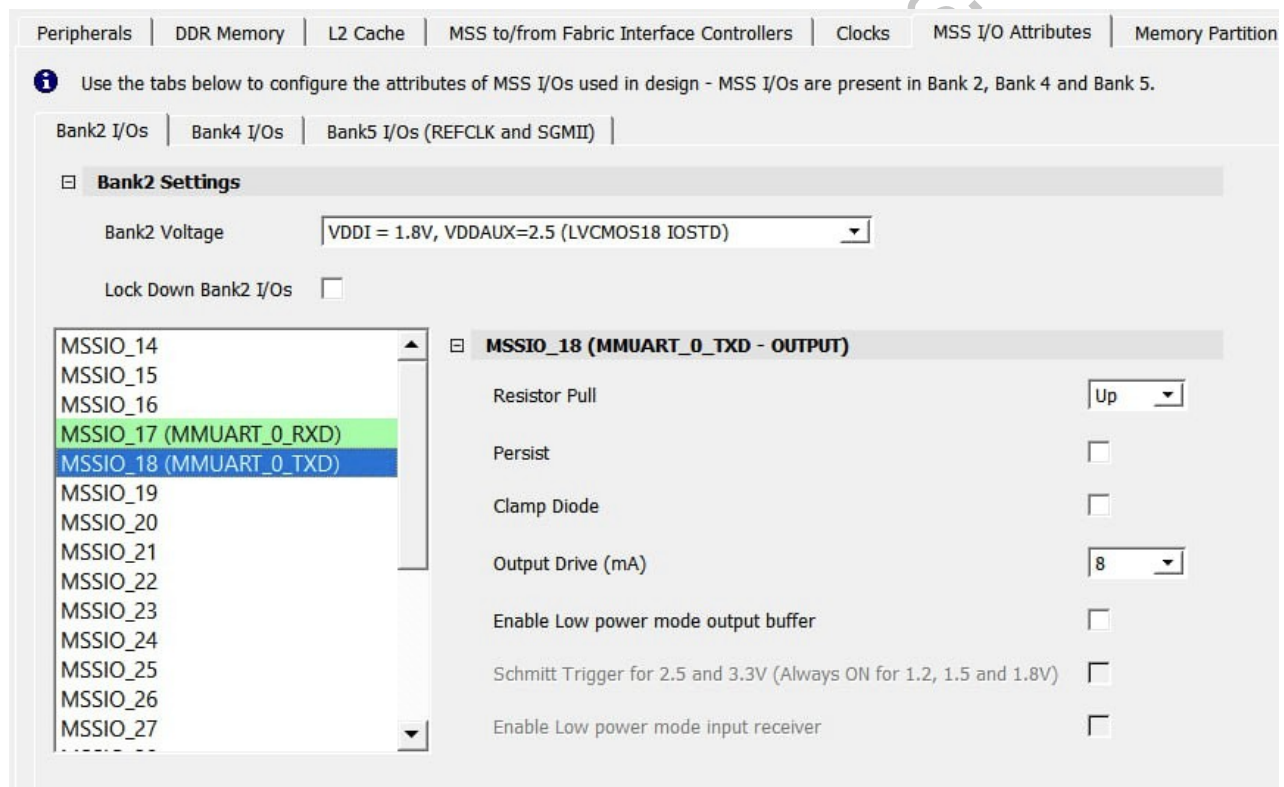
125

The diagram illustrates the MSS Clock Controller architecture. A 125 MHz Dedicated I/O clock is input to the MSS PLL (600,000 MHz). The MSS PLL outputs a 600,000 MHz clock to the CPU, AXI & L2, and AHB/APB. The MSS PLL also outputs a 600,000 MHz clock to the eMMC/SD/SIO, CAN, DORC + Phy, and SGMI Phy. The MSS PLL is also connected to the DORC PLL and the SGMI PLL.

The SoC has its own clock by default, in the image you can see that it has a clock connected to the **MSS_REFCLK_IN_P/N** pins of 125MHz.



The next one is to configure the properties of the MSS pins used, such as if they are in Pull-Up, Pull-Down, etc.



The next step is to configure the different memory zones, this is a bit complex to handle, because the SoC has at least 5 internal cores: 1 RV64IMAC and 4 RV64GC.

Peripherals | DDR Memory | L2 Cache | MSS to/from Fabric Interface Controllers | Clocks | MSS I/O Attributes | Memory Partition and Protection | Misc

☐ Use Processor PMP and AXI Switch MPU Configurations

DDR Memory Partition | Processor PMP | AXI Switch MPU

☐ Setup Physical DDR Offset Address manually

	MSS Offset Address:	Range	MSS High Address:	hysical DDR Offset Address	hysical DDR High Address
Cached 1GB	0x8000_0000	128 MB	0x87FF_FFFF	0x0000_0000	0x7FF_FFFF
Cached 16GB	0x10_0000_0000	896 MB	0x10_37FF_FFFF	0x800_0000	0x3FFF_FFFF
Non-Cached 256MB	0xC000_0000	128 MB	0xC7FF_FFFF	0x4000_0000	0x47FF_FFFF
Non-Cached 16GB	0x14_0000_0000	896 MB	0x14_37FF_FFFF	0x4800_0000	0x7FFF_FFFF

Note1: Range selection for any Cached or Non-Cached memory region must be a multiple of 16 MB.
Note2: Memory is not allocated in DDR when Range is set to 0.

The last tab is used to configure some final options of the SoC, such as if it has an external pin for interrupts.

Peripherals | DDR Memory | L2 Cache | MSS to/from Fabric Interface Controllers | Clocks | MSS I/O Attributes | Memory Partition and Protection | Misc

☐ **Debug Trace**

Expose MSS UltraSoC Trace ports to Fabric ☐

Expose JTAG Trace/Debug ports to Fabric ☐

Expose JTAG Trace/Debug Control via Fabric ☐

☐ **Interrupt**

Expose Interrupt ports to Fabric ☐

GPIO Interrupt register setting

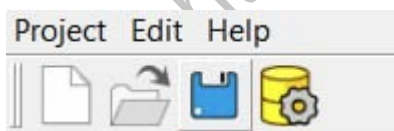
☐ **Boot Status**

Expose Boot Status ports ☐

☐ **MSS Feedback and Debug Ports**

Expose Feedback ports to Fabric ☐

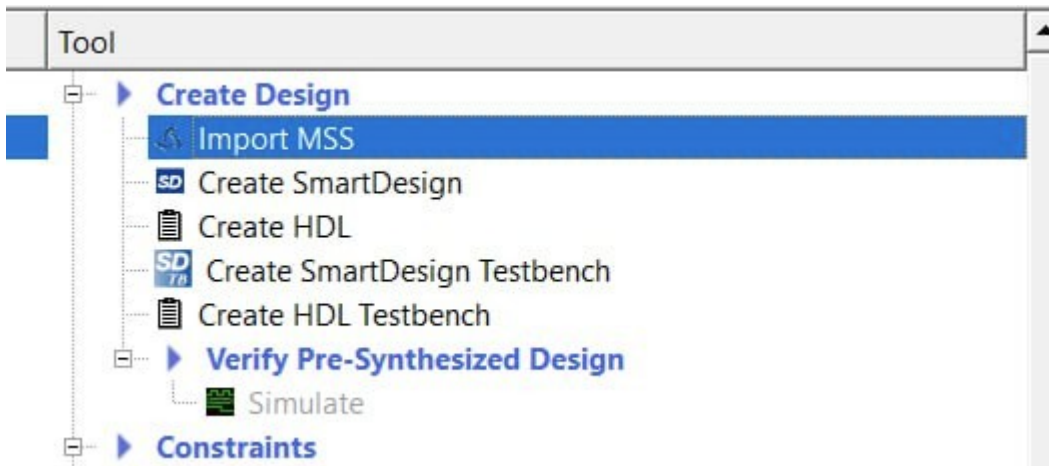
Now what you have to do is generate the component and save the configuration.



You have to create a .cxz file to use it in Libero.

Libero

The next step is in Libero. We open Libero and go to the Import MSS option.



This option asks us for a .cxz file like the one generated by the *PFSoc MSS Configurator*.

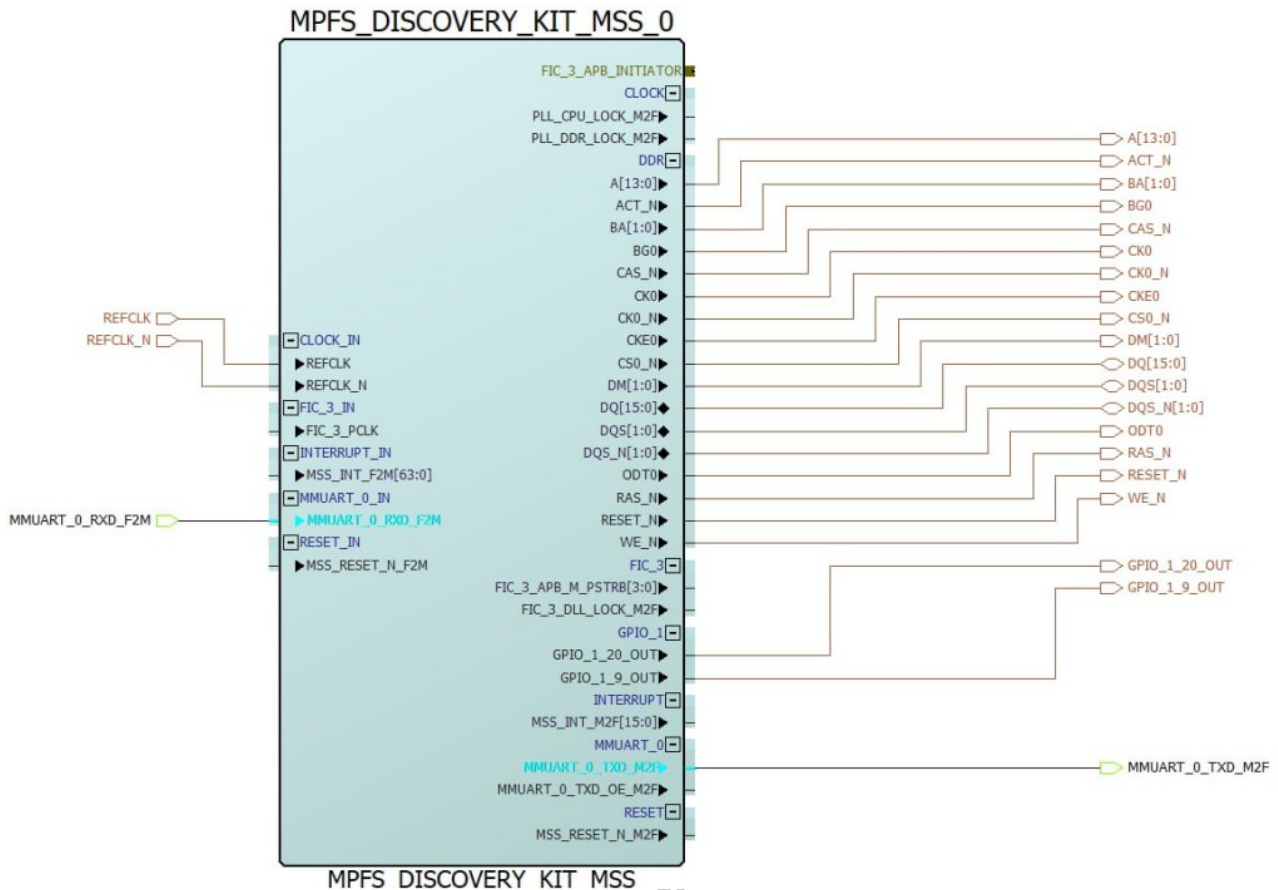
Nombre	Fecha de modificación	Tipo	Tamaño
MPFS_DISCOVERY_KIT_MSS.cxz	06/01/2025 22:43	Archivo CXZ	81 KB

When we import it, it appears in the hierarchy.



Now we just have to create a SmartDesign and add it.

When we add it, we can see that there are ports that are already selected. A differential clock appears, the APB interface selected to couple IP blocks, the RAM memory pins also appear, two GPIOs selected to be output through the MSS pins, and a UART with external pins also appears (this UART has 3 pins, the TXD_OE pin does not need to be connected).



The hierarchy is as follows.



Now the only thing left to do is to generate the design with the desired IP blocks, or directly generate the bitstream, and continue with the SoftConsole.

NOTE: the UART that comes out through *FABRIC*, Libero asks us to assign the pins we want.

GPIO_1_20_OUT	OUTPUT	LVC MOS33	B2	<input checked="" type="checkbox"/>	IOPAD_TRI
MMUART_0_RXD_F2M	INPUT	LVC MOS18	Y21	<input checked="" type="checkbox"/>	INBUF
MMUART_0_TXD_M2F	OUTPUT	LVC MOS18	W21	<input checked="" type="checkbox"/>	OUTBUF
ODT0	OUTPUT	HSTL12I	T2	<input checked="" type="checkbox"/>	IOPAD_TRI

Drivers

Peripheral drivers can be obtained through SoftConsole 2022's internal build system (this option is quite difficult to obtain) or from this GitHub repository (there may be drivers that don't quite work):

<https://github.com/polarfire-soc/polarfire-soc-bare-metal-library>

<https://soceame.wordpress.com/2025/03/11/configuring-the-peripherals-of-a-polarfire-soc/>

NOTE: Some working drivers can also be found in the examples provided by SoftConsole upon installation, for both SoftConsole 2021 and 2022.

<https://soceame.wordpress.com/>