How to connect an IP block in Libero

Created by: David Rubio G.

Blog post: https://soceame.wordpress.com/2025/03/11/how-to-connect-an-ip-block-in-libero/

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

GitHub: https://github.com/DRubioG

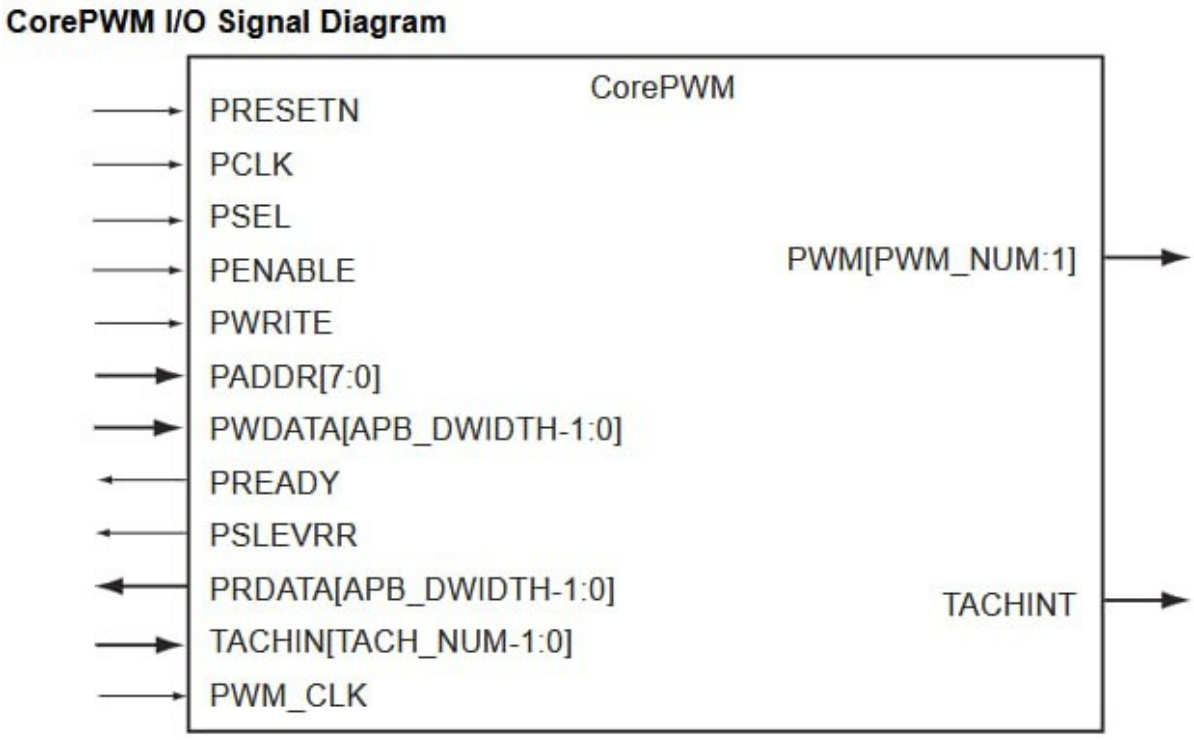
Last modification date: 11/03/25

In this post I will explain how to connect an IP block in Libero (within the limits allowed by Microchip tools).

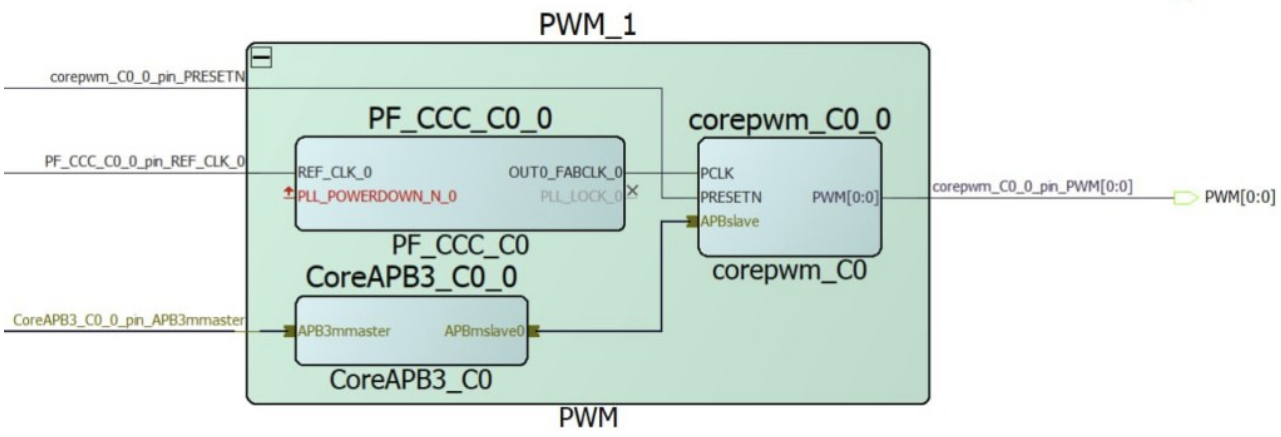
To connect an IP block the first thing you have to know is the type of interface to use. In a previous post I already explained how to create an interface.

https://soceame.wordpress.com/2025/03/11/how-to-create-a-bif-interface-in-libero/

In our case we will use an IP block called CorePWM included in Libero.

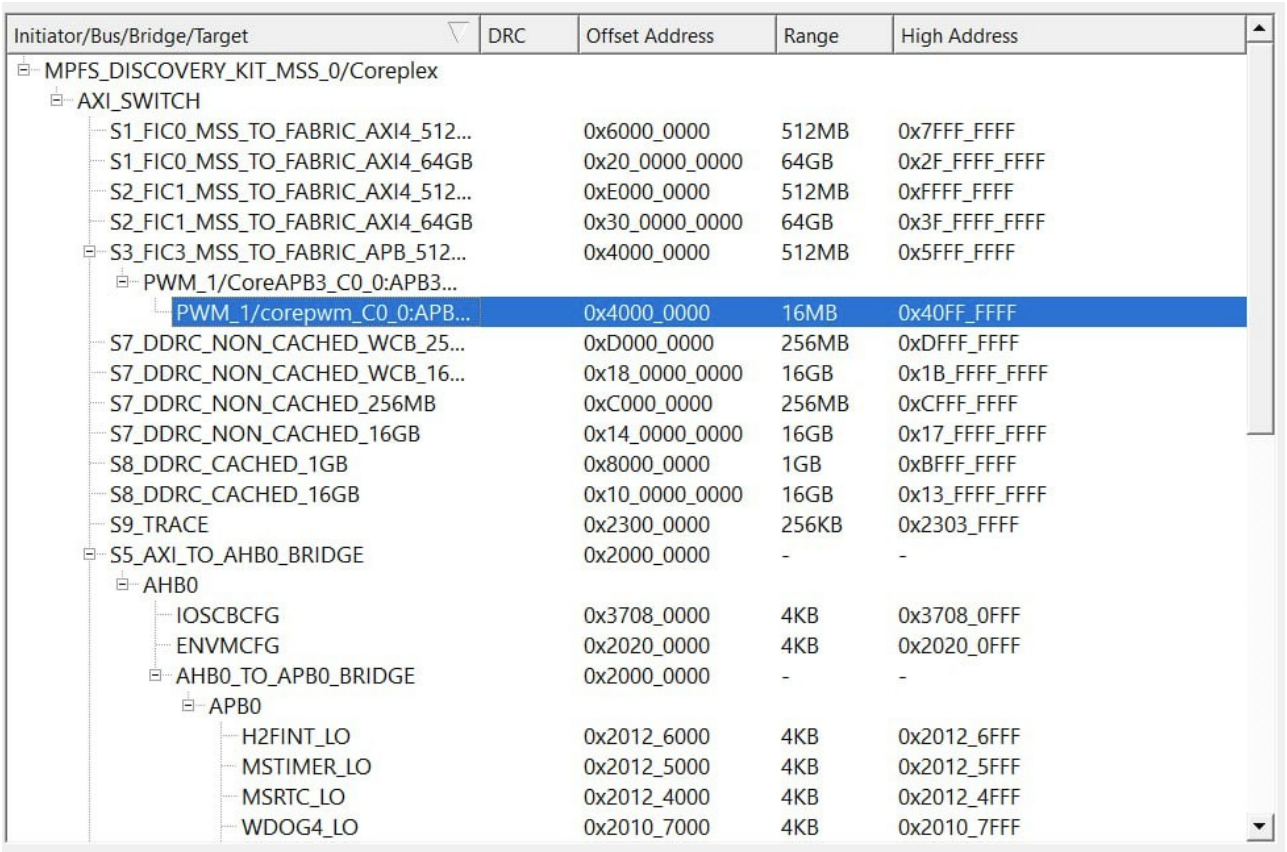


This block has an APB type interface so to be able to connect it to a SmartFusion2 or a PolarFire SoC a **CoreAPB3** type interface is needed. This interface allows communication through memory addresses with the IP block.



So, if we look at the memory address of the IP block, we see that it has the memory address 0x40000000.

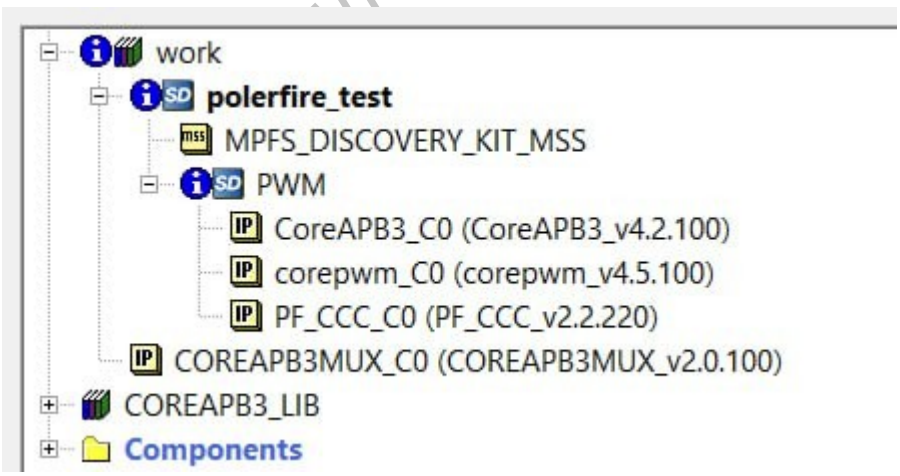
https://soceame.wordpress.com/2025/03/11/how-to-access-ip-block-memory-addresses-in-libero/



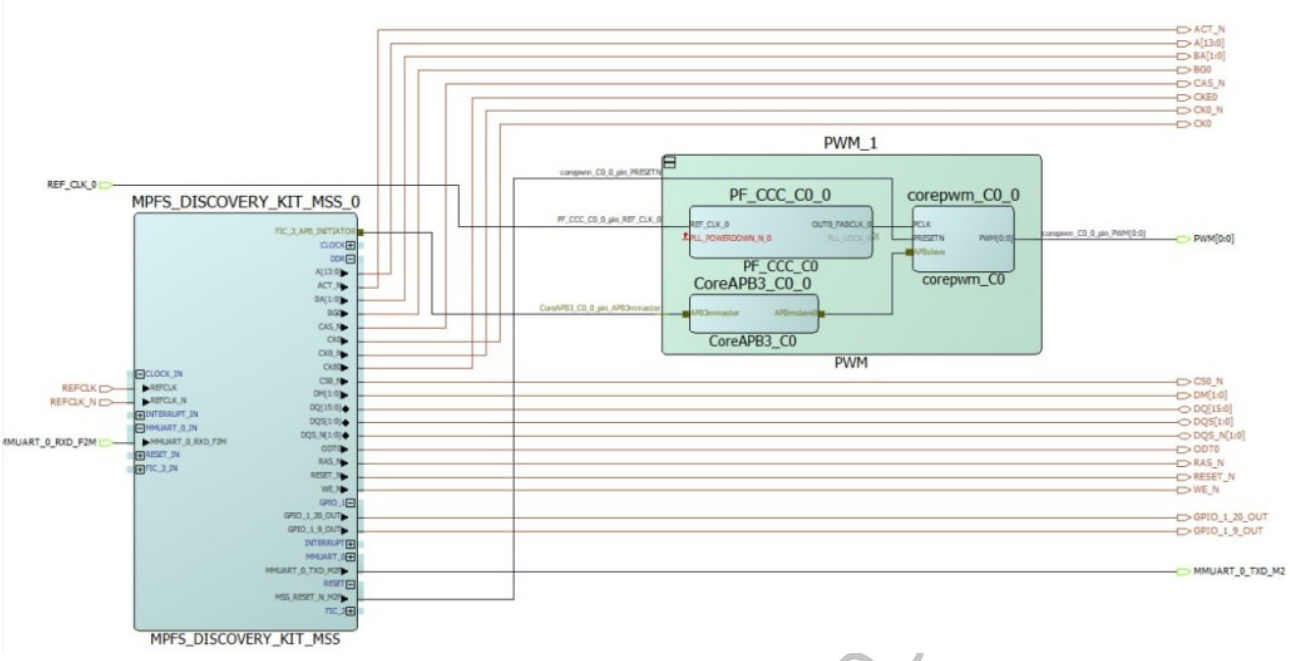
Initiator/Bus/Bridge/Target	DRC	Offset Address	Range	High Address
MPFS_DISCOVERY_KIT_MSS_0/Coreplex				
AXI_SWITCH				
S1_FIC0_MSS_TO_FABRIC_AXI4_512...		0x6000_0000	512MB	0x7FFF_FFFF
S1_FIC0_MSS_TO_FABRIC_AXI4_64GB		0x20_0000_0000	64GB	0x2F_FFFF_FFFF
S2_FIC1_MSS_TO_FABRIC_AXI4_512...		0xE000_0000	512MB	0xFFFF_FFFF
S2_FIC1_MSS_TO_FABRIC_AXI4_64GB		0x30_0000_0000	64GB	0x3F_FFFF_FFFF
S3_FIC3_MSS_TO_FABRIC_APB_512...		0x4000_0000	512MB	0x5FFF_FFFF
PWM_1/CoreAPB3_C0_0:APB3...				
PWM_1/corepwm_C0_0:APB...		0x4000_0000	16MB	0x40FF_FFFF
S7_DDRC_NON_CACHED_WCB_25...		0xD000_0000	256MB	0xDFFF_FFFF
S7_DDRC_NON_CACHED_WCB_16...		0x18_0000_0000	16GB	0x1B_FFFF_FFFF
S7_DDRC_NON_CACHED_256MB		0xC000_0000	256MB	0xCFFF_FFFF
S7_DDRC_NON_CACHED_16GB		0x14_0000_0000	16GB	0x17_FFFF_FFFF
S8_DDRC_CACHED_1GB		0x8000_0000	1GB	0xBFFF_FFFF
S8_DDRC_CACHED_16GB		0x10_0000_0000	16GB	0x13_FFFF_FFFF
S9_TRACE		0x2300_0000	256KB	0x2303_FFFF
S5_AXI_TO_AHB0_BRIDGE		0x2000_0000	-	-
AHB0				
IOSCBCFG		0x3708_0000	4KB	0x3708_0FFF
ENVMCFG		0x2020_0000	4KB	0x2020_0FFF
AHB0_TO_APB0_BRIDGE		0x2000_0000	-	-
APB0				
H2FINT_LO		0x2012_6000	4KB	0x2012_6FFF
MSTIMER_LO		0x2012_5000	4KB	0x2012_5FFF
MSRTC_LO		0x2012_4000	4KB	0x2012_4FFF
WDOG4_LO		0x2010_7000	4KB	0x2010_7FFF

The IP block also requires a clock to run, which can be the one from the MSS or another one. And a reset, which can be the one generated by the MSS.

The structure of the project would be something like this.



So now all we have to do is connect the CoreAPB to the MSS APB FIC interface.

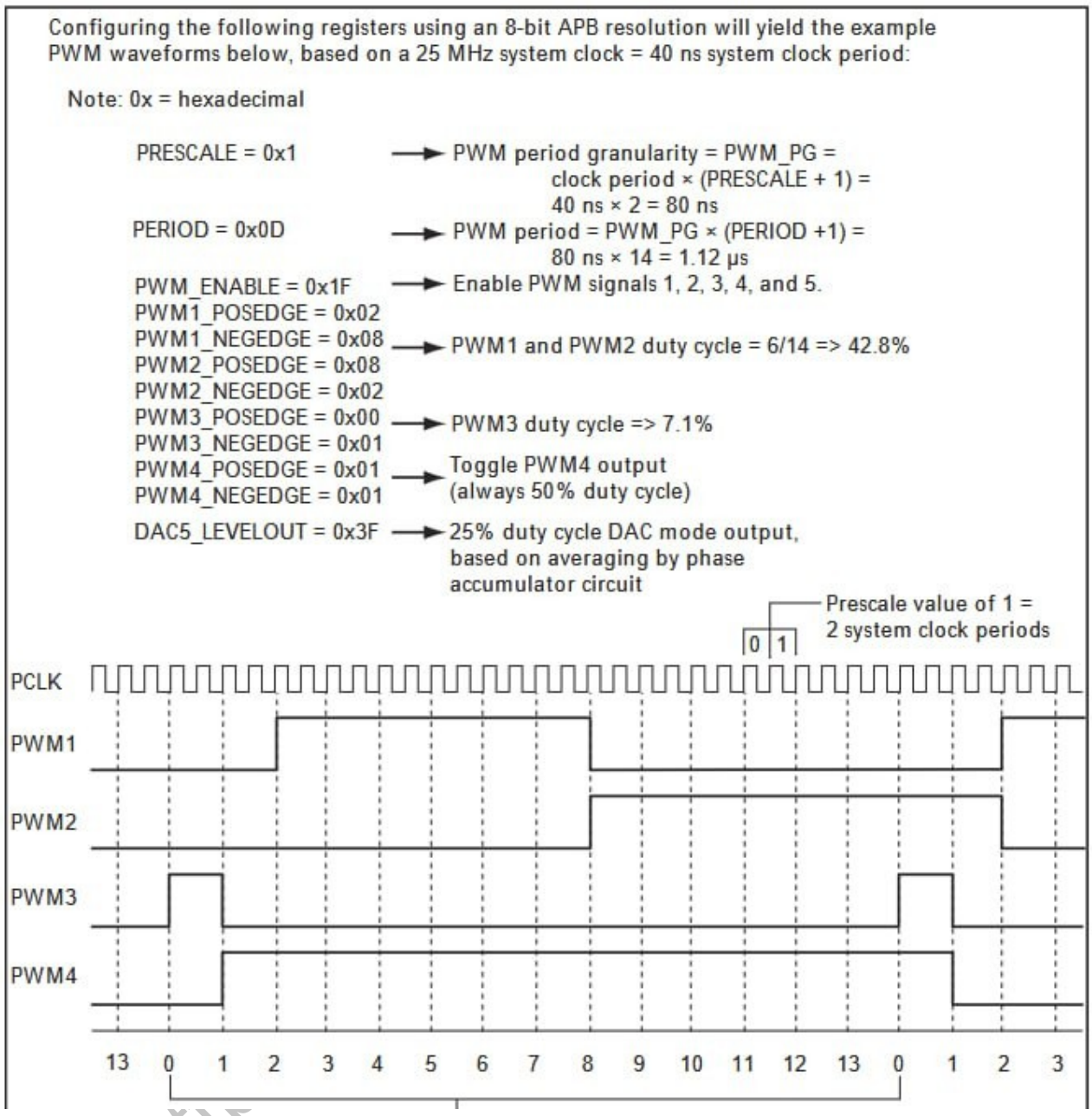


Now with the SoC memory address, all we need to do is go to the SoftConsole to implement the IP block configuration.

To do this, we use the internal memory table provided by Microchip.

Register Name	Paddr[7:0]	Description	Type	Default
PRESCALE	0x00	PWM MODE: The system clock cycle is multiplied with the PRESCALE value resulting in the minimum PERIOD count timebase. DAC MODE: The Prescale and Period Registers could be used in conjunction with the shadow register to synchronize DAC LEVELOUT.	R/W	0X08
PERIOD	0x04	PWM MODE: The PRESCALE value is multiplied with the PERIOD value yielding the PWM waveform cycle.	R/W	0x08
PWM_ENABLE_0_7	0x08	Bitwise channel enables for PWM/DAC channels 1 through 8.	R/W	0x00
PWM_ENABLE_8_15	0x0C	Bitwise channel enables for PWM/DAC channels 9 through 16.	R/W	0x00
SYNC_UPDATE	0xE4	SYNC_UPDATE: When this bit is set to “1” and SHADOW_REG_EN is selected, all POSEDGE and NEGEDGE registers are updated synchronously. Synchronous updates to the PWM waveform occur only when SHADOW_REG_EN is asserted and SYNC_UPDATE is set to "1". When this bit is set to "0", all the POSEDGE and NEGEDGE registers are updated asynchronously.	R/W	0x00
PWM1_POSEDGE	0x10	PWM MODE: Sets the positive edge of the output with respect to the PERIOD resolution. When APB writes to this register, all the channels are updated.	R/W	0x00
PWM1_NEGEDGE DAC1_LEVELOUT	0x14	PWM MODE: Sets the negative edge of the output with respect to the PERIOD resolution. DAC MODE: Sets the desired output level, from 0-100%.	R/W	0x00

It also provides a mini example that will be used for the SoC.



With this example, we develop the application that makes the PWM work.

```
uint8_t *value = (uint8_t *)0x40000000;
*(value+0x00) = 0x1;
*(value+0x04) = 0x0D;
*(value+0x10) = 0x02;
*(value+0x14) = 0x08;
*(value+0x07) = 0x1F;
```

NOTE: Because Libero SoCs are poorly documented and the examples they provide are too complex, the previous project does not work because the processor crashes. Even so, the workflow is well defined to configure Microchip SoCs.