Cómo conectar un bloque IP en Libero

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2025/01/07/como-conectar-un-bloque-ip-en-libero/

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

GitHub: https://github.com/DRubioG

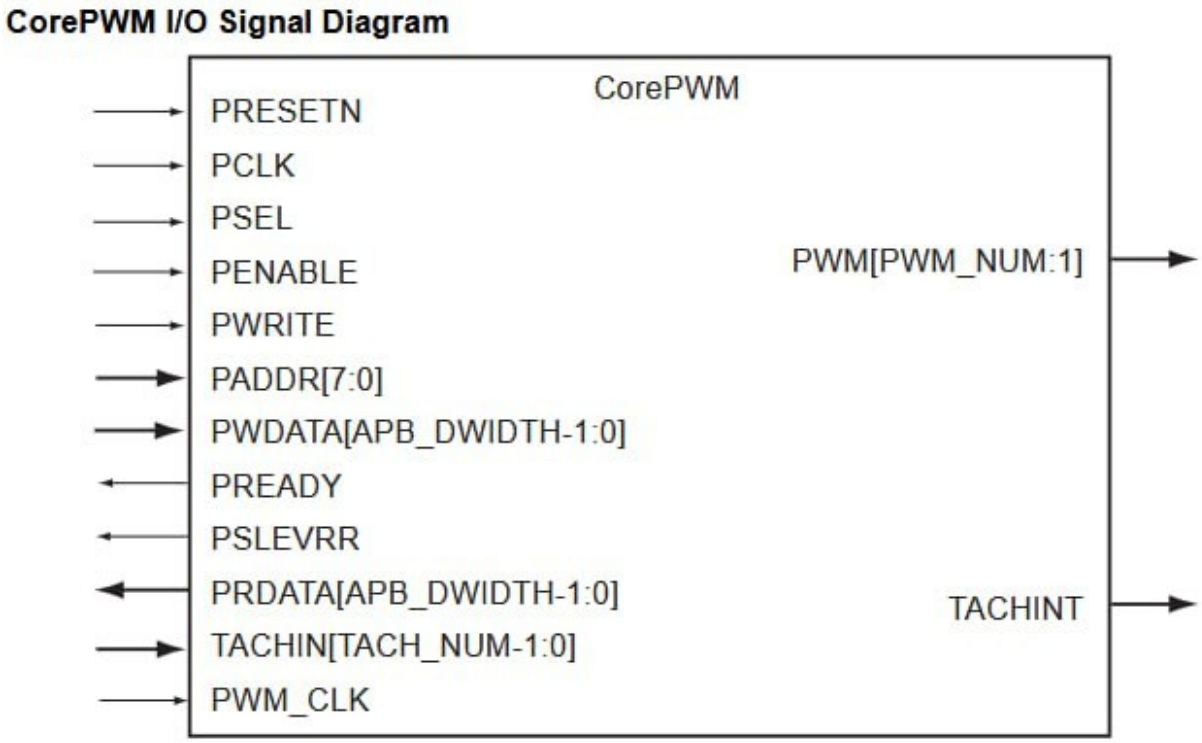
Fecha última modificación: 24/02/2025

En esta entrada voy a explicar cómo se conecta un bloque IP en Libero (dentro de los límites que permiten las herramientas de Microchip).

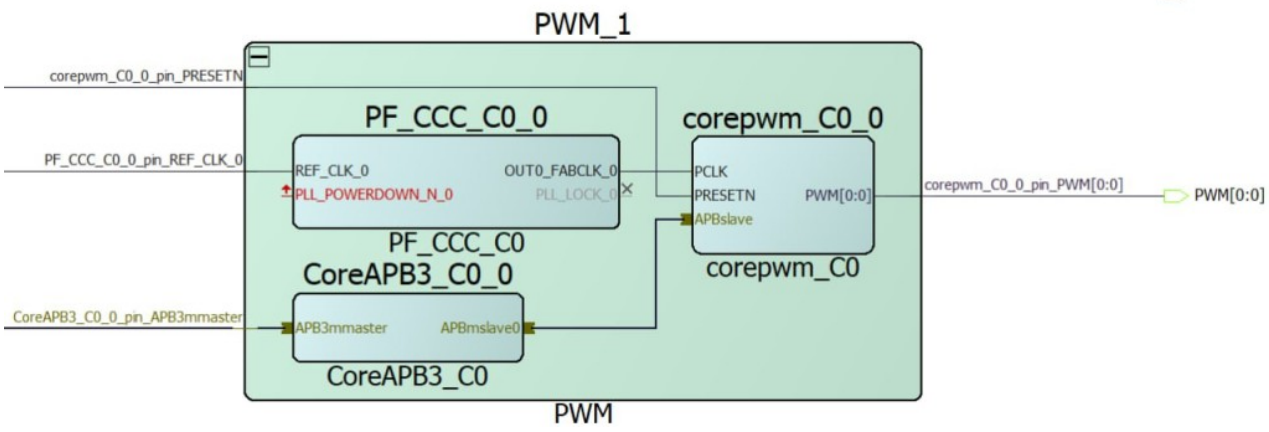
Para conectar un bloque IP lo primero que se tiene que conocer es el tipo de interfaz que utilizar. En una entrada anterior ya explico cómo se puede crear una interfaz.

https://soceame.wordpress.com/2024/12/31/como-crear-una-interfaz-bif-en-libero/

En nuestro caso utilizaremos un bloque IP llamado CorePWM incluido en Libero.

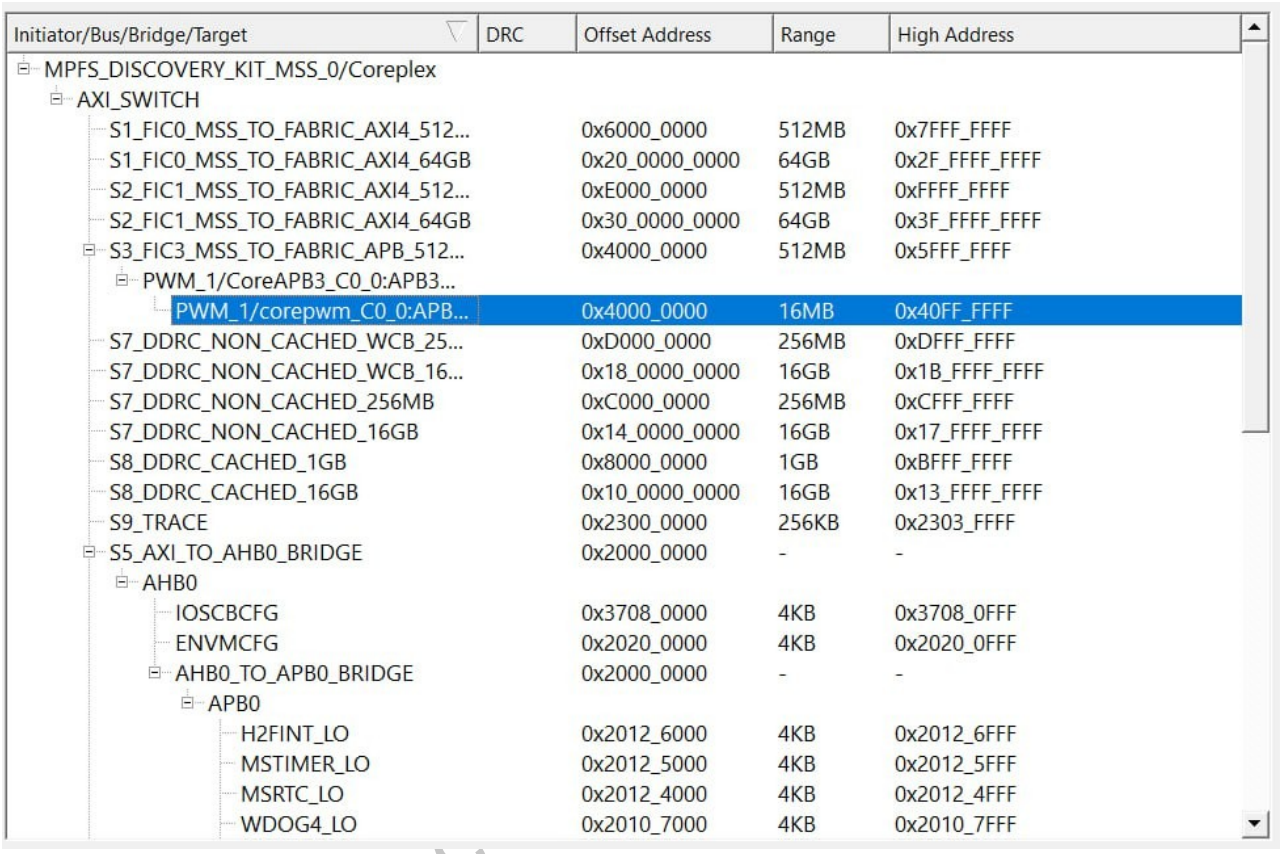


Este bloque tiene una interfaz de tipo APB por lo que para poder conectarlo a una SmartFusion2 o una PolarFire SoC se necesita de una interfaz de tipo **CoreAPB3**. Esta interfaz lo que hace es permitir la comunicación mediante direcciones de memoria con bloque IP.



Entonces, si miramos la dirección de memoria del bloque IP, vemos que tiene adjudicada la dirección de memoria 0x40000000.

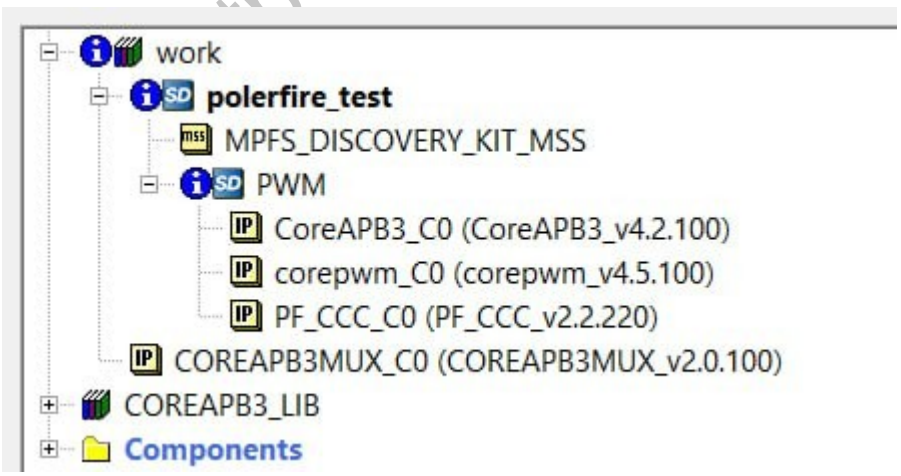
https://soceame.wordpress.com/2025/01/07/como-acceder-a-las-direcciones-de-memoria-de-los-bloques-ip-en-libero/



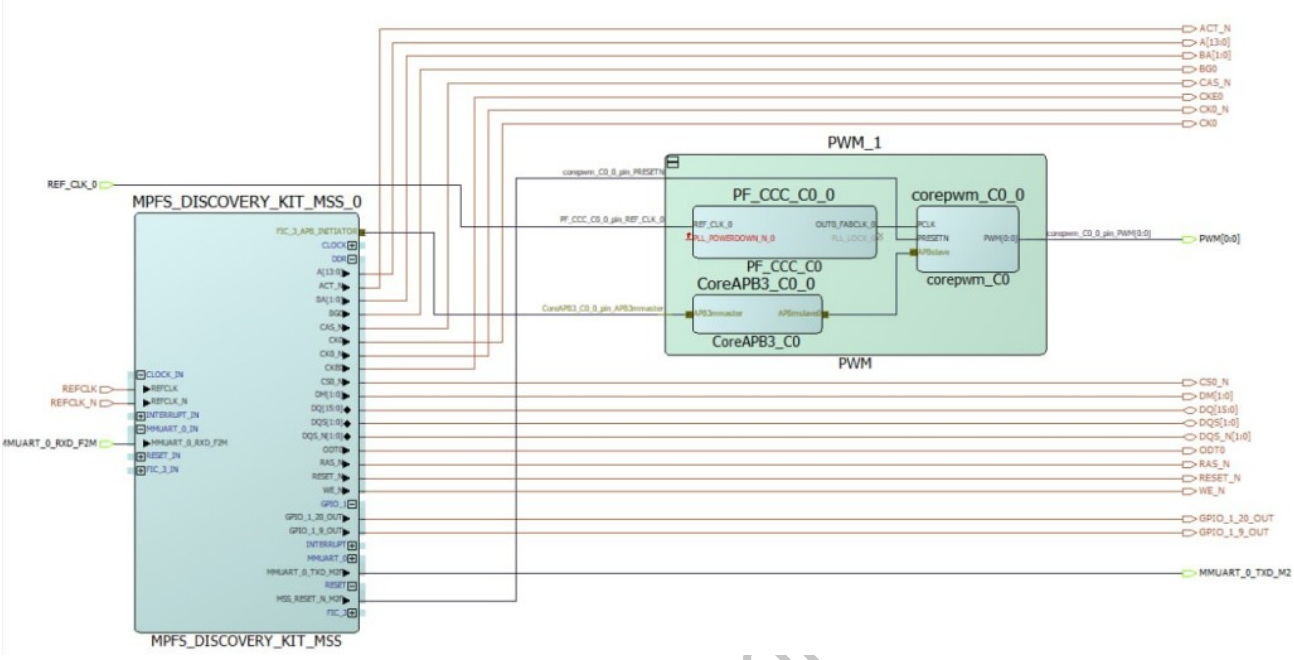
Initiator/Bus/Bridge/Target	DRC	Offset Address	Range	High Address
MPFS_DISCOVERY_KIT_MSS_0/Coreplex				
AXI_SWITCH				
S1_FIC0_MSS_TO_FABRIC_AXI4_512...		0x6000_0000	512MB	0x7FFF_FFFF
S1_FIC0_MSS_TO_FABRIC_AXI4_64GB		0x20_0000_0000	64GB	0x2F_FFFF_FFFF
S2_FIC1_MSS_TO_FABRIC_AXI4_512...		0xE000_0000	512MB	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

También el bloque IP requiere de un reloj para ejecutarse, que puede ser el del MSS u otro. Y un reset, que puede ser el que genera el MSS.

La estructura del proyecto sería algo así.



Entonces, ahora lo único que tenemos que hacer es conectar el *CoreAPB* con la interfaz FIC para APB del MSS.

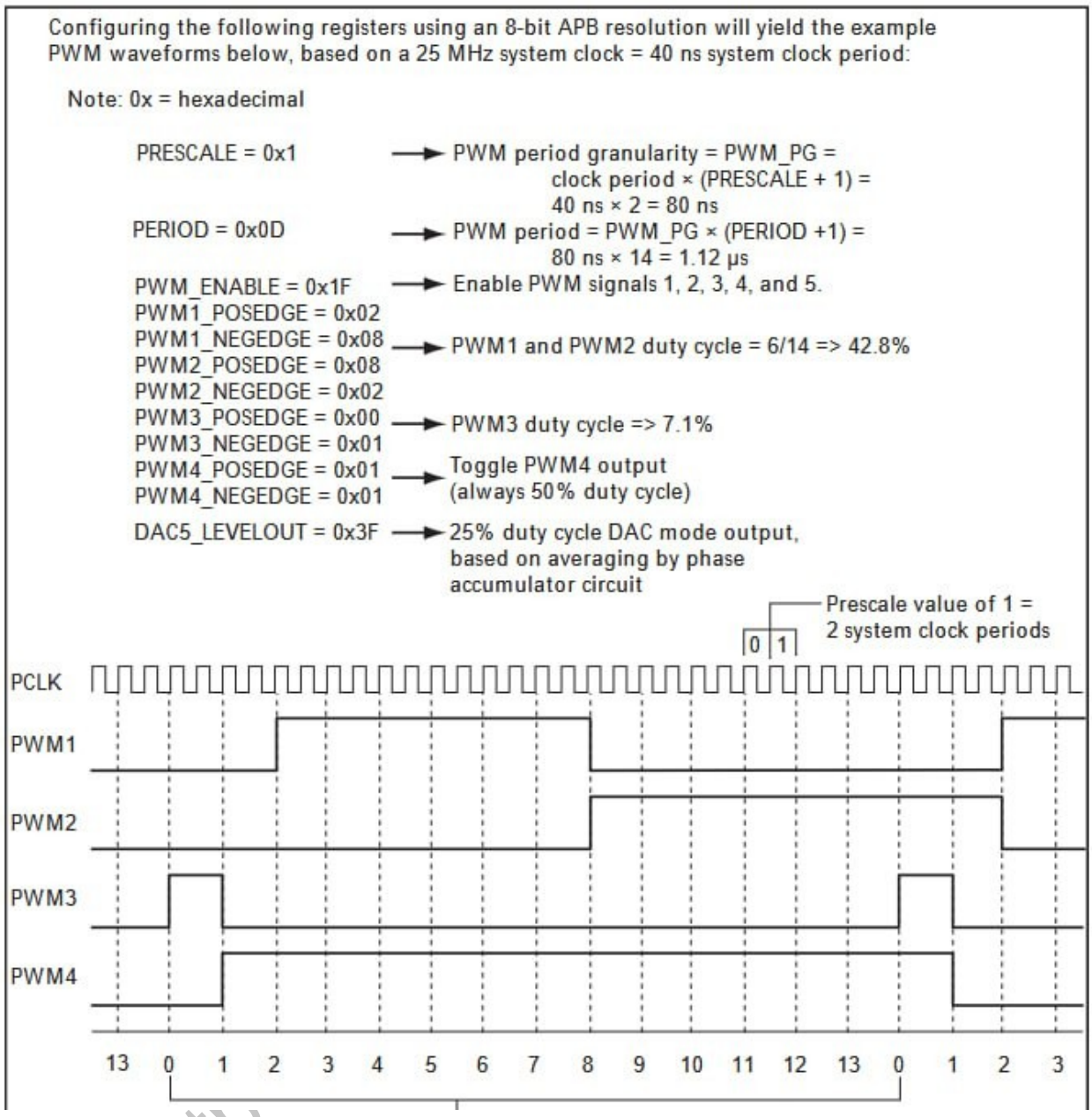


Ahora con la dirección de memoria el SoC, lo único que necesitamos es irnos al SoftConsole para implementar la configuración del bloque IP.

Para ello se utiliza la tabla de memoria interna que provee 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

También provee de un mini ejemplo que es el que se utilizará para el SoC.



Con este ejemplo se desarrolla la aplicación que hace funcionar la PWM.

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

NOTA: debido a que los SoCs de Libero están poco documentados y que los ejemplos que proveen son demasiado complejos, el proyecto anterior no funciona porque se bloquea el procesador. Aún así el flujo de trabajo queda bien definido para configurar los SoCs de Microchip.