The following example shows which SPU instance to use for SAADC peripheral to configure the peripheral permissions using PERIPH[*n*].PERM:

```
#define SPU_CORTEX_ADDRESS_REGION    (0x50000000)

uint32_t perip_addr = NRF_SAADC_S_BASE;


uint32_t apb_bus_number = (perip_addr & 0x00FC0000);
uint32_t apb_slave_index = (perip_addr & 0x0003F000) >> 12;

// Get the address to the SPU instance
NRF_SPU_Type *p_spu = (NRF_SPU_Type*)(SPU_CORTEX_ADDRESS_REGION |
apb_bus_number);

// Configure PERIPH[n].PERM.SECATTR to secure for SAADC
p_spu->PERIPH[apb_slave_index].PERM =
(p_spu->PERIPH[apb_slave_index].PERM &
~SPU_PERIPH_PERM_SECATTR_Msk) |
(SPU_PERIPH_PERM_SECATTR_Secure <<
SPU_PERIPH_PERM_SECATTR_Pos)
```

See Instantiation on page 216 to find the value of SLAVE_BITS for each SPU instance.

SPU supports secure and non-secure accesses based on TrustZone. On each access to a peripheral address, the security state of the master initiating the transaction is verified against the SPU security attribute configuration of the peripheral. The following figure shows a simplified view of the SPU registers controlling several internal modules.

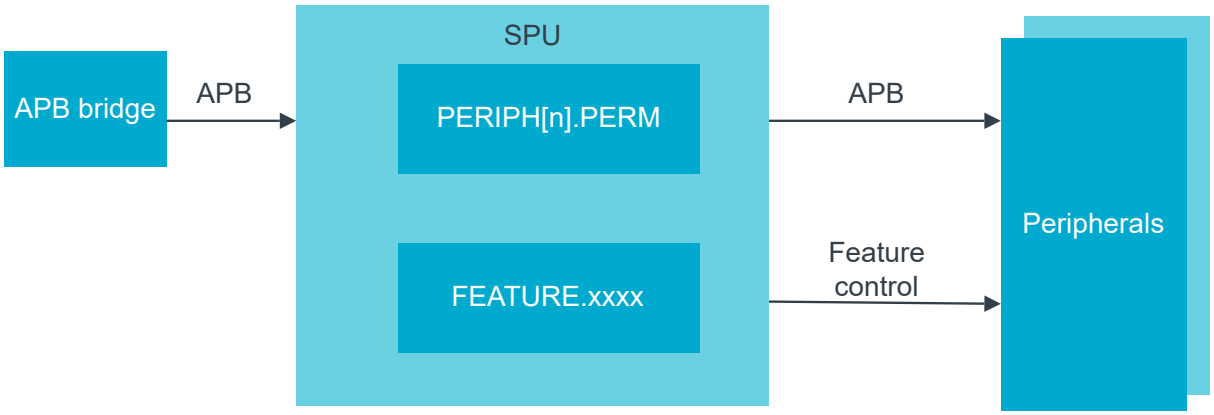

Figure 34: Simplified view of peripherals and peripheral features using SPU

The protection logic implements a read-as-zero/write-ignore (RAZ/WI) policy:

- A read operation that is not allowed by the SPU will always return a zero value on the bus, preventing information leak.
- A write operation that is not allowed by the SPU will be ignored.

An access error on peripherals managed by an SPU result in the PERIPHACCERR event on the SPU.

7.8.5.2 Peripheral access control

Peripheral access control depends on the security attributes.