life.augmented

TN1311

Technical note

PowerPC core memory protection unit

Introduction

This document is intended for software and hardware developers who want to understand the SPC5 core memory protection unit (CMPU), how to configure and handle it.

Main goal of this document is to clarify the core memory protection unit and to provide reference code to configure and manage the core memory protection unit as well as the execution scenario.

The device under analysis is the SPC58xC.

1 Overview

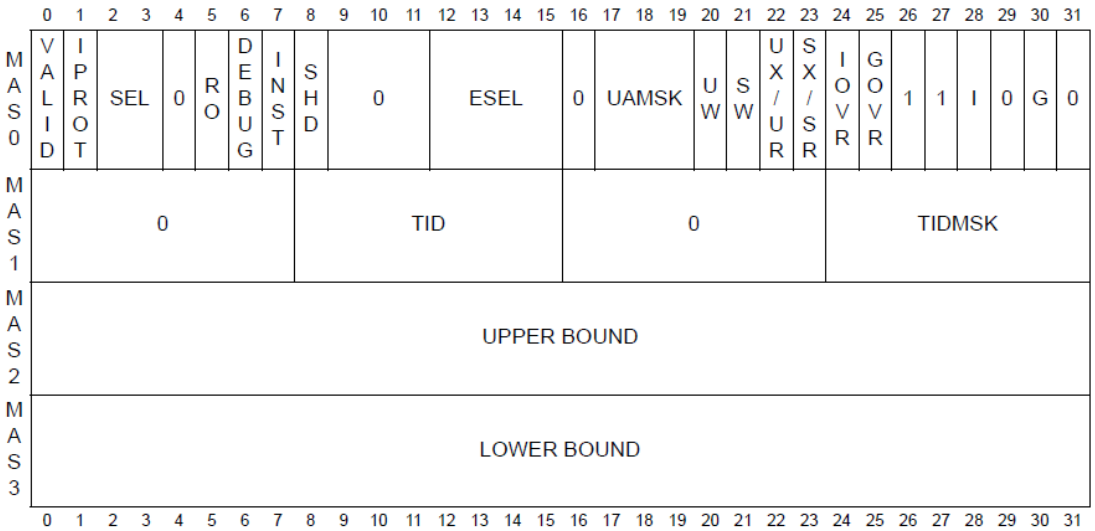
The core memory protection unit (CMPU) is a mechanism included in each core to protect address ranges against access by software. The CMPU is typically used by the operating system to ensure inter-task interference protection. In particular, the protection of the memory regions is based on the following features:

- 24-entry region descriptor table with support for 6 arbitrary-sized instruction memory regions, 12 arbitrary-sized data memory regions and 6 additional arbitrary-sized regions programmable as instruction or data memory regions
- Ability to set access permissions and memory attributes on a per-region basis
- Process ID aware, with per-bit masking of TID values
- Capability for masking upper address bits in the range comparison
- Capability of bypassing permissions checking for selected access types
- Per-entry write-once logic for entry protection
- Hardware flash invalidation support and per-entry invalidation protection controls
- Ability to optionally utilize region descriptors for generating debug events and watch points

The MPU entries are accessed indirectly through 4 MPU Assist (MAS) registers. Software can write and read the MPU Assist registers with **mtspr** and **mfspr** instructions. These registers contain information related to reading and writing a given entry within the MPU. Data is read from the MPU into the MAS registers with a **mpure** (MPU read entry) instruction. Data is written to the MPU from the MAS registers with a **mpuwe** (MPU write entry) instruction.

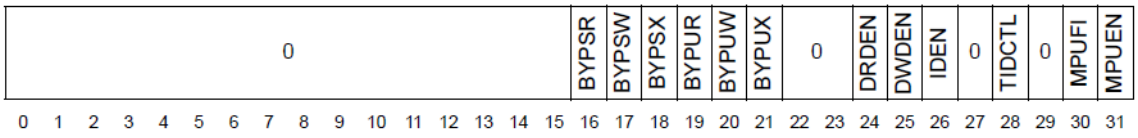
The MAS registers are summarized in the figure below

Figure 1. MPU assist registers summary



Moreover, the MPU0 control and status register 0 (MPU0CSR0) control the operation of the MPU. The MPU0CSR0 register is shown in the next figure.

Figure 2. **MPU0 Control and status register 0 (MPU0CSR0)**

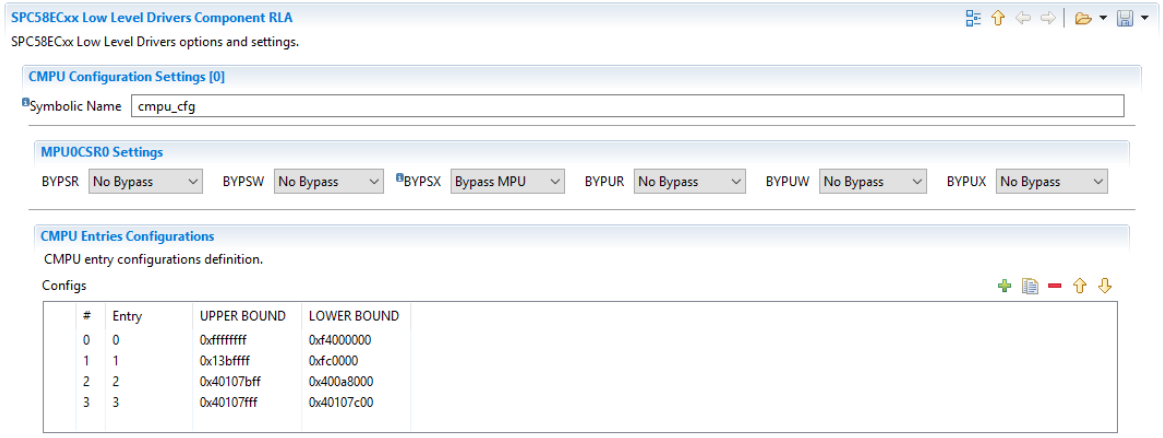


2 SPC5Studio CMPU support

SPC5Studio includes a specific CMPU low level driver that allows to configure the core memory protection unit. It is possible to define more CMPU configurations and dynamically set one of them as active during the application execution. Each configuration includes:

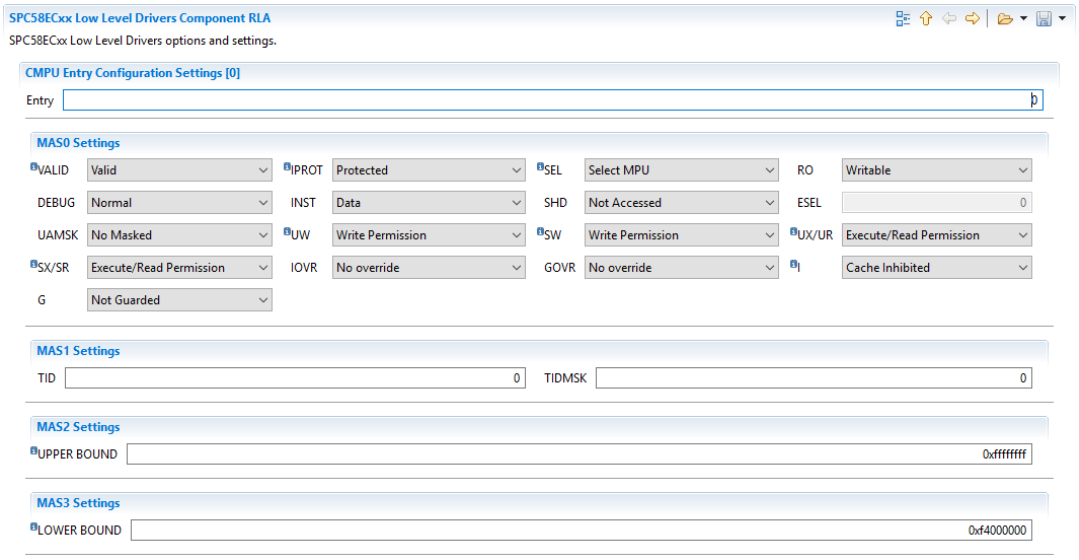
- the configuration of the MPU0CSR0 register
- the configuration of up to 24 CMPU region descriptor entries

Figure 3. SPC5Studio CMPU configuration



The configuration of the MPU0CSR0 register allows to specify if the MPU protections for supervisor/user read/write/instruction accesses, have to be bypassed or not. The configuration of each region descriptor entry (see next figure) allows specifying the lower and upper bounds of the memory region, the access permissions, the entry protection, the region ID and the region mask which allows masking some bits within the TID.

Figure 4. SPC5Studio CMPU entries configuration



Next figure shows the implementation of the main API function (***cmpu_lld_start***) of the SPC5Studio CMPU low-level driver. It is composed of two specific sections, the first one to configure all the CMPU entries defined in the CMPU configuration through the MAS registers and the second one to configure the content of the MPU0CSR0 register and enable the CMPU.

Figure 5. SPC5Studio CMPU configuration API

Configure the CMPU Region Descriptor Entries

Set MPU protection bypass and enable MPU

```
void cmpu_lld_start(CMPUDriver *cmpudp, const CMPUConfig *config) {

  uint8_t i;
  uint32_t mpu0csr0;

  cmpudp->config = config;

  for (i = 0; i < SPC5_CMPU_NUM_OF_ENTRIES; i++) {
    if (cmpudp->config->entry[i] != NULL) {
      /* Set MPU Assist Registers 0 (MAS0).*/
      mtspr(SPC5_SPR_MAS0, cmpudp->config->entry[i]->mas0);

      /* Set MPU Assist Registers 1 (MAS1) - TID and TIDMSK.*/
      mtspr(SPC5_SPR_MAS1, cmpudp->config->entry[i]->mas1);

      /* Set MPU Assist Registers 2 (MAS2) - Upper bound.*/
      mtspr(SPC5_SPR_MAS2, cmpudp->config->entry[i]->mas2);

      /* Set MPU Assist Registers 3 (MAS3) - Lower bound.*/
      mtspr(SPC5_SPR_MAS3, cmpudp->config->entry[i]->mas3);

      mpuwe();
    }
  }

  /* Enable MPU.*/
  mpu0csr0 = cmpudp->config->mpu0csr0 | 0x00000001UL;
  mtspr(SPC5_SPR_MPU0CSR0, mpu0csr0);

  mpusync();
}
```

SPC5Studio also contains an example (***SPC58ECxx_RLA CMPU Test Application***) that shows how to use the CMPU low-level driver. It configures four memory regions via CMPU as shown in table below

Table 1. SPC58ECxx_RLA CMPU test application memory regions

Region	Lower bound	Upper bound	Region type	Access
0	0x00FC0000	0x013BFFFF	Flash	Full access
1	0xF4000000	0xFFFFFFFF	Registers	Full access
2	0x400A8000	0x40107BFF	RAM (383KB)	Full access
3	0x40107C00	0x40107FFF	RAM (1KB)	No access

Then the application attempts to access to the region 3. Since region 3 is protected, an IVOR2 will occur and a specific led will blink to highlight the test has been passed. The led code is implemented in the related callback function.

Revision history

Table 2. **Document revision history**

Date	Version	Changes
23-Jul-2020	1	Initial release.

Contents