Application Report

J721E DDR Firewall Example

Kip Broadhurst

ABSTRACT

This application report focuses on the programming of the Jacinto™ 7 firewalls to isolate A72 / A53 originated DDR transactions from undesired accesses to defined DDR memory locations. General information is also included on where to find firewall related documentation, and an understanding of how regional firewalls can be configured.

Project collateral and code mentioned in this document can be downloaded from the following URL: https://www.ti.com/lit/zip/spracx6.

Table of Contents

List of Figures

List of Tables

Trademarks

Jacinto™ is a trademark of Texas Instruments.
All trademarks are the property of their respective owners.

1 Introduction

Firewalls can be used to protect both data and device configuration by managing the access to defined memory ranges. The isolation provided firewall functionality is useful when considering both safety and security solutions.

The Jacinto 7 family of devices all use the same approach to firewalls. This document focuses on the J721E device and focuses on the Processor SDK 7.1 release. The examples provide are expected to be conceptually the same, in programming and code location, across the Jacinto 7 family and the Processor SDK releases.

The sample code and information in the document is a guideline only, full testing and understanding of why the system under test is being firewalled is required for any implementation.

2 Firewall Documentation

Documentation for Jacinto 7 implementation of firewalls is available in multiple locations, and touches on many different components on the Jacinto 7 device and SDK S/W architecture. This document references and borrows from the available TI material.

2.1 Technical Reference Manual (TRM)

The TRM for the device under test, has a section called *System Interconnect*. Within that section there is a sub-section on *Interconnect Firewall*. To get a feel for the Jacinto 7 firewall solution, TI encourages that you start by referencing this section.

TRMs are available for each TI device on ti.com.

2.2 SDK TISCI Documentation

You can program the firewalls by using TI System Controller Interface (TISCI) APIs. Direct programming of firewalls is not permitted on the J7 family of devices.

The TISCI documentation on firewalls is recommended reading before beginning any work with the Jacinto 7 firewalls. It provides a background on terminology, and overall concepts.

- Firewall TISCI Description

  The message set for TISCI used for firewall configuration is small. The complexity arises in designing how the system is desired to be configured. Messages available from TISCI for Firewalls are:

  - Set Firewall Region:
    - Sets, control, permissions and address region for specified firewall Id & region.
  - Get Firewall Region:
    - Queries current configuration for specified firewall Id and region
  - Change Firewall Owner:
    - Change owner of specified owner for the specified firewall Id and region.

2.3 SDK Firewall Documentation

The TI Processor SDKs for RTOS contain documentation on all components of the SDK including firewalls. Below are some example links for the Processor SDK RTOS 7.1 release. Each of the pages should exist for the Jacinto 7 family device that is under test. Links provided below will go to the J721E device latest SDK release.

- J721E Firewall Descriptions
- J721E Host Descriptions
- Firewall FAQ

2.4 TI NDA Firewall Slide Sets

For customers under NDA with Texas Instruments, a slide set going over firewall setup is available. For more information, contact your local TI representative for NDA access.

3 Firewall Definitions and Terms

For an understanding of firewalls, when reading TI documentation, the below terms also need to be understood. The Firewall FAQ is a great reference for further understanding.

Region A defined memory range, against which firewall permission and control attributes are stored. These regional permissions / attributes are used to filter interconnect transactions for a module. Each firewall can have 1 to 24 regions. Each region has following registers:

- Control
- Permission
- Start / End Address

Host Id Host Id is a software concept used by SYSFW and is used in TISCI. A host id represents a processing entity. Host IDS for the Jacinto 7 device are listed both in the TRM and SDK documentation, or can be viewed in header files.

Priv Id The privilege ID is a hardware level identifier. Every host maps to a priv Id. A Priv ID can represent one or more Host Ids. Priv IDs are listed in the SDK documentation or can be viewed in SDK header files.

Firewall Id An identifier used to uniquely define each firewall.

System Firmware System Firmware is a collective term used to describe the TI Foundational Security (TIFS) and Resource Management (RM)/ Power Management (PM) services.

DMSC Security Manager and Device Manager Core (DMSC). System firmware executes on the DMSC.

4 SysConfig Tool

TI provides an offline resource management tool called SysConfig. SysConfig can be used to auto generate generic code that can then be used in the SDK for programming of firewalls. General usage for the firewalls would be as listed below:

1. Use the TI SysConfig tool to define the various firewalls that are to be programmed
2. The SysConfig tool will generate a .c file.
3. Integrate the contents from the autogenerated .c file into the boot flow of choice

For instructions on installing and using the SysConfig, reference the embedded link.

The output format from SysConfig can be re-used in the sample code included in this document.

5 Master Firewall versus Slave Firewall

All modules and subsystems on the Jacinto7 interconnect can be classified into two categories: masters and slaves. Masters are capable of initiating read and write transfers in the system. The slaves on the other hand depend on the masters to perform transfers to and from them.

5.1 Slave Firewalls

This is not covered in the document, but as the format from the SysConfig tool can be re-used in the sample code below, it is a simple step to also add slave firewalls. A slave firewall will filter transactions at the slave end of the connection, not at the master.

5.2 Master Firewalls

For Master firewall transactions, the transactions are filtered before going to the interconnect.

Looking at the J721e device, there are three firewalls present, A72, C7x and DRU. The A72 master firewalls will be looked at more closely in this document.

5.3 A72 Master Firewall

The A72 itself is a master on the interconnect and has a master side firewall capable of filtering outgoing transactions. Figure 5-1 (available in the Technical Reference for the DRA829 / TDA4 device) shows where the firewall to be programmed is located.

The master side firewall for the A72 has a Firewall Id of 257, as seen in below code example as `CSL_MSTR_FW_A72SS0_CORE0_CPU_0_CPU_0_MSMC_ID`.

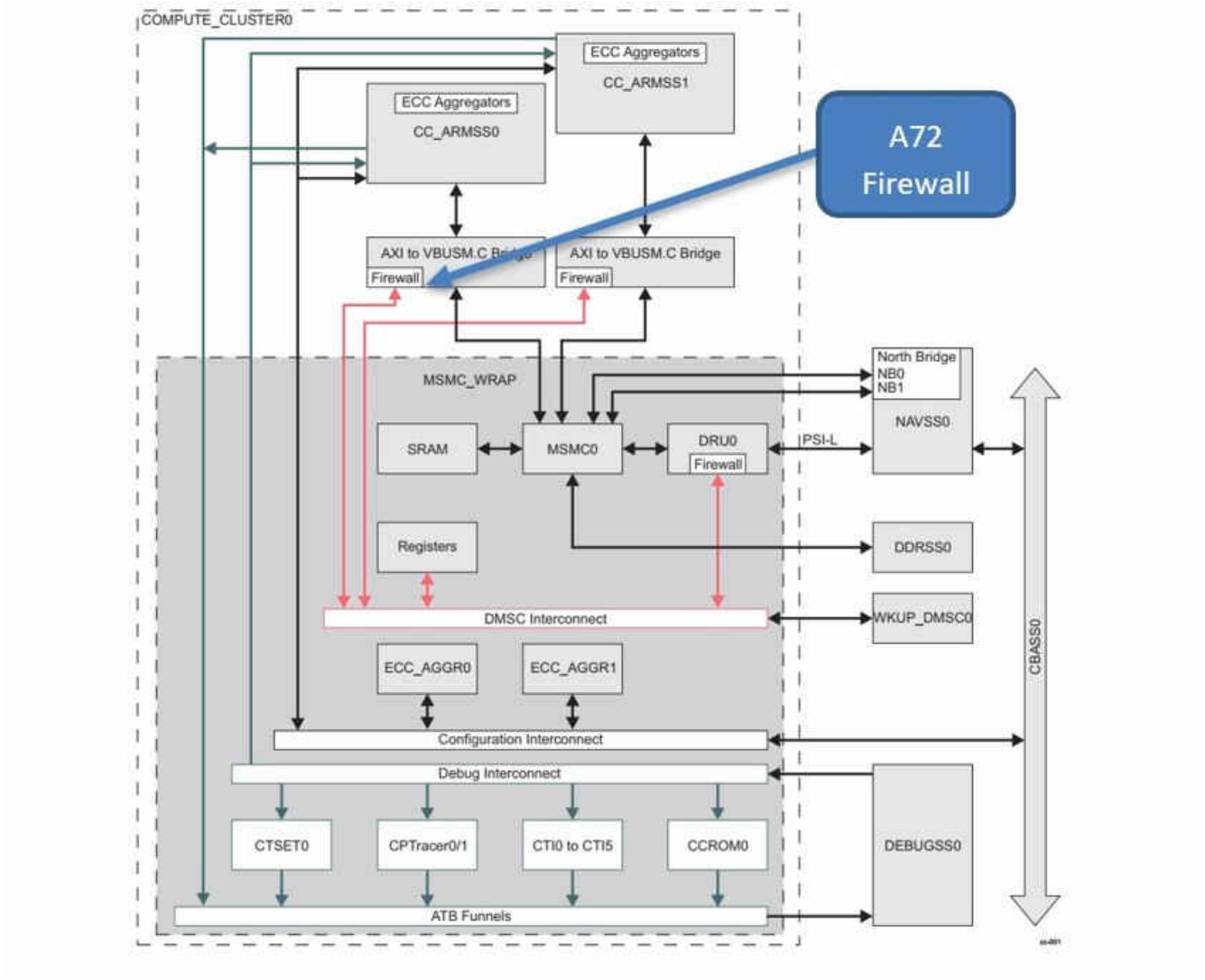

Figure 5-1. COMPUTE_CLUSTER0 Overview

6 Where to Firewall

What DDR memory ranges to firewall will differ from customer to customer, and from use case to use case. Taking a default TI SDK release as an example, some easy first steps can be seen.

Each release of the PSDK, when built for vision_apps solutions, will generate a system memory map. The system memory map shows memory that is specific to each, core, and memory that shared between cores. This memory map is a recommended resource for determining which areas should be firewalled for A72 access.

6.1 Example

When customizing for a system, the memory map should be reviewed to identify any regions that must be protected as well as reviewing for memory regions that should be protected. If the A72 does not require access, then that memory can be optionally firewalled.

Referencing the system memory map from Processor SDK QNX 7.1, Table 6-1 can be generated. In this table, the memory regions can be reviewed to identify memory ranges that make sense to firewall, to prevent A72 access.

With these firewalls in place all the Vision Apps SDK demos would continue to function. If A72 software inadvertently attempts to access any of the firewalled memory regions, and exception will occur.

Table 6-1. Memory Regions Used to Identify Memory Ranges

Name	Start Addr	End Addr	Size	Attributes	Description
L2RAM_C66x_1	0x00800000	0x00837FFF	224.00 KB	RWIX	L2 for C66x_1
L2RAM_C66x_2	0x00800000	0x00837FFF	224.00 KB	RWIX	L2 for C66x_2
MAIN_OCRAM_ MCU2_0	0x03600000	0x0361FFFF	128.00 KB	RWIX	Main OCRAM for MCU2_0
MAIN_OCRAM_ MCU2_1	0x03620000	0x0363FFFF	128.00 KB	RWIX	Main OCRAM for MCU2_1
L2RAM_C7x_1	0x64800000	0x64877FFF	480.00 KB	RWIX	L2 for C7x_1
L1RAM_C7x_1	0x64E00000	0x64E03FFF	16.00 KB	RWIX	L1 for C7x_1
MSMC_MPU1	0x70000000	0x7001FFFF	128.00 KB	RWIX	MSMC reserved for MPU1 for ATF
MSMC_C7x_1	0x70020000	0x707E7FFF	7.78 MB	RWIX	MSMC for C7x_1
MSMC_DMSC	0x707F0000	0x707FFFFF	64.00 KB	RWIX	MSMC reserved for DMSC IPC
DDR_MCU1_0_IPC	0xA0000000	0xA00FFFFF	1024.00 KB	RWIX	DDR for MCU1_0 for Linux IPC
DDR_MCU1_0_ RESOURCE_TABLE	0xA0100000	0xA01003FF	1024 B	RWIX	DDR for MCU1_0 for Linux resource table
DDR_MCU1_0	0xA0100400	0xA0FFFFFF	15.00 MB	RWIX	DDR for MCU1_0 for code/data
DDR_MCU2_0_IPC	0xA1000000	0xA10FFFFF	1024.00 KB	RWIX	DDR for MCU2_0 for Linux IPC
DDR_MCU2_0_ RESOURCE_TABLE	0xA1100000	0xA11003FF	1024 B	RWIX	DDR for MCU2_0 for Linux resource table
DDR_MCU2_0	0xA1100400	0xA2FFFFFF	31.00 MB	RWIX	DDR for MCU2_0 for code/data
DDR_MCU2_1_IPC	0xA3000000	0xA30FFFFF	1024.00 KB	RWIX	DDR for MCU2_1 for Linux IPC
DDR_MCU2_1_ RESOURCE_TABLE	0xA3100000	0xA31003FF	1024 B	RWIX	DDR for MCU2_1 for Linux resource table
DDR_MCU2_1	0xA3100400	0xA4FFFFFF	31.00 MB	RWIX	DDR for MCU2_1 for code/data
DDR_MCU3_0_IPC	0xA5000000	0xA50FFFFF	1024.00 KB	RWIX	DDR for MCU3_0 for Linux IPC
DDR_MCU3_0_ RESOURCE_TABLE	0xA5100000	0xA51003FF	1024 B	RWIX	DDR for MCU3_0 for Linux resource table

Table 6-1. Memory Regions Used to Identify Memory Ranges (continued)

Name	Start Addr	End Addr	Size	Attributes	Description
DDR_MCU3_0	0xA5100400	0xA57FFFFF	7.00 MB	RWIX	DDR for MCU3_0 for code/data
DDR_MCU3_1_IPC	0xA5800000	0xA58FFFFF	1024.00 KB	RWIX	DDR for MCU3_1 for Linux IPC
DDR_MCU3_1_ RESOURCE_TABLE	0xA5900000	0xA59003FF	1024 B	RWIX	DDR for MCU3_1 for Linux resource table
DDR_MCU3_1	0xA5900400	0xA5FFFFFF	7.00 MB	RWIX	DDR for MCU3_1 for code/data
DDR_C66x_2_IPC	0xA6000000	0xA60FFFFF	1024.00 KB	RWIX	DDR for C66x_2 for Linux IPC
DDR_C66x_1_ RESOURCE_TABLE	0xA6100000	0xA61003FF	1024 B	RWIX	DDR for C66x_1 for Linux resource table
DDR_C66x_1_BOOT	0xA6200000	0xA62003FF	1024 B	RWIX	DDR for C66x_1 for boot section
DDR_C66x_1	0xA6200400	0xA6FFFFFF	14.00 MB	RWIX	DDR for C66x_1 for code/data
DDR_C66x_1_IPC	0xA7000000	0xA70FFFFF	1024.00 KB	RWIX	DDR for C66x_1 for Linux IPC
DDR_C66x_2_ RESOURCE_TABLE	0xA7100000	0xA71003FF	1024 B	RWIX	DDR for C66x_2 for Linux resource table
DDR_C66x_2_BOOT	0xA7200000	0xA72003FF	1024 B	RWIX	DDR for C66x_2 for boot section
DDR_C66x_2	0xA7200400	0xA7FFFFFF	14.00 MB	RWIX	DDR for C66x_2 for code/data
DDR_C7x_1_IPC	0xA8000000	0xA80FFFFF	1024.00 KB	RWIX	DDR for C7x_1 for Linux IPC
DDR_C7x_1_ RESOURCE_TABLE	0xA8100000	0xA81003FF	1024 B	RWIX	DDR for C7x_1 for Linux resource table
DDR_C7x_1_BOOT	0xA8200000	0xA82003FF	1024 B	RWIX	DDR for C7x_1 for boot section
DDR_C7x_1_VECS	0xA8400000	0xA8403FFF	16.00 KB	RWIX	DDR for C7x_1 for vecs section
DDR_C7x_1_ SECURE_VECS	0xA8600000	0xA8603FFF	16.00 KB	RWIX	DDR for C7x_1 for secure vecs section
DDR_C7x_1	0xA8604000	0xA8FFFFFF	9.98 MB	RWIX	DDR for C7x_1 for code/data
IPC_VRING_MEM	0xAA000000	0xABFFFFFF	32.00 MB		Memory for IPC Vring's. MUST be non-cached or cache-coherent
APP_LOG_MEM	0xAC000000	0xAC03FFFF	256.00 KB		Memory for remote core logging
TIOVX_OBJ_ DESC_MEM	0xAC040000	0xADFDFFFF	31.62 MB		Memory for TI OpenVX shared memory. MUST be non-cached or cache-coherent
PCIE_QUEUE_ SHARED_MEM	0xADFE0000	0xADFEFFFF	64.00 KB		Memory for IPC over PCIe using shared memory. MUST be non-cached or cache-coherent
PCIE_QUEUE_ MIRROR_REMOTE_ SHARED_MEM	0xADFF0000	0xADFFFFFF	64.00 KB		Reserved Memory for RAT mapping of remote PCIe IPC shared memory. MUST be non-cached or cache-coherent

Table 6-1. Memory Regions Used to Identify Memory Ranges (continued)

Name	Start Addr	End Addr	Size	Attributes	Description
DDR_SHARED_MEM	0xAE000000	0xCDFFFFFF	512.00 MB		Memory for shared memory buffers in DDR
DDR_MCU2_0_NON_CACHE	0xCE000000	0xCE00FFFF	64.00 KB	RWIX	DDR for MCU2_0 for non-cached heap
DDR_MCU2_1_NON_CACHE	0xCE010000	0xD1FFFFFF	63.94 MB	RWIX	DDR for MCU2_1 for non-cached heap
DDR_MCU1_0_LOCAL_HEAP	0xD2000000	0xD21FFFFF	2.00 MB	RWIX	DDR for MCU1_0 for local heap
DDR_MCU1_1_LOCAL_HEAP	0xD2200000	0xD23FFFFF	2.00 MB	RWIX	DDR for MCU1_1 for local heap
DDR_MCU2_0_LOCAL_HEAP	0xD2400000	0xD2BFFFFF	8.00 MB	RWIX	DDR for MCU2_0 for local heap
DDR_MCU2_1_LOCAL_HEAP	0xD2C00000	0xD3BFFFFF	16.00 MB	RWIX	DDR for MCU2_1 for local heap
DDR_MCU3_0_LOCAL_HEAP	0xD3C00000	0xD3DFFFFF	2.00 MB	RWIX	DDR for MCU3_0 for local heap
DDR_MCU3_1_LOCAL_HEAP	0xD3E00000	0xD3FFFFFF	2.00 MB	RWIX	DDR for MCU3_1 for local heap
DDR_C66X_1_LOCAL_HEAP	0xD4000000	0xD4FFFFFF	16.00 MB	RWIX	DDR for c66x_1 for local heap
DDR_C66X_1_SCRATCH	0xD5000000	0xD7FFFFFF	48.00 MB	RWIX	DDR for c66x_1 for Scratch Memory
DDR_C66X_2_LOCAL_HEAP	0xD8000000	0xD8FFFFFF	16.00 MB	RWIX	DDR for c66x_2 for local heap
DDR_C66X_2_SCRATCH	0xD9000000	0xDBFFFFFF	48.00 MB	RWIX	DDR for c66x_2 for Scratch Memory
DDR_C7X_1_LOCAL_HEAP	0xDC000000	0xEBFFFFFF	256.00 MB	RWIX	DDR for c7x_1 for local heap
DDR_C7X_1_SCRATCH	0xEC000000	0xF9FFFFFF	224.00 MB	RWIX	DDR for c7x_1 for Scratch Memory
TIOVX_LOG_RT_MEM	0xFA000000	0xFAFFFFFF	16.00 MB		Memory for TI OpenVX shared memory for Run-time logging. MUST be non-cached or cache-coherent
DDR_MCU1_1_IPC	0xFB000000	0xFB0FFFFF	1024.00 KB	RWIX	DDR for MCU1_1 for Linux IPC
DDR_MCU1_1_RESOURCE_TABLE	0xFB100000	0xFB1003FF	1024 B	RWIX	DDR for MCU1_1 for Linux resource table
DDR_MCU1_1	0xFB100400	0xFBFFFFFF	15.00 MB	RWIX	DDR for MCU1_1 for code/data

Using this table and combining adjacent memory regions to firewalled, the below 2 x DDR ranges,cover all the memory locations that the A72 should NOT be accessing.

Start Address	End Address
0XA0000000	0XA8FFFFFF
0xCE000000	0xFBFFFFFF

To prevent the A72 from accessing the 2 memory ranges, a firewall Region for each memory must be configured, where the region permissions are specified to not allow any A72 access. In the following example code, Region 0 of the A72 master firewall is set to allow all accesses from A72, while Region 1 and Region 2, are set to prevent A72 access to the memory ranges that need to be firewalled. The resulting view of accessible memory from A72 perspective is show pictorially below.

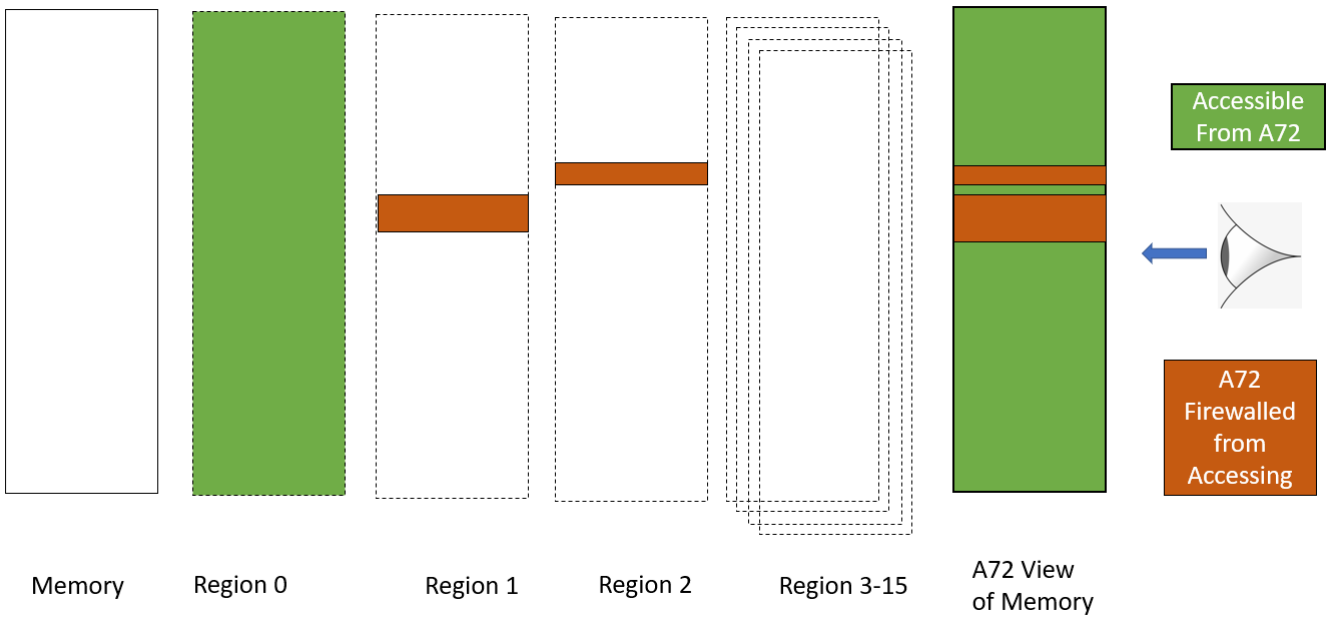

Figure 6-1. Example Code

7 Programming Firewalls

Note that all programming of firewall register is done via TISCI. There is no direct register programming of firewall registers on the Jacinto 7 family of devices.

Not all firewalls are user programmable. For firewalls that are programmable, you can use available TI documentation or make use of the sample code below.

There are numerous examples in the Processor SDK showing firewalls being programmed, as well, the aforementioned TI Documentation has lots of information.

The sample code below is targeted at preventing A72 from accessing certain DDR memory regions. There will be many different use cases, to allow or prevent access to memory locations or modules. Each of these scenarios can be handled by the below sample framework, by simply expanding the table entries.

7.1 Sample SBL Code

When adding a firewall, no transactions can be in transit that would hit the firewall. As such, adding firewalls during initialization is recommended. Below is one option showing functions that can be added to the SBL boot flow to program the A72 Master firewall.

As mentioned above, this same framework can be re-used for Slave firewalls and other Master firewalls. Only the table entries need to be updated. Table entries can be generated using the SysConfig tool.

7.1.1 Create a Table

Create a table with entries in which each entry represents a firewall region. This format is the same format as the .c output used by SysConfig tool. In the example below, three regions are created for the A72 Master firewall.

```
struct ti_sci_msg_fwl_region {
    uint16_t            fwl_id;
    uint16_t            region;
    uint32_t            n_permission_regs;
    uint32_t            control;
    uint32_t            permissions[FWL_MAX_PRIVID_SLOTS];
    uint64_t        start_address;
    uint64_t        end_address;
} __attribute__((__packed__));

void J721E_Set_Firewall(uint32_t isBuildHs)
{
    int32_t status = CSL_EFAIL;
    struct ti_sci_msg_fwl_region j721e_fwl_data[] = {

        /* compute_cluster Master firewall  - background region 0 */
        {
            .fwl_id = CSL_MSTR_FW_A72SS0_CORE0_CPU_0_CPU_0_MSMC_ID,
            .region = 0,
            .n_permission_regs = 1,
            .control = 0x30A,
            .start_address = 0x00000000,
            .end_address = 0xFFFFFFFFF,
            .permissions = { 0x1FFFF }, // PrivId 1U
        },
        /* compute_cluster Master firewall  - region 1 */
        {
            .fwl_id = CSL_MSTR_FW_A72SS0_CORE0_CPU_0_CPU_0_MSMC_ID,
            .region = 1,
            .n_permission_regs = 1,
            .control = 0x20A,
            .start_address = 0xa0000000,
            .end_address = 0xa8ffffff,
            .permissions = { 0x10000 }, // PrivId 1U
        },
        /* compute_cluster Master firewall  - region 2 */
        {
            .fwl_id = CSL_MSTR_FW_A72SS0_CORE0_CPU_0_CPU_0_MSMC_ID,
            .region = 2,
            .n_permission_regs = 1,
            .control = 0x20A,
            .start_address = 0xce000000,
            .end_address = 0xfbffffff,
            .permissions = { 0x10000 }, // PrivId 1U
        },
    };
```

- Region 0 as a background region, giving A72 full access to memory range.
- Region 1 and 2, then introduce restrictions, ensuring that A72 does not have any permissions on the memory ranges defined for those regions.
- Note that for all three of the regions, the Privid on the .permissions entry, indicates which originator the permissions should be applied to. When bit [16] is set to ‘1’, this indicates that the permissions are to be applied to transactions originated from the A72.

7.1.2 Parse the Table of Firewall Regions

For each entry in the table of firewall regions

1. set the ownership, using the defined TISCI APIs
2. program the region, using the defined TISCI APIs

The ownership in this case is set to the MCU Boot Island where the SBL code will be running.

```
uint32_t i = 0;
    uint32_t j = 0;
    uint32_t j721e_fwl_count = 3;  // Number of entries

    struct tisci_msg_fwl_set_firewall_region_resp respFwCtrl = {0};
    struct tisci_msg_fwl_set_firewall_region_req reqFwCtrl;

    for (i = 0; i < j721e_fwl_count; i++)
    {
        /* Setting Owner */
        struct tisci_msg_fwl_change_owner_info_req req;
        req.fwl_id = (uint16_t)j721e_fwl_data[i].fwl_id;
        req.region = (uint16_t) j721e_fwl_data[i].region;
        req.owner_index = (uint8_t) HOST_ID_MCU_0_R5_1; // Cortex R5 context 1 on MCU island(Boot)

        struct tisci_msg_fwl_change_owner_info_resp resp = {0};


        status = Sciclient_firewallChangeOwnerInfo(&req, &resp, SCICLIENT_SERVICE_WAIT_FOREVER);
        if (status != CSL_PASS)
        {
            SBL_log(SBL_LOG_ERR,"Firewall Unable to change Owner, %d\n", i);
            J721E_dump_owner_req(&req);
        }

        /* Setting Region */
        reqFwCtrl.fwl_id = (uint16_t) j721e_fwl_data[i].fwl_id;
        reqFwCtrl.region = (uint16_t) j721e_fwl_data[i].region;
        reqFwCtrl.n_permission_regs = (uint32_t) j721e_fwl_data[i].n_permission_regs;
        reqFwCtrl.control = (uint32_t) j721e_fwl_data[i].control;
        for(j = 0; j < reqFwCtrl.n_permission_regs; j++)
        {
            reqFwCtrl.permissions[j] = (uint32_t) j721e_fwl_data[i].permissions[j];
        }
        reqFwCtrl.start_address = j721e_fwl_data[i].start_address;
        reqFwCtrl.end_address = j721e_fwl_data[i].end_address;

        status = Sciclient_firewallSetRegion(&reqFwCtrl, &respFwCtrl,
SCICLIENT_SERVICE_WAIT_FOREVER);
        if (status != CSL_PASS)
        {
            SBL_log(SBL_LOG_ERR,"Firewall entry/%d, region set failed.\n", i);
            J721E_dump_region_req(&reqFwCtrl);
        }
    }
```

7.1.3 Utility Functions

New code is not always successful on the first try. Below utility functions can dump out the requests being sent via SCI, allowing for some readable output on the console port.

```
void J721E_dump_owner_req(struct tisci_msg_fwl_change_owner_info_req *req)
{
    uint32_t i = 0;

    SBL_log(SBL_LOG_ERR,"\n");
    SBL_log(SBL_LOG_ERR,"Ownership Request:\n");
    SBL_log(SBL_LOG_ERR,"req.fwl_id                = 0x%x\n",req->fwl_id);
    SBL_log(SBL_LOG_ERR,"req.owner_index    = 0x%x\n",req->owner_index);
    SBL_log(SBL_LOG_ERR,"req.region                = 0x%x\n",req->region);

    uint8_t *bPtr = (uint8_t *) req;
    for (i = 0; (i < sizeof(struct tisci_msg_fwl_change_owner_info_req)); i++)
    {
        SBL_log(SBL_LOG_ERR,"%02x ", bPtr[i]);
    }
    SBL_log(SBL_LOG_ERR,"\n");
    return;
}


void J721E_dump_region_req(struct tisci_msg_fwl_set_firewall_region_req *req)
{
    uint32_t i = 0;
    uint32_t *startAddr;
    uint32_t *endAddr;

    startAddr = (uint32_t *) &req->start_address;
    endAddr = (uint32_t *) &req->end_address;

    SBL_log(SBL_LOG_ERR,"\n");
    SBL_log(SBL_LOG_ERR,"Region Set Request:\n");
    SBL_log(SBL_LOG_ERR,"req.fwl_id            = %d\n",   req->fwl_id);
    SBL_log(SBL_LOG_ERR,"req.region            = %d\n",   req->region);
    SBL_log(SBL_LOG_ERR,"req.control           = 0x%x\n", req->control);
    SBL_log(SBL_LOG_ERR,"req.start_address     = 0x%x%x\n", startAddr[1], startAddr[0]);
    SBL_log(SBL_LOG_ERR,"req.end_address       = 0x%x%x\n", endAddr[1], endAddr[0]);
    SBL_log(SBL_LOG_ERR,"req.n_permission_regs = 0x%x\n", req->n_permission_regs);
    for(i = 0; i < req->n_permission_regs; i++)
    {
        SBL_log(SBL_LOG_ERR, "req.permissions[%d]    = 0x%x\n", i, req->permissions[0]);
    }
    SBL_log(SBL_LOG_ERR, "\n");
    return;
}
```

7.1.4 Processor SDK 7.1 SBL Example

Attached in a zip file is an example modification done to SBL boot flow in Processors SDK 7.1, using the above sample code.

The sample code in the attached zip file is based on the Processor SDK memory map provided in Processor SDK 7.1. The firewall memory ranges would likely need to be customized for the platform under test.

To see the changes, the two directories in the zip file can be compared:

- ti-processor-sdk-rtos-j721e-evm-07_01_00_11_baseline
- ti-processor-sdk-rtos-j721e-evm-07_01_00_11_firewalls

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265