J7VCL MCAL - SoC User Manual

Document Version : 23
Document Owner : Nikki Shah

TI Confidential - NDA Restrictions

Copyright ©2023 **Texas Instruments Incorporated**

Document Revision History

Revision	Date	Author	Description
0.1	07 Mar 2022	Nikki Shah	Initial Version
0.2	09 Dec 2022	Ashok Kumawat	SoC raleted updates
v.23	13 Feb 2023	Rohit Tiwari	Updated the log for CddIPC communication for linux as host

1 1.1. Introduction

SoC's such as J7VCL (J7200) family of devices, integrates an MicroController Unit Subsystem (MCU SS) as an chip-in-chip. It operates using a separate voltage supply, clock sources and resets and includes the components needed for device management. This allows the MCUSS to function continuously regardless of the state of the rest of the device. MCU SS has one or more DUAL core Cortex R5F.

2 1.2. Purpose

This document is a supplement to the CSP provided with the MCAL product.

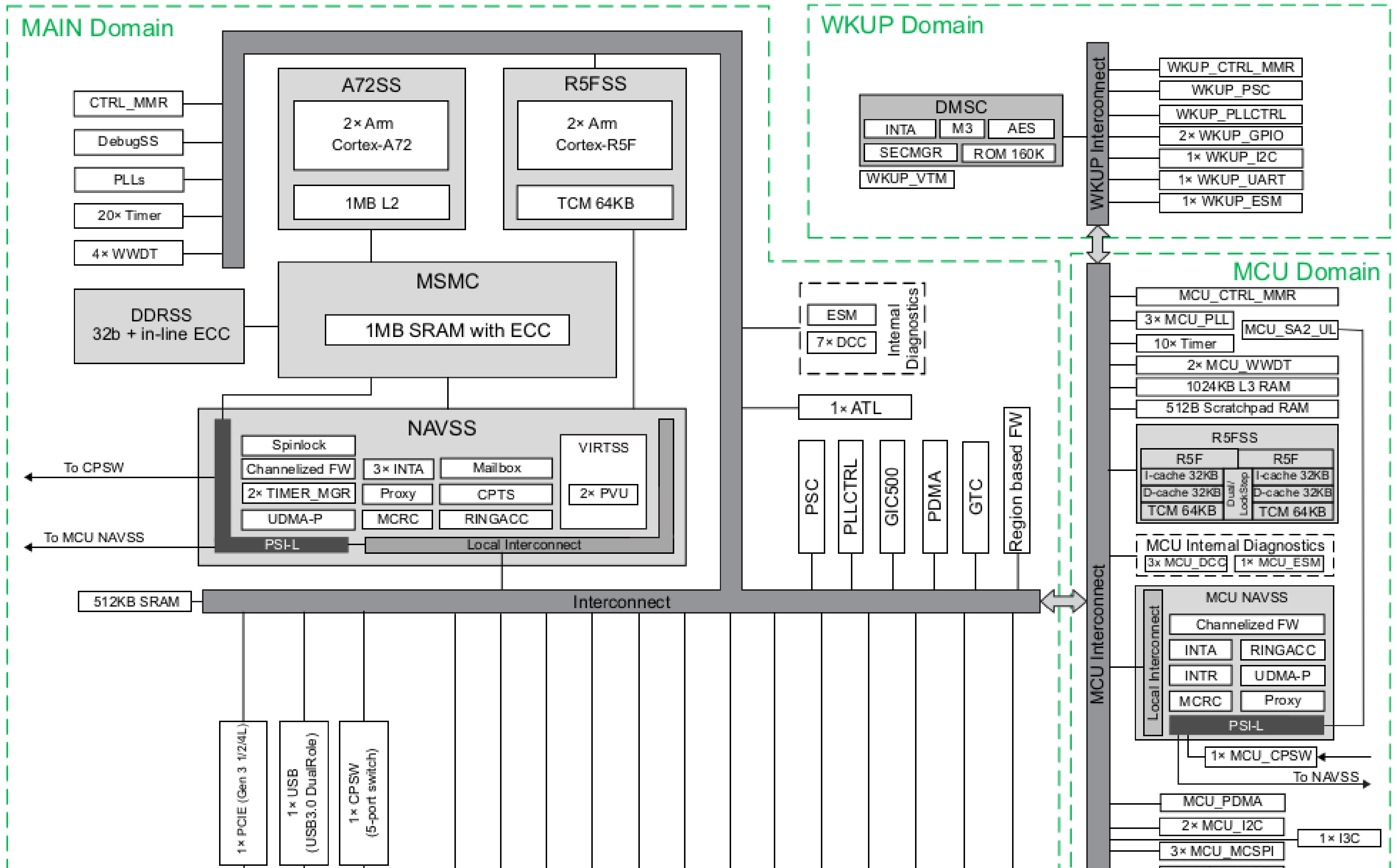
The purpose of this document is to highlight J7200 SoC integration details, SoC specific differences and deviations from MCAL Architecture Document, Module User Guide, Design Documents, Test Strategy Document, etc.

3 1.3. SoC Architecture

Reference: J7200 DRA821 Processors Silicon Revision 1.0 TRM, Revision: January 2021, J7200 DRA821 Processor SR 1.0

J7VCL device integrates MicroControllerUnit (MCU) dedicated to provide isolated processing entity. This isolated processing entity could be employed to realize an ASIL system (upto ASIL D). The block diagrams of J7VCL is as shown below, the integrated MCU is referred as MCU SS. The AUTOSAR is expected to be hosted on MCU SS (or WAKEUP/MUC of J7VCL) and other R5F in Main domain, in case of J7VCL class of devices. The MCU SS will have dual core lock-step R5F processor. The MCAL drivers provided will use peripheral within the MCU SS and shared peripherals within the SoC.

J7VCL SoCs are expected to be used in various automotive ECUs (such as front-camera ADAS, SRV, Gate Way, Head-Unit, etc...). AUTOSAR is the industry standard software architecture that allows OEM's software applications to scale to different vehicles, different SoCs, while maintaining safety and functional requirements. DRA821 SoC dedicates MCU SS to host AUTOSAR stack and provide a isolated processing entity.


MAIN Domain
CTRL_MMR
DebugSS
PLLs
20× Timer
4× WWDT
DDRSS
32b + in-line ECC
A72SS
2× Arm
Cortex-A72
1MB L2
R5FSS
2× Arm
Cortex-R5F
TCM 64KB
MSMC
1MB SRAM with ECC
NAVSS
Spinlock
Channelized FW
3× INTA
Mailbox
2× TIMER_MGR
Proxy
CPTS
UDMA-P
MCRC
RINGACC
VIRTSS
2× PVU
PSI-L
Local Interconnect
To CPSW
To MCU NAVSS
512KB SRAM
Interconnect
ESM
7× DCC
Internal Diagnostics
1× ATL
PSC
PLLCTRL
GIC500
PDMA
GTC
Region based FW
1× PCIE (Gen 3 1/2/4L)
1× USB (USB3.0 DualRole)
1× CPSW (5-port switch)
WKUP Domain
DMSC
INTA
M3
AES
SECMGR
ROM 160K
WKUP_VTM
WKUP Interconnect
WKUP_CTRL_MMR
WKUP_PSC
WKUP_PLLCTRL
2× WKUP_GPIO
1× WKUP_I2C
1× WKUP_UART
1× WKUP_ESM
MCU Domain
MCU Interconnect
MCU_CTRL_MMR
3× MCU_PLL
MCU_SA2_UL
10× Timer
2× MCU_WWDT
1024KB L3 RAM
512B Scratchpad RAM
R5FSS
R5F
R5F
I-cache 32KB
I-cache 32KB
D-cache 32KB
D-cache 32KB
TCM 64KB
TCM 64KB
Dual/LockStep
MCU Internal Diagnostics
3x MCU_DCC
1× MCU_ESM
MCU NAVSS
Local Interconnect
Channelized FW
INTA
RINGACC
INTR
UDMA-P
MCRC
Proxy
PSI-L
1× MCU_CPSW
To NAVSS
MCU_PDMA
2× MCU_I2C
1× I3C
3× MCU_MCSPI

4 1.4. MCAL Scope and HW Peripheral Details

	MCAL Driver	HW Peripheral Details	Device Specific Recommendations
1	ADC	2x ADC Instances	NA
2	DIO	7x GPIO Instances	NA
3	GPT	16x DM Timers	NA
4	WDG	12x RTI Instances	NA
5	SPI	11 McSPI Devices	Below SPI instances for TDA4x have limitations: 1. MCSPI7 from the main domain does not support master mode. 2. MCSPI17 from the main domain and MCU_MCSPI2 from MCU domain. Below SPI instances of device variants TDA4x does not support master mode and are not pinned out externally.

	MCAL Driver	HW Peripheral Details	Device Specific Recommendations
6	ICU	3x ECAP Instances	NA
7	FLS	OSPI External Flash Device: S28HS512T	NA
8	PWM (Timer)	16x DM Timers	NA
9	EPWM	6x EPWM Instances	NA
10	CDD IPC		
11	ETH	TBD	
12	MCU	TBD	NA

Below table outlines specific HW IP usage per module:

	SW Modules	HW IP
1	ADC	adc12_16ffc_10_rel.1.0.x
2	DIO	gpio_144_10_rel.1.5.x
3	WDG	rti_10_rel.0.0.x
4	CAN	mcanss_10_rel.1.1.x
5	SPI	spi_10_rel.2.11.x
6	GPT	dmtimer_dmc1ms_10_rel.1.0.x
7	PWM	dmtimer_dmc1ms_10_rel.1.0.x
8	EPWM	ehrpwm_10_rel.1.3.x
9	ICU	ecap_10_rel.1.0.x

	SW Modules	HW IP
10	FLS	ospi_10_rel.1.0.x
11	CDD IPC	ksipc_mailbox_rel.1.0.x

5 1.5. MCAL Integration Details

J7VCL MCAL is delivered as a component within Processor SDK and part of MCUSW.

J7VCL MCAL examples are dependent on an SDK and will require that proper dependencies are met.

5.1 1.5.1. Dependency:

5.1.1 **1.5.1.1. Hardware Dependency on J7200 EVM**

EVM:

Contact your FAE for documents describing the EVM

J7200 EVM NO Boot Mode / CCS:

J7200 EVM MMC/SD Boot Mode:

J7200 EVM OSPI Boot Mode:

Emulator:

J7200 EVM includes an on-board XDS110 USB emulator, which could be used with CCS. Please contact your FAE for documents describing the EVM.

An external emulator such as Spectrum Digital XDS560V2 could be used. While creating the target, please select the emulator that is being used. Refer to MCUSW User Guide for details.

5.1.2 **1.5.1.2. Compiler**

The SDK with which MCUSW is expected to be used, packages all required compiler and code generation tools required by MCUSW. The Configurator would be an exception, please refer to MCUSW User Guide for details.

MCUSW has now migrated to the TI Arm Clang compiler. Some benefits of this migration include:

- Excellent C/C++ standards support
- Improved code size over.
- Ease of use with fast compiles and expressive diagnostic messages.

5.1.3 **1.5.1.3. PDK**

"PDK" is a component within Processor SDK. Following section list the sub-components of PDK that are used / required by MCAL modules.

Please check release note that came with this release for the compatible version of PDK/SDK.

UDMA:

UDMA is used to move data between peripherals and memory.

- The Eth MCAL module relies on UDMA driver.

- SPI Driver relied on UDMA driver for DMA mode.

MCAL Example Application

- Applications rely on SCI Client to request interrupt number as resource.
- Applications rely on OSAL to register MCAL modules interrupts.
- Applications rely on UART driver to print on console.
- For MCU21 applications, please note that sciserver_testapp needs to be run on mcu1_0 core.

6 1.6. Running Examples:

6.1 1.6.1. CCS

IDE (CCS)

Code Composer Studio is an integrated development environment (IDE) that supports TI's Microcontroller and Embedded Processors portfolio.

- CCS Link
- Download

J7200

1. Supported CCS version is detailed in SDK Release Notes
2. Installation and configuration of GEL files is detailed in SDK How To

6.1.1 1.6.1.1. Load Example Binaries

1. Ensure boot mode of the EVM is configured as described in No Boot mode.
2. UART / Console for demo application logs / messages
    - J7200 EVM had 2 UART ports
    - UART port named **MCU UART** would be used when demo applications are hosted on MCU R5F (mcu 1 0)
    - UART port named **Main UART** would be used when demo applications are hosted on MAIN R5F (mcu 2 1)
3. CCS Setup & Steps to run from CCS Refer the SDK Release Notes user guide for generic test setup details and steps to run the examples/demos using CCS/SBL.
4. **Reset MCU_Cortex_R5_0_0 core**
5. In core MCU_Cortex_R5_0_0, load binary **(driver name)_app_mcu1_0_(release or debug).xer5f**

    - J7200 MCAL Binaries is available at ($PSDKRA_INSTALL_PATH)/mcusw/binary/(driver name)_app/bin/j7200_evm/
    - Some of the example applications (ipc) would have more than 1 binaries. The name of the binaries specify the core that it's intended to hosted on
6. Run example
    - Expect to see prints on CCS console or UART console. See "Setup Build Environment" in MCUSW User Guide.
7. On Core MCU 2 1
    - UART port on which prints are displayed is different, ensure to connect UART port named **UART** on the EVM
    - J7200 MCAL Binaries is available at ($PSDKRA_INSTALL_PATH)/mcusw/binary/(driver name)_app/bin/j7200_evm/(driver name)_app_**mcu2_1**_(release or debug).xer5f
    - Connect to MAIN_Cortex_R5_0_1
    - Load binaries and run

6.2 1.6.2. SBL

SD/MMC

To build the SBL binary for SD/MMC, please use the following command:

$cd ($PSDKRA_INSTALL_PATH)/pdk_jacinto_07_xx_xx/packages/ti/build $make sbl_mmcsd_img

Post compilation of SBL, the SBL binary can be found at ($PSDKRA_INSTALL_PATH)/(pdk-install-folder)/packages/ti/boot/sbl/binary/j7200_evm/(boot-media)/bin/*.tiimage

To bo able to boot from SD card copy the following to the SD card boot partition (FAT32)

- Copy SBL binary **sbl_mmcsd_img_mcu1_0_release.tiimage** as **tiboot3.bin**
- Copy the tifs.bin form **($PSDKRA_INSTALL_PATH)/(pdk-install-folder)/packages/ti/drv/sciclient/soc/V1/tifs.bin as tifs.bin in case of J721E**
- Copy the tifs.bin form **($PSDKRA_INSTALL_PATH)/(pdk-install-folder)/packages/ti/drv/sciclient/soc/V2/tifs.bin as tifs.bin in case of J7200**
- Copy the application from ($PSDKRA_INSTALL_PATH)/mcusw/binary/(driver name)_app/bin/j7200_evm)/.*appimage to the SD card boot partition as **app**
- MMC SD: Ensure The bootmode switches are configured as described in MMC/SD Boot Mode.

OSPI

Steps below highlight the steps required to program OSPI with binary image

Software Prerequisites

- Download and install the latest Uniflash version 6.1 from https://www.ti.com/tool/UNIFLASH

Board Setup for Flashing OSPI

- Configure SW3 on CP board for below values
  SW3: 1-ON, 2-OFF, 3-OFF, 4-OFF, 5-OFF, 6-OFF, 7-OFF, 8-OFF
- Configure Boot switches to 'UART' mode
  SW8: 1-OFF, 2-OFF, 3-OFF, 4-OFF, 5-OFF, 6-OFF, 7-OFF, 8-OFF
  SW9: 1-OFF, 2-ON, 3-ON, 4-ON, 5-OFF, 6-OFF, 7-OFF, 8-OFF
- Connect micro USB cable to MCU UART port (J43) and host PC – Confgure serial console application on host PC to use MCU UART port with '115200 8N1' configuration

Procedure for Flashing OSPI

1. Load the uart flash writer binary uart_j7200_evm_flash_programmer_release.tiimage @ 0th location.
   - Flash programmer is available as part of Uniflash at - (Path to Uniflash Install Directory)/processors/FlashWriter/j7200_evm
   - Goto Uniflash Install Directory and run the following commands from that path.
     Eg: dslite.bat --mode processors -c COM55 -f C:\ti\uniflash_6.1.0\processors\FlashWriter\j7200_evm\uart_j7200_evm_flash_programmer_release.tiimage -i 0
2. Make sure the character 'C' is getting displayed on the serial console. Make a note of the COM port number.
3. Close all the serial console applications on host PC, disconnect and reconnect micro USB cable connected to MCU UART port (J43)
4. Run the below command to flash the SBL to OSPI flash dslite.bat –mode processors -c (COM Port#) -f (Boot Image) -d 3 -o 0
   - COM port#: COM port number on which J7 MCU UART serial port is connected
   - Boot Image: SBL image (sbl_ospi_img_mcu1_0_release.tiimage) with full path.

Eg: dslite.bat --mode processors -c COM55 -f C:\ti\j7_evm_repo\pdk\packages\ti\boot\sbl\binary\j7200_evm\ospi\bin\sbl_ospi_img_mcu1_0_release.tiimage -d 3 -o 0

5. Run the below command to flash the system firmware to OSPI flash dslite.bat –mode processors -c (COM Port#) -f (SYSFW Image) -d 3 -o 80000
    - COM port#: COM port number on which J7 MCU UART serial port is connected
    - SYSFW Image: System firmware image with full path.
      Eg: dslite.bat --mode processors -c COM55 -f C:\ti\j7_evm_repo\pdk\packages\ti\drv\sciclient\soc\V2\tifs.bin -d 3 -o 80000
6. Run the below command to flash the app image to OSPI flash dslite.bat –mode processors -c (COM Port#) -f (App Image) -d 3 -o 100000
    - COM port#: COM port number on which J7 MCU UART serial port is connected
    - App Image: Application image with full path name.
      Eg: dslite.bat --mode processors -c COM55 -f C:
      \ti\j7_evm_repo\pdk\packages\ti\binary\udma_memcpy_testapp\bin\j7200_evm\udma_memcpy_testapp_mcu1_0_release.appimage -d 3 -o 100000
7. Run the below command to flash OSPI PHY tuning binary. In case of J7200 flash at location 3FC0000 and for J721E at 3FE0000 dslite.bat –mode processors -c (COM Port#) -f (nor_spi_patterns.bin) -d 3 -o 3FE0000
    - COM port#: COM port number on which J7 MCU UART serial port is connected
    - OSPI Phy Bin : nor_spi_patterns.bin.
      Eg: dslite.bat --mode processors -c COM55 -f C:\ti\j7_evm_repo\pdk\packages\ti\board\src\flash\nor\ospi\nor_spi_patterns.bin -d 3 -o 3FE0000

Note : In Windows, during flashing if you get any error "Unknown response from the target", please disconnect and reconnect micro USB cable and then try to flash again. **Note** : For J7200 platform, during flashing please select from uniflash_6.1.0/processors/FlashWriter/j7200_evm/uart_j7200_evm_flash_programmer_release.tiimage.

Procedure for Verifying OSPI Boot

- After successful flashing, power OFF the board and configure it for OSPI boot.
- Connect micro USB cable to MCU UART port (J43) and host PC – Confgure serial console application on host PC to use MCU UART port with '115200 8N1' configuration
- Power ON the board and confirm the boot logs on serial console

Mode	Switch Settings
UART	SW8: 0000_0000, SW9: 0111_0000
OSPI (J721E)	SW8: 0000_0000, SW9: 0100_0000
OSPI (J7200)	SW8: 1000_0010, SW9: 0011_0000
OSPI(J721S2)	SW8: 0000_1010, SW9: 0110_0000

7 1.7. MCAL Module User Guide Info:

7.1 **1.7.1. ADC**

Sample Log

J7200

```
ADC_APP: Sample Application - STARTS !!!
ADC_APP: Variant - Post Build being used !!!

ADC_APP: GROUPs 1: HWUNIT 1: LOOP COUNT 5: NUM STREAMS 1:!!!

ADC_APP:  ADC MCAL Version Info
ADC_APP: ---------------------
ADC_APP:  Vendor ID           : 44
ADC_APP:  Module ID           : 123
ADC_APP:  SW Major Version    : 2
ADC_APP:  SW Minor Version    : 0
ADC_APP:  SW Patch Version    : 0

ADC_APP: ADC INTERNAL DIAGNOSTIC DEBUG MODE Test
ADC_APP: ADC_UNIT_0 Internal Diagnostic Debug Mode - Enabled

 Result Buffer Content
 ---------------------
 ADC Group 0 Log:
 ----------------
 Channel    Stream    HW_CH       ADC Value    Volt
 ----------------------------------------------------
 0          0         ADC_IN0     0x00000ff6  1796 mV

 Read Buffer Content
 --------------------
 ADC Group 0 Log:
 ----------------
 Channel    Set Idx   HW_CH       ADC Value    Volt
 ----------------------------------------------------
 0          0         ADC_IN0     0x00000ff6  1796 mV
 0          1         ADC_IN0     0x00000ff6  1796 mV
 0          2         ADC_IN0     0x00000ff6  1796 mV
 0          3         ADC_IN0     0x00000ff6  1796 mV
 0          4         ADC_IN0     0x00000ff6  1796 mV

ADC_APP: ADC_UNIT_0 Internal Diagnostic Debug Mode - Disabled

ADC_APP: ADC Functional Operation Test

 Result Buffer Content
 ---------------------
 ADC Group 0 Log:
```

```
 ----------------
 Channel    Stream    HW_CH       ADC Value    Volt
 ----------------------------------------------------
 0          0         ADC_IN0     0x00000e4d  1609 mV

 Read Buffer Content
 -------------------
 ADC Group 0 Log:
 ----------------
 Channel    Set Idx   HW_CH       ADC Value    Volt
 ----------------------------------------------------
 0          0         ADC_IN0     0x00000d1a  1474 mV
 0          1         ADC_IN0     0x00000e50  1610 mV
 0          2         ADC_IN0     0x00000e57  1613 mV
 0          3         ADC_IN0     0x00000e53  1611 mV
 0          4         ADC_IN0     0x00000e4d  1609 mV
 Warning: ADC Group 0 values are out of range!!

ADC_APP: ADC Stack Usage: 2812 bytes
ADC_APP: ADC Test Passed!!!
ADC_APP: Sample Application - DONE !!!
```

7.2 **1.7.2. DIO**

Pin Mapping J7200

In case of J7200, each Dio instance supports 9 banks of 16 DIO signals/pins or channels (2 in WKUP and 4 in Main domain). Please note in each instance there are some pins that are not pinned out and are reserved.
The mapping of pins in the different instances is shown in the following table. For this implementation the absolute numbering of channel ids starts from the Dio instance in the wakeup domain. Please refer TRM for more details.

Instance	**Pin Number**	**ChannelId**	**Port ID**	**Available/Not Available**
WKUP_GPIO 0	0	0	0	Available
:	:	:	:	Available
WKUP_GPIO 0	31	31	0	Available
WKUP_GPIO 0	32	32	1	Available
:	:	:	:	Available
WKUP_GPIO 0	63	63	1	Available

Instance	Pin Number	ChannelId	Port ID	Available/Not Available
WKUP_GPIO 0	64	64	2	Available
:	:	:	:	Available
WKUP_GPIO 0	83	83	2	Available
WKUP_GPIO 0	84	x	x	**Not Available**
:	:	:	:	**Not Available**
WKUP_GPIO 0	143	x	x	**Not Available**
WKUP_GPIO 1	0	144	3	Available
:	:	:	:	Available
WKUP_GPIO 1	31	175	3	Available

Instance	**Pin Number**	**ChannelId**	**Port ID**	**Available/Not Available**
WKUP_GPIO 1	32	176	4	Available
:	:	:	:	Available
WKUP_GPIO 1	63	207	4	Available
WKUP_GPIO 1	64	208	5	Available
:	:	:	:	Available
WKUP_GPIO 1	83	227	5	Available
WKUP_GPIO 1	84	x	x	**Not Available**
:	:	:	:	**Not Available**
WKUP_GPIO 1	143	x	x	**Not Available**

Instance	Pin Number	ChannelId	Port ID	Available/Not Available
GPIO 0	0	288	6	Available
:	:	:	:	Available
GPIO 0	31	319	6	Available
GPIO 0	32	320	7	Available
:	:	:	:	Available
GPIO 0	63	351	7	Available
GPIO 0	64	352	8	Available
:	:	:	:	Available
GPIO 0	95	383	8	Available

Instance	Pin Number	ChannelId	Port ID	Available/Not Available
GPIO 0	96	384	9	Available
:	:	:	:	Available
GPIO 0	127	415	9	Available
GPIO 0	128	x	x	**Not Available**
:	:	:	:	**Not Available**
GPIO 0	143	x	x	**Not Available**
GPIO 2	0	576	12	Available
:	:	:	:	Available
GPIO 2	31	607	12	Available

Instance	Pin Number	ChannelId	Port ID	Available/Not Available
GPIO 2	32	608	13	Available
:	:	:	:	Available
GPIO 2	63	639	13	Available
GPIO 2	64	640	14	Available
:	:	:	:	Available
GPIO 2	95	671	14	Available
GPIO 2	96	672	15	Available
:	:	:	:	Available
GPIO 2	127	703	15	Available

Instance	Pin Number	ChannelId	Port ID	Available/Not Available
GPIO 2	128	x	x	**Not Available**
:	:	:	:	**Not Available**
GPIO 2	143	x	x	**Not Available**
GPIO 4	0	864	18	Available
:	:	:	:	Available
GPIO 4	31	895	18	Available
GPIO 4	32	896	19	Available
:	:	:	:	Available
GPIO 4	63	927	19	Available

Instance	Pin Number	ChannelId	Port ID	Available/Not Available
GPIO 4	64	928	20	Available
:	:	:	:	Available
GPIO 4	95	959	20	Available
GPIO 4	96	960	21	Available
:	:	:	:	Available
GPIO 4	127	991	21	Available
GPIO 4	128	x	x	**Not Available**
:	:	:	:	**Not Available**
GPIO 4	143	x	x	**Not Available**

Instance	Pin Number	ChannelId	Port ID	Available/Not Available
GPIO 6	0	1152	25	Available
:	:	:	:	Available
GPIO 6	31	1183	25	Available
GPIO 6	32	1184	26	Available
:	:	:	:	Available
GPIO 6	63	1215	26	Available
GPIO 6	64	1216	27	Available
:	:	:	:	Available
GPIO 6	95	1247	27	Available

Instance	Pin Number	ChannelId	Port ID	Available/Not Available
GPIO 6	96	1248	28	Available
:	:	:	:	Available
GPIO 6	127	1279	28	Available
GPIO 6	128	x	x	**Not Available**
:	:	:	:	**Not Available**
GPIO 6	143	x	x	**Not Available**

Sample Log

J7200

```
Sample Application - STARTS !!!
DIO MCAL Version Info
---------------------
Vendor ID          : 44
Module ID          : 120
SW Major Version   : 2
SW Minor Version   : 0
SW Patch Version   : 0

Test A. Write and Read Channel
------------------------------
Channels written
Channel read DIO_PinLevel[0] = 0
Channel read DIO_PinLevel[1] = 1
Channel read DIO_PinLevel[2] = 0
DIO Service API Read-back Channel Succeeds !!!
Main Domain Channels written
Channel read DIO_PinLevel[0] = 0
Channel read DIO_PinLevel[1] = 1
Channel read DIO_PinLevel[2] = 0
DIO Service API Read-back Channel Main Domain Succeeds !!!

DIO Test A :Service API: Write/Read Channel completed
```

```
Test B. Write and Read Channel Group
------------------------------------
DIO Service Read/Write Channel Group Read-back Succeeds !!!

DIO Test B : Service API : Write/Read Channel Group completed

Test C. Write and Read Port
---------------------------
DIO Service API Read-Back Port succeeds !!!
DIO Service API Read-Back Port Main Domain succeeds !!!

DIO Test C : Service API: Write/Read Port completed

Test D. Flip Channel
----------------------
Pin Value Before Flip: 0
Pin Value After Flip: 1

DIO Test D : Service API: Flip Channel completed
Pin Value Before Flip: 1
Pin Value After Flip: 0

DIO Test D : Service API: Flip Channel Main Domain completed
```

```
Test E. Dio_RegisterReadback
---------------------
DIO register readback compare Passed!!
DIO register(WKUP_GPIO0_B01_Ch8) direction is Input !!
DIO register readback compare Main Domain Passed!!
DIO register direction(GPIO0_B01_Ch0) is Input !!
DIO register(GPIO0_B01_Ch22) direction is Output !!

DIO Test E : Service API: Register Read-back completed
DIO Stack Usage: 2764 bytes

---------------------------------

DioApp: Sample Application - Completes successfully !!!
```

7.3 1.7.3. WDG

Sample Log

J7200

User Input : To generate an interrupt (via ESM)

```
Starting WDG test  !!!
WDG MCAL Version Info
---------------------
Vendor ID           : 44
Module ID           : 102
SW Major Version    : 2
SW Minor Version    : 0
SW Patch Version    : 0

On Expiry of WDG timeout Please enter :
0 To generate an interrupt (via ESM)
1 To drive MCU_SAFETY_ERROR Pin low (via ESM)
0
Variant - Pre Compile being used !!!
WDG timeout is configured for 2000 millisecs
Number of times WDG will be serviced : 5
WDG ESM Interrupt will be generated after 2000 * 5 i.e 10000 millisecs
WDG Elapsed and generated an event to ESM
ESM has generated an Interrupt
WDG App Completed as expected!!!
Please refer user guide on usage details
WDG Stack Usage : 2764 bytes
WDG Test Passed!!!
```

7.4 **1.7.4. SPI**

Spi Instance ID mapping and ISR mapping

11 spi instances are supported by this driver implementation (8 instances in Main Domain & 3 in MCU Domain in case of J721E/J7200/J721S2). The following table lists the mapping between instance of MCSPI and SpiChannelId of the configurator

Spi HwUnit Id	Spi Instance	Associated ISR
0	MCU SPI 0	Spi_IrqUnitMcuMcspi0TxRx
1	MCU SPI 1	Spi_IrqUnitMcuMcspi1TxRx
2	MCU SPI 2	Spi_IrqUnitMcuMcspi2TxRx
3	SPI 0	Spi_IrqUnitMcspi0TxRx
4	SPI 1	Spi_IrqUnitMcspi1TxRx
5	SPI 2	Spi_IrqUnitMcspi2TxRx

Spi HwUnit Id	Spi Instance	Associated ISR
6	SPI 3	Spi_IrqUnitMcspi3TxRx
7	SPI 4	Spi_IrqUnitMcspi4TxRx
8	SPI 5	Spi_IrqUnitMcspi5TxRx
9	SPI 6	Spi_IrqUnitMcspi6TxRx
10	SPI 7	Spi_IrqUnitMcspi7TxRx

Sample Log

J7200

```
SPI_APP: Variant - Post Build being used !!!
SPI_APP: Interrupt List Completed !!!
SPI_APP: Sample Application - STARTS !!!

SPI MCAL Version Info
---------------------
Vendor ID           : 44
```

```
Module ID            : 83
SW Major Version     : 2
SW Minor Version     : 0
SW Patch Version     : 0

SPI_APP: CH 1: JOBS 1: SEQ 1: Max HWUNIT 1: NUM OF WORDS 10000:!!!
SPI_APP: SPI Async transmit in progress!!
SPI_APP: SPI Hwunit 0 configuration Register Readback values
SPI_APP: MCSPI_HL_REV                     : 0x40301a0b
SPI_APP: MCSPI_HL_HWINFO                  : 0x9
SPI_APP: MCSPI_HL_SYSCONFIG               : 0x4
SPI_APP: MCSPI_REVISION                           : 0x2b
SPI_APP: MCSPI_SYSSTATUS                  : 0x1
SPI_APP: MCSPI_SYST                       : 0x0
SPI_APP: MCSPI_MODULCTRL                  : 0x1
SPI_APP: MCSPI_SYSCONFIG                  : 0x308
SPI_APP: MCSPI_CH0CONF                    : 0x20050f80
SPI_APP: MCSPI_CH1CONF                    : 0x60000
SPI_APP: MCSPI_CH2CONF                    : 0x60000
SPI_APP: MCSPI_CH3CONF                    : 0x60000
SPI_APP: MCSPI_IRQENABLE                  : 0x0
SPI_APP: SPI Stack Usage: 2844 bytes
SPI_APP: SPI Loopback Test Passed!!!
```

7.5 1.7.5. CAN

Sample Log

J7200

```
CAN_APP: Sample Application - STARTS !!!
CAN_APP: Variant - Pre Compile being used !!!
CAN_APP: Successfully Enabled CAN Transceiver MCU MCAN0!!!
CAN_APP: Successfully Enabled CAN Transceiver MCU MCAN1!!!
CAN_APP: Message Id Received 400000a0 Message Length is 64
CAN_APP: Can Controller Instance MCAN 0          Internal LoopBack Mode Test Passed
CAN_APP: Message Id Received c00000b0 Message Length is 64
CAN_APP: Can Controller Instance MCAN 1          Internal LoopBack Mode Test Passed
CAN_APP: Message Id Received 400000c0 Message Length is 64
CAN_APP: Can Controller Instance MCAN 2          Internal LoopBack Mode Test Passed
```

```
CAN_APP: Safety Diagnostic API functionality test for controller MCAN 0
CAN_APP: Can_EnableIntr() API test
CAN_APP: Successfully Enabled TOO interrupt
CAN_APP: Successfully Enabled PEA interrupt
CAN_APP: Can_GetIntrStatus() API test
CAN_APP: No error event occured
CAN_APP: Can_ClearIntrStatus() API test
CAN_APP: Interrupt Status Clear Successful
CAN_APP: Can_DisableIntr() API test
CAN_APP: Successfully Disable TOO interrupt
CAN_APP: Successfully Disable PEA interrupt
CAN_APP: Can_RegisterReadback() API test
CAN_APP: Can_RegisterReadback Successful
```

```
CAN_APP: Safety Diagnostic API functionality test for controller MCAN 1
CAN_APP: Can_EnableIntr() API test
CAN_APP: Successfully Enabled TOO interrupt
CAN_APP: Successfully Enabled PEA interrupt
CAN_APP: Can_GetIntrStatus() API test
CAN_APP: No error event occured
CAN_APP: Can_ClearIntrStatus() API test
CAN_APP: Interrupt Status Clear Successful
CAN_APP: Can_DisableIntr() API test
CAN_APP: Successfully Disable TOO interrupt
CAN_APP: Successfully Disable PEA interrupt
CAN_APP: Can_RegisterReadback() API test
CAN_APP: Can_RegisterReadback Successful
```

```
CAN_APP: Safety Diagnostic API functionality test for controller MCAN 2
CAN_APP: Can_EnableIntr() API test
CAN_APP: Successfully Enabled TOO interrupt
CAN_APP: Successfully Enabled PEA interrupt
CAN_APP: Can_GetIntrStatus() API test
CAN_APP: No error event occured
CAN_APP: Can_ClearIntrStatus() API test
CAN_APP: Interrupt Status Clear Successful
CAN_APP: Can_DisableIntr() API test
CAN_APP: Successfully Disable TOO interrupt
CAN_APP: Successfully Disable PEA interrupt
CAN_APP: Can_RegisterReadback() API test
CAN_APP: Can_RegisterReadback Successful
CAN Stack Usage: 2836 bytes
CAN_APP: CAN Test Passed!!!
```

7.6 1.7.6. GPT

Sample Log

J7200

```
GPT_APP: Building Interrupt List !!!
GPT_APP: Variant - Post Build being used !!!
GPT_APP: Interrupt List Completed !!!

GPT_APP: Sample Application - STARTS !!!
GPT_APP: Variant - Post Build being used !!!

GPT_APP GPT MCAL Version Info
GPT_APP---------------------
GPT_APP Vendor ID          : 44
GPT_APP Module ID          : 100
GPT_APP SW Major Version   : 2
GPT_APP SW Minor Version   : 0
GPT_APP SW Patch Version   : 0


GPT_APP------------------------------------------
GPT_APP: GPT Channel 11 configuration register values
GPT_APP TIMER_TIDR        : 0x50003900
GPT_APP TIMER_TTGR        : 0xffffffff
GPT_APP TIMER_TSICR       : 0x0
GPT_APP TIMER_TIOCP_CFG   : 0xa
GPT_APP TIMER_TCLR        : 0x0
----------------------------------------
GPT_APP: Running GPT Test for channel 11
----------------------------------------
GPT_APP: Enabled notification for channel [11]
GPT_APP: Started Timer Channel [11]
GPT_APP: Elapsed Time Value = adb42
GPT_APP: Wait for notification(approx. 6 seconds)
GPT_APP: GPT Notification received for channel 11 !!!

GPT_APP: Disable channel notification for this channel
```

```
GPT_APP: Stopped for channel 11
GPT_APP: Enable wakeup for this channel
GPT_APP: Started timer channel [11]
GPT_APP: check if this channel is wakeup source for any wakeup event
 EcuM : Wakeup event received for wakeupSource =0 !!!
GPT_APP: Woken up for channel [11]
GPT_APP: Stop timer
GPT_APP: GPT example passed for channel =11 !!!

GPT_APP-----------------------------------------
GPT_APP: GPT Channel 6 configuration register values
GPT_APP TIMER_TIDR        : 0x50003900
GPT_APP TIMER_TTGR        : 0xffffffff
GPT_APP TIMER_TSICR       : 0x0
GPT_APP TIMER_TIOCP_CFG   : 0xa
GPT_APP TIMER_TCLR        : 0x0
-----------------------------------------
GPT_APP: Running GPT Test for channel 6
-----------------------------------------
GPT_APP: Enabled notification for channel [6]
GPT_APP: Started Timer Channel [6]
GPT_APP: Elapsed Time Value = adafa
GPT_APP: Wait for notification(approx. 6 seconds)
GPT_APP: GPT Notification received for channel 6 !!!
```

```
GPT_APP: Disable channel notification for this channel
GPT_APP: Wait till timer overflows, no notification should be received
GPT_APP: Time Elapsed Value = 0x3692e8
GPT_APP: Time Remaining Value = 0x8ae36fa
GPT_APP: Waiting for timer to overflow
GPT_APP: Overflow happened no notification received
```

```
GPT_APP: Stopped for channel 6
GPT_APP: Enable wakeup for this channel
GPT_APP: Started timer channel [6]
GPT_APP: check if this channel is wakeup source for any wakeup event
 EcuM : Wakeup event received for wakeupSource =0 !!!
GPT_APP: Woken up for channel [6]
GPT_APP: Stop timer
GPT_APP: GPT example passed for channel =6 !!!

GPT_APP----------------------------------------
GPT_APP: GPT Channel 9 configuration register values
GPT_APP TIMER_TIDR       : 0x50003900
GPT_APP TIMER_TTGR       : 0xffffffff
GPT_APP TIMER_TSICR      : 0x0
GPT_APP TIMER_TIOCP_CFG  : 0xa
GPT_APP TIMER_TCLR       : 0x0
----------------------------------------
GPT_APP: Running GPT Test for channel 9
----------------------------------------
GPT_APP: Enabled notification for channel [9]
GPT_APP: Started Timer Channel [9]
GPT_APP: Elapsed Time Value = adafa
GPT_APP: Wait for notification(approx. 6 seconds)
GPT_APP: GPT Notification received for channel 9 !!!
```

```
GPT_APP: Disable channel notification for this channel
GPT_APP: Wait till timer overflows, no notification should be received
GPT_APP: Time Elapsed Value = 0x3692b8
GPT_APP: Time Remaining Value = 0x8ae3757
GPT_APP: Waiting for timer to overflow
GPT_APP: Overflow happened no notification received
```

```
GPT_APP: Stopped for channel 9
GPT_APP: Enable wakeup for this channel
GPT_APP: Started timer channel [9]
GPT_APP: check if this channel is wakeup source for any wakeup event
 EcuM : Wakeup event received for wakeupSource =0 !!!
GPT_APP: Woken up for channel [9]
GPT_APP: Stop timer
GPT_APP: GPT example passed for channel =9 !!!

GPT_APP----------------------------------------
GPT_APP: GPT Channel 15 configuration register values
GPT_APP TIMER_TIDR        : 0x50003900
GPT_APP TIMER_TTGR        : 0xffffffff
GPT_APP TIMER_TSICR       : 0x0
GPT_APP TIMER_TIOCP_CFG   : 0xa
GPT_APP TIMER_TCLR        : 0x0
----------------------------------------
GPT_APP: Running GPT Test for channel 15
----------------------------------------
GPT_APP: Enabled notification for channel [15]
GPT_APP: Started Timer Channel [15]
GPT_APP: Elapsed Time Value = b261e
GPT_APP: Wait for notification(approx. 6 seconds)
GPT_APP: GPT Notification received for channel 15 !!!
```

```
GPT_APP: Disable channel notification for this channel
GPT_APP: Wait till timer overflows, no notification should be received
GPT_APP: Time Elapsed Value = 0x3728f1
GPT_APP: Time Remaining Value = 0x8ada11b
GPT_APP: Waiting for timer to overflow
GPT_APP: Overflow happened no notification received
```

```
GPT_APP: Stopped for channel 15
GPT_APP: Enable wakeup for this channel
GPT_APP: Started timer channel [15]
GPT_APP: check if this channel is wakeup source for any wakeup event
 EcuM : Wakeup event received for wakeupSource =0 !!!
GPT_APP: Woken up for channel [15]
GPT_APP: Stop timer
GPT_APP: GPT example passed for channel =15 !!!

GPT_APP----------------------------------------
GPT_APP: GPT Channel 17 configuration register values
GPT_APP TIMER_TIDR        : 0x50003900
GPT_APP TIMER_TTGR        : 0xffffffff
GPT_APP TIMER_TSICR       : 0x0
GPT_APP TIMER_TIOCP_CFG   : 0xa
GPT_APP TIMER_TCLR        : 0x0
----------------------------------------
GPT_APP: Running GPT Test for channel 17
----------------------------------------
GPT_APP: Enabled notification for channel [17]
GPT_APP: Started Timer Channel [17]
GPT_APP: Elapsed Time Value = b261e
GPT_APP: Wait for notification(approx. 6 seconds)
GPT_APP: GPT Notification received for channel 18 !!!
```

```
GPT_APP: Disable channel notification for this channel
GPT_APP: Wait till timer overflows, no notification should be received
GPT_APP: Time Elapsed Value = 0x372903
GPT_APP: Time Remaining Value = 0x8ada109
GPT_APP: Waiting for timer to overflow
GPT_APP: Overflow happened no notification received

GPT_APP: Stopped for channel 17
GPT_APP: Enable wakeup for this channel
GPT_APP: Started timer channel [17]
GPT_APP: check if this channel is wakeup source for any wakeup event
 EcuM : Wakeup event received for wakeupSource =0 !!!
GPT_APP: Woken up for channel [17]
GPT_APP: Stop timer
GPT_APP: GPT example passed for channel =17 !!!
GPT_APP: GPT example Completed !!!
GPT_APP: GPT Stack Usage 2864 bytes
GPT_APP: GPT Test Passed!!!
```

7.7 1.7.7. PWM

Example Application

PwmApp_Gpt

J7200 EVM

The timer pins J5F Pin F15 can be probed to check the
output PWM signals,

PwmApp_Epwm

The ePWM in Main domain test pin 11 or 2 on GESI board (A or B)on J7200 EVM can be probed to check the output PWM signals. These pins will change based on the EVM used.

7.7.1 1.7.7.1. PWM_GPT

Sample Log

J7200

```
PWM_APP_GPT: Sample Application - STARTS !!!
PWM_APP_GPT: Variant - Pre Compile being used !!!

PWM_APP_GPT: PWM CHANNEL 10 configuration Register Readback values
PWM_APP_GPT: PWM_REV                : 0x50003900
PWM_APP_GPT: PWM_TTGR               : 0xffffffff
PWM_APP_GPT: PWM_TMSYNCTRL          : 0x0
PWM_APP_GPT: PWM Channel Initialized
PWM_APP_GPT: PWM Duty cycle: 50 Percent, PWM Period: 1 sec
PWM_APP_GPT: Probe TIMER in Main domain(J5F Pin F15)in J7200 EVM
```

```
PWM_APP_GPT: This example waits for 10 seconds please probe

PWM_APP_GPT: Changing the Duty cycle from 50 Percent to 80 Percent
PWM_APP_GPT: This example waits for 10 seconds Please probe

PWM_APP_GPT: Setting Output to Idle state
PWM_APP_GPT: This example waits for 10 seconds Please probe

PWM_APP_GPT: Changing PWM Period from 1s to 500 ms and Duty Cycle to 50%
PWM_APP_GPT: This example waits for 10 seconds Please probe

PWM_APP_GPT: This example waits for 10 seconds please probe
PWM_APP_GPT: Pwm Isr Count: 20

 WM_APP_GPT: Disabling Notifications for PWM channel
PWM_APP_GPT: PWM De-initialized
PWM_APP_GPT: Pwm Stack Usage 2756 bytes
PWM_APP_GPT: PWM Test Passed!!!
```

7.7.2 **1.7.7.2. PWM_EPWM**

Sample Log

J7200

```
 PWM_APP_EPWM: Sample Application - STARTS !!!
PWM_APP_EPWM: EPWM being used with Channel # 1!!!
PWM_APP_EPWM: Variant - Pre Compile being used !!!

PWM_APP_EPWM: PWM Channel Initialized
PWM_APP_EPWM: PWM Duty cycle: 60 Percent, 1000Hz
PWM_APP_EPWM: PWM CHANNEL 1 configuration Register Readback values
PWM_APP_EPWM: EPWM_TBCTL                   : 0x8032
PWM_APP_EPWM: EPWM_TBPHS                   : 0x0
PWM_APP_EPWM: EPWM_TBCNT            : 0x4b7f
PWM_APP_EPWM: EPWM_AQCTLA           : 0xf2
PWM_APP_EPWM: EPWM_AQCTLB                  : 0xf02
PWM_APP_EPWM: EPWM_DBCTL            : 0x0
PWM_APP_EPWM: EPWM_TZSEL                   : 0x0
PWM_APP_EPWM: EPWM_TZCTL                   : 0x0
PWM_APP_EPWM: EPWM_PCCTL            : 0x0
PWM_APP_EPWM: EPWM_TBSTS            : 0x1
PWM_APP_EPWM: EPWM_TBPRD            : 0xf424
PWM_APP_EPWM: EPWM_AQSFRC           : 0xd2
PWM_APP_EPWM: EPWM_AQSCFRC          : 0x0
PWM_APP_EPWM: EPWM_Aqsfrc           : 0x0
PWM_APP_EPWM: EPWM_Aqsfrc           : 0x0
PWM_APP_EPWM: EPWM_Aqsfrc           : 0x0
PWM_APP_EPWM: Probe EPWM in Main domain(Test Connector Pin 11 or 2 on GESI Board (A or B) )in J721E EVM and J7200 EVM
```

```
PWM_APP_EPWM: This example waits for 30 seconds please probe

PWM_APP_EPWM: Changing the Duty cycle from to 60 to 80 percent
PWM_APP_EPWM: This example waits for 30 seconds Please probe

PWM_APP_EPWM: Changing the Duty cycle to 100 Percent
PWM_APP_EPWM: This example waits for 10 seconds Please probe

PWM_APP_EPWM: Setting Output to Idle state
PWM_APP_EPWM: This example waits for 10 seconds Please probe

PWM_APP_EPWM: Changing Frequency from 1kHz to 10KHz and Duty Cycle to 50%
PWM_APP_EPWM: period is set to 6250
PWM_APP_EPWM: This app again waits for 30 seconds please probe
PWM_APP_EPWM: Pwm Isr Count: 300000
PWM_APP_EPWM: App Run time: 8460830 micro secs

 WM_APP_EPWM: Disabling Notifications for PWM channel
PWM_APP_EPWM: Changing Frequency from 10kHz to 1.25MHz and Duty Cycle to 40.5%
PWM_APP_EPWM: period is set to 50
PWM_APP_EPWM: This app again waits for 30 seconds     please probe
PWM_APP_EPWM: Pwm Stack Usage 2772 bytes
PWM_APP_EPWM: PWM Test Passed!!!
```

7.8 1.7.8. CDDIPC

Interrupt to ISR mapping

The following table lists the interrupt details, required for applications to register ISR to receive interrupt on the core that hosts MCAL/IPC

J721E / J7200 / J721S2:

CDD IPC Example on MCU 10 (deprecated because baremetal IPC app cant be supported on MCU 10)
Please note the SCI Client / DMSC Firmware API are invoked to route interrupt to MCU 10 (via routers or no routers)

Note: Remote core Application on MPU 1_0 Core is deprecated because FreeRTOS is not supported on that core.

Host Core	Remote Core	Cluster	User	Int No on MCU 10	Comments
MCU 1 0	MCU 2 0	7	0	377	ISR **Cdd_IpcIrqMbxFromMcu_20**
MCU 1 0	MCU 2 1	7	0	377	ISR **Cdd_IpcIrqMbxFromMcu_21**

J721E / J7200 / J721S2: CDD IPC Example on MCU 21

Host Core	Remote Core	Cluster	User	Int No on MCU 10	Comments
MCU 2 1	MCU 2 0	7	0	377	ISR **Cdd_IpcIrqMbxFromMcu_20**

J721E: CDD IPC Example for LINUX host (MPU 10)

Host Core	Remote Core	Cluster	User	Int No on MCU 10	Comments
MCU 2 1	MPU 1 0	7	0	248	ISR **Cdd_IpcIrqMbxFromMpu_10**

Running the example application via CCS

J721E / J7200 / J721S2

- **MCU 2 1**

The steps below allows one to run example application on J7200 EVM

Steps to run

MCU 2 1 with Linux Host

MCU 1 0 with Linux Host

Sample Log

J7200 - MCU 1 0 Linux communication

J7200 - MCU 2 1 Linux communication

J7200 - MCU 2 1

```
CDD_IPC_APP :
CDD_IPC_APP : Sample Application - STARTS !!!

CDD_IPC_APP : CDD IPC MCAL Version Info
CDD_IPC_APP :---------------------
CDD_IPC_APP : Vendor ID         : 44
CDD_IPC_APP : Module ID         : 255
CDD_IPC_APP : SW Major Version    : 2
CDD_IPC_APP : SW Minor Version    : 0
CDD_IPC_APP : SW Patch Version    : 0

CDD_IPC_APP :
CDD_IPC_APP : Sample Application - STARTS !!!
CDD_IPC_APP : Received ti.ipc4.ping-pong as ctrl MSG from MCU 2 0
CDD_IPC_APP : Received ping 0 Iteration 10 from MCU 2 0
CDD_IPC_APP : Received ping 1 Iteration 9 from MCU 2 0
CDD_IPC_APP : Received ping 2 Iteration 8 from MCU 2 0
CDD_IPC_APP : Received ping 3 Iteration 7 from MCU 2 0
CDD_IPC_APP : Received ping 4 Iteration 6 from MCU 2 0
CDD_IPC_APP : Received ping 5 Iteration 5 from MCU 2 0
CDD_IPC_APP : Received ping 6 Iteration 4 from MCU 2 0
CDD_IPC_APP : Received ping 7 Iteration 3 from MCU 2 0
CDD_IPC_APP : Received ping 8 Iteration 2 from MCU 2 0
CDD_IPC_APP : Received ping 9 Iteration 1 from MCU 2 0

CDD_IPC_APP :-----------------------------------------
CDD_IPC_APP :: IPC Channel 3 configuration register values
CDD_IPC_APP : MAILBOX_REVISION        : 0x66fc8900
CDD_IPC_APP : MAILBOX_SYSCONFIG       : 0x0
CDD_IPC_APP : MAILBOX_IRQ_EOI         : 0x0
CDD_IPC_APP : MAILBOX_MESSAGE         : 0x0
CDD_IPC_APP : MAILBOX_FIFOSTATUS      : 0x0
CDD_IPC_APP : MAILBOX_MSGSTATUS       : 0x0
CDD_IPC_APP : MAILBOX_IRQSTATUS_RAW    : 0xaaaaaaaa
CDD_IPC_APP : MAILBOX_IRQSTATUS_CLR    : 0x0
CDD_IPC_APP : MAILBOX_IRQENABLE_CLR    : 0x400
```

7.9 1.7.9. FLS

Constraints:

Feature Combination Constraints-

- If operating in XIP mode, dacEnable has to be enabled, and Interrupt mode has to be set to STD_OFF. For J7200-EVM, PHY mode has to be disabled for XIP enable flag to work.
- On J7200, write and erase is not functional in DAC mode. Only Read is possible in DAC mode. Please use INDAC mode for writing and erasing flash for J7200.

Example Log for J7200

FLS_APP_DAC

```
  FLS_APP_DAC:
-------------------FLS Sample Application - STARTS !!! ------------------
FLS_APP_DAC:
 Running on J7 VCL
FLS_APP_DAC: FLS spi_test Initiating and Starting.
FLS_APP_DAC: Variant - Pre Compile being used !!!
FLS_APP_DAC: Configuring Clocks.
FLS_APP_DAC: Clock Configured at 166666666Hz

FLS MCAL Version Info
---------------------
```

```
Vendor ID         : 44
Module ID         : 92
SW Major Version  : 2
SW Minor Version  : 0
SW Patch Version  : 0

FLS_APP_DAC: DATA SIZE TEST is 0x100000
FLS_APP_DAC: Offset is 0x0
FLS_APP_DAC: Reading
FLS_APP_DAC: Job Processing in Progress.
FLS_APP_DAC: Job Ends: SUCCESS
FLS_APP_DAC:
 Read 1048576 bytes at transfer rate 68451 Kbps
FLS_APP_DAC: Reading
FLS_APP_DAC: Job Processing in Progress.
FLS_APP_DAC: Job Ends: SUCCESS
FLS_APP_DAC:
 Read 1048576 bytes at transfer rate 71593 Kbps
FLS_APP_DAC: DONE!
```

FLS_APP_INDAC

```
FLS_APP_INDAC:
------------------FLS Sample Application - STARTS !!! ------------------
FLS_APP_INDAC: FLS spi_test Initiating and Starting.
FLS_APP_INDAC: Variant - Pre Compile being used !!!
FLS_APP_INDAC: Configuring Clocks.
FLS_APP_INDAC: Clock Configured at 166666666Hz

FLS MCAL Version Info
---------------------
Vendor ID          : 44
Module ID          : 92
SW Major Version   : 2
SW Minor Version   : 0
SW Patch Version   : 0

FLS_APP_INDAC: DATA SIZE TEST is 0x100000
FLS_APP_INDAC: Offset is 0x0
FLS_APP_INDAC: Writing PHY Tune Data to last sector in memory
FLS_APP_INDAC: Erasing
FLS_APP_INDAC: Job Processing in Progress.
FLS_APP_INDAC: Job Ends: SUCCESS
FLS_APP_INDAC: Blank Checking
FLS_APP_INDAC: Job Processing in Progress.
FLS_APP_INDAC: Job Ends: SUCCESS
FLS_APP_INDAC: Writing
```

```
FLS_APP_INDAC: Job Processing in Progress.
FLS_APP_INDAC: Job Ends: SUCCESS
FLS_APP_INDAC: Comparing
FLS_APP_INDAC: Job Processing in Progress.
FLS_APP_INDAC: Job Ends: SUCCESS
FLS_APP_INDAC: Writing PHY Tune Data complete
FLS_APP_INDAC: Erasing
FLS_APP_INDAC: Job Processing in Progress.
FLS_APP_INDAC: Job Ends: SUCCESS
FLS_APP_INDAC: Blank Checking
FLS_APP_INDAC: Job Processing in Progress.
FLS_APP_INDAC: Job Ends: SUCCESS
FLS_APP_INDAC: Writing
FLS_APP_INDAC: Job Processing in Progress.
FLS_APP_INDAC: Job Ends: SUCCESS
FLS_APP_INDAC:
 Write 1048576 bytes at transfer rate 2847 Kbps
FLS_APP_INDAC: Reading
FLS_APP_INDAC: Job Processing in Progress.
FLS_APP_INDAC: Job Ends: SUCCESS
FLS_APP_INDAC:
 Read 1048576 bytes at transfer rate 43399 Kbps
FLS_APP_INDAC: Comparing
FLS_APP_INDAC: Job Processing in Progress.
FLS_APP_INDAC: Job Ends: SUCCESS
FLS_APP_INDAC:
 Changing mode to DAC
FLS_APP_INDAC: DONE!
```

FLS_app_xip

```
FLS_APP_XIP:
   ------------------FLS Sample Application - STARTS !!! ------------------
   FLS_APP_XIP:
    Running on J7 VCL
   FLS_APP_XIP: FLS spi_test Initiating and Starting.
   FLS_APP_XIP: Variant - Pre Compile being used !!!
   FLS_APP_XIP: Configuring Clocks.
   FLS_APP_XIP: Clock Configured at 166666666Hz

   FLS MCAL Version Info
   ---------------------
   Vendor ID          : 44
   Module ID          : 92
   SW Major Version   : 2
   SW Minor Version   : 0
   SW Patch Version   : 0
```

```
FLS_APP_XIP: DATA SIZE TEST is 0x1000
FLS_APP_XIP: Offset is 0x0
FLS_APP_XIP: Erasing
FLS_APP_XIP: Job Processing in Progress.
FLS_APP_XIP: Job Ends: SUCCESS
FLS_APP_XIP: Blank Checking
FLS_APP_XIP: Job Processing in Progress.
FLS_APP_XIP: Job Ends: SUCCESS
FLS_APP_XIP: Writing
FLS_APP_XIP: Job Processing in Progress.
FLS_APP_XIP: Job Ends: SUCCESS
FLS_APP_XIP: Calling XIP app
MCU1_0 running
MCU1_0 reports: All tests have passed
FLS_APP_XIP: Returning from XIP app, returned value is 0xFEEDFACE
FLS_APP_XIP: Reading
FLS_APP_XIP: Job Processing in Progress.
FLS_APP_XIP: Job Ends: SUCCESS
FLS_APP_XIP: Comparing
FLS_APP_XIP: Job Processing in Progress.
FLS_APP_XIP: Job Ends: SUCCESS
FLS_APP_XIP: DONE!
```

7.10 1.7.10. ICU

Functional Description:

The Icu driver uses ECAP module to capture events. There are three ECAP instances available to work with (ECAP0 - ECAP2) on J721E and J7200.

Icu Channel ID , Instance mapping and ISR mapping:

The Icu module is implemented using the ECAP instances on the device.

Three ECAP instances are supported by this driver implementation (three instances present in MAIN domain on J721E & J7200). The following table lists the mapping between instance of
ECAP and IcuChannelId of the configurator.

IcuChannelId	ECAP Instance	Associated ISR (if notification is enabled)
0	ECAP0	Icu_ch0Notify
1	ECAP1	Icu_ch1Notify
2	ECAP2	Icu_ch2Notify

Build and Running the Example Application:

Example Application:

To run the icu_app:

1. J7200: Uses instance ECAP0 instance and EPWM1 instance
User need to connect EPWM output (Pin 2 on J22 on GESI Expansion Board) to ECAP input (TP97 on base Board).

Example Log

J7200

```
                              ICU_APP: Sample Application - STARTS !!!                              ICU_APP: ICU MCAL Version Info

   ---------------------
   ICU_APP: Vendor ID        : 44
   ICU_APP: Module ID        : 122
   ICU_APP: SW Major Version   : 2
   ICU_APP: SW Minor Version   : 0
   ICU_APP: SW Patch Version   : 0                              ICU_APP: Variant - Pre Compile being used !!!
      ICU_APP: EPWM Channel Initialized
   ICU_APP: EPWM Duty cycle: 60 Percent, 1000Hz
   ICU_APP: Use EPWM (Pin 2 in J22 on GESI Board) as input to ECAP test point (TP97 on base board) for J7200 EVM
   ICU_APP: Edge Detect Mode!
   ICU_APP: input state is ICU_IDLE
   ICU_APP: SignalNotification for Double Edge Detection Reached in 1 sec: 2000
   ICU_APP: SignalNotification for Single Edge Detection Reached in 1 sec: 1000
   ICU_APP: input state is ICU_ACTIVE                              ICU_APP: Changing EPWM Frequency from 1000Hz to 25000Hz and Duty Cycle
to 50%
   ICU_APP: Edge Detect Mode!
   ICU_APP: input state is ICU_IDLE
   ICU_APP: SignalNotification for Single Edge Detection Reached in 1 sec: 25002
   ICU_APP: input state is ICU_ACTIVE
   ICU_APP: SignalNotification for after disabling notification Edge Detection Reached: 0
   ICU_APP: Calling Deinit
   ICU_APP: Icu Stack Usage 1328 bytes
   ICU_APP: ICU Test Passed!!!
```

7.11 1.7.11. MCU

Example Log

J7200

```
 McuApp: Sample Application - STARTS !!!

MCU MCAL Version Info
---------------------
Vendor ID        : 44
Module ID        : 101
SW Major Version   : 2
SW Minor Version   : 0
SW Patch Version   : 0

 Module clock request successful !!!
 Module clock frequency set. Freq = 19200000 !!!
MCU Reset Reason:5
MCU Reset Reason Raw Value:0x100000
MCU Stack Usage: 2788 bytes
MCU_APP: MCU Test Passed!!!
```

8 1.8. Build Env:

<TO-DO>

Add info on how to build example with SDK integration

9 1.9. Steps to Build in Windows:

By default CORE SDK RTOS JACINTO support to be built in Linux environment. All the required tools (compilers, OS, etc...) are packaged in CORE SDK RTOS, which enables MCUSW to built without any modifications.

The components MCUSW and PDK can be built in windows environment, with right version of tools.

List below details the steps required to build MCAL (MCUSW) examples in windows environment

9.1 1.9.1.1. Step 1 : Download the windows version of required tools

- CORE SDK RTOS JACINTO source is installed and accessible from windows machine, which would be used to build
- Download the windows version of the tools
    - XDC download TI Website
    - ARM Compiler TI Website
    - DSP Compiler TI Website
    - **Please ensure to download the versions used in CORE SDK RTOS JACINTO**
- Create a folder "CORE_SDK_RTOS_JACINTO_XXYYZZ" where XXYYZZ is release number

9.2 1.9.1.2. Step 2 : Install tools and copy the components

- Install the downloaded tools in ${Path}/CORE_SDK_RTOS_JACINTO_XXYYZZ/
    - Ensure the version of downloaded tools match the versions used in CORE SDK RTOS
- Copy the components mcusw and pdk
    - Ensure that the mcusw & pdk naming conventions is same as in CORE SDK RTOS

9.3 1.9.1.3. Step 4 : Disable generation of cust SBL

- Custom SBL is part of PDK package and relies on Linux based tools
- This requires to be excluded
- In file CORE_SDK_RTOS_JACINTO_XXYYZZ\pdk\packages\ti\boot\sbl\sbl_component.mk
- Comment out / delete
    - **sbl_lib_cust** from sbl_LIB_LIST
    - All statements under **# SBL Custom LIB**
- Without these steps the PDK library compilation would fail

9.4 1.9.1.4. Step 3 : Build

- Follow the steps listed in Build to build MCAL examples

9.5 1.9.1.5. Examples NOT supported in windows build

Core	Examples Not Supported	Comments
MCU 1 0	Multi-Core Boot Application	As demo reuqires Linux/QNX, C7x & C66 apps
MPU 1 0	IPC Remote Client Application	Not Yet supported

Core	Examples Not Supported	Comments
MCU 1 0	can_profile_xip_app	Creation of .bin image is not supported
MCU 1 0	fls_xip	Creation of .bin image is not supported