MCAL User Manual for Irq

32-bit TriCore™ AURIX™ TC3xx microcontroller

About this document

Scope and purpose

This User Manual is intended to enable users to integrate the Microcontroller Abstraction Layer (MCAL) software for the TriCore™ AURIX™ family of 32-bit microcontrollers.

This document describes responsibilities of integrator in-charge of integrating MCAL software with the basic software (BSW) stack. This document also provides detailed information on safety, configuration and functions along with examples of usage of significant features.

Note: Detailed information about package installation, safety and other generic information that are common across all modules are provided in MCAL User Manual General.

Intended audience

This document is intended for anyone using the Irq module of the TC3xx MCAL software.

Document conventions

Table 1 Conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, table and figure captions, screen names, windows, dialog boxes, menus, sub-menus
Italics	Denotes variable(s) and reference(s)
`Courier New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
>	Indicates that a cascading sub-menu opens when you select a menu item
[cover parentID=<alpha numeric value>]	Used for traceability completeness. Reader should ignore these.

Reference documents

This User Manual should be read in conjunction with the following documents:

- AURIX™ TC3xx MCAL User Manual General

Table of contents

About this document . . . 1
Table of contents . . . 2
1 IRQ driver . . . 4
1.1 User information . . . 4
1.1.1 Description . . . 4
1.1.2 Hardware-software mapping . . . 4
1.1.3 File structure . . . 5
1.1.3.1 C file structure . . . 5
1.1.3.2 Code generator plugin files . . . 6
1.1.4 Integration hints . . . 7
1.1.4.1 Integration with AUTOSAR stack . . . 7
1.1.4.2 Multicore and Resource Manager . . . 8
1.1.4.3 MCU support . . . 8
1.1.4.4 PORT support . . . 9
1.1.4.5 DMA support . . . 9
1.1.4.6 Interrupt connections . . . 9
1.1.4.7 Example usage . . . 9
1.1.4.7.1 Configuring the driver . . . 9
1.1.4.7.2 Initializing interrupts . . . 10
1.1.4.7.3 Clearing interrupts . . . 10
1.1.5 Key architectural considerations . . . 11
1.2 Assumptions of Use (AoUs) . . . 11
1.3 Reference information . . . 11
1.3.1 Configuration interfaces . . . 11
1.3.1.1 Container: IrqGeneral . . . 11
1.3.1.1.1 IrqOSekEnable . . . 11
1.3.1.2 Container: Irq<ModuleName>Config . . . 12
1.3.1.2.1 Container: Irq<ModuleName>CatConfig . . . 12
1.3.1.2.2 Container: Irq<ModuleName>PrioConfig . . . 13
1.3.1.2.3 Container: Irq<ModuleName>TosConfig . . . 13
1.3.1.3 Container: CommonPublishedInformation . . . 14
1.3.1.3.1 ArMajorVersion . . . 14
1.3.1.3.2 ArMinorVersion . . . 14
1.3.1.3.3 ArPatchVersion . . . 15
1.3.1.3.4 SwMajorVersion . . . 15
1.3.1.3.5 SwMinorVersion . . . 16
1.3.1.3.6 SwPatchVersion . . . 16
1.3.1.3.7 ModuleId . . . 16
1.3.1.3.8 VendorId . . . 17

Table of contents

1.3.2 Functions – Type definitions . . . 17
1.3.3 Functions - APIs . . . 17
1.3.3.1 Interrupt init functions . . . 17
1.3.3.2 Irq_ClearAllInterruptFlags . . . 18
1.3.4 Notifications and callbacks . . . 18
1.3.5 Scheduled functions . . . 18
1.3.6 Interrupt service routines . . . 18
1.3.7 Callout . . . 18
1.3.8 **Errors Handling** . . . 19
1.3.9 Deviations and limitations . . . 19
1.3.9.1 Deviations . . . 19
1.3.9.2 Limitations . . . 19
Revision history . . . 19
Disclaimer . . . 20

1 IRQ driver

1.1 User information

1.1.1 Description

The IRQ driver provides the necessary configuration parameters and APIs for interrupt configuration, initialization and handling.

The driver is responsible for:

- Configuration of priority number for service requests
- Runtime APIs for initialization of service request nodes with configured priority and service provider (CPUx, DMA where x varies from 0 to number of available cores)
- Runtime APIs for initialization of general service request nodes with configured priority and service provider which can be used for software trigger service requests
- Runtime APIs for clearing of service request flags for SRNs
- If CAT2 is selected, operating system should take care of interrupt handling

The IRQ driver is implemented as Pre-Compile variant.

1.1.2 Hardware-software mapping

This section describes the system view of IRQ driver and peripherals administered by it.

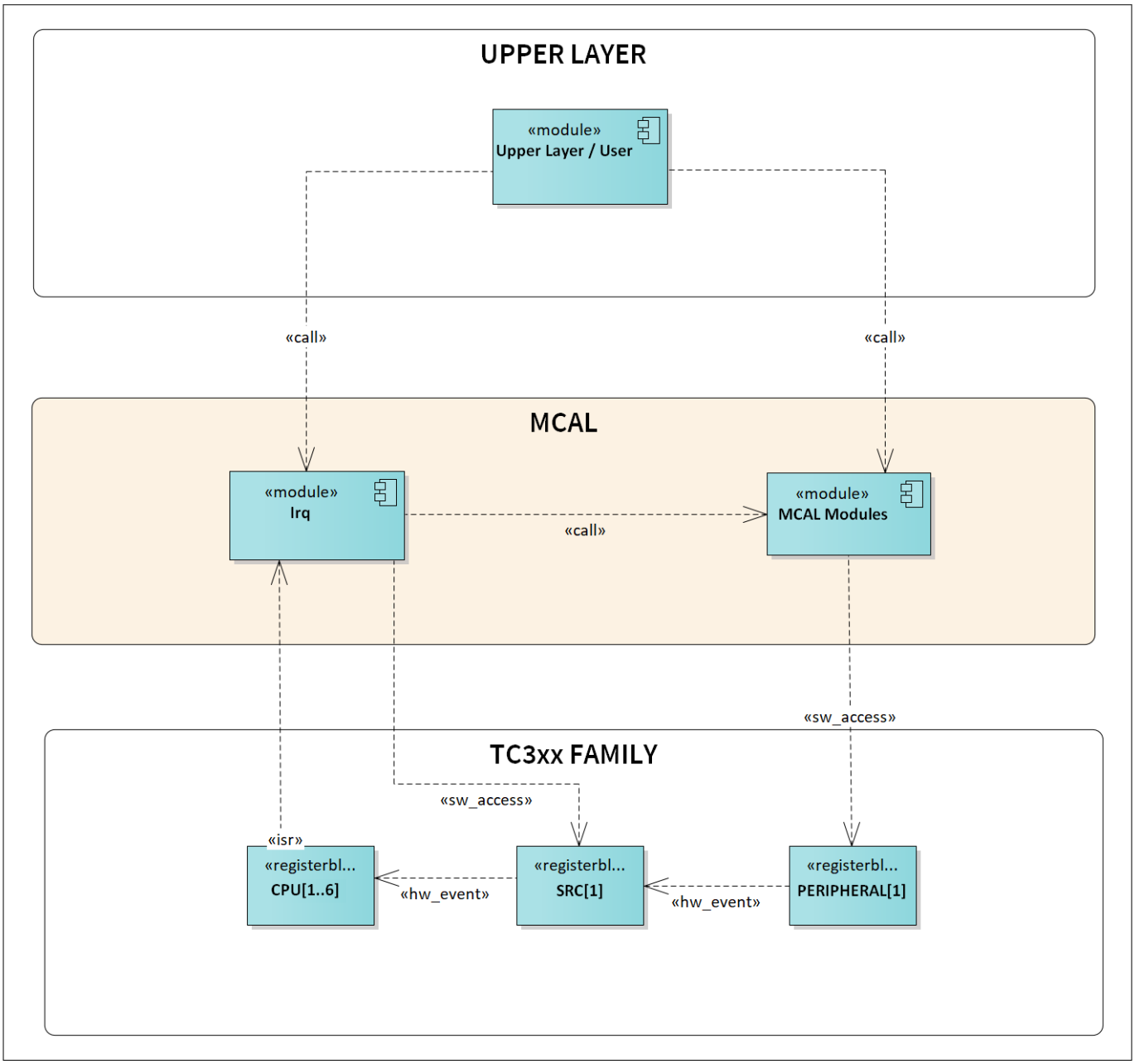

Figure 1 Mapping of hardware-software interfaces

The IRQ driver accesses the registers of interrupt controller. The driver initializes the peripheral-specific SRNs priority. On receiving an interrupt, the interrupt system jumps to the appropriate interrupt handler frame within the IRQ driver and the corresponding interrupt handler function is called. In the MCAL drivers, the interrupt handler functions are located within the software driver.

1.1.3 File structure

1.1.3.1 C file structure

This section provides details of the C files of the IRQ driver.

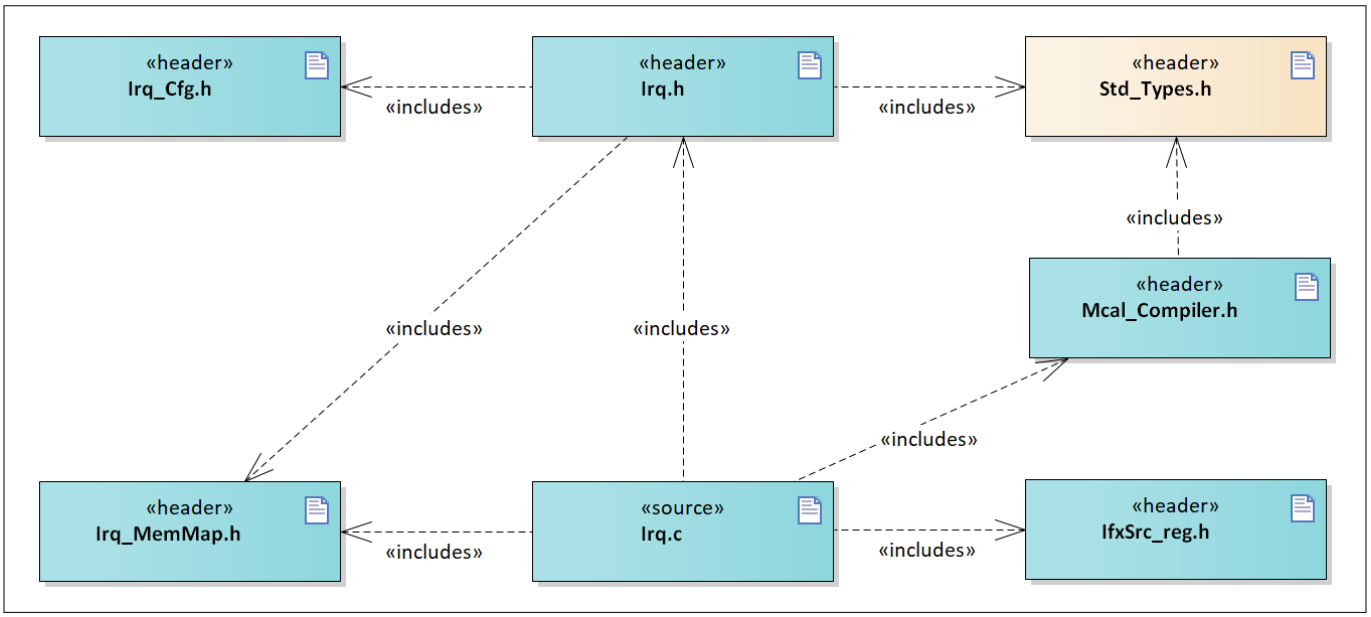

Figure 2 C file structure

Table 2 C file structure

File name	Description
`Irq.h`	Header file (static) defining prototypes of data structures and APIs
`Irq.c`	File (static) containing implementation of APIs
`Irq_Cfg.h`	Generated header file containing macros and configuration data of interrupt priority and interrupt service providers
`Irq_MemMap.h`	File (static) containing the memory section definitions used by the IRQ driver
`IfxSrc_reg.h`	SRC register definition file
`Std_Types.h`	Standard data types to be used are declared here
`Mcal_Compiler.h`	Compiler abstraction of TriCore™ instructions

1.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the IRQ driver.

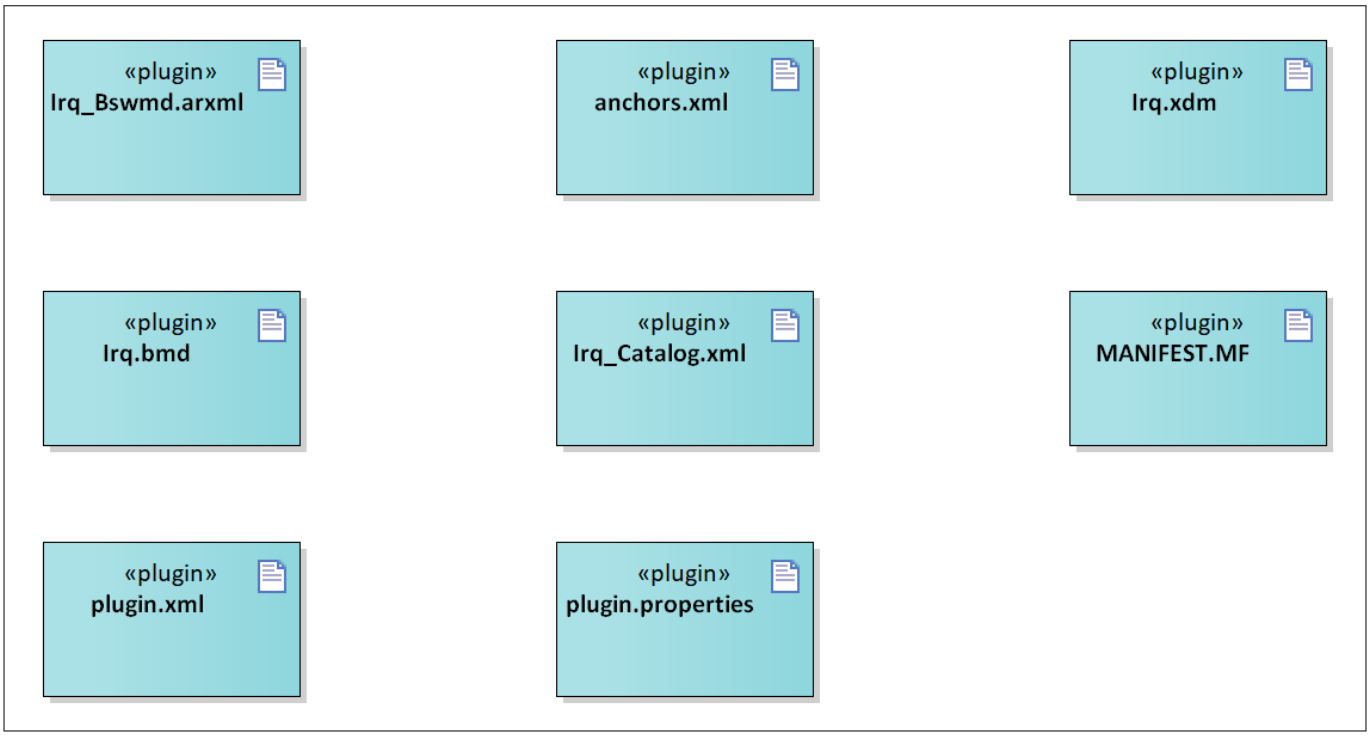

Figure 3 Code generator plugin files

Table 3 Code generator plugin files

File name	Description
`anchors.xml`	Tresos anchors support file for the IRQ driver
`Irq.xdm`	Tresos format XML data model schema file
`Irq.bmd`	AUTOSAR format XML data model schema file (for each device)
`MANIFEST.MF`	Tresos plugin support file containing the metadata for the IRQ driver
`plugin.proprties`	Tresos plugin support file for the IRQ driver
`plugin.xml`	Tresos plugin support file for the IRQ driver
`Irq_Bswmd.arxml`	AUTOSAR format module description file
`Irq_Catalog.xml`	AUTOSAR format catalog file

1.1.4 Integration hints

This section lists the key points that an integrator or user of the IRQ driver must consider.

1.1.4.1 Integration with AUTOSAR stack

This section lists the modules that are not part of the MCAL but are required to integrate the IRQ driver.

- **EcuM**

The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers. The EcuM

module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

- **DET**

DET module is not required for integrating the IRQ driver.

- **DEM**

DEM module is not required for integrating the IRQ driver.

- **SchM**

SchM is not required for integrating the IRQ driver.

- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows re-location of text, variables, constants and configuration data to user specific memory regions. In order to achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Irq_MemMap.h` file.

The `Irq_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are relocated to the correct memory region. A sample implementation listing the memorysection macros is depicted below.

```
/* Code Sections */
#if defined IRQ_START_SEC_CODE_QM_GLOBAL
    /* User Pragma here */
    #undef IRQ_START_SEC_CODE_QM_GLOBAL
    #undef MEMMAP_ERROR
#elif defined IRQ_STOP_SEC_CODE_QM_GLOBAL
    /* User Pragma here */
    #undef IRQ_STOP_SEC_CODE_QM_GLOBAL
    #undef MEMMAP_ERROR
#endif
```

Safety error

The IRQ driver does not report any safety error.

Notifications and callbacks

The IRQ driver does not provide any notification or callback functions.

Operating system

The OS or application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application.

Operating system files provided by MCAL package is only an example code and must be updated by the integrator with the actual OS files for the desired function.

1.1.4.2 Multicore and Resource Manager

The IRQ driver does not support execution on multiple cores in parallel.

1.1.4.3 MCU support

The IRQ driver is dependent on the MCU driver for the clock service. Therefore, the MCU initialization must be completed prior to IRQ initialization.

1.1.4.4 PORT support

The IRQ driver does not use any services provided by the PORT driver.

1.1.4.5 DMA support

The IRQ driver does not use any services provided by the DMA driver.

1.1.4.6 Interrupt connections

Defining the interrupt vector table entries and invoking the interrupt handlers must be done by the user. A sample invocation for GTM TOM is depicted as follows:

GTM TOM interrupts definition and invoking:

```
/* SRC_GTMTOM0SR0_ISR*/
ISR(GTMTOM0SR0_ISR)
{
    /* Enable global interrupts */
    ENABLE();
    /* Parameter is channel number */
    Mcu_17_Gtm_TomChannelIsr (TOM_MODULE_0 , TOM_CHANNEL_0);
    /* TOM_MODULE_0 = 0, TOM_CHANNEL_0 = 0*/
}
```

1.1.4.7 Example usage

1.1.4.7.1 Configuring the driver

Use the following to configure the module:

- IRQ general
  - Configure IrqOSekEnable to indicate the use of OS. If this parameter is enabled then user is allowed to configure the CAT2 interrupts
  - Configuring IrqOSekEnable is shown in the following figure:

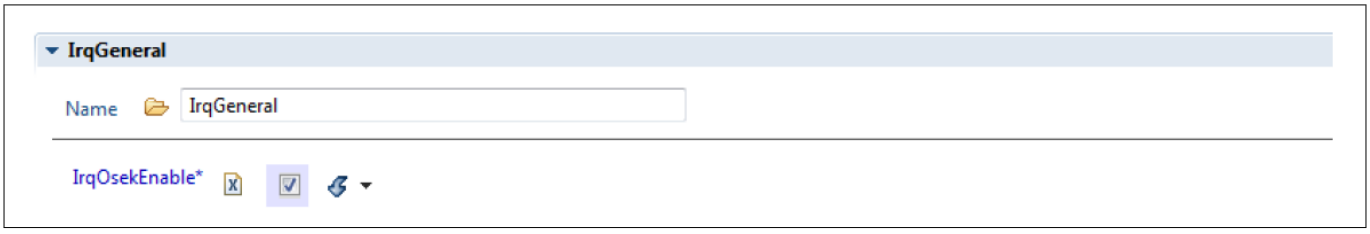

Figure 4 Configuring IrqOSekEnable

- Module interrupt setting
  - Configure Irq<ModuleName>SRN##Cat to indicate the category of the corresponding SRN
  - Configure Irq<ModuleName>SRN##Prio to indicate the priority of the corresponding SRN
  - Configure Irq<ModuleName>SRN##Tos to indicate the category of the corresponding SRN
  - Example for configuring module interrupt is shown in the following figure:

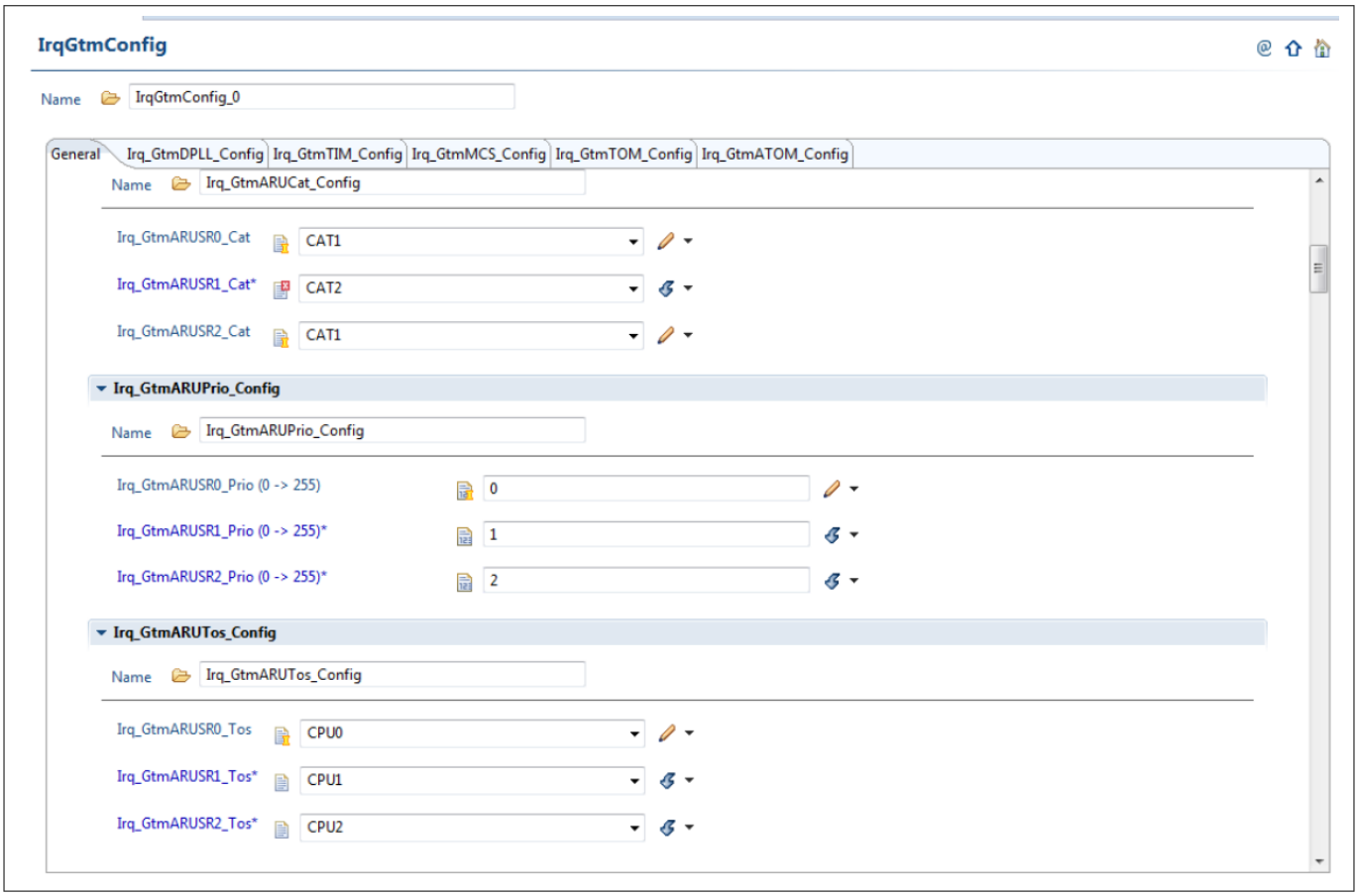

Figure 5 Interrupt configuration

1.1.4.7.2 Initializing interrupts

The module-specific service request control (SRC) registers are initialized with type of service and priority by `<ModuleName>_Init` functions. Additionally, user should enable module interrupts in interrupt configuration register.

For example, GTM initialization is depicted as follows:

```
/* Initialize Module Interrupt Priority and Service Providers */
IrqGtm_Init();
/* Enable module interrupts using SRE Bit*/
SRC_GTMTOM00.B.SRE = 1U;
```

1.1.4.7.3 Clearing interrupts

The `Irq_ClearAllInterruptFlag()` API clears the SRC registers of all modules.

```
/* Clear Interrupt Flags */
Irq_ClearAllInterruptFlags();
```

1.1.5 Key architectural considerations

None.

1.2 Assumptions of Use (AoUs)

There are no AoU for the IRQ driver.

1.3 Reference information

1.3.1 Configuration interfaces

The following diagram depicts the hierarchy along with the extensions provided for IRQ module.

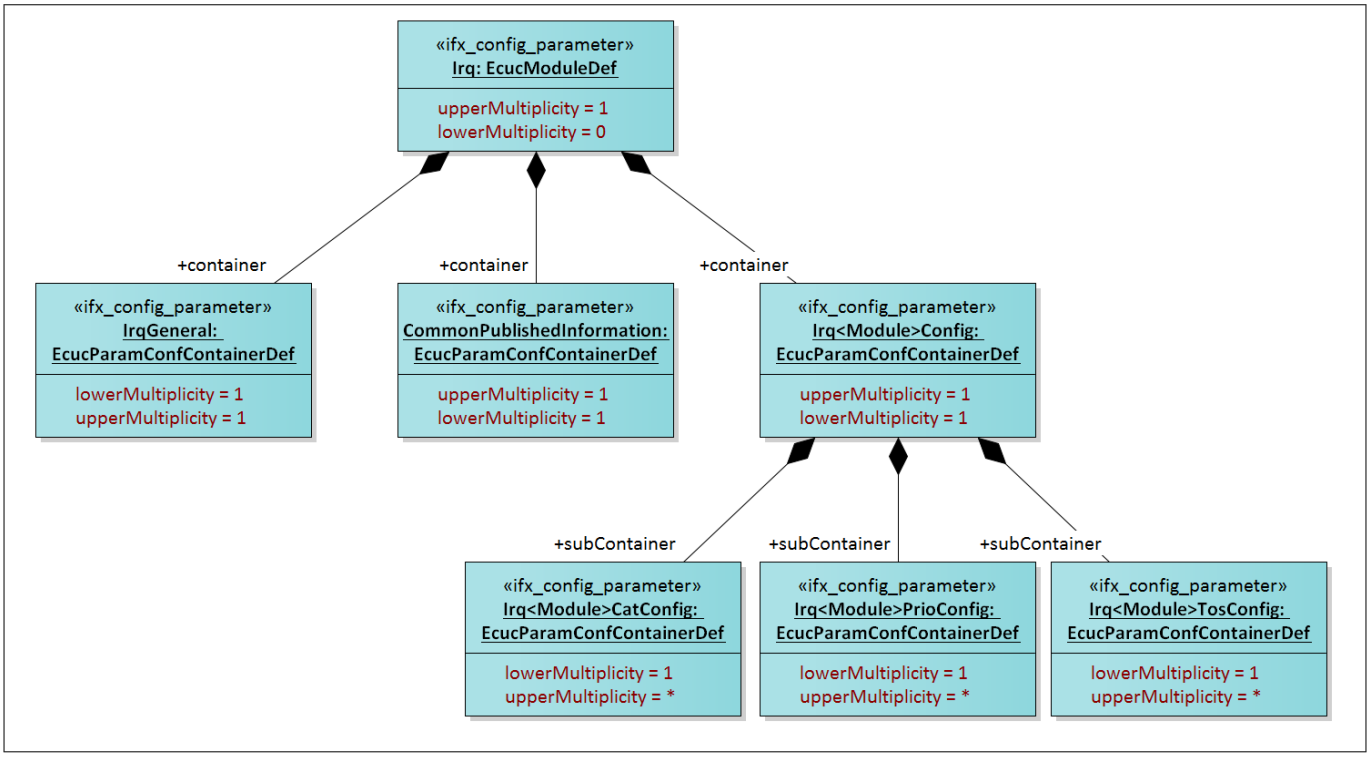

Figure 6 IRQ module configuration

1.3.1.1 Container: IrqGeneral

This container includes general configuration parameters of the IRQ driver.

1.3.1.1.1 IrqOSekEnable

Table 4 Specification for IrqOSekEnable

Name	IrqOSekEnable		
Description	Parameter to identify the use of OS. This controls the available configuration options for interrupt category.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef

Table 4 Specification for IrqOSekEnable (continued)

Range	TRUE: OSEK OS is used. This allows the user to configure the interrupt category as CAT1 or CAT2 FALSE: OSEK OS is not used. All interrupt must be of category CAT1.		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.2 Container: Irq<ModuleName>Config

This container includes module-specific interrupt configuration parameters. Where <ModuleName> stand for Ccu6, Gpt, Can, Adc, Gtm, GpsrGroup, Flexray, Spi, Asclin, Ethernet, Dma, Scu, Dmu, Hssl, Sent, Stm, I^2C and Dsadc.

The name of the container Irq<ModuleName>Config is affecting the macro generation. Hence, the container name should not be modified.

Note: The list of available modules is provided in the Release Notes.

1.3.1.2.1 Container: Irq<ModuleName>CatConfig

This container includes the parameter to configure the category of all the SRN supported by the module.

The name of the container Irq<ModuleName>CatConfig is affecting the macro generation. Hence, the container name should not be modified.

Irq<ModuleName><SrnName>Cat

Table 5 Specification for Irq<ModuleName><SrnName>Cat

Name	Irq<ModuleName><SrnName>Cat		
Description	The interrupt category setting of the corresponding SRN of the module. The SRN number indicates the SRN configured. Depending upon the category the interrupt frame is different. <SrnName> stands for name of the service request node.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	CAT1 : Category 1 interrupt is selected CAT2 : Category 2 interrupt is selected		
Default value	CAT 1		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

Table 5 Specification for Irq<ModuleName><SrnName>Cat (continued)

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	Only the option CAT1 is allowed to select if the IrqOSekEnable parameter is set to false.		

1.3.1.2.2 Container: Irq<ModuleName>PrioConfig

This container includes the parameter to configure the priority of all the SRN supported by the module.

The name of the container Irq<ModuleName>PrioConfig is affecting the macro generation. Hence, the container name should not be modified.

Irq<ModuleName><SrnName>Prio

Table 6 Specification for Irq<ModuleName><SrnName>Prio

Name	Irq<ModuleName><SrnName>Prio		
Description	The interrupt priority setting of the corresponding SRN of the module. Each SRN should have different priority number.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 to 255		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.2.3 Container: Irq<ModuleName>TosConfig

This container includes the parameter to configure the type of service of all the SRN supported by the module.

The name of the container Irq<ModuleName>TosConfig is affecting the macro generation. Hence, the container name should not be modified.

Irq<ModuleName><SrnName>Tos

Table 7 Specification for Irq<ModuleName><SrnName>Tos

Name	Irq<ModuleName><SrnName>Tos
Description	The type of service setting of the corresponding SRN of the module. This defines the interrupt control unit, which service the ISR.

Table 7 Specification for Irq<ModuleName><SrnName>Tos (continued)

Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	CPUx: where x depends on derivate DMA		
Default value	CPU0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3 Container: CommonPublishedInformation

This container includes published information about vendor and versions.

1.3.1.3.1 ArMajorVersion

Table 8 Specification for ArMajorVersion

Name	ArMajorVersion		
Description	This parameter specifies AUTOSAR major release version.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 to 255		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3.2 ArMinorVersion

Table 9 Specification for ArMinorVersion

Name	ArMinorVersion		
Description	This parameter specifies AUTOSAR minor release version.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 to 255		

Table 9 Specification for ArMinorVersion (continued)

Default value	2		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3.3 ArPatchVersion

Table 10 Specification for ArPatchVersion

Name	ArPatchVersion		
Description	This parameter specifies AUTOSAR patch release version.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 to 255		
Default value	2		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3.4 SwMajorVersion

Table 11 Specification for SwMajorVersion

Name	SwMajorVersion		
Description	This parameter specifies software major release version.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 to 255		
Default value	2		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3.5 SwMinorVersion

Table 12 Specification for SwMinorVersion

Name	SwMinorVersion		
Description	This parameter specifies software minor release version.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 to 255		
Default value	2		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3.6 SwPatchVersion

Table 13 Specification for SwPatchVersion

Name	ArPatchVersion		
Description	This parameter specifies software patch release version.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 to 255		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3.7 ModuleId

Table 14 Specification for ModuleId

Name	ModuleId		
Description	This parameter specifies module identification number.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 to 65535		
Default value	255		

Table 14 Specification for ModuleId (continued)

Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.1.3.8 VendorId

Table 15 Specification for VendorId

Name	VendorId		
Description	This parameter specifies vendor identification number.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 to 65535		
Default value	17		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

1.3.2 Functions – Type definitions

No datatype are defined in IRQ.

1.3.3 Functions - APIs

This section lists all the APIs of the IRQ driver.

1.3.3.1 Interrupt init functions

Table 16 Specification for Irq<ModuleName>_Init API

Syntax	`void Irq<ModuleName>_Init(void)`
Service ID	None
Sync/Async	Synchronous
Reentrancy	Non-reentrant
Parameters (in)	None
Parameters (out)	None

Table 16 Specification for Irq<ModuleName>_Init API (continued)

Parameters (in-out)	None
Return	None
Description	Initializes the priority and type of service of SRNs for modules.
Source	IFX
Error handling	-
Configuration dependencies	-

1.3.3.2 Irq_ClearAllInterruptFlags

Table 17 Specification for Irq_ClearAllInterruptFlags API

Syntax	`void Irq_ClearAllInterruptFlags(void)`
Service ID	None
Sync/Async	Synchronous
Reentrancy	Non-reentrant
Parameters (in)	None
Parameters (out)	None
Parameters (in-out)	None
Return	None
Description	The service clears the service request (SRR), interrupt trigger overflow clear bit (IOVCLR) and software sticky clear bit (SWSCLR) flags for available modules SRN.
Source	IFX
Error handling	-
Configuration dependencies	-

1.3.4 Notifications and callbacks

The IRQ driver does not provide any notification or callbacks.

1.3.5 Scheduled functions

The IRQ driver does not provide any scheduled functions.

1.3.6 Interrupt service routines

The IRQ driver does not provide any interrupt handlers.

1.3.7 Callout

The IRQ driver does not provide any Callout functions.

1.3.8 Errors Handling

The IRQ driver does not report any error.

1.3.9 Deviations and limitations

1.3.9.1 Deviations

The IRQ driver does not have any deviations.

1.3.9.2 Limitations

The IRQ driver does not support SRNs with different service providers to have the same priorities.

Revision history

Major changes since the last revision.

Date	Version	Description
2020-08-11	1.0	Document is released.
2020-08-10	0.1	• Initial version • IRQ driver chapter moved from TC3xx_SW_MCAL_UM_DEMO to this document.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-08-11

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-wla1597125371962

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”) .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.