Integration Manual

for S32K14X OCU Driver

Document Number: IM2OCUASR4.2 Rev0002R1.0.2
Rev. 1.0

Contents

Section number **Title** **Page**

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	26/04/2019	NXP MCAL Team	Updated version for ASR 4.2.2S32K14XR1.0.2

Chapter 2
Introduction

This integration manual describes the integration requirements for Ocu Driver for S32K14X microcontrollers.

2.1 Supported Derivatives

The software described in this document is intented to be used with the following microcontroller devices of NXP Semiconductors .

Table 2-1. S32K14X Derivatives

NXP Semiconductors	s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100

All of the above microcontroller devices are collectively named as S32K14X .

2.2 Overview

AUTOSAR (AUTomotive Open System ARchitecture) is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.

- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".
- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

Boldface type: Bold is used for important terms, notes and warnings.

Italic font: Italic typeface is used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
BSW	Basic Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
ISR	interrupt Service Routine
OCU	Output Compare Unit
OS	Operating System
RAM	Random Access Memory
ROM	Read-only Memory
N/A	Not Applicable
MCU	Microcontroller Unit
GUI	Graphical User Interface
EcuM	ECU state Manager

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
API	Application Programming Interface
PB Variant	Post Build Variant
PC Variant	Pre Compile Variant
SWS	Software Specification
VLE	Variable Length Encoding
XML	Extensible Markup Language
FTM	FlexTimer Module

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	Specification of Ocu Driver	AUTOSAR Release 4.2.2
2	S32K14X Reference Manual	Reference Manual, Rev. 9, 9/2018
3	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	30/11/2017
4	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	30/11/2017
5	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	30/11/2017
6	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	25/10/2018
7	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	07/01/2019

Chapter 3
Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar Ocu driver for NXP SemiconductorsS32K14X . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The Ocu driver files are compiled using

- Green Hills Multi 7.1.4 / Compiler 2017.1.4
- (Linaro GCC 6.3-2017.06~dev) 6.3.1 20170509 (Wed Jan 24 16:21:45 CST 2018 build.sh rev=g27a1317 s=L631 Earmv7 -V release_g27a1317_build_Fed_Earmv7)
- IAR: V8.11.2

The compiler, linker flags used for building the driver are explained below:

Note

The TS_T40D2M10I2R0 plugin name is composed as follow:

TS_T = Target_Id

D = Derivative_Id

M = SW_Version_Major

I = SW_Version_Minor

R = Revision

(i.e. Target_Id = 40 identifies CORTEXM architecture and Derivative_Id = 2 identifies the S32K14X)

3.1.1 GHS Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-ansi	Specifies ANSI C with extensions. This mode extends the ANSI X3.159-1989 standard with certain useful and compatible constructs.
-Osize	Optimize for size.
-dual_debug	Enables the generation of DWARF, COFF, or BSD debugging information in the object file
-G	Generates source level debugging information and allows procedure call from debugger's command line.
--no_exceptions	Disables support for exception handling
-Wundef	Generates warnings for undefined symbols in preprocessor expressions
-Wimplicit-int	Issues a warning if the return type of a function is not declared before it is called
-Wshadow	Issues a warning if the declaration of a local variable shadows the declaration of a variable of the same name declared at the global scope, or at an outer scope
-Wtrigraphs	Issues a warning for any use of trigraphs
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
--prototype_errors	Generates errors when functions referenced or called have no prototype
--incorrect_pragma_warnings	Valid #pragma directives with wrong syntax are treated as warnings
-noslashcomment	C++ like comments will generate a compilation error
-preprocess_assembly_files	Preprocesses assembly files
-nostartfile	Do not use Start files
--short_enum	Store enumerations in the smallest possible type
-c	Produces an object file (called input-file.o) for each source file.
--no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup.
-keeptempfiles	Prevents the deletion of temporary files after they are used. If an assembly language file is created by the compiler, this option will place it in the current directory instead of the temporary directory. Produces an object file (called input-file.o) for each source file.
-list	Creates a listing by using the name of the object file with the .lst extension. Assembler option
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DDISABLE_MCAL_INTERMODULE_ASR_CHECK	-D defines a preprocessor symbol to disable the inter-module version check for AR_RELEASE versions. DISABLE_MCAL_INTERMODULE_ASR_CHECK: By default in the package, drivers are compiled to perform the inter-module version check as per Autosar BSW004. When the inter-module version check needs to be disabled then the DISABLE_MCAL_INTERMODULE_ASR_CHECK global define must be added to the list of compiler options.
-DGHS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GHS preprocessor symbol.

Table 3-2. Assembler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-c	Produces an object file (called input-file.o) for each source file.
-preprocess_assembly_files	Preprocesses assembly files
-asm=list	Creates a listing by using the name of the object file with the .lst extension. Assembler option

Table 3-3. Linker Options

Option	Description
-Mn	Map file numeric ordering
-delete	Removal from the executable of functions that are unused and unreferenced
-v	Display removed unused functions
-ignore_debug_references	Ignores relocations from DWARF debug sections when using -delete.
-map	Creates a detailed map file
-keepmap	Keep the map file in the event of a link error
-lstartup	Link libstartup library -Run-time environment startup routines
-lsys	Link libsys library -Run-time environment system routines
-larch	Link libarch library -Target-specific run-time support. Any file produced by the Green Hills Compiler may depend on symbols in this library.
-lansi	Link libansi library -the standard C library
-L(/lib/thumb2)	Link thumb2 library
-lutf8_s32	Include utf8_s32.a to use the Wide Character Functions

3.1.2 IAR Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
--endian=little	Specifies the endianess of core: little endian.
-Ohz	Sets the optimization level to High, favoring size.
-c	Produces an object file (called input-file.o) for each source file.
--no_clustering	Disables static clustering optimizations.
--no_mem_idioms	Makes the compiler to not optimize code sequences that clear, set, or copy a memory region.
--no_explicit_zero_opt	Places the zero initialized variables in data section instead of bss.
--debug	Makes the compiler include information in the object modules.

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
--diag_suppress=Pa050	Suppresses diagnostic messages (warnings) about non-standard line endings.
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DIAR	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the IAR preprocessor symbol.
--require_prototypes	Forces the compiler to verify that all functions have proper prototypes.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by compiler.
--no_system_include	Disables the automatic search for system include files.
-e	Enables language extensions. This option is needed by FLS driver which uses _packed structures.

Table 3-5. Assembler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
-g	Use this option to disable the automatic search for system include files.

Table 3-6. Linker Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--map filename	Produces a map file.
--no_library_search	Disables automatic runtime library search.
--entry _start	Treats the symbol _start as a root symbol and as the start of the application.
--enable_stack_usage	Enables stack usage analysis.
--skip_dynamic_initialization	Suppress dynamic initialization during system startup.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by linker.
--config	Specifies the configuration file to be used by the linker.

3.1.3 GCC Compiler/Linker/Assembler Options

Table 3-7. Compiler Options

Option	Description
-c	Produces an object file (called input-file.o) for each source file.
-Os	Use optimization for size.
-ggdb3	Produce debugging information for use by GDB. Level 3 includes extra information, such as all the macro definitions present in the program.
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-ansi	Specifies ANSI C with extensions.
-mlittle-endian	Generate code for a processor running in little-endian mode.
-fomit-frame-pointer	Removes the frame pointer for all functions, which might make debugging harder.
-msoft-float	Use software floating-point instructions.
-fno-common	Specifies that the compiler should place uninitialized global variables in the data section of the object file, rather than generating them as common blocks.
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
-Wextra	Enables some extra warning flags that are not enabled by '-Wall'.
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types.
-Wno-sign-compare	Do not warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned.
-fstack-usage	Geneates an extra file that specifies the maximum amount of stack used, on a per-function basis.
-fdump-ipa-all	Enables all inter-procedural analysis dumps.
-Werror=implicit-function-declaration	Generates an error when the prototype of the function is not defined..
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DGCC	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GCC preprocessor symbol.

Table 3-8. Assembler Options

Option	Description
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-c	Produces an object file (called input-file.o) for each source file.
-mthumb	This option specifies that the assembler should start assembling Thumb instructions.
-x assembler-with-cpp	Indicates that the assembly code contains C directives and the C preprocessor must be run.

Table 3-9. Linker Options

Option	Description
-Map=filename	Print a link map to the file mapfile.
-T scriptfile	Use scriptfile as the linker script. This script replaces ld's default linker script(rather than adding to it), so commandfile must specify everything necessary to describe the output file.
--disable-newlib-supplied-syscalls -specs=nosys.specs	These options support for using newlib on core M0+
-u _printf_float -u _scanf_float	These options support generating profile report.
-nostartfiles	Do not use the standard system startup files when linking
-e _start	Specify that the program entry point is _start
-static	The --static flag tells the linker to link a static, not a dynamically linked
-lc	The -lc flag tells the linker to link this binary against the C library, which is newlib in our case.
-lnosys	The -lnosys flag tells the linker to link this binary against the "nosys" library
$(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib/thumb/v6-m $(TOOLCHAIN_DIR)/lib/gcc/arm-none-eabi/6.3.1/thumb/v6-m	Library for core M0+
$(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib/thumb $(TOOLCHAIN_DIR)/arm-none-eabi/newlib/lib)	Library for core M4

3.2 Files required for Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the Ocu driver for S32K14X microcontrollers.

To avoid integration of incompatible files, all the include files from other modules shall have the same AR_MAJOR_VERSION and AR_MINOR_VERSION, i.e. only files with the same AUTOSAR major and minor versions can be compiled.

Ocu Files

- ..\Ocu_TS_T40D2M10I2R0\include\Ocu.h
- ..\Ocu_TS_T40D2M10I2R0\include\Ocu_Irq.h
- ..\Ocu_TS_T40D2M10I2R0\include\Ocu_Types.h
- ..\Ocu_TS_T40D2M10I2R0\include\Ocu_EnvCfg.h
- ..\Ocu_TS_T40D2M10I2R0\include\Ocu_Ftm.h
- ..\Ocu_TS_T40D2M10I2R0\include\Ocu_Ftm_Types.h
- ..\Ocu_TS_T40D2M10I2R0\include\Ocu_Ftm_Irq.h
- ..\Ocu_TS_T40D2M10I2R0\include\Ocu_Ipw.h
- ..\Ocu_TS_T40D2M10I2R0\include\Ocu_Ipw_Irq.h

- ..\Ocu_TS_T40D2M10I2R0\include\Ocu_Ipw_Types.h
- ..\Ocu_TS_T40D2M10I2R0\src\Ocu.c
- ..\Ocu_TS_T40D2M10I2R0\src\Ocu_Ftm.c
- ..\Ocu_TS_T40D2M10I2R0\src\Ocu_Ipw.c

Ocu Generated Files

- Ocu_PBCfg<VariantNo>.c files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB)
- Ocu_Cfg.h - For driver compilation, this file should be generated by the user using a configuration tool

Note: As a deviation from standard:

- Ocu_PBCfg<VariantNo>.c files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB)
- Ocu_Cfg.h - This file contaians all the elements that are not variant aware, configured and generated only once (defines). For driver compilation, this file should be generated by the user using a configuration tool

Files from Base common folder

- ..\Base_TS_T40D2M10I2R0\include\Compiler.h
- ..\Base_TS_T40D2M10I2R0\include\Compiler_Cfg.h
- ..\Base_TS_T40D2M10I2R0\include\ComStack_Types.h
- ..\Base_TS_T40D2M10I2R0\include\Ocu_MemMap.h
- ..\Base_TS_T40D2M10I2R0\include\Mcal.h
- ..\Base_TS_T40D2M10I2R0\include\Platform_Types.h
- ..\Base_TS_T40D2M10I2R0\include\Std_Types.h
- ..\Base_TS_T40D2M10I2R0\include\Reg_eSys.h
- ..\Base_TS_T40D2M10I2R0\include\Soc_Ips.h
- ..\Base_TS_T40D2M10I2R0\include\SilRegMacros.h

Files from Det folder:

- ..\Det_ TS_T40D2M10I2R0 \include\Det.h

Files from Rte folder:

- ..\Rte_ TS_T40D2M10I2R0 \include\SchM_Ocu.h

Files from Mcl folder:

- ..\Mcl_ TS_T40D2M10I2R0 \include\Ftm_Common_Types.h
- ..\Mcl_ TS_T40D2M10I2R0 \include\Reg_eSys_Ftm.h
- ..\Mcl_ TS_T40D2M10I2R0 \src\Ftm_Common.c

3.3 Setting up the Plug-ins

The Ocu driver was designed to be configured by using the EB Tresos Studio (version EB tresos Studio 23.0.0 b170330-0431 or later.)

Location of various files inside the FR module folder:

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\ Ocu _ TS_T40D2M10I2R0 \abc\Ocu.xdm
  - ..\ Ocu _ TS_T40D2M10I2R0 \abc\Mcu.xdm
  - ..\ Ocu _ TS_T40D2M10I2R0 \abc\Resource.xdm
  - ..\ Ocu _ TS_T40D2M10I2R0 \abc\Mcl.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\ Ocu _ TS_T40D2M10I2R0 \autosar\Ocu<subderivative_name>.epd
  - ..\ Ocu _ TS_T40D2M10I2R0 \autosar\Mcu.epd
  - ..\ Ocu _ TS_T40D2M10I2R0 \autosar\Resource_<subderivative_name>.epd
  - ..\ Ocu _ TS_T40D2M10I2R0 \autosar\Mcl.epd
- Code Generation Templates for parameters:
  - ..\ Ocu _ TS_T40D2M10I2R0 \Ocu_PBcfg.c
  - ..\ Ocu _ TS_T40D2M10I2R0 \Ocu_Cfg.h
  - ..\ Ocu _ TS_T40D2M10I2R0 \Mcu_Cfg.h
  - ..\ Ocu _ TS_T40D2M10I2R0 \Mcl_Cfg.h

Steps to generate the configuration:

1. Copy the module folders Ocu _ TS_T40D2M10I2R0 ,DEM_TS_T40D2M10I2R0 , Base_ TS_T40D2M10I2R0 , Resource_ TS_T40D2M10I2R0 , Mcu_ TS_T40D2M10I2R0 , Mcl_ TS_T40D2M10I2R0 into the Tresos plugins folder.
2. Set the desired Tresos Output location folder for the generated sources and header files.
3. Use the EB tresos Studio GUI to modify ECU configuration parameters values.
4. Generate the configuration files.

Dependencies

- **MCU** is required to use System Clock when clock source is used as Peripheral clock source to generate Ocu Segment values.
- **MCL** is required to provide some common files used by the FTM peripheral
- **RESOURCE** is required to select processor derivative. Current Ocu driver has support for the following derivatives, everyone having attached a Resource file: s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100,

s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100 .

- **ECUC** is needed to allows users to configure multiple configuration.
- **DET** is required for signaling the development error detection (parameters out of range, null pointers, etc).

Chapter 4
Function calls to module

4.1 Function Calls during Start-up

Ocu shall be initialized during STARTUP phase of EcuM initialization. The API to be called for this is Ocu _Init(). The MCU module should be initialized before the Ocu is initialized.

4.2 Function Calls during Shutdown

During shutdown phase, Ocu_DeInit() function can be called. Calling this function depends on the initialization-deinitialization strategy deployed by user.

4.3 Function Calls during Wake-up

During Wake-up phase, Ocu_Init() function may be called but only if during a previous phase Ocu_DeInit() was called. Calling this function depends on the initialization deinitialization strategy deployed by user.

Chapter 5
Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

In the current implementation, OCU is using the services of Schedule Manager (SchM) for entering and exiting the exclusive areas.

The following critical regions are used in the OCU driver:

In Ocu_Ftm.c

In function Ocu_Ftm_StartChannel : **OCU_EXCLUSIVE_AREA_08**

In function Ocu_Ftm_StopChannel : **OCU_EXCLUSIVE_AREA_09**

In function Ocu_Ftm_SetPinAction : **OCU_EXCLUSIVE_AREA_10**

In function Ocu_Ftm_SetPinState : **OCU_EXCLUSIVE_AREA_11**

In function Ocu_Ftm_SetRelativeThreshold : **OCU_EXCLUSIVE_AREA_12**

In function Ocu_Ftm_SetAbsoluteThreshold : **OCU_EXCLUSIVE_AREA_13**

In function Ocu_Ftm_SetClockMode : **OCU_EXCLUSIVE_AREA_14**

In function Ocu_Ftm_ProcessCommonInterrupt : **OCU_EXCLUSIVE_AREA_15**

Table 5-1. Exclusive Areas

	OCU_EXCLUSIVE_AREA_08	OCU_EXCLUSIVE_AREA_09	OCU_EXCLUSIVE_AREA_10	OCU_EXCLUSIVE_AREA_11	OCU_EXCLUSIVE_AREA_12	OCU_EXCLUSIVE_AREA_13	OCU_EXCLUSIVE_AREA_14	OCU_EXCLUSIVE_AREA_15	Interrupt Service Routines Critical Regions(composed diagram)

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

OCU_EXCLUSIVE_AREA_08	X	X	X	X				X	
OCU_EXCLUSIVE_AREA_09	X	X	X					X	
OCU_EXCLUSIVE_AREA_10	X	X	X					X	
OCU_EXCLUSIVE_AREA_11	X			X					
OCU_EXCLUSIVE_AREA_12					X	X			
OCU_EXCLUSIVE_AREA_13					X	X			
OCU_EXCLUSIVE_AREA_14							X		
OCU_EXCLUSIVE_AREA_15	X	X	X					X	
Interrupt Service Routines Critical Regions(composed diagram)									

5.2 Peripheral Hardware Requirements

For S32K14X controllers, Ocu functionality is provided by the and FTM modules.

5.3 ISR to configure within OS – dependencies

The following ISR's are used by the Ocu driver:

Table 5-2. FTM ISR's

ISR Name	CM4 Hardware interrupt vector
For FTM0	
ISR(FTM_0_CH_0_CH_1_ISR)	99
ISR(FTM_0_CH_2_CH_3_ISR)	100
ISR(FTM_0_CH_4_CH_5_ISR)	101
ISR(FTM_0_CH_6_CH_7_ISR)	102
For FTM1	
ISR(FTM_1_CH_0_CH_1_ISR)	105
ISR(FTM_1_CH_2_CH_3_ISR)	106
ISR(FTM_1_CH_4_CH_5_ISR)	107
ISR(FTM_1_CH_6_CH_7_ISR)	108
For FTM2	
ISR(FTM_2_CH_0_CH_1_ISR)	111
ISR(FTM_2_CH_2_CH_3_ISR)	112
ISR(FTM_2_CH_4_CH_5_ISR)	113
ISR(FTM_2_CH_6_CH_7_ISR)	114
For FTM3	
ISR(FTM_3_CH_0_CH_1_ISR)	117
ISR(FTM_3_CH_2_CH_3_ISR)	118
ISR(FTM_3_CH_4_CH_5_ISR)	119
ISR(FTM_3_CH_6_CH_7_ISR)	120
For FTM4	
ISR(FTM_4_CH_0_CH_1_ISR)	123
ISR(FTM_4_CH_2_CH_3_ISR)	124
ISR(FTM_4_CH_4_CH_5_ISR)	125
ISR(FTM_4_CH_6_CH_7_ISR)	126
For FTM5	
ISR(FTM_5_CH_0_CH_1_ISR)	129
ISR(FTM_5_CH_2_CH_3_ISR)	130
ISR(FTM_5_CH_4_CH_5_ISR)	131
ISR(FTM_5_CH_6_CH_7_ISR)	132
For FTM6	
ISR(FTM_6_CH_0_CH_1_ISR)	135
ISR(FTM_6_CH_2_CH_3_ISR)	136
ISR(FTM_6_CH_4_CH_5_ISR)	137
ISR(FTM_6_CH_6_CH_7_ISR)	138
For FTM7	

Table continues on the next page...

Table 5-2. FTM ISR's (continued)

ISR Name	CM4 Hardware interrupt vector
ISR(FTM_7_CH_0_CH_1_ISR)	141
ISR(FTM_7_CH_2_CH_3_ISR)	142
ISR(FTM_7_CH_4_CH_5_ISR)	143
ISR(FTM_7_CH_6_CH_7_ISR)	144

5.4 ISR Macro

MCAL drivers use the ISR macro to define the functions that will process hardware interrupts. Depending on whether the OS is used or not, this macro can have different definitions:

a. OS is not used - AUTOSAR_OS_NOT_USED is defined:

i. If USE_SW_VECTOR_MODE is defined:

```
#define ISR(IsrName) void IsrName(void)
```

In this case, drivers' interrupt handlers are normal C functions and the prolog/epilog handle the context save and restore.

ii. If USE_SW_VECTOR_MODE is not defined:

```
#define ISR(IsrName) INTERRUPT_FUNC void IsrName(void)
```

In this case, drivers' interrupt handlers must save and restore the execution context.

Custom OS is used - AUTOSAR_OS_NOT_USED is not defined

```
#define ISR(IsrName) void OS_isr_##IsrName()
```

In this case, OS is handling the execution context when an interrupt occurs. Drivers' interrupt handlers are normal C functions.

Other vendor's OS is used - AUTOSAR_OS_NOT_USED is not defined. Please refer to the OS documentation for description of the ISR macro.

5.5 Other AUTOSAR modules - dependencies

Development Error Tracer:

This module is necessary for enabling Development error detection. The API function used is Det_ReportError(). The activation / deactivation of Development error detection is configurable using the

'OcuDevErrorDetect' configuration parameter.

Diagnostic Event Manager:

This module is necessary for enabling Production error detection. The API function used is Dem_ReportErrorStatus ().

Mcu:

MCU module shall be initialized before using Ocu.

This module is required for setting the FTM global Pre-scalar value and clock.

Mcl:

Mcl module shall be initialized before using Ocu.

This module is used to obtain the common interrupts sources.

Port:

PORT module shall configure the FlexOcu and FTM channels which are used by the Ocu driver.

EcuC:

This module is necessary for handling Postbuild Variant. This module allows users to configure multiple configuration

Configuration dependency to other module:

Care must be used not to allocate the same FTM channels to other MCAL drivers (ICU/ GPT).

5.6 Data cache restriction

None

5.7 User Mode support

There is no retriction when running from user mode for all OCU IPs. Therefore no further actions are needed in OCU driver.

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler

None.

6.2 API Requirements

None.

6.3 Calls to Notification Functions, Callbacks, Callouts

The Ocu Driver provides a notification per channel that is called whenever the selected edges are generated. The notifications can be configured as pointers to user defined functions. If notification is not desired for a specific channel then 'NULL_PTR' or 'NULL' shall be configured. The syntax of this function is as follows: void Ocu_Notification_#channel(void) An extern declaration of the notification functions is available in Ocu_PBCfg.c. The notification functions have to be implemented by the user.

Chapter 7
Memory Allocation

7.1 Sections to be defined in MemMap.h

Tables descibe Sections to be defined in MemMap.h:

Table 7-1. Sectionto be define

<Section name>	Tyep of section	Description
OCU_START_SEC_CONFIG_DATA_<ALIGNMENT>	Configuration Data	Start of Memory Section for Config Data.
OCU_STOP_SEC_CONFIG_DATA_<ALIGNMENT>	Configuration Data	End of Memory Section for Config Data.
OCU_START_SEC_CODE	Code	Start of memory Section for Code in Flash.
OCU_STOP_SEC_CODE	Code	Stop of memory Section for Code in Flash.
OCU_START_SEC_RAMCODE	Code	Start of memory Section for Code in Ram.
OCU_STOP_SEC_RAMCODE	Code	Stop of memory Section for Code in Ram.
OCU_START_SEC_VAR_< INIT_POLICY>_<ALIGNMENT>	Variables	Start of memory Section for Variables.
OCU_STOP_SEC_VAR_ <INIT_POLICY>_<ALIGNMENT>	Variables	Stop of memory Section for Variables.
OCU_START_SEC_CONST_<ALIGNMENT>	Constant data	Start of memory Section for Constant.
OCU_STOP_SEC_CONST_<ALIGNMENT>	Constant data	Stop of memory Section for Constant.

Which the shortcut '<ALIGNMENT >' means the variable alignment. In order to avoid memory gaps in the allocation variables are allocated according their size. Possible ALIGNMENT postfixes are described in the table at the end of this section.

The shortcut '<INIT_POLICY>' means the initialization policy of variables. Possible '<INIT_POLICY>' postfixes are described in the table at the end of this section.

Tables descibe value range of shortcut ALIGMENT, INIT_POLICY:

Table 7-2. Range of <ALIGNMENT>

<ALIGMENT>	Description
BOOLEAN	Used for variables and constants of size 1 bit
8	Used for variables and constants which have to be aligned to 8 bit. For instance used for variables of size 8 bit or used for composite data types: arrays, structs and unions containing elements of maximum 8 bits
16	Used for variables and constants which have to be aligned to 16 bit. For instance used for variables of size 16 bit or used for composite data types: arrays, structs and unions containing elements of maximum 16 bits
32	Used for variables and constants which have to be aligned to 32 bit. For instance used for variables of size 32 bit or used for composite data types: arrays, structs and unions containing elements of maximum 32 bits
UNSPECIFIED	Used for variables, constants, structure, array and unions when SIZE (alignment) does not fit the criteria of 8,16 or 32 bit. For instance used for variables of unknown size

Table 7-3. Range of <INIT_POLICY>

<INIT_POLICY>	Description
NO-INIT	Used for variables that are never cleared and never initialized by start up code (BSS)
INIT	Used for variables that are initialized with values after every reset

7.2 Linker command file

Memory shall be allocated for every section defined in Ocu_MemMap.h

Chapter 8
Configuration parameters considerations

Configuration parameter class for Autosar Ocu driver fall into the following variants as defined below:

8.1 Configuration Parameters

Specifies whether the configuration parameter shall be of configuration class Post Build.

Table 8-1. Configuration Parameters

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
Ocu	IMPLEMENTATION_CONFIG_VARIANT	Pre Compile parameter for all Variants of Configuration	Pre Compile
OcuConfigurationOfOptApiServices	OcuDeInitApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	OcuGetCounterApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	OcuNotificationSupported	Pre Compile parameter for all Variants of Configuration	Pre Compile
	OcuSetAbsoluteThresholdApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	OcuSetPinActionApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	OcuSetPinStateApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	OcuSetRelativeThresholdApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	OcuVersionInfoApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
OcuGeneral	OcuDevErorDetect	Pre Compile parameter for all Variants of Configuration	Pre Compile
	OcuEnableDualClockMode	Pre Compile parameter for all Variants of Configuration	Pre Compile or Post Build
OcuConfigSet/OcuChannel	OcuChannelId	VariantPC or VariantPB	Pre Compile

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	OcuAssignedHardwareChannel	VariantPC or VariantPB	Pre Compile or Post Build
	OcuDefaultThreshold	VariantPC or VariantPB	Pre Compile or Post Build
	OcuHardwareTriggeredAdc	VariantPC or VariantPB	Current implementation does not support
	OcuHardwareTriggeredDMA	VariantPC or VariantPB	Current implementation does not support
	OcuMaxCounterValue	VariantPC or VariantPB	Pre Compile or Post Build
	OcuNotification	VariantPC or VariantPB	Pre Compile or Post Build
	OcuOuptutPinUsed	VariantPC or VariantPB	Pre Compile or Post Build
	OcuOutputPinDefaultState	VariantPC or VariantPB	Pre Compile or Post Build
	OcuOutputPinAction	VariantPC or VariantPB	Pre Compile or Post Build
	OcuUseMcuReferencePoint	VariantPC or VariantPB	Pre Compile or Post Build
	OcuChannelTickDuration	VariantPC or VariantPB	Pre Compile or Post Build
	OcuMcuClockReferencePoint	VariantPC or VariantPB	Pre Compile or Post Build
OcuChannelConfigSet/ OcuHwIntrruptConfigList	OcuIsrHwId	VariantPC or VariantPB	Pre Compile or Post Build
	OcuIsrEnable	VariantPC or VariantPB	Pre Compile or Post Build
CommonPublishedInformation	ArReleaseMajorVersion	VariantPC or VariantPB	Pre Compile or Post Build
	ArReleaseMinorVersion	VariantPC or VariantPB	Pre Compile or Post Build
	ArReleaseRevisionVersion	VariantPC or VariantPB	Pre Compile or Post Build
	ModuleId	VariantPC or VariantPB	Pre Compile or Post Build
	SwMajorVersion	VariantPC or VariantPB	Pre Compile or Post Build
	SwMinorVersion	VariantPC or VariantPB	Pre Compile or Post Build
	SwPatchVersion	VariantPC or VariantPB	Pre Compile or Post Build
	VendorApiInfix	VariantPC or VariantPB	Pre Compile or Post Build
	VendorId	VariantPC or VariantPB	Pre Compile or Post Build

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating Output Compare Unit :

- Generate the required Ocu configurations. For more details refer to section Files required for Compilation
- Allocate proper memory sections in Ocu_MemMap.h and linker command file. For more details refer to section
- Compile & build the Ocu with all the dependent modules. For more details refer to section Building the Driver

Chapter 10
ISR Reference

None.

Chapter 11
External Assumptions for OCU driver

The section presents requirements that must be complied with when integrating OCU driver into the application.

[SWS_Ocu_00004]

<< Std_Types.h shall include Compiler.h and Platform_Types.h. >>

NOTE

Not part of Ocu driver

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Document Number IM2OCUASR4.2 Rev0002R1.0.2
Revision 1.0

AUTOSAR