Integration Manual

for S32K14X DIO Driver

Document Number: IM2DIOASR4.2 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 Dio 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
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
DEM	Diagnostic Event Manager
DET	Development Error Tracer
C/CPP	C and C++ Source Code
VLE	Variable Length Encoding
N/A	Not Applicable
MCU	Micro Controller Unit
DIO	Digital Input Output

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	Specification of Dio 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 Dio driver for NXP SemiconductorsS32K14X . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The Dio 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 GCC Compiler/Linker/Assembler Options

Table 3-4. 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.

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
-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 3-5. 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 3-6. 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.

Table continues on the next page...

Table 3-6. Linker Options (continued)

Option	Description
-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

3.1.3 IAR Compiler/Linker/Assembler Options

Table 3-7. 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.
--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 3-8. 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 3-9. 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.

3.2 Files required for Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the Dio 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.

Dio Files

- ..\Dio_TS_T40D2M10I2R0\include\Dio.h
- ..\Dio_TS_T40D2M10I2R0\include\Dio_EnvCfg.h
- ..\Dio_TS_T40D2M10I2R0\include\Dio_Gpio.h
- ..\Dio_TS_T40D2M10I2R0\include\Dio_Ipw.h
- ..\Dio_TS_T40D2M10I2R0\include\Reg_eSys_Gpio.h
- ..\Dio_TS_T40D2M10I2R0\src\Dio.c
- ..\Dio_TS_T40D2M10I2R0\src\Dio_Gpio.c

Dio Generated Files

- Dio_Cfg.c - This file should be generated by the user using a configuration tool for compilation.
- Dio_Cfg.h - This file should be generated by the user using a configuration tool for compilation.

As a deviation from standard:

- Dio_Cfg.c - This file will contain the definition for all configuration structures containing only variables that are not variant aware, configured and generated only once. This file alone does not contain the whole structure needed by Port_Init function to configure the driver. Based on the number of variants configured in the EcuC, there can be more than one configuration structure for one module even for PreCompile variant.

Files from Base common folder

- ..\Base_TS_T40D2M10I2R0\include\Compiler.h
- ..\Base_TS_T40D2M10I2R0\include\Compiler_Cfg.h
- ..\Base_TS_T40D2M10I2R0\include\CompilerDefinition.h
- ..\Base_TS_T40D2M10I2R0\include\ComStack_Cfg.h
- ..\Base_TS_T40D2M10I2R0\include\ComStack_Types.h
- ..\Base_TS_T40D2M10I2R0\include\Dio_MemMap.h
- ..\Base_TS_T40D2M10I2R0\include\Mcal.h
- ..\Base_TS_T40D2M10I2R0\include\Platform_Types.h
- ..\Base_TS_T40D2M10I2R0\include\Reg_eSys.h
- ..\Base_TS_T40D2M10I2R0\include\RegLockMacros.h
- ..\Base_TS_T40D2M10I2R0\include\SilRegMacros.h
- ..\Base_TS_T40D2M10I2R0\include\Soc_Ips.h
- ..\Base_TS_T40D2M10I2R0\include\Std_Types.h
- ..\Base_TS_T40D2M10I2R0\include\StdRegMacros.h

Files from Det folder:

- ..\Det_TS_T40D2M10I2R0\include\Det.h
- ..\Det_TS_T40D2M10I2R0\src\Det.c

Files from Rte folder:

- ..\Rte_TS_T40D2M10I2R0\include\SchM_Dio.h
- ..\Rte_TS_T40D2M10I2R0\src\SchM_Dio.c

3.3 Setting up the Plug-ins

The Dio 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 module folder:

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\Dio_TS_T40D2M10I2R0\config\Dio.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k118_lqfp48.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k118_lqfp64.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k142_lqfp48.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k142_lqfp64.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k142_lqfp100.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k144_lqfp48.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k144_lqfp64.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k144_lqfp100.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k144_mapbga100.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k146_lqfp64.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k146_lqfp100.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k146_mapbga100.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k146_lqfp144.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k148_lqfp100.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k148_mapbga100.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k148_lqfp144.epd
  - ..\Dio_TS_T40D2M10I2R0\autosar\Dios32k148_lqfp176.epd
- Code Generation Templates for parameters without variation points:
  - ..\Dio_TS_T40D2M10I2R0\generate_PC\include\Dio_Cfg.h
  - ..\Dio_TS_T40D2M10I2R0\generate_PC\src\Dio_Cfg.c

Steps to generate the configuration:

1. Copy the module folders Dio_TS_T40D2M10I2R0, Base_TS_T40D2M10I2R0, Resource_TS_T40D2M10I2R0, Det_TS_T40D2M10I2R0, EcuC_TS_T40D2M10I2R0, Rte_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.

Chapter 4
Function calls to module

4.1 Function Calls during Start-up

None.

4.2 Function Calls during Shutdown

None.

4.3 Function Calls during Wake-up

None.

Chapter 5
Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

In the current implementation, DIO is using the services of Run-Time Environment (RTE) for entering and exiting the critical regions. RTE implementation is done by the integrators of the MCAL using OS or non-OS services.

For testing the Dio driver, stubs are used for RTE.

5.2 Peripheral Hardware Requirements

The Dio driver uses GPIO peripheral.

Port pins that are available on a particular package are descibed in the S32K14X Reference Manual.

The formula for calculating port and channel number for PINx is:

PORT = PINx / 32

CHANNEL = PINx % 32

5.3 ISR to configure within OS – dependencies

None.

5.4 ISR Macro

None.

5.5 Other AUTOSAR modules - dependencies

- **PORT:** The PORT module is used to configure the port pins with the needed modes, before they are used by the DIO module
- **DET:** The DET module is used for enabling Development error detection. The API function used is Det_ReportError(). The activation / deactivation of Development error detection is configurable using the DioDevErrorDetect configuration parameter.
- **BASE:** The BASE module contains the common files/definitions needed by all MCAL modules.
- **RESOURCE:** The RESOURCE module is used to select microcontroller's derivatives.
- **RTE:** The RTE module is used to manage the exclusive area inside DIO driver.
- **ECUC:** The ECUC module is used for ECU configuration. MCAL modules need ECUC to retrieve the variant information.
- **MCU:** The MCU driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required by other MCAL software modules. The clocks need to be initialized prior to using the DIO driver

5.6 Data cache restriction

None

5.7 User Mode support

No special measures need to be taken to run **DIO** module in user mode. The Dio driver code can be executed at any time in both supervisor and user mode.

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

None.

Chapter 7
Memory Allocation

7.1 Sections to be defined in Dio_MemMap.h

Table 7-1. MemMap sections present in the Dio driver code

Section name	Section type	Description
DIO_START_SEC_CODE	Code	Start of Memory Section for Code
DIO_STOP_SEC_CODE	Code	End of Memory Section for Code
DIO_START_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	Start of Memory Section for Config Data
DIO_STOP_SEC_CONFIG_DATA_UNSPECIFIED	Configuration Data	End of Memory Section for Config Data

7.2 Linker command file

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

Chapter 8
Configuration parameters considerations

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

8.1 Configuration Parameters

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

Table 8-1. Configuration Parameters

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
DioGeneral			
	DioDevErrorDetect	Pre Compile	Pre Compile
	DioVersionInfoApi	Pre Compile	Pre Compile
	DioReversePortBits	Pre Compile	Pre Compile
	DioFlipChannelApi	Pre Compile	Pre Compile
	DioReadZeroForUndefinedPortPins	Pre Compile	Pre Compile
	DioMaskedWritePortApi	Pre Compile	Pre Compile
	DioEnableUserModeSupport	Pre Compile	Pre Compile
DioPort			
	DioPortId	Pre Compile	Pre Compile
DioChannel			
	DioChannelId	Pre Compile	Pre Compile
DioChannelGroup			
	DioChannelGroupIdentification	Pre Compile	Pre Compile
	DioPortBitNumber	Pre Compile	Pre Compile
	DioPortOffset	Pre Compile	Pre Compile
	DioPortMask	Pre Compile	Pre Compile
CommonPublishedInformation			

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	ArReleaseMajorVersion	Pre Compile	Pre Compile
	ArReleaseMinorVersion	Pre Compile	Pre Compile
	ArReleaseRevisionVersion	Pre Compile	Pre Compile
	ModuleId	Pre Compile	Pre Compile
	SwMajorVersion	Pre Compile	Pre Compile
	SwMinorVersion	Pre Compile	Pre Compile
	SwPatchVersion	Pre Compile	Pre Compile
	VendorApiInfix	Pre Compile	Pre Compile
	VendorId	Pre Compile	Pre Compile

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating Digital Input Output :

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

Chapter 10
ISR Reference

None.

Chapter 11
External Assumptions for DIO driver

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

[SMCAL_CPR_EXT172]

<< If the platform supports configuring the port pins as Input-Output then the pins for which the application intends to use Dio_FlipChannel() function at run-time should be configured as Input-Output as in this case FlipChannel() function writes the output buffer and returns the value from the input buffer. If the platform supports configuring the port pins as either Input or Output then the pins for which the application intends to use Dio_FlipChannel() function at run-time should be configured as Output and in this case FlipChannel() function writes the output buffer and returns the value from the output buffer. >>

[SWS_Dio_00061]

<< The Dio module shall not provide APIs for overall configuration and initialization of the port structure which is used in the Dio module. These actions are done by the PORT Driver Module. >>

NOTE

DIO module implementation shall be made independent of PORT configuration. (DIO_SW001)

[SWS_Dio_00102]

<< The Dio module's user shall only use the Dio functions after the Port Driver has been initialized. Otherwise the Dio module will exhibit undefined behavior. >>

[SWS_Dio_00127]

<< The Port module shall configure a DIO channel as input or output [SWS_Dio_00001 and SWS_Dio_00002]. >>

[SWS_Dio_00001]

<< The Dio module shall not provide an interface for initialization of the hardware. The Port Driver performs this. >>

NOTE

DIO module implementation shall be made independent of PORT configuration. (DIO_SW001)

[SWS_Dio_00002]

<< The PORT driver shall provide the reconfiguration of the port pin direction during runtime. >>

NOTE

DIO module implementation shall be made independent of PORT configuration. (DIO_SW001)

[SWS_Dio_00017]

<< For parameter values of type Dio_ChannelType, the Dio's user shall use the symbolic names provided by the configuration description.Furthermore, SWS_Dio_00103 applies to the type Dio_ChannelType. >>

NOTE

If supported by the external application, parameter validation at production time is no more needed.

[SWS_Dio_00020]

<< For parameter values of type Dio_PortType, the user shall use the symbolic names provided by the configuration description.Furthermore, SWS_Dio_00103 applies to the type Dio_PortType. >>

NOTE

If supported by the external application, parameter validation at production time is no more needed.

[SWS_Dio_00022]

<< For parameter values of type Dio_ChannelGroupType, the user shall use the symbolic names provided by the configuration description.Furthermore, SWS_Dio_00056 applies to the type Dio_ChannelGroupType. >>

NOTE

If supported by the external application, parameter validation at production time is no more needed.

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 IM2DIOASR4.2 Rev0002R1.0.2
Revision 1.0