
User Manual

For S32K1XX MCAL Sample Application

Document Number: UMSAASR4.2R1.0.1
Rev. 1.1



Contents

Section number	Title	Page
Chapter 1		
Revision History		
Chapter 2		
About this Manual		
2.1	Acronyms and Definitions.....	7
2.2	Reference List.....	7
Chapter 3		
Installation Steps		
3.1	Hardware Installation.....	9
3.2	Software Installation.....	10
3.2.1	Tresos Project Installation.....	11
3.2.2	MCAL Application Configuration.....	13
Chapter 4		
Sample Application Example Description		
4.1	The application software functionality.....	15
4.2	Description of the LEDs and Buttons functionality.....	16
Chapter 5		
Building the Sample Application Example		
5.1	Building the Sample Application example.....	19
5.2	Building with different compilers.....	19
5.3	Building for different run-modes.....	20
5.4	Clean Object and Linker Output Files.....	20
5.5	Modifying the Configuration in Tresos Studio.....	21

Chapter 1

Revision History

Table 1-0. Revision History

Revision	Date	Author	Description
1.0	04/06/2018	Stefan Tataru	1.0.0 Release
1.1	13/06/2018	Stefan Tataru	1.0.1 Release

Chapter 2

About this Manual

This User Manual describes utilization of the sample application for S32K1XX microcontroller with Autosar MCAL 4.2 version RTM 1.0.1.

2.1 Acronyms and Definitions

Table 2-1. Acronyms and Definitions

Abbreviation / Acronym	Description
DIO	Digital Input Output Driver
PORT	Port Driver
BSW	Basic Software
ADC	Analog Digital Converter
FEE	Flash EEPROM Emulation
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
ISR	Interrupt Service Routine
OS	Operating System
GUI	Graphical User Interface
API	Application Programming Interface
EcuM	ECU state Manager
WDG	Watchdog Driver
PLL	Phase Lock Loop
LED	Light Emitting Diode
PB Variant	Post Build Variant
LT Variant	Link Time Variant
PC Variant	Pre Compile Variant

Reference List

2.2 Reference List

Table 2-2. Reference List

#	Items	Version
1	S32K1XX Microcontroller Reference Manual	S32K1XX_RM_Rev8.1

Chapter 3

Installation Steps

3.1 Hardware Installation

The hardware installation describes setup of the two Evaluation Boards.

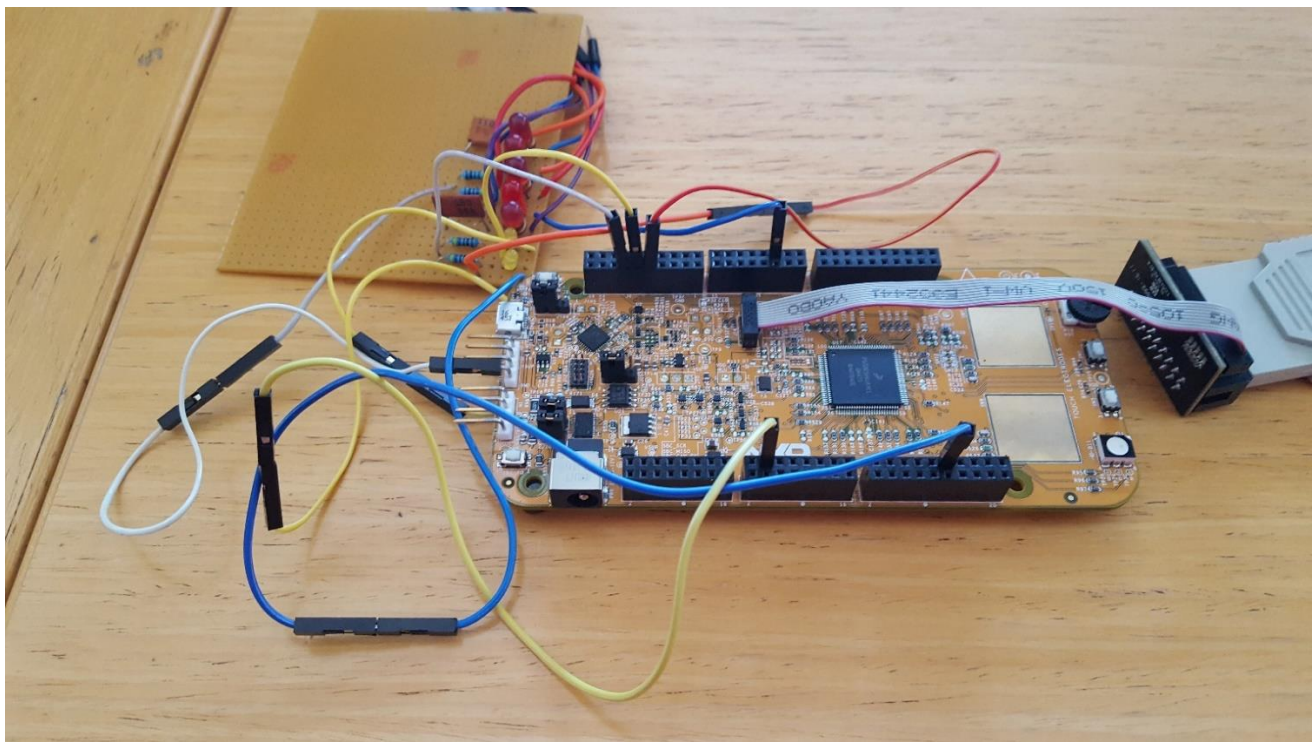


Figure 3-1. SCH-29248 Rev B1 Evaluation Board

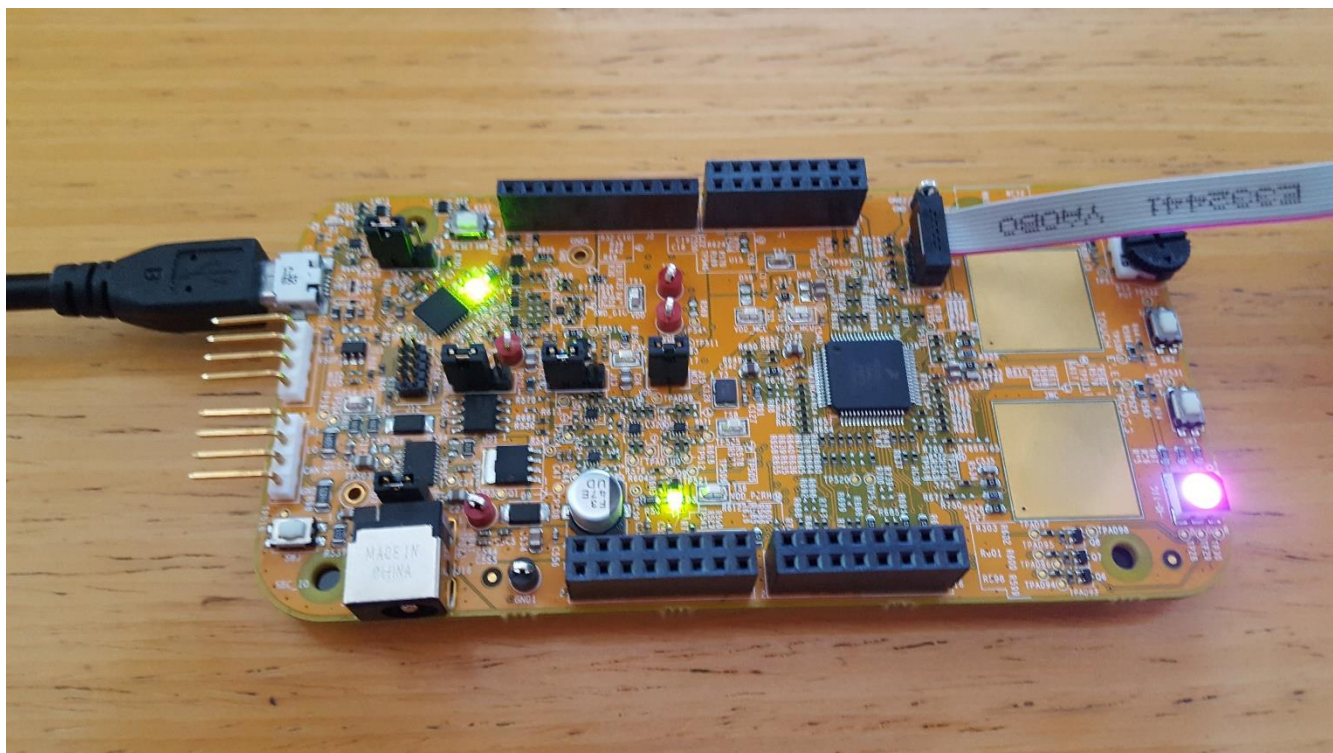


Figure 3-2. SCH-29945 Rev B Evaluation Board

3.2 Software Installation

Please install the MCAL package on your computer. The Integration Framework Sample Application package is delivered as a MCAL-type plugin:

IntegrationFramework_TS_T40D2M10I1R0

The Application plugin has the following folder structure:

Chapter 3 Installation Steps

Folder or file	Description
-autosar	contains the IntegrationFramework.epd file
-auxiliary	contains files and folders required to build and start the framework application
-build	Contains the cmm folder and the batch files required to start build system
-bin subfolder	generated object files and linker output files are stored into this folder
-cmm subfolder	contains Lauterbach T32 cmm script files
-toolchains subfolder	contains linker-scrips folder, make folder and startup folder
-linkfiles subfolder	contains linker-scrips files
-make folder	contains make-files needed to build the application for all available CPU cores and compilers
-startup folder	contains source files and headers needed to start the application (startup code and interrupt vector definitions for each CPU type and available compiler)
-config folder	contains the configuration XDM template file for EB tresos.
-generate_PC	contains the configuration generation templates
- src folder	contains the source code files for all components of the framework application
- include folder	contains header files for all components of the framework application
- makefile file	the framework application makefile
- make.bat file	launches the make command
- launch.bat file	contains path to the Tresos Studio installation and launches the make.bat file
- Tresos folder/workspace	contains the Tresos project with the application configuration

Note: Since the application framework is NOT production code it is not delivered in the same plugins folder as the rest of the MCAL drivers. In order to build and run the application the user must copy the application plugin folder

“IntegrationFramework_TS_T40D2M10I1R0” in the same folder where the rest of the MCAL plugins are located.

3.2.1 Tresos Project Installation

The following procedure requires that the user has EB Tresos Studio installed.

Procedure:

1. Make sure that all MCAL plugins are already installed in the Tresos Studio pluginsdirectory
2. Open Tresos Studio
3. Import Sample application project
 - a. Click on "File" and select "Import"
 - b. Select "Existing Projects into Workspace" and click on "Next" button as shown in Figure 4-3. Import Window - the First View
 - c. Next steps are depicted in Figure 4-4. Import Window - the Second View
 - Select "Select root directory" and click on "Browse"
 - Select the location of the [project] folder in the installed Sample application packagefolder (Tresos/workspace/[project])
 - Select "Copy projects into workspace"
 - Click on "Finish" button

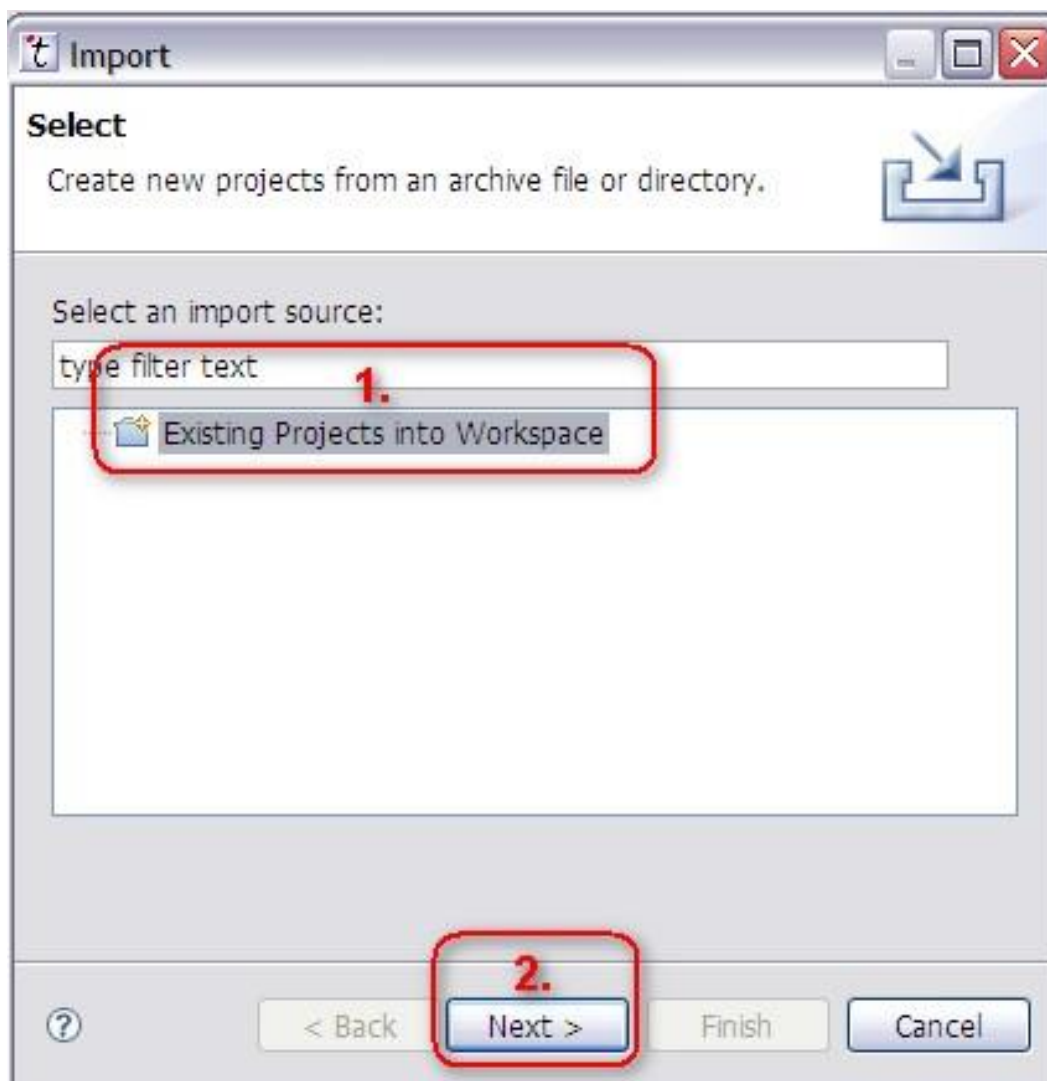


Figure 3-3. Import Window - the First View

Chapter 3 Installation Steps

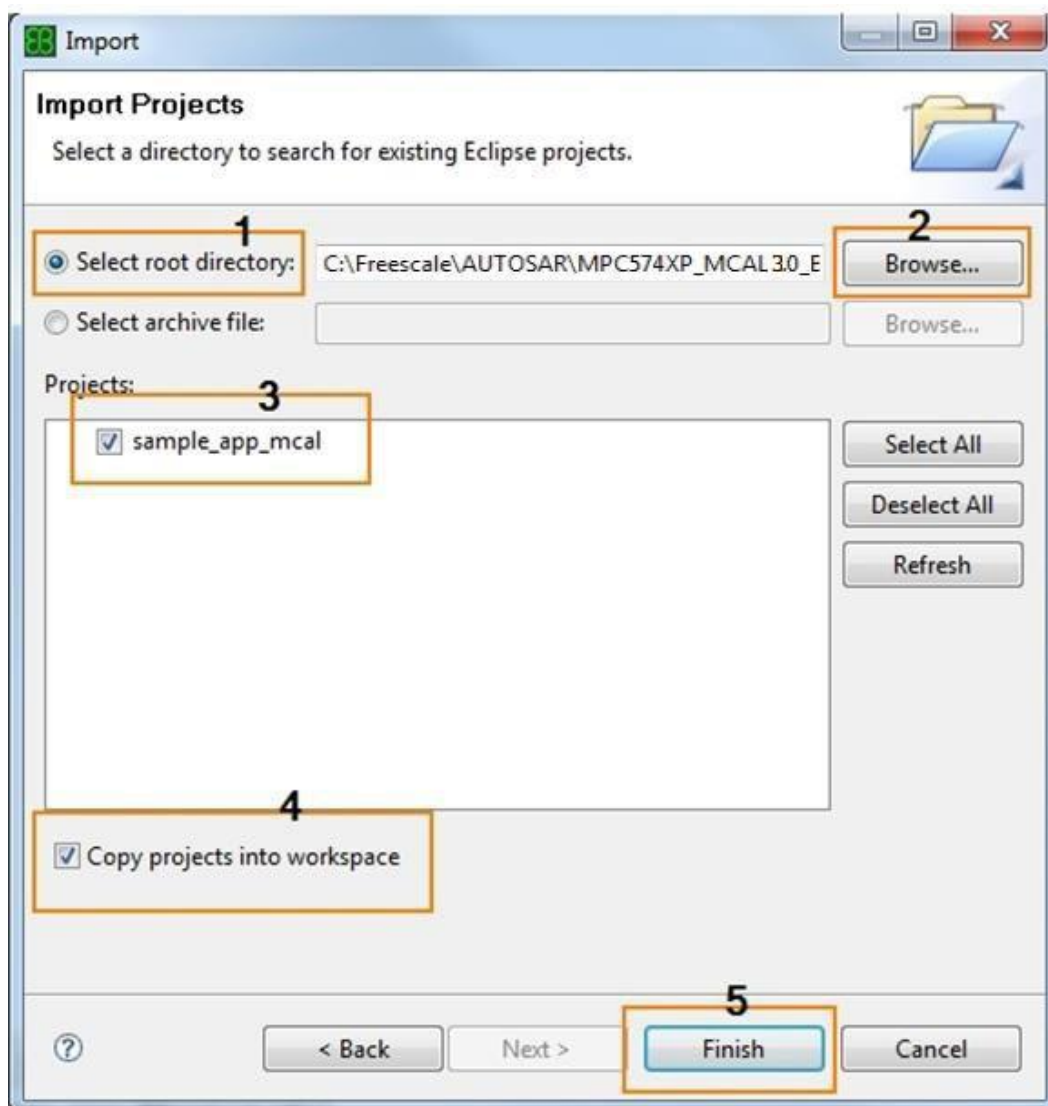


Figure 3-4. Import Window - the Second View

3.2.2 MCAL Application Configuration

The following procedure requires that the user has EB Tresos Studio installed and the toolchains versions specified in the MCAL Release Notes.

The toolchain that will be used needs to be installed for correct operation and the path to the installation location shall be added into the system environment variable(s):

- GHS_DIR

Ex: SET GHS_DIR= C:/tools/ghs/ARM_MULTI_7.1.4COMPILER_2017.1.4

- LINARO_DIR

Ex: SET LINARO_DIR= C:/tools/GCC/gcc-6.3-arm32-eabi

- IAR_DIR

Ex: SET IAR_DIR= C:/tools/IARSystem/EmbeddedWorkbench8.0/arm

- TRESOS_DIR for setting up the path to installed EB Tresos folder

Ex: TRESOS_DIR=C:/Tools/EB/v23.0.0

- PLUGINS_DIR for defining the path to all source file to be build

Ex: SET PLUGINS_DIR=C:/Tools/EB/v23.0.0/plugins

- TRESOS_WORKSPACE_DIR for defining the path to all generated configuration files

Ex: TRESOS_WORKSPACE_DIR=C:/Tools/EB/v23.0.0/workspace/

lighting_S32K144_4.2_RTM1.0.1 /output

Note There will be two workspaces depending on the Platform in use (S32K118/S32K144)

Note The path to the toolchain must not contain spaces. In case the compiler is installed into a path with spaces, the variable must be set with the "short" folder name (8.3 version of the file name that can be displayed with dir /X in command prompt)

Procedure:

1. Open launch.bat file in a text editor and specify the EB Tresos Studio location in the TRESOS_DIR parameter as shown in Figure 4-5. Configuration of the Tresos Studio Location.
2. Make sure that installation location of the compiler is added in the system environment variable (GHS, GCC, IAR)
3. Setup the plugins folder location if the plugins are not installed in the Tresos plugins folder (PLUGINS_DIR)

4. Setup the workspace folder location plugins folder
(TRESOS_WORKSPACE_DIR).

```
:: uncomment line below if you do not set TRESOS_DIR over environment
:: SET TRESOS_DIR=
:: SET GHS_DIR=
:: SET IAR_DIR=
:: SET LINARO_DIR=
:: SET PLUGINS_DIR=
```

Chapter 4

Sample Application Example Description

This application demonstrates an example of usage for the MCAL modules. It is not part of the production code deliverables

4.1 The application software functionality

Initializes MCU module

(Only for S32K144)

- Initializes PLL and configures it to 80MHz.
- Checks whether PLL is locked
- Activates the PLL clock to the MCU clock distribution

(Only for S32K118)

- System clock is set to FIRC

Initializes WDG driver and configures it to OFF MODE.

Initializes PORT module. Pins configuration is show in section PORT and DIO Modules - Pin Configuration and DioChannel Assignment for keys and leds, and in PORT Configuration excluding Leds and Keys

Initialize DIO driver.

Initialize the GPT driver. Gpt channel 0 (LPIT0_CH1) is used to trigger the internal round-robin scheduler.

Initializes the PWM driver and Sample application specific data for this driver.

Initializes the OCU driver. In the configuration provided by the sample application framework the OCU driver is used to trigger ADC at user-configured moments of time.

Initialize the ADC driver and Sample application specific data for this driver

Initializes Integration framework components and starts internal round-robin scheduler. The internal scheduler

Once the scheduler is started the following tasks are executed cyclically:

- Basic Software – IO Driver Abstraction task is called every 10 ms.
- Basic Software – System Driver Abstraction task is called every 10 ms.
- Lighting Application task is called every 20 ms.

a. Basic Software - IO Driver Abstraction task

IODAL component main purpose is to handle and control all I/O capable drivers: ADC, DIO, OCU, ICU, PWM.

IODAL handles Digital I/O's and PWM channels by calling the underlying drivers directly.

ADC Channels are handled by using SW trigger conversion which is executed at precise moments of time define by using a Time-trigger table in OCU. For each ADC group two time-events are required:

- Start conversion event: defined in configuration as a Time Trigger scheduling point
- Read results event: define in configuration as “conversion time”

Note. IoDal configuration is based around IO channel descriptors. For analog channels each descriptor is defined as a unique combination of an ADC group and an ADC channel; for digital/pwm channels each descriptor is define by a single dio/pwm channel.

b. Basic Software - System Driver Abstraction task

SYSDAL component main purpose is to implement core system functionalities (i.e. the internal round-robin scheduler, user interrupt enablement) and to implement the power-up and power-down sequences (similarly to ECUM from AUTOSAR).

User Interrupts and Power Modes are also, defined and configured in SysDal.

The round-robin scheduler is based around a GPT timer and is used to execute application functionality at predefined moments of time. (i.e. every 10 ms call IoDal_Main()...)

c. Lighting Application task

Application task is used to process command inputs signals and calculate outputs values for the all output channels of each lighting instance using the following logic.

- If Output signal is PWM and If Input command signal is ANALOG_INPUT, the duty cycle value for all outputs for that application instance is proportional to the value read from the analog input (i.e. Analog value read from a Potentiometer)
- If Output signal is PWM and If Input command signal is DIGITAL_INPUT, the duty cycle value for all outputs for that application instance is increased by 12.5% every-time the digital input is set from LOW to HIGH. When the duty-cycle value exceeds 100% the duty-cycle is reset to 0% and the cycle will start from that. Basically every-time a button is pressed the duty is increased by 12.5% until it reaches 100%.
- If Output signal is DIGITAL_OUTPUT and If Input command signal is DIGITAL_INPUT, the output signal shall be toggled between LOW and HIGH every time the digital input is set from LOW to HIGH. Basically every-time a button is pressed the output is toggled between LOW and HIGH.

4.2 Description of the LEDs and Buttons functionality

The detailed description of the LEDs and Buttons functionality is depicted in the following table:

PortPin Name	Pin ID (PCR ID)	Pin Mode	Pin Direction	Pin Level	Connected HW	Channel Assignment
PortPin_DOUT0	96	GPIO	Out	Low	LED1	DioChannel_1
PortPin_DOUT1	111	GPIO	Out	Low	LED2	DioChannel_2
PortPin_PWM0	44	FTM0_CH0	Out	Low	LED3	-
PortPin_PWM1	45	FTM0_CH1	Out	Low	LED4	-
PortPin_PWM2	106	FTM2_CH0	OUT	Low	LED5	-
PortPin_DIG_KEY	76	GPIO	In	Low	SW2	Dio_Key
PortPin_ADC0_SE12	78	ADC0_SE12	In	-	POT	-

Table 4-1. PORT and DIO Modules - Pin Configuration and DioChannel Assignment for S32K144 Derivative

PortPin Name	Pin ID (PCR ID)	Pin Mode	Pin Direction	Pin Level	Connected HW	Channel Assignment
PortPin_RGB_GREEN	111	GPIO	Out	Low	RGB_LED	Dio_RGB_Green
PortPin_PWM0	136	FTM0_CH6	Out	Low	RGB_LED	-
PortPin_PWM1	116	FTM0_CH1	Out	Low	RGB_LED	-
PortPin_DIG_KEY1	101	GPIO	In	Low	SW3	Dio_Key1
PortPin_DIG_KEY2	99	GPIO	In	Low	SW2	Dio_Key2

Table 4-1. PORT and DIO Modules - Pin Configuration and DioChannel Assignment for S32K118 Derivative

LEDs and Buttons	Functionality
LED5	Each time SW3 is pressed the duty cycle for LED5 increase with 12.5% until it reach 100% and then reset
LED3	It's duty cycle is control by the Analog Potentiometer(POT)
LED4	It's duty cycle is control by the Analog Potentiometer(POT)
LED1	Output controlled by SW3 (On/Off)
LED2	Output controlled by SW3 (On/Off)
SW2	Input for controlling LED1 LED2 (On/Off) and duty cycle on LED5
POT	Input for controlling the duty cycle of LED3 and LED4

Table 4-2. LEDs and Buttons Functionality for S32K144 Derivative

LEDs and Buttons	Functionality
RGB_LED BLUE	Each time SW2 is pressed the duty cycle for DS2 increase with 12.5% until it reach 100% and then reset
RGB_LED RED	It's duty cycle is control by the Analog Potentiometer(POT)
RGB_LED GREEN	When SW3 is pressed the led turns on, when SW3 is pressed again led turn off

SW2	Input for controlling Led RGB_LED BLUE
SW3	Input for controlling Led RGB_LED GREEN
POT	Controlling the duty cycle of RGB_LED RED

Table 4-2. LEDs and Buttons Functionality for S32K118 Derivative

Chapter 5

Building the Sample Application Example

This section describes the build procedure.

5.1 Building the Sample Application example

Procedure:

1. Open the Windows command prompt window
2. Change the current directory to the sample application folder
3. To build the sample, execute the following command to run launch.bat: launch.bat
4. The object files and linker output file (sample_app_mcal.elf) shall be generated in the /bin subdirectory
5. To execute the sample application, load the executable file placed in the /bin subdirectory to the evaluation board using the Lauterbach debugger and run.cmm script.

Note

The launch.bat file calls the make.bat file and then the GNU make utility is called from the Tressos Studio bin directory.

5.2 Building with different compilers

To build the sample application with a different compiler, use the following parameter for the launch command:

launch.bat TOOLCHAIN=[toolchain]

where [toolchain] can have the values:

- * ghs - default - use the GreenHills Multi compiler
- * linaro - use the Gcc compiler
- * iar - use the Iar compiler

5.3 Building for different run-modes

To build the sample application for a different run-mode, use the following parameter for the launch command:

launch.bat MODE=[run_mode] where

[run_mode] can have the values: * SUPR

- default - run in Supervisor mode

* USER - un in User mode

Note

In order to run in USER mode, all drivers that need to be executed in this mode should have the "Enable User Mode Support" parameter set to 'true' and their configuration files regenerated from Tresos.

Note

In order to run in USER mode, AUTOSAR OS should not be used since it does not allow other run-modes except Supervisor

5.4 Building for different controller derivatives

To build the sample application for a either 3M or 6M derivative, use the following parameter for the launch command:

launch.bat DERIV=[derivative] where [derivative] can have the values:

- * 118 – build for S32K118 variant
- * 144 – build for S32K144 variant

Note

In order to run for either of the variants the user should make sure that the correct resource was selected in Tresos and configurations were correctly generated for that derivative.

Note

It is possible to build with different option by using a command line with more than one parameter.

For instance: running “*launch.bat* TOOLCHAIN=ghs MODE=SUPR DERIV=144 “ will build the code for GHS compiler, Supervisor Mode on S32K144 derivative.

5.5 Clean Object and Linker Output Files

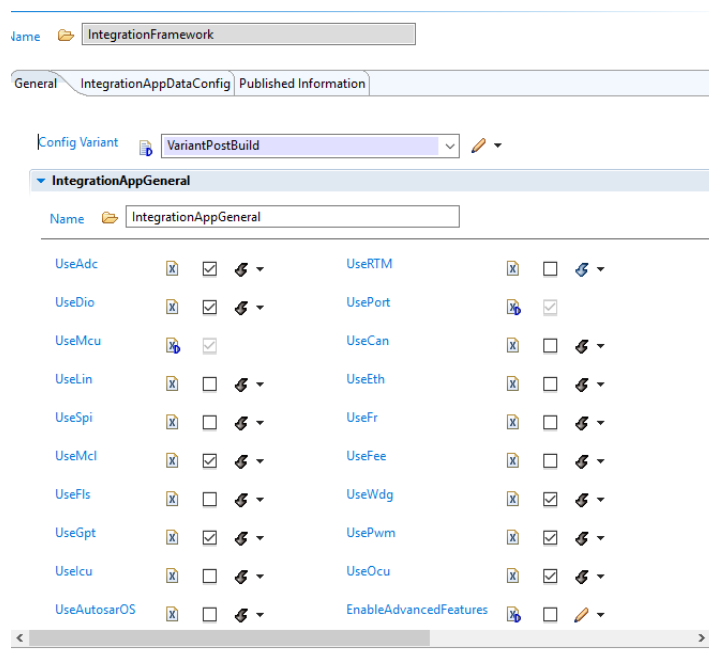
To clean the object and linker output files from the folder /bin, execute the following steps

Procedure:

1. Open the Windows command prompt window
2. Change the current directory to sample application folder
3. Execute the following command *launch.bat clean*
4. The object files and linker output files shall be cleared from the /bin and from the sample application root folders.

5.6 Framework Configuration Parameters

1. **IntegrationAppGeneral** Container – holds driver enablement parameters.



Note: For this version of application only Adc, Dio, Mcu, Gpt, Port, Pwm and Ocu drivers are supported.

2. **IntegrationAppFeaturesConfig** Container – holds application enablement parameters.

IntegrationApp_FeatureList

Name

IntegrationApp_FeatureList

EnableVdrApp		<input type="checkbox"/>		EnableLightingApp		<input checked="" type="checkbox"/>	
EnableMotorControlApp		<input type="checkbox"/>		EnableRteBuffers		<input type="checkbox"/>	
EnableSingleTaskAppRunnable		<input checked="" type="checkbox"/>					

IntegrationAppFeaturesConfig

Note: For this version of application only Lighting App is supported.

3. **AppConfig/LightingControlApp** Container – holds parameters for configuring each lighting instance.

LightingControlApp

Name

LightingControlApp_0

Sample App IO Features

LightingChannels

Output Signal Type

PWM

Sample App IO Features

Name

LightCtrlUserInterface

Comand (input) Signal Type

ANALOG_INPUT


LightControlButton

LightControlCanMsg

LightControlPot

tegrationAppDataConfig_0/BswConfig_0/GeneralConfig_0/IoDAL_AN_POT

- LightCtrlOutputSignalType: specifies what type of output current instance shall use (supported values: PWM, DIGITAL_OUTPUT)
- LightCtrlInputSignalType: specifies what type of input current instance shall use (supported values: ANALOG, DIGITAL_INPUT)
- **LightingChannels** Container – holds parameters for configuring each output channel for the current instance (and their corresponding feedback channels)

Name  LightingControlApp_0

Sample App IO Features LightingChannels

 LightingChannels

Index	Name	IoPwmOutput	Channel...	IoChanne...
0	LightingChannels_0	@ /IntegrationFramework/IntegrationFramework...	X ✓	@ /Integratio...
1	LightingChannels_1	@ /IntegrationFramework/IntegrationFramework...	X ✓	@ /Integratio...

4. **BswConfig/ GeneralConfig/IoDAL** Container holds IoDal channel descriptor configuration for each BSW IO signal.

General

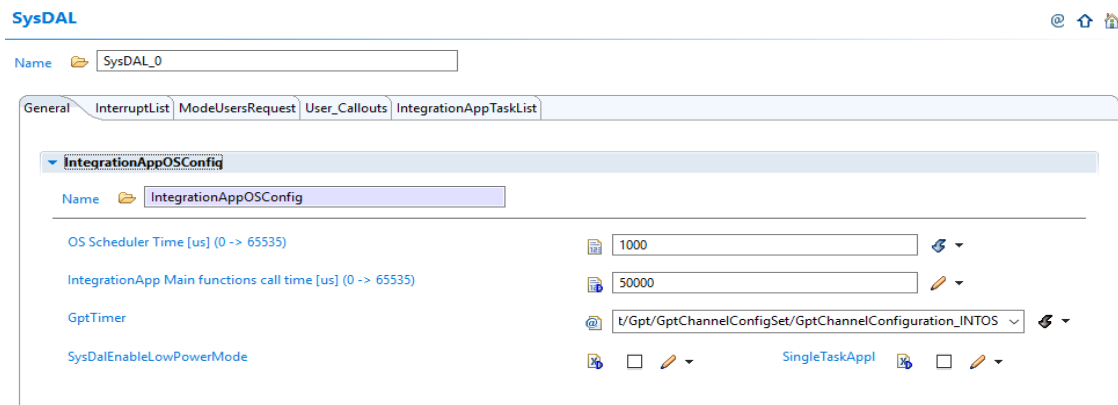
Referenced Signal Direction	Input
Referenced Signal Type	Analog
UseExternalDevice	<input type="checkbox"/>
DigitalSignalLevel	STD_LOW
Analog Group Reference	@ /Adc/AdcConfigSet/AdcHwUnit_0/AdcPotGroup
Analog Channel Reference	@ /dc/AdcConfigSet/AdcHwUnit_0/AdcPotChannel
Analog channel conversion time (in ticks) (0 -> 4294967295)	50
Analog channel trigger type	SW
SW trigger for Analog input	@ /Config_0/TimeTriggerTable_0/SchedulePoints_0
DioRef	
PwmRef	
PWM Type	NORMAL
Sync PWM Channel Mask (0 -> 4294967295)	0
OcuRef	

- ReferenceDirection: determines I/O channel direction (input/output)
- ReferenceType: determines signal type (analog, digital, pwm)

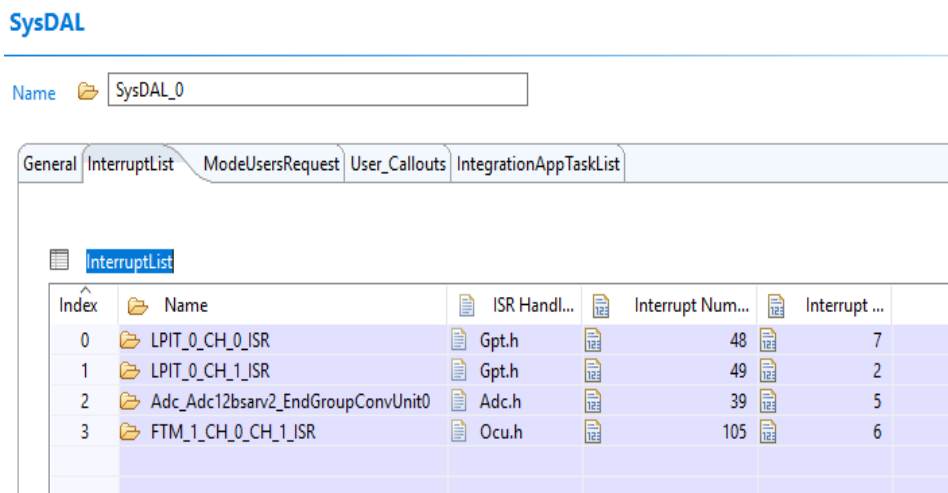
For analog signals, each descriptor is determined by unique combination of references to an Analog Channel and an Analog Group. Also for analog signals conversion time, trigger type and time are required.

For pwm and digital signals only references to underlying drivers are needed.

5. **BswConfig/ GeneralConfig/SysDAL/General** Container holds SysDal global configuration, including the configuration on the internal scheduler.
- 6.



7. **BswConfig/ GeneralConfig/SysDAL/InterruptList** Container holds the interrupt handlers that need to be activated at system level. (in our case only GPT, OCU and ADC interrupts are enabled)



8. **BswConfig/ GeneralConfig/SysDAL/ModeUserRequest** Container holds the configuration for the user mode (not used for current version).
9. **BswConfig/ GeneralConfig/SysDAL/ModeUserRequest** Container hold the list of API's that need to be called by SysDal during initialization and/or de-initialization.

General InterruptList ModeUsersRequest User_Callouts IntegrationAppTaskList					
User_Callouts					
Index	Name	Parameters	Callout H...	Callout A...	
0	Gpt_Init	&GptChan...	Gpt.h	AllDriverInit	
1	Port_Init	&PortConf...	Port.h	AllDriverInit	
2	Det_Init		Det.h	AllDriverInit	
3	Det_Start		Det.h	AllDriverInit	
4	Ocu_Init	&OcuConf...	Ocu.h	AllDriverInit	
5	Adc_Init	&AdcConf...	Adc.h	AllDriverInit	
6	Pwm_Init	&PwmCha...	Pwm.h	AllDriverInit	
7	Pwm_SelectCommonTimeb...	0, 3	Pwm.h	AllDriverInit	
8	IoDal_Init	&IoDal_Co...	IoDal.h	AllDriverInit	
9	Gpt_EnableNotification	1	Gpt.h	AllDriverInit	

10. **BswConfig/ GeneralConfig/SysDAL/IntegrationAppTaskList** Container hold the configuration of the tasks for the internal round-robin scheduler. Up to 7 cyclic tasks and one sigle-shot pre-hook task can be enabled.

SysDAL

Name

General InterruptList ModeUsersRequest User_Callouts IntegrationAppTaskList					
IntegrationAppTaskList					
Index	Name	Sample A...	Task Sche...		
0	SysDal	INTAPP_TA...	10000		
1	IoDal	INTAPP_TA...	10		
2	Rte_Init	INTAPP_PR...	0		
3	Rte	INTAPP_TA...	20		

- For each defined task the user can configure a list of API's that will be called by it.

5.6 Modifying the Configuration in Tresos Studio

Users may change the application configuration according to their needs.

Procedure:

1. Open the EB Tresos Studio GUI
2. Open previously imported Sample Application project
3. Use the Tresos Studio GUI to modify configuration parameter values and save the changes. The value of the External Crystal Frequency parameter can be changed as depicted in Figure 5-1: Modifying the External Crystal Frequency
4. Select the Sample Application project and click on "Generate" button to generate the configuration files.
5. Copy the generated configuration files from workspace/[project]/output/includedirectory into the Sample Application folder /cfg/include.

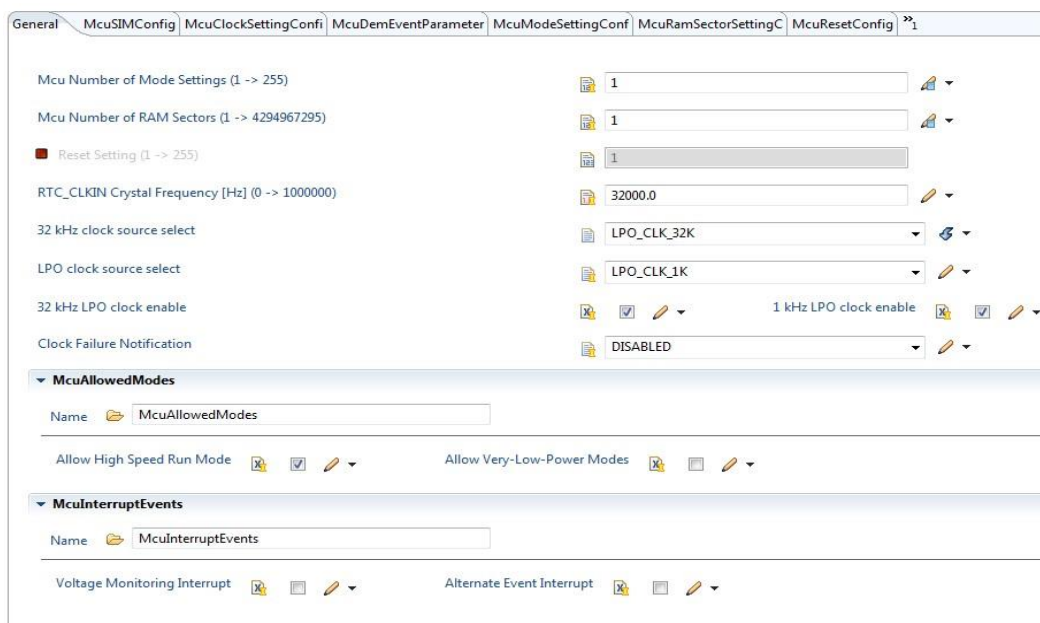


Figure 5-1. Modifying the External Crystal Frequency

**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.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and μ Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2017 NXP B.V.

Document Number UM5AASR4.2R1.0.0
Revision 1.0