Processor SDK RTOS Migration Guide for StarterWare 02.00.xx to 02.01.xx

Introduction

StarterWare 02.01.xx provides a no-OS platform support for AM335x and AM437x. StarterWare provides Device Abstraction Layer (DAL) libraries and peripheral/board level examples that demonstrate the capabilities of the peripherals. Module is being retained for backward compatibility reason in Processor SDK. Document includes migration details across different versions.

Overview of differences

The below table gives a high level overview of differences. Additional details are given in later sections.

Feature	**StarterWare 02.00.xx**	**StarterWare 02.01.xx**
Devices supported	Single SoC only within a single package	Multiple SoCs within a single package
Board support	Different binary for each board / EVM	Single binary for multiple boards of a single SoC
Build system	build: Contains all makefiles, project files etc. for building all modules & examples	build: Only contains common makerules, scripts etc. for build system.
Make-based build system	Needs to be built from within build folder	Can be built from root folder
Data Types	Basic data types (e.g. unsigned int)	C99 fixed size data types from stdint.h (e.g. uint32_t)
Function/type names	Inconsistent conventions	utils are suffixed by Utils, apps by Apps, and DAL functions with no suffix.
SoC specific information	Distributed across, including within examples	Contained within soc folder
Board specific information	Distributed across, including within examples	Contained within board folder
Examples	Categorized based on board - one for each soc-board combination	Categorized based on IP / use case, single for all SoCs & boards.
Pinmux configuration	Only few pinmux configurations as demonstrated in examples.	Auto-generated pinmux database with easy-to-use GUI pinmux tool. Generic pinmux APIs
PRCM configuration	Only few power sequences can be demonstrated. Remaining have to be figured out by user	Auto-generated PRCM/clock database with generic APIs to enable all scenarios. Generic PRCM APIs
Interrupt controllers in different SoCs	User calls different APIs for interrupt configuration	Single generic interrupt header file
Cache controller / mmu configuration	User calls different APIs for different SoCs	Single generic Cache / MMU header file
Header files	All header files, exported/internal in include folder. Difficult to differentiate module-specific headers and internal/exported.	All exported header files only in include/<module>, internal in module itself. Source files include as <module>/header.h
API naming across DAL, utils, examples	No convention followed. Difficult to find out what is in DAL, utils, or examples.	Utils modules have Utils in name (e.g. I2CUtils). Apps have App in name (e.g. GPIOApp). Dals are just IP name (e.g. GPIO)
Macros: Distributed across hw file & DAL.	Some APIs require user to shift & mask before passing to DAL, others do this in DAL.	Consistency across DALs, with shift & mask within DALs only.
Macro usage	Hw file macros are directly exposed. Some macros are defined in DAL header.	DAL redefines and classifies macros into enums to abstract out IP specific differences

API parameters	User chooses from macros in hw generated header file and/or dal header file	Enums are defined in dal header file to group together parameters that can go to specific function.
Doxygen API reference documentation	Doxygen documentation for functions in the 'c' file	Complete doxygen documentation in the header file. Enables complete definition of the interface within the header files.

The below table gives a high level overview of changes in the folder structure

StarterWare 02.00.xx module/feature	**StarterWare 02.01.xx module/feature**
Folder name: drivers	Folder name: dal
Folder name: system_config	Folder name: soc
Folder name: platform	Folder name: board
Folder name: NA	Folder name: device
libs at root - e.g. mmcsdlib, nandlib etc.	libs within library folder - e.g. mmcsd, nand etc.

Interface differences

CACHE

The below table lists the file name differences between the versions

StarterWare 02.00.xx	**StarterWare 02.01.xx**	**Reason for change**
\include\armv7a\cache.h	include\cache.h	The cache API's are made generic across different architectures.
NA	include\armv7a\pl310.h	Added new file which contains API's for PL310 outer cache controller

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	**StarterWare 02.01.xx: API Name**	**StarterWare 02.01.xx: Functionality**	**StarterWare 02.01.xx: Changes**
CacheEnable	CACHEEnable	Enables cache	
CacheDisable	CACHEDisable	Disables cache	
CacheInstInvalidateAll	CACHEInstInvAll	Invalidate complete instruction cache	
CacheDataCleanInvalidateAll	CACHEDataCleanInvAll	Clean and invalidate complete data cache	
CacheDataCleanAll	CACHEDataCleanAll	Clean complete data cache	
CacheDataInvalidateAll	CACHEDataInvAll	Invalidate complete data cache	
CacheDataCleanBuff	CACHEDataCleanRange	Clean part of data cache	
CacheDataInvalidateBuff	CACHEDataInvRange	Invalidate part of data cache	
CacheDataCleanInvalidateBuff	CACHEDataCleanInvRange	Clean and invalidate part of data cache	
--NA--	CACHERegisterOuterCache	Register outer cache	New API
--NA--	CACHELockByWay	Lock cache contents by WAY	New API
--NA--	CACHEUnlockByWay	Unlock cache contents by WAY	New API
CacheInstInvalidateBuff	--NA--	This API is deprecated in the current release	Deprecated

The below table lists the deprecated APIs and their equivalent APIs in new starterware

Deprecated API	Equivalent API
CacheInstInvalidateAll	CACHEInstInvAll
CacheDataCleanInvalidateAll	CACHEDataCleanInvAll
CacheDataCleanAll	CACHEDataCleanAll
CacheDataInvalidateAll	CACHEDataInvAll
CacheDataCleanBuff	CACHEDataCleanRange
CacheDataInvalidateBuff	CACHEDataInvRange
CacheDataCleanInvalidateBuff	CACHEDataCleanInvRange

MMU

The below table lists the file name differences between the versions

StarterWare 02.00.xx	StarterWare 02.01.xx	Reason for change
\include\armv7a\mmu.h	include\mmu.h	The cache API's are made generic across different architectures.

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.01.xx: Functionality	StarterWare 02.01.xx: Changes
MMUInit	MMUInit	Initialize the MMU	The parameters to this API is changed to make it more generic.
--NA--	MMUDisable	Disable MMU	New API

DMTIMER

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.01.xx: Functionality	StarterWare 02.01.xx: Changes
DMTimerEnable	DMTIMEREnable	This API will start/stop the timer depending on the *enableDmtimer* parameter.	Added param ***enableDmtimer*** so as to start or stop the dmtimer.
DMTimerDisable	DMTIMEREnable	This API will start/stop the timer depending on the *enableDmtimer* parameter.	Added param ***enableDmtimer***so as to start or stop the dmtimer.

GPIO

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.00.xx: Functionality	StarterWare 02.01.xx : Changes
GPIOModuleEnable	GPIOModuleEnable	This API is used to enable the GPIO module.	The functionality of both enabling and disabling of GPIO module is combined into a single API. This new API takes a one more input parameter **enableModule** which controls enabling and disabling.
GPIOModuleDisable	GPIOModuleEnable	This API is used to disable the GPIO module.	Same as above.
GPIORawIntStatus	GPIOIntrRawStatus	This API determines the raw interrupt status of the specified GPIO pins in the instance corresponding to the specified interrupt line.	This new API accepts the gpio pin number as parameter and returns the raw status corresponding to that pin, where as old API is accepting input mask where status of more than one pin can be returned.
GPIOAutoIdleModeControl	GPIOAutoIdleModeEnable	This API is used to control(enable/disable) the Auto-Idle mode for GPIO.	This new API accepts the macros **TRUE, FALSE** for the parameter **enableAutoIdle,** where as old API is accepting the macros **GPIO_AUTO_IDLE_MODE_ENABLE, GPIO_AUTO_IDLE_MODE_DISABLE.**
GPIODebounceFuncControl	GPIODebounceFuncEnable	This API enables/disables debouncing feature for a specified input GPIO pin.	The second parameter is renamed from **controlFlag** to **debounceEnable** and accepts the macros **TRUE** or**FALSE**.

The below table lists the deprecated APIs and their equivalent APIs in new starterware

Deprecated API	Equivalent API	Reason for deprecating the API
GPIOIntTypeGet	GPIOGetIntrType	---
GPIODirModeGet	GPIOGetDirMode	---

PWMSS

EPWM

The below table lists the DAL file name differences between the versions

StarterWare 02.00.xx	StarterWare 02.01.xx:	Reason for change
ehrpwm.c	epwm.c	Since EPWM is the module name,
ehrpwm.h	ehrpwm.h	Same as above.
hw_ehrpwm.h	hw_pwmss_epwm.h	Same as above.
epwm.h	----	This file is redundant file and is removed.

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.00.xx: Functionality	StarterWare 02.01.xx: Changes
EHRPWMPWMOpFreqSet	EPWMTbPwmFreqCfg	This API configures the PWM Time base counter Frequency/Period.	module name is updated from EHRPWM -> EPWM. The macros that can be passed to the parameter **counterDir** has been grouped into an enum **epwmTbCounterDir_t** and macros that can be passed to the parameter **enableShadowWrite** are grouped into an enum **epwmShadowRegCtrl_t**.
EHRPWMTBEmulationModeSet	EPWMTbSetEmulationMode	This API configures emulation mode. This setting determines the behaviour of Timebase counter during emulation (debugging)	The macros that can be passed to the parameter **mode** are grouped into an enum **epwmTbEmuMode_t**.
EHRPWMTimebaseSyncEnable	EPWMTbSyncEnable	This API enables the synchronization of time base sub-module and also configures the phase count value to be loaded after sync event, counter direction after sync event.	The following parameter name is changed from **phsCountDir** to **counterDir** and the macros that can be passed to this are grouped in to an enum **epwmTbCntDirAftSync_t.**
EHRPWMSyncOutModeSet	EPWMTbSetSyncOutMode	This API selects the source of the synchronization output signal. It determines on which of the supported events sync-out has to be generated.	The macros that are passed to the parameter **syncOutMode** are grouped into an enum **epwmTbSyncOutEvt_t**.
EHRPWMTBStatusGet	EPWMTbGetStatus	This API gets the Time Base status as indicated by the tbStatusMask parameter.	The macros that are passed to the parameter **tbStatusMask** are grouped into an enum **epwmTbSts_t**.
EHRPWMTBClearStatus	EPWMTbStatusClear	This API clears the Time base status bits.	The following parameter name is changed from **tbStatusMask** to **tbStatusClrMask** and the macros that can be passed to this are **EPWM_TB_STS_CTR_MAX**, **EPWM_TB_STS_SYNCI**.

EHRPWMLoadCMPA	EPWMCounterComparatorCfg	This API loads the CMPA value.	Comparator configuration function is made generic such that it can accept pwm channel (either A or B) as a parameter and single API can be used for both comparator A and comparator B configurations. Following is the list of parameters and their associated enums which contains possible macro values for that parameter: **cmpType : epwmCcCmp_t** **enableShadowWrite : epwmShadowRegCtrl_t** **shadowToActiveLoadTrigger : epwmCcCmpLoadMode_t** **overwriteShadow : TRUE or FALSE**
EHRPWMLoadCMPB	EHRPWMLoadCMPB	This API loads the CMPB value.	Same as above.
EHRPWMConfigureAQActionOnA	EPWMAqActionOnOutputCfg	This API configures the action to be taken on A by the Action qualifier module upon receiving the events.	This API is made generic such that it can accept pwm channel (either A or B) as a parameter and single API can be used for both for A and B channel action qualifier configuration. The parameter **pwmOutputCh** selects the pwm channel which accepts the macro vales from the enum **epwmOutputCh_t**. All the action qualifier configuration parameters are grouped into a single structure **epwmAqActionCfg_t** and a pointer to this structure is passed to the API.
EHRPWMConfigureAQActionOnB	EPWMAqActionOnOutputCfg	This API configures the action to be taken on B by the Action qualifier module upon receiving the events.	Same as above.
EHRPWMSWForceA	EPWMAqSwTriggerOneTimeAction	This API triggers the SW forced single event on A. This can be used for testing the AQ sub-module. Every call to this API will trigger a single event.	The API made generic to accept pwm channel (either A or B) as a parameter and single API can be used to force the software events on both A and B channels. This new API accepts following two new parameters and their associated enums: **pwmOutputCh : epwmOutputCh_t**. **swTrigAction : epwmAqSwTrigOtAction_t**.
EHRPWMSWForceB	EPWMAqSwTriggerOneTimeAction	This API triggers the SW forced single event on B**.** This can be used for testing the AQ sub-module. Every call to this API will trigger a single event.	Same as above.

EHRPWMAQContSWForceOnA	EPWMAqSwTriggerContAction	This API forces a value continuously on A output channel.	API is made generic to accept the pwm channel (either A or B)as a parameter and single API can be used for both A and B channel outputs. The new API accepts the following parameters and their associated enums which contain the supported macro values for that parameters: **pwmOutputCh : epwmOutputCh_t** **swTrigAction : epwmAqSwTrigContAction_t** **activeRegReloadMode : epwmAqCsfrcRegReload_t**
EHRPWMAQContSWForceOnB	EPWMAqSwTriggerContAction	This API forces a value continuously on A output channel.	Same as above
EHRPWMDBSourceSelect	EPWMDeadbandCfg	This API selects the source for delay blocks in dead band sub-module.	All the dead band related configurations are done in a single API. This new API performs following dead band configurations: deadband source select, polarity select, output mode, rising edge delay, falling edge delay. All the deadband configuration parameters are grouped into a single structure **epwmDeadbandCfg_t** and the pointer to the structure is passed to the new API.
EHRPWMDBPolaritySelect	EPWMDeadbandCfg	This API selects the polarity. This allows to selectively invert one of the delayed signals before it is sent out of the dead-band sub-module.	Same as above.
EHRPWMDBOutput	EPWMDeadbandCfg	This API selects output mode.	Same as above.
EHRPWMDBConfigureRED	EPWMDeadbandCfg	This API sets the raising edge delay.	Same as above.
EHRPWMDBConfigureFED	EPWMDeadbandCfg	This API sets the Falling edge delay.	Same as above.
--NA---	EPWMDeadbandBypass	This API bypasses the Dead-band sub-module	New API.
EHRPWMConfigureChopperDuty	EPWMChopperCfg	This API configures the chopper duty cyce.	All the chopper submodule configurations are done in a single API. This new API performs following chopper configurations: Chopper duty cycle, frequency, one shot pulse width. All the chopper related configuration parameters are grouped in to a single structure **epwmChopperCfg_t** and a pointer to the structure is passed to this API.
EHRPWMConfigureChopperFreq	EPWMChopperCfg	This API configures the chopper frequency.	Same as above.

EHRPWMConfigureChopperOSPW	EPWMChopperCfg	This API configures one shot pulse width.	Same as above.
EHRPWMChopperEnable	EPWMChopperEnable	This API enables the PWM chopper sub-module	The enable and disable functionality has been combined into single API, which accepts the enableFlag as a parameter.
EHRPWMChopperDisable	EPWMChopperEnable	This API disables the PWM chopper sub-module	Same as above.
EHRPWMTZTripEventEnable	EPWMTzTripEventEnable	This API enables the trip event	This new API accepts one more parameter **'pinNum'** which controls on which Tripzone pin the event has to be enabled. The parameter **tzEventType** which selects the type of trip zone event accepts values from the following enum **epwmTzEvent_t**.
EHRPWMTZTripEventDisable	EPWMTzTripEventDisable	This API disable the trip event.	Same as above.
EHRPWMTZForceAOnTrip	EPWMTzTriggerTripAction	This API configures the o/p on PWM A channel when a trip event is recognized.	The new API is made generic such that it accepts the pwm channel (either A or B) as an input parameter and hence single API can be used to configure trip actions of both A and B channel. For the following parameters possible values are grouped into enums: **tripAction : epwmTzTripAction_t**. **pwmOutputCh : epwmOutputCh_t**.
EHRPWMTZForceBOnTrip	EPWMTzTriggerTripAction	This API configures the o/p on PWM B channel when a trip event is recognized.	Same as above.
EHRPWMTZFlagGet	EPWMTzEventStatus	This API returns the selected trip zone event status	The parameter **flagToRead** is renamed as **eventMask** and accepts the values from the following enum **epwmTzStsFlg_t.**
EHRPWMTZFlagClear	EPWMTzEventStatusClear	This API clears the selected trip zone event status	The parameter **flagToClear** is renamed as **eventMask** and accepts the values from the following enum **epwmTzStsFlg_t.**
EHRPWMTZSWFrcEvent	EPWMTzTriggerSwEvent	This API enables to generate Software forced trip condition	The parameter **osht_CBC** is renamed as **tzEventType** and accepts the values from the following enum **epwmTzEvent_t**.
EHRPWMTZIntEnable	EPWMTzIntrEnable	This API enables the trip interrupt.	The parameter **osht_CBC** is renamed as **tzEventType** and accepts the values from the following enum **epwmTzEvent_t**.
EHRPWMTZIntDisable	EPWMTzIntrDisable	This API disables the trip interrupt.	The parameter **osht_CBC** is renamed as **tzEventType** and accepts the values from the following enum **epwmTzEvent_t**.
EHRPWMETIntDisable	EPWMEtIntrDisable	This API disables the ePWM Event interrupt	
EHRPWMETIntEnable	EPWMEtIntrEnable	This API enables the ePWM Event interrupt	

EHRPWMETIntSourceSelect	EPWMEtIntrCfg	This API selects the interrupt source	All the configurations corresponding to the event trigger interrupt configuration have been combined into single API. This API configures interrupt source and interrupt period. The following are the configuration parameters and thier respective enums which contains possible macro values for that parameter: **intrEvtSrc : epwmEtIntrEvt_t.** **intrPrd : epwmEtIntrPeriod_t.**
EHRPWMETIntPrescale	EPWMEtIntrCfg	This API prescales the event on which interrupt is to be generated	Same as above
EHRPWMLoadCMPAHR	EPWMHrLoadCmpAHrValue	This API loads Counter-Comparator A high resolution value	The parameter **ShadowToActiveLoadTrigger** takes values from the enum **epwmHrRegActLoad_t**.

The below table lists the deprecated APIs and their equivalent APIs in new starterware

Deprecated API	Equivalent API	Reason for deprecating the API
EHRPWMClockEnableStatusGet	EPWMClockEnableStatusGet	--NA--
EHRPWMClockDisableStatusGet	EPWMClockDisableStatusGet	--NA--

ECAP

The below table lists the DAL file name differences between the versions

StarterWare 02.00.xx:	StarterWare 02.01.xx:	Reason for change
ecap.c	ecap.c	No Change.
ecap.h	ecap.h	Same as above.
hw_ecap.h	hw_pwmss_ecap.h	Renamed to reflect the subsystem and submodule names.

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.00.xx: Functionality	StarterWare 02.01.xx: Changes
ECAPClockEnable	ECAPClkEnable	This API enables the clock for ECAP sub-module in the PWM Subsystem.	Enable and Disable functionality are handled within the same API. It accepts **enableClk** as a parameter to control enable/disable of the clocks.
ECAPClockDisable	ECAPClkEnable	This API disables the clock for ECAP sub-module in the PWM Subsystem.	Enable and Disable functionality are handled within the same API. It accepts **enableClk** as a parameter to control enable/disable of the clocks.
ECAPCaptureLoadingEnable	ECAPCaptureLoadEnable	This API controls enabling of the Loading of the capture registers with the time stamp value on a capture event.	Enable and Disable functionality are handled within the same API. It accepts **enableCaptLoad** as a parameter to control enable/disable of the clocks.

ECAPCaptureLoadingDisable	ECAPCaptureLoadEnable	This API controls disabling of the Loading of the capture registers with the time stamp value on a capture event.	Enable and Disable functionality are handled within the same API. It accepts **enableCaptLoad** as a parameter to control enable/disable of the clocks.
ECAPOperatingModeSelect	ECAPSetOperMode	This API configures the operating mode of ECAP to either Capture or Auxiliary PWM mode.	Function name is updated. The macros that are passed to the parameter **operMode** are grouped into an enum **ecapOperMode_t**
ECAPContinousModeConfig	ECAPSetCaptureMode	This API configures the ECAP in continuous mode.	Function name is updated. The macros that are passed to the parameter **captMode** are grouped into an enum **ecapCaptMode_t**.
ECAPOneShotModeConfig	ECAPSetCaptureMode	This API configures the ECAP in one-shot mode.	Function name is updated. The ECAPSetCaptureMode API configures ECAP in one-shot mode, the setting of StopWrap value is done through ECAPSetStopWrapVal API
ECAPOneShotModeConfig	ECAPSetStopWrapVal	This function sets the Stop/Wrap Value for a capture Event in One-shot and Continuous modes.	Function name is updated. The feature of setting StopWrap value is handled within this API. The user has to configure the ECAP to capture before calling this API.
ECAPCapeEvtPolarityConfig	ECAPSetCaptEvtPol	This function sets the configures the capture event polarity for the specific event number.	Function name is updated. The API now accepts **captEvtNum** and **evtPol** as the parameters to configure the polarity of a particular event.
ECAPCaptureEvtCntrRstConfig	ECAPCaptEvtCounterResetConfig	This API enables/disables reset of the time stamp counter after a capture event.	Function name is updated. The API now accepts **captEvtNum** and **counterCfg** as the parameters to enable reset or no reset of the Capture event counter.
ECAPAPWMPolarityConfig	ECAPSetApwmOutputPol	This function configures the output polarity of the APWM pulse output.	Function name is updated. The API **outputPol** as a parameter which accepts the polarity setting to be done from the following enum **ecapAPWMOutputPol_t**
ECAPCounterControl	ECAPTimeStampCounterEnable	This API controls the time-stamp counter by programming it to run in free run mode or stop it.	Same as above. The **enableCounter** parameter is updated to accept TRUE or FALSE as input to control the running of the timestamp counter.
ECAPSyncInOutSelect	ECAPSyncInEnable	This API controls enabling/disabling the Sync-In select mode..	Function name is updated. The **enableSyncIn** parameter takes TRUE or FALSE as the input to control enabling/disabling of the SyncIn feature.
ECAPSyncInOutSelect	ECAPSyncOutSignalCtrl	This API configures the Sync-Out Event features like Period Equal event to be Sync-out signal etc.	The Sync-Out signal features which was handled within ECAPSyncInOutSelect API is now handled separtely with **ECAPSyncOutSignalCtrl** API. The **syncOutCtrl** takes values from the following enum **ecapSyncOutCfg_t** for different synout signal properties.
ECAPIntEnable	ECAPIntrEnable	This function enables the specified Interrupts for ECAP module.	Function name is updated. The API accepts masks value of the interrupts to be enabled from the following enum **ecapIntrMask_t**

ECAPIntDisable	ECAPIntrDisable	This function enables the specified Interrupts for ECAP module.	Function name is updated. The API accepts masks value of the interrupts to be disabled from the following enum **ecapIntrMask_t**
ECAPIntStatusClear	ECAPIntrClear	This API Clears the status of specified interrupts.	Function name is updated. The API accepts interrupt mask value as the parameter from the following enum **ecapIntrMask_t.** It also clears the Global Interrupt bit to enable generation of further interrupts.
ECAPIntStatus	ECAPIntrStatus	This API returns the status of all active and enabled interrupts.	Function name is updated. The API accepts interrupt mask value as the parameter from the following enum **ecapIntrMask_t**.
--NA---	ECAPIntrTrigger	This API triggers an interrupt request for a specified event.	New API.
EcapContextSave	ECAPContextSave	This API can be used to save the Register context of ECAP	Same as above. The Interrupt Clear register is not saved in the context. The capture event3 and event4 registers are added to the existing registers which are saved.
EcapContextRestore	ECAPContextRestore	This API can be used to restore the Register context of ECAP	Same as above.

The below table lists the deprecated APIs and their equivalent APIs in new starterware

Deprecated API	Equivalent API	Reason for deprecating the API
ECAPClockEnableStatusGet	--NA--	--NA--
ECAPClockDisableStatusGet	--NA--	--NA--

I2C

The below table lists the DAL file name differences between the versions

StarterWare 02.00.xx	StarterWare 02.01.xx:	Reason for change
hsi2c.c	i2c.c	I2C IP integrated in the AM335x and AM43xx does not support the High Speed Mode of I2C. The file is renamed to avoid ambiguity the DAL file is renamed to i2c.c
hsi2c.h	i2c.h	Same as above.
hw_hsi2c.h	hw_i2c.h	Same as above.

The below table lists API differences between the versions.

StarterWare 02.00.xx: API Name	StarterWare 02.00.xx Functionality	StarterWare 02.01.xx: API Name	StarterWare 02.01.xx: Change
I2CMasterEnable	The API enables the I2C module.	I2CModuleEnable	The features of Enabling/Disabling the Module is combined into a single API "I2CModuleEnable". The New API takes one of the following macros TRUE/FALSE as a parameter to enable/disable the module.
I2CMasterDisable	The API disables the I2C module	I2CModuleEnable	Same as above
I2CMasterIntEnableEx	The API is used to enable specific interrupts of the module.	I2CIntrEnable	The name of the API is changed. Macros that can be passed to the API are updated and grouped together in an enum **i2cIntr_t**.
I2CMasterIntDisableEx	The API is used to disable specific interrupts of the module	I2CIntrDisable	Functionality remains the same. Rest of the change Same as applicable for I2CIntrEnable API.
I2CMasterIntStatusEx	The API is used to return the status of specific interrupts specified as a flag through one of the input parameter.	I2CIntrStatus	The feature of returning the Status of specific interrupt is merged into **I2CIntrStatus** API.
I2CMasterIntRawStatusEx	The API is used to return the Status of specific interrupts (both enabled and not enabled).	I2CIntrRawStatus	The feature of getting the status of specific interrupts is merged into I2CIntrRawStatus.
I2CMasterIntClearEx	The API clears the status of specific interrupts according to the macro value passed by the user.	I2CIntrClear	The API name is updated. It accepts the macros from the enum **i2cIntr_t** as an input parameter.
I2CMasterIntRawStatusClearEx	The API can be used to trigger an interrupt event for debug purpose.	I2CIntrTrigger	The API name is updated to reflect the functionality it provides. It accepts the macros from the enum **i2cIntr_t** as an input parameter.
I2CDMATxEventEnable	The API enables the Transmit DMA channel of the module.	I2CDMATxEventEnable	The new API is updated to accept a macro (**TRUE/FALSE**) as a user parameter to enable/disable the Tx DMA channel of the I2C.
I2CDMATxEventDisable	The API disables the Transmit DMA channel of the module.	I2CDMATxEventEnable	The new API combines the feature to disable the Tx DMA channel which can be used by passing the macro (**FALSE**) to the API.
I2CDMARxEventEnable	The API enables the Receive DMA channel of the module.	I2CDMARxEventEnable	The new API is updated to accept a macro (**TRUE/FALSE**) as a user parameter to enable/disable the Receive DMA channel of the I2C.
I2CDMARxEventDisable	The API disables the Receive DMA channel of the module.	I2CDMARxEventEnable	The new API combines the feature to disable the Rx DMA Channel which can be used by passing the macro (**FALSE**) to the API.
I2COwnAddressSet	This API configures any one of the own address field out of four present in I2C controller.	I2CSlaveSetOwnAddr	The API parameters are updated to accept the address to be set, addr mode and addr index separately. The addr mode and addr index accept values from the following enums **i2cAddrMode_t** and **i2cOwnAddrIndex_t**

I2CClockBlockingControl	This API blocks or unblocks the clock for any of the module's four own addresses.	I2CSlaveClockBlockingEnable	The API parameters are updated to accept addr index and a user flag to verify whether Clock Blocking should be enabled/disabled.
I2CAutoIdleEnable	Enables the AutoIdle feature	I2CAutoIdleEnable	The API is updated to enable/disable the AutoIdle mode according to the parameter passed by the user. It accepts one of the two values (**TRUE/FALSE**).
I2CAutoIdleDisable	Disables the AutoIdle feature	I2CAutoIdleEnable	Same as above.
I2CMasterControl	Configures the module in different modes as per the user passed parameters.	I2CSetMode	The API name is updated. It takes macro values from the following enum **i2cMode_t**
--NA--	--NA--	I2CStartByteModeEnable	This API enables the Start Byte mode of the I2C.

The below table lists the deprecated APIs and their equivalent APIs in new starterware

Deprecated API	Equivalent API	Reason for deprecating the API
I2CMasterBusy	I2CMasterBusy	---

WDT

The below table lists the DAL file name differences between the versions

StarterWare 02.00.xx	StarterWare 02.01.xx:	Reason for change
watchdog.c	wdt.c	WDT is followed for file, API, macro and enum names. Hence the name is changed.
watchdog.h	wdt.h	Same as above.
hw_watchdog.h	hw_wdt.h	Same as above.

The below table lists API differences between the versions.

StarterWare 02.00.xx: API Name	StarterWare 02.00.xx Functionality	StarterWare 02.00.xx API	StarterWare 02.01.xx : Changes
WatchdogTimerEnable	The API enables the WDT.	WDTEnable	The API is updated. Functionality change is this API also can disable the WDT by passing the parameter FALSE.
WatchdogTimerDisable	The API disables the WDT	WDTEnable	This API is removed since disable is also done using WDTEnable API.
WatchdogTimerPreScalerClkEnable	The API enables/configures and also can disable the prescaler clock of WDT.	WDTPrescalerClkEnable	The API is updated. Functionality change is the same API is used to disable the prescaler clock.
WatchdogTimerPreScalerClkDisable	The API disables the prescaler clock of WDT.	WDTPrescalerClkDisable	Same as above.
WatchdogTimerWritePostedStatusGet	This API is discarded.	NA	The functionality of this API is now handled using macro internally in an API.
NA	This API will set the idle mode for WDT.	WDTSetIdleMode	This API is newly added.

NA	This API will enable/disable the emulation mode of WDT.	WDTEmulationEnable	This API is newly added.

Note

The API WatchdogTimerIntEnableStatusGet() has been discarded since it is not longer needed.

TSCADC

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.00.xx: Functionality	StarterWare 02.01.xx: Changes
TSCADCIntStatusClear	TSCADCIntrClear	Clear the interrupt status	Intr is the convention to be followed for interrupts. Also status is not used in Interrupt clear APIs.
TSCADCIntStatus	TSCADCIntrStatus	Get the interrupt status	Intr is the convention to be followed for interrupts
TSCADCTSTransistorConfig	NA	Enable/Disable touchsceen transistors	This API is removed and is not needed. Transistor configuration is automatically handled when touchscreen mode is used.
TSCADCStepConfigProtectionDisable	TSCADCStepConfigWrProtectEnable	Disable the step config write protection feature.	Disable feature is incorporated in TSCADCStepConfigProtectionEnable. FALSE needs to be passed as the second parameter.
TSCADCStepIDTagConfig	TSCADCStepIdTagEnable	Enable/disable the step id tag	ID -> Id. TRUE/FALSE values as the parameters.
TSCADCModuleStateSet	TSCADCEnable	Enable/Disable TSCADC	API name was not matching the functionality.
TSCADCSequencerFSMBusyStatus	TSCADCStatus	Check the TSCADC FSM busy status	The new API will send the status of entire register and it is responsibility of application to check the required status. In the case the app can use the entry TSCADC_STATUS_FSM_BUSY from tscAdcStatus_t enum.
TSCADCSequencerCurrentStepID	TSCADCStatus	Check the TSCADC current step ID	The new API will send the status of entire register and it is responsibility of application to check the required status. In the case the app can use the entry TSCADC_STATUS_STEP_ID from tscAdcStatus_t enum.
TSCADCSequencerPenIrqStatusRead	TSCADCStatus	Check the TSCSDC IRQ0 or IRQ1 status	The new API will send the status of entire register and it is responsibility of application to check the required status. In the case the app can use the entry TSCADC_PEN_IRQ0 and TSCADC_PEN_IRQ1 from tscAdcStatus_t enum.
TSCADCConfigureAFEClock	TSCADCClkDivConfig	Configure the TSCADC clock divider.	The new API represents the functionality better.

TSCADCRawIntStatusRead	TSCADCIntrRawStatus	Read the TSCADC Raw interrupt status. This API would read the interrupt status given by flag. However this is not followed and hence the TSCADCIntrRawStatus API needs to be used which shall pass the overall register contents and application needs to check the specific status.	This API is removed.
TSCADCRawIntStatus	TSCADCIntrRawStatus	Read the TSCADC Raw interrupt status	
TSCADCIntStatusRead	TSCADCIntrStatus	Read the TSCADC interrupt status. This API would read the interrupt status given by flag. However this is not followed and hence the TSCADCIntrRawStatus API needs to be used which shall pass the overall register contents and application needs to check the specific status.	This API is removed.
TSCADCDMAFIFODisable	TSCADCDmaFifoEnable	Disable the DMA for FIFO.	This API is discarded. Disable functionality is handled in TSCADCDMAFIFOEnable API.
TSCADCIsDMAFIFOEnabled	NA	Check the DMA status	This API is discarded. Not used anywhere. Not required.
TSCADCIdleStepOperationModeControl	TSCADCTsIdleStepConfig	Configure the mode for TSCADC. Either single ended or differential.	All Idle step configurations are combined into a single APIs. Also API is made to touchscreen specific.
TSCADCIdleStepConfig	TSCADCTsIdleStepConfig	Configure the Idle step of TSCADC.	API name is changed to make it Touch screen specific and also the API parameter order is updated.
TSCADCIdleStepAnalogSupplyConfig	TSCADCTsIdleStepConfig	Provide analog supply for the Idle step.	All Idle step configurations are combined into a single APIs. Also API is made to touchscreen specific.
TSCADCIdleStepAnalogGroundConfig	TSCADCTsIdleStepConfig	Provide analog ground for the idle step.	All Idle step configurations are combined into a single APIs. Also API is made to touchscreen specific.
TSCADCChargeStepOperationModeControl	TSCADCTsChargeStepConfig	Configure the mode for TSCADC. Either single ended or differential.	API is discarded and all charge step functionality is added in a single API.

TSCADCChargeStepConfig	TSCADCTsChargeStepConfig	Configure the charge step of ADC	API name is changed to make it Touchscreen specific
TSCADCChargeStepAnalogSupplyConfig	TSCADCTsChargeStepConfig	Provide analog supply for the charge step	API is discarded and all charge step functionality is added in a single API.
TSCADCChargeStepAnalogGroundConfig	TSCADCTsChargeStepConfig	Provide analog ground for the charge step	API is discarded and all charge step functionality is added in a single API.
TSCADCChargeStepOpenDelayConfig	TSCADCTsChargeStepDelayConfig	Configure the open delay value for Charge step	API name is changed and also to make it Touchscreen specific
TSCADCTSStepOperationModeControl	TSCADCStepConfig	Configure the step for single ended or differential.	API is discarded and all step functionality is added in a single API.
TSCADCTSStepConfig	TSCADCStepConfig	Configure the TSCADC step	API name is updated to not make it Touchscreen specific. Paramter order is updated.
TSCADCTSStepAnalogSupplyConfig	TSCADCStepConfig	Configure the Analog supply for the step	API is discarded and all step functionality is added in a single API.
TSCADCTSStepAnalogGroundConfig	TSCADCStepConfig	Configure the Analog ground supply for the step	API is discarded and all step functionality is added in a single API.
TSCADCTSStepOutOfRangeCheckEnable	TSCADCStepRangeCheckEnable	Enable/Disable out of range check	API name is updated to not make it Touchscreen specific. TRUE needs to be used to enable.
TSCADCTSStepFIFOSelConfig	TSCADCStepFifoConfig	Configure the FIFO for the corresponding step.	API name is updated to not make it Touchscreen specific.
TSCADCTSStepAverageConfig	TSCADCStepSamplesAvg	Configure the number of samples to be configured.	API name is updated to not make it Touchscreen specific.
TSCADCTSStepOutOfRangeCheckDisable	TSCADCStepOutOfRangeCheckEnable	Enable/Disable out of range check	API name is updated to not make it Touchscreen specific. FALSE needs to be used to disable.
TSCADCTSStepModeConfig	TSCADCStepMode	Configure the Step mode	API name is updated to not make it Touchscreen specific.
TSCADCTSStepSampleDelayConfig	TSCADCSetStepDelay	Configure the sample ad open delay for the step	API name is updated to not make it Touchscreen specific and the API takes 2 parameters which can configure sample and open delay.
TSCADCTSStepOpenDelayConfig	TSCADCSetStepDelay	Configure the sample ad open delay for the step	API name is updated to not make it Touchscreen specific and the API takes 2 parameters which can configure sample and open delay.

MCSPI

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.00.xx: Functionality	StarterWare 02.01.xx: Changes
McSPICSEnable	MCSPICsEnable	Enable/Disable the chip select pin.	Same API is used to enable/disable the chip select by passing TRUE/FALSE respectively.
McSPICSTimeControlSet	MCSPISetCsTimeControl	Configure the chip select time control value.	API name updated. Also parameter order is changed.
McSPIStartBitEnable	MCSPIStartBitEnable	Enable/Disable start bit for SPI transfer.	Same API shall be used to enable/disable the functionality by passing TRUE/FALSE respectively
McSPIStartBitPolarityConfig	MCSPISetStartBitPol	Configure the start bit polarity.	API name updated. Also parameter order is changed.
McSPIMasterModeEnable	MCSPIModeConfig	Configure the MCSPI controller in Master/Slave modes and other required parameters of the mode.	Master/Slave mode can be configured using this API. API name is updated accordingly.
McSPISlaveModeEnable	MCSPIModeConfig	Configure the MCSPI controller in Slave modes and other required parameters of the mode.	Slave mode can be configured using this API. API name is updated accordingly.
McSPISlaveModeConfig	MCSPIModeConfig	Configure the MCSPI controller in Slave modes and other required parameters of the mode.	Slave mode can be configured using this API. API name is updated accordingly.
McSPIChannelEnable	MCSPIChEnable	Enable/Disable the channel of MCSPI controller.	Same API shall be used to enable/disable the functionality by passing TRUE/FALSE respectively. API name updated.
McSPIReset	MCSPIReset	Reset the MCSPI controller.	API name updated.
McSPITurboModeEnable	MCSPITurboModeEnable	Enable/Disable TURBO mode of operation.	Same API is used to enable/disable the functionality by passing TRUE/FALSE respectively
McSPIDMAEnable	MCSPIDmaEnable	Enable/Disable the MCSPI DMA events.	Same API shall be used to enable/disable the functionality by passing TRUE/FALSE respectively
McSPICSPolarityConfig	MCSPISetCsPol	Set the chip select polarity.	API name is updated. Parameter order is changed.

The below table lists the deprecated APIs and their equivalent APIs in new starterware

Deprecated API	Equivalent API	Reason for deprecating the API
McSPICSDisable	MCSPICsEnable	Single API to enable/disable a feature
McSPIStartBitDisable	MCSPIStartBitEnable	Single API to enable/disable a feature
McSPIChannelDisable	MCSPIChEnable	Single API to enable/disable a feature
McSPITurboModeDisable	MCSPITurboModeEnable	Single API to enable/disable a feature
McSPIDMADisable	MCSPIDmaEnable	Single API to enable/disable a feature

UART

The below table lists the DAL file name differences between the versions

StarterWare 02.00.xx	StarterWare 02.01.xx	Reason for change
uart_irda_cir.c	uart.c	The IRDA and CIR features are not supported by StarterWare and the file is renamed to reflect the same.
uart_irda_cir.h	uart.h	Same as above.
hw_uart.h	hw_uart.h	Same as above.
uart.h and uart.c	uart_v1.h and uart_v1.c	These files related to AM1808 are redundant and are renamed.

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.01.xx: Functionality	StarterWare 02.01.xx: Changes
UARTRegConfigModeEnable	UARTSetRegAccessMode	This API configures the specified Register Access mode of UART to allow access to certain registers.	Function name is updated. The API accepts the register mode value as a parameter from the following enum **uartRegAccessMode_t**
UARTDivisorLatchWrite, UARTDivisorLatchEnable, UARTDivisorLatchDisable and UARTDivisorValCompute	UARTSetBaudRate	This API sets the baud rate value for the UART operation according to the value passed by the user.	Function name is updated. The API combines the features of enabling access to divisor latch register, computing divisor value and setting baudRate in to a single API. The API takes baudrate input from the following enum **uartBaudRate_t**
UARTIntEnable	UARTIntrEnable	This API disables the synchronization. Even if sync-in event occurs the count value will not be reloaded.	Function name is updated. The API returns one of the macros from the following enum **uartIntrMask_t** to indicate the interrupt to be enabled.
UARTIntDisable	UARTIntrDisable	This API disables the specified interrupts in the UART mode of operation.	Function name is updated. The API returns one of the macros from the following enum **uartIntrMask_t** to indicate the interrupt to be disabled.
UARTIntIdentityGet	UARTIntrStatus	This API returns the status of UART Interrupts.	Function name is updated. The API returns one of the macros from the following enum **uartIntrMask_t** to indicate the interrupt status.

UARTIntPendingStatusGet	UARTIsIntrPending	This API determines whether any UART interrupt condition is pending to be serviced.	Function name is updated. The API returns TRUE/FALSE to indicate if an interrupt is pending or not.
UARTTxEmptyIntControl	UARTTxEmptyIntrEnable	This API is used to enable/disable the feature to get Transmit interrupts when both the Tx FIFO and the Shift register become empty.	Function name is updated. The API accepts **enableTxEmptyIntr** as a parameter which accepts TRUE/FALSE to control the feature.
UARTRxErrorGet	UARTLineStatus	This API returns the Line status of UART including errors and FIFO status.	Function name is updated. The API returns the Line Status of UART Line with the macros from the following enum **uartLineSts_t** .
UARTLineCharacConfig	UARTLineCharacteristicConfig	The API configures the line characteristic required for a UART frame transmission/reception.	Function name is updated. The API accepts stop bits and character length as a separate parameter compared to the previous API.
UARTFIFOConfig	UARTFifoConfig	This API configures the FIFO settings for the UART instance. This API provides a common interface for configuring the FIFO related settings through both the FCR and TLR/TCR registers. The following Fifo related configuration are done by the API. - Enables the FIFO mode of UART - Transmitter and Receiver FIFO Trigger Level granularity setting - Transmitter and Receiver FIFO Trigger Level configuration - Clearing of the Tx and Rx FIFOs - Enables/Disables the DMA mode of operation	The API accepts a structure **uartFifoCfg_t** as a parameter for configuring the FIFO in different settings. This API does not provide the option of configuring FIFO trigger level settings only through the FCR register. The FIFO trigger levels are configured either through TLR or a combination of FCR and TLR registers.
UARTFIFOEnableStatusGet	UARTIsFifoEnabled	This API determines whether FIFO mode of operation is enabled for the UART instance or not.	Function name is updated. The API returns TRUE/FALSE to indicate if the FIFO mode is enabled or not.
UARTWakeUpEventsEnable	UARTWakeupEventEnable	This API enable the Wakeup capability for the specified events.	Function name is updated. The API parameter **wakeupEvent** takes one of the macros as input from the following enum **uartWakeupEvent_t**
UARTWakeUpEventsDisable	UARTWakeupEventDisable	This API disable the Wakeup capability for the specified events.	Function name is updated. The API parameter **wakeupEvent** takes one of the macros as input from the following enum **uartWakeupEvent_t**

NA	UARTWakeupIntrEnable	This API enables/disables the wakeup interrupt feature..	Function name is updated.
UARTAutoIdleModeControl	UARTAutoIdleEnable	This API is used to control(enable/disable) the Auto-Idle mode of operation of the UART.	Function name is updated. The API parameter takes TRUE/FALSE to enable/disable the idle mode.
UARTIdleModeConfigure	UARTSetIdleMode	This API sets UART to one of the idle mode mechanism.	Function name is updated. The API parameter **idleMode** takes any of the macros from the following enum **uartIdleMode_t**
UARTLoopbackModeControl	UARTLoopbackEnable	This API controls(enables/disables) the Loopback mode of operation for the UART instance.	SaFunction name is updated. The API parameter **enableLoopBack** takes TRUE or FALSE to control the Loopback mode.
UARTBreakCtl	UARTBreakCtrlEnable	This API is used to enable/disable a Break condition where the transmitter output goes low to alert the communication terminal.	Function name is updated. The API parameter **enableBrkCtrl** takes TRUE or FALSE to control the Break Control feature.
UARTTxFIFOLevelGet	UARTGetFifoLevel	This API determines the current level of the Transmitter FIFO.	Function name is updated. The API returns the level of FIFO in bytes.
UARTTxFIFOFullStatusGet	UARTIsTxFifoFull	This API determines whether the Transmitter FIFO is full or not.	Function name is updated. The API returns TRUE/FALSE to indicate the status of the Tx FIFO.
UARTCharsAvail	UARTIsCharAvail	This API checks if the RX FIFO has atleast one byte of data to be read.	Function name is updated. The API returns TRUE/FALSE to indicate that there is atleast one byte of data to be read.
UARTSpaceAvail	UARTIsSpaceAvail	This API checks whether the TX FIFO is empty or not.	Function name is updated. The API returns TRUE/FALSE to indicate that there is space available in the FIFO to read or not.
UARTCharGet	UARTGetChar	This API reads a character from the UART.	The **UARTGetChar** API provides a common interface to read a character from UART in the following different ways 1: Blocking - it waits indefinitely until a character arrives 2: Non-Blocking - it checks once if a character arrived and returns the character or returns error. 3: Timeout - it waits til the occurence of a timeout for the arrival of a character and returns the character else returns a Timeout error.

UARTCharPut	UARTPutChar	This API puts a character to the UART FIFO for transmission.	The **UARTPutChar** API provides a common interface to write a character to UART in the following different ways 1: Blocking - it waits indefinitely until there is space to write a character. 2: Non-Blocking - it checks once if there is space in the FIFO to write a character. 3: Timeout - it waits til the occurence of a timeout for a space in the FIFO to write a character or else returns a timeout error.
UARTCharGetNonBlocking, UARTCharGet, UARTCharPutNonBlocking	UARTGetChar and UARTPutChar	Refer above explaination for UARTGetChar and UARTPutChar	Refer above explaination for **UARTGetChar** and **UARTPutChar**.
UARTFIFOWrite	UARTFifoWrite	This API copies the requested amount of data from the specified data block to the UART Transmit FIFO.	Function name is updated. The API does not return any value indicating the number of bytes returned, as the API does not check emptiness of the FIFO before writing.
UARTDMAEnable	UARTDmaEnable	This API Enables the DMA mode and configures the DMA mode for the UART module.	Function name is updated. The API takes **dmaMode** as a parameter from the following enum uartDmaMode_t
UARTTxDMAThresholdControl	UARTDmaTxThresholdEnable	This function enables/disables the setting the TX_DMA_THRESHOLD register which holds Transmit DMA Threshold value to be used.	Function name is updated. The **enableTxThreshold** parameter takes TRUE/FALSE to control enabling/disabling of the feature.
UARTDMACounterResetControl	UARTDmaCounterResetEnable	This API controls enabling/disabling the DMA Counter Reset when the FIFO reset option is selected.	Function name is updated. The **enableCounterReset** parameter takes TRUE/FALSE to control enabling/disabling of the feature.
UARTAutoRTSAutoCTSControl	UARTAutoRtsEnable	This API enables/disables the Auto-RTS (Request to Send) hardware flow control feature.	Function name is updated. The **enableRtsCtrl** parameter can take either TRUE/FALSE to control enabling/disabling of the Auto-RTS feature.
UARTAutoRTSAutoCTSControl	UARTAutoCtsEnable	This API enables/disables the Auto-CTS (Clear to Send) hardware flow control feature.	Function name is updated. The **enableCtsCtrl** parameter can take either TRUE/FALSE to control enabling/disabling of the Auto-CTS feature.
UARTFlowCtrlTrigLvlConfig	UARTSetFlowCtrlTriggerLvl	This API configures the Receiver FIFO trigger levels to start/stop transmission when flow control feature of UART is enabled.	Function name is updated. The API parameters **rxHaltTriggerLvl** and **rxStartTriggerLvl** to configure software flow control.

UARTSoftwareFlowCtrlOptSet	UARTSwFlowCtrlConfig	This API configures the different Software Flow Control combinations.	Function name is updated. The API parameter **flowCtrlCfg** takes values from the following enum **uartSwFlowCtrlCfg_t**
UARTXONAnyFeatureControl	UARTXonAnyFeatureEnable	This API Enables/Disables the XON any feature of Software Flow Control.	Function name is updated. The API parameter **enableXonAnyFeature** takes TRUE/FALSE to control enabling/disabling of the Special Character Detect feature.
UARTSpecialCharDetectControl	UARTSpecialCharDetectEnable	This API controls enabling/disabling the feature of detecting Special Character through the UART receiver.	Function name is updated. The API takes TRUE/FALSE to control enabling/disabling of the Special Character Detect feature.
UARTPulseShapingControl	UARTPulseShapingEnable	This API controls enabling/disabling the Pulse Shaping feature of the UART.	Function name is updated. The API takes TRUE/FALSE to control enabling/disabling of the Pulse Shaping feature.
UARTModemControlSet	UARTModemCtrlEnable	This API switches the specified Modem Control Signals to active state depending on the option passed by user.	Function name is updated. The API takes **modemPin** and **enableCtrlPin** as the parameter to set the modem ctrl pins to active and inactive state.
UARTModemControlClear	UARTModemCtrlEnable	This API switches the specified Modem Control Signals to inactive state depending on the option passed by user.	Same as above
UARTModemStatusGet	UARTModemStatus	This API provides the current state of the control lines from the modem, data set or peripheral device.	Function name is updated. The function returns the status of UART Modem control lines through mask values grouped together as the following enum **uartModemSts_t**
UARTModemStatusChangeCheck	UARTModemStatus	Same as above	This functionality is provided through **UARTModemStatus** API.
UARTDSRInterruptControl	UARTDsrPinIntrEnable	This API enables/disables the DSR(Data Set Ready) pin status Interrupt feature.	Function name is updated. The API accepts TRUE/FALSE macro through enableDsrIntr parameter to control the interrupt feature.
UARTRXCTSDSRTransitionStatusGet	UARTRxCtsDsrTransitionStatus	This API returns the status of falling edge signal status on the Receive, CTSn(Clear to Send) and DSRn(Data Set Ready) lines.	Function name is updated. The API returns a TRUE/FALSE depending on occurence of an event on CTS, DSR lines.
UARTOperatingModeSelect	UARTEnableAutobaudMode	This API enables/disables the Auto-Baud mode of the UART module.	Function name is updated. The API **UARTEnableAutobaudMode** takes enableAutoBaud as a parameter and enables/disables the AutoBaud mode.

The below table lists the deprecated APIs and their equivalent APIs in new starterware

Deprecated API	Reason for deprecating the API
UARTSubConfigTCRTLRModeEn	The functionality is handled within the other APIs
UARTSubConfigXOFFModeEn	The functionality is handled within the other APIs
UARTSubConfigMSRSPRModeEn	The functionality is handled within the other APIs
UARTTCRTLRBitValRestore	The functionality is handled within the other APIs
UARTEnhanFuncEnable	The functionality is handled within the other APIs
UARTEnhanFuncBitValRestore	The functionality is handled within the other APIs

RTC

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.00.xx: Functionality	StarterWare 02.01.xx: Changes
RTCWriteProtectEnable	RTCWriteProtectEnable	This API enables/disables the write-protection for RTC registers	Single API to enable & disable WriteProtect, also takes an additional parameter to select enable/disable
RTCWriteProtectDisable	RTCWriteProtectEnable	This API enables/disables the write-protection for RTC registers	Single API to enable & disable WriteProtect, also takes an additional parameter to select enable/disable
RTC32KClkSourceSelect	RTCClkSrcSelect	This API selects the clock source for the RTC module	Same as above.
RTCRun	RTCEnable	This API enables/disables the RTC module.	Only one API to enable and disable RTC.The new API also takes a parameter enable/disable.
RTCStop	RTCEnable	This API enables/disables the RTC module.	Only one API to enable and disable RTC.The new API also takes a parameter enable/disable.
RTCDisable	RTCEnable	This API enables/disables the RTC module.	Only one API to enable and disable RTC.The new API also takes a parameter enable/disable.
RTCEnable	RTCEnable	This API enables/disables the RTC module.	Only one API to enable and disable RTC.The new API also takes a parameter enable/disable.
RTCOscillatorStateControl	RTCXtalOutPinEnable	This API is used to configure the XTALOUT Pin to connect it to an external crystal.	Same as above
RTCIntTimerDisable	RTCTimerIntrDisable	This API disables the periodic timer interrupts.	Function name is updated.
RTCMinRoundingDisable	RTCRoundingEnable	This API Enables/Disables the feature of minute rounding.	Single API to enable and disable minute rounding feature. New API takes an additional parameter to select enable/disable.

RTCTimeSet	RTCSetTime	This API programs the specified time information the Time registers.	Function name is updated, Single API to set all 'Time' related information.
RTCHourModeSet	RTCSetTime	This API programs the specified time mode information in the Time registers.	Function name is updated, Single API to set all 'Time' related information.
RTCSecondSet	RTCSetTime	This API programs the specified time information the Time registers.	Function name is updated, Single API to set all 'Time' related information.
RTCMinuteSet	RTCSetTime	This API programs the specified time information the Time registers.	Function name is updated, Single API to set all 'Time' related information.
RTCHourSet	RTCSetTime	This API programs the specified time information the Time registers.	Function name is updated, Single API to set all 'Time' related information.
RTCMeridiemSet	RTCSetTime	This API programs the specified time meridiem mode information the Time registers.	Function name is updated, Single API to set all 'Time' related information.
RTCTimeGet	RTCGetTime	This API reads the current time from the registers holding time information.	Function name is updated, Single API to get all 'Time' related information.
RTCHourModeGet	RTCGetTime	This API reads the current time from the registers holding time information.	Function name is updated, Single API to get all 'Time' related information.
RTCSecondGet	RTCGetTime	This API reads the current time from the registers holding time information.	Function name is updated, Single API to get all 'Time' related information.
RTCMinuteGet	RTCGetTime	This API reads the current time from the registers holding time information.	Function name is updated, Single API to get all 'Time' related information.
RTCHourGet	RTCGetTime	This API reads the current time from the registers holding time information.	Function name is updated, Single API to get all 'Time' related information.
RTCMeridiemGet	RTCGetTime	This API reads the current time mode from the registers holding time information.	Function name is updated, Single API to get all 'Time' related information.
RTCDayOfMonthSet	RTCSetDate	This API sets the specified date information in registers holding date information..	Function name is updated, Single API to set all 'Date' related information
RTCMonthSet	RTCSetDate	This API sets the specified date information in registers holding date information..	Function name is updated, Single API to set all 'Date' related information
RTCYearSet	RTCSetDate	This API sets the specified date information in registers holding date information..	Function name is updated, Single API to set all 'Date' related information
RTCDayOfTheWeekSet	RTCSetDate	This API sets the specified date information in registers holding date information..	Function name is updated, Single API to set all 'Date' related information
RTCCalendarSet	RTCSetDate	This API sets the specified date information in registers holding date information..	Function name is updated, Single API to set all 'Date' related information

RTCDayOfMonthGet	RTCGetDate	This API reads the calendar information from relevant registers holding date information.	Function name is updated, Single API to get all 'Date' related information
RTCMonthGet	RTCGetDate	This API reads the calendar information from relevant registers holding date information.	Function name is updated, Single API to get all 'Date' related information
RTCYearGet	RTCGetDate	This API reads the calendar information from relevant registers holding date information.	Function name is updated, Single API to get all 'Date' related information
RTCDayOfTheWeekGet	RTCGetDate	This API reads the calendar information from relevant registers holding date information.	Function name is updated, Single API to get all 'Date' related information
RTCCalendarGet	RTCGetDate	This API reads the calendar information from relevant registers holding date information.	Function name is updated, Single API to get all 'Date' related information
RTCAlarm2TimeSet	RTCSetAlarm	This API sets the specified alarm register with Alarm information including Time and Date.	Function name is updated, Single API to set all alarm time & date related information.
RTCAlarm2CalendarSet	RTCSetAlarm	This API sets the specified alarm register with Alarm information including Time and Date.	Function name is updated, Single API to set all alarm time & date related information.
RTCAlarmSecondSet	RTCSetAlarm	This API sets the specified alarm register with Alarm information including Time and Date.	Function name is updated, Single API to set all alarm time & date related information.
RTCAlarmMinuteSet	RTCSetAlarm	This API sets the specified alarm register with Alarm information including Time and Date.	Function name is updated, Single API to set all alarm time & date related information.
RTCAlarmHourSet	RTCSetAlarm	This API sets the specified alarm register with Alarm information including Time and Date.	Function name is updated, Single API to set all alarm time & date related information.
RTCAlarmHourMeridiemSet	RTCSetAlarm	This API sets the specified alarm register with Alarm information including Time and Date.	Function name is updated, Single API to set all alarm time & date related information.
RTCAlarmTimeSet	RTCSetAlarm	This API sets the specified alarm register with Alarm information including Time and Date.	Function name is updated, Single API to set all alarm time & date related information.
RTCAlarmDayOfMonthSet	RTCSetAlarm	This API sets the specified alarm register with Alarm information including Time and Date.	Function name is updated, Single API to set all alarm time & date related information.

RTCAlarmMonthSet	RTCSetAlarm	This API sets the specified alarm register with Alarm information including Time and Date.	Function name is updated, Single API to set all alarm time & date related information.
RTCAlarmYearSet	RTCSetAlarm	This API sets the specified alarm register with Alarm information including Time and Date.	Function name is updated, Single API to set all alarm time & date related information.
RTCAlarmCalendarSet	RTCSetAlarm	This API sets the specified alarm register with Alarm information including Time and Date.	Function name is updated, Single API to set all alarm time & date related information.
RTCAlarmSecondGet	RTCGetAlarm	This API reads the programmed Alarm information including Time and Date from the specified alarm register.	Function name is updated, single API to get all alarm time & date related information.
RTCAlarmMinuteGet	RTCGetAlarm	This API reads the programmed Alarm information including Time and Date from the specified alarm register.	Function name is updated, single API to get all alarm time & date related information.
RTCAlarmHourGet	RTCGetAlarm	This API reads the programmed Alarm information including Time and Date from the specified alarm register.	Function name is updated, single API to get all alarm time & date related information.
RTCAlarmHourMeridiemGet	RTCGetAlarm	This API reads the programmed Alarm information including Time and Date from the specified alarm register.	Function name is updated, single API to get all alarm time & date related information.
RTCAlarmTimeGet	RTCGetAlarm	This API reads the programmed Alarm information including Time and Date from the specified alarm register.	Function name is updated, single API to get all alarm time & date related information.
RTCAlarmDayOfMonthGet	RTCGetAlarm	This API reads the programmed Alarm information including Time and Date from the specified alarm register.	Function name is updated, single API to get all alarm time & date related information.
RTCAlarmMonthGet	RTCGetAlarm	This API reads the programmed Alarm information including Time and Date from the specified alarm register.	Function name is updated, single API to get all alarm time & date related information.
RTCAlarmYearGet	RTCGetAlarm	This API reads the programmed Alarm information including Time and Date from the specified alarm register.	Function name is updated, single API to get all alarm time & date related information.
RTCAlarmCalendarGet	RTCGetAlarm	This API reads the programmed Alarm information including Time and Date from the specified alarm register.	Function name is updated, single API to get all alarm time & date related information.
RTCWakeUpAlarmEventControl	RTCWakeupEnable	This API is used to Enable the Alarm and Timer event Wakeup signal to the CPU.	Function name is updated.

RTCWakeUpTimerEventControl	RTCWakeupEnable	This API is used to Enable the Alarm and Timer event Wakeup signal to the CPU.	Function name is updated.
RTCWakeUpAlarmEventControl	RTCWakeupDisable	This API is used to Disable the Alarm and Timer event Wakeup signal to the CPU.	Function name is updated.
RTCWakeUpTimerEventControl	RTCWakeupDisable	This API is used to Disable the Alarm and Timer event Wakeup signal to the CPU.	Function name is updated.
RTCConfigPmicExtWake	RTCPmicExtWakeupEnable	This API Enables the PMIC External Wakeup feature.	2 new APIs for enable & disable.
RTCConfigPmicExtWake	RTCPmicExtWakeupDisable	This API Disables the PMIC External Wakeup feature.	2 new APIs for enable & disable.
RTCConfigPmicExtWakeDebounce	RTCPmicExtWakeupDebounceEnable	This API Enables the External Wakeup Debounce feature.	Separate APIs for enable & disable.
RTCConfigPmicExtWakeDebounce	RTCPmicExtWakeupDebounceDisable	This API Disables the External Wakeup Debounce feature	Separate APIs for enable & disable.
-NA-	RTCSetDebounceTime	This API sets the debounce time for the RTC module.	New API
RTCAutoCompEnable RTCTestModeControl	RTCTestModeEnable	This API is used to Enable/Disable the Test mode of the RTC.	Single API to both enable and disable.
RTCAutoCompDisable RTCTestModeControl	RTCTestModeEnable	This API is used to Enable/Disable the Test mode of the RTC.	Single API to both enable and disable.
RTCSet32CounterEnable	RTCCompensationEnable	This API Enable/Disable Crystal Compensation feature to account for any inaccuracy in 32K oscillator.	Single API to enable/disable RTC compensation.
RTCSet32CounterDisable	RTCCompensationEnable	This API Enable/Disable Crystal Compensation feature to account for any inaccuracy in 32K oscillator.	Single API to enable/disable RTC compensation.

The below table lists the deprecated APIs and their equivalent APIs in new starterware

Deprecated API	Equivalent API	Reason for deprecating the API
RTCEnableStatus	None	---
RTCRunStatusGet	None	---

Note

1: The following APIs have been removed as these are valid only on AM1808 SOC and are not supported on AM335x and AM43xx SOCs.

RTCSplitPwrEnable RTCSplitPwrDisable RTCSoftwareReset

DCAN

The below table lists the DAL file name differences between the versions

StarterWare 02.00.xx	StarterWare 02.01.xx:	Reason for change
dcan.c	dcan.c	No change.
dcan.h	dcan.h	No change.
hw_dcan.h	hw_dcan.h	No change.

The below table lists API differences between the versions.

StarterWare 02.00.xx: API Name	StarterWare 02.00.xx Functionality	StarterWare 02.01.xx API	StarterWare 02.01.xx : Change
DCANInitModeSet	Configure DCAN in Initialization mode.	DCANSetMode	API name is changed. "DCANSetMode" API shall be used to configure DCAN in Init/Normal mode.
DCANNormalModeSet	Configure DCAN in Normal mode.	DCANSetMode	API name is changed. "DCANSetMode" API shall be used to configure DCAN in Init/Normal mode.
DCANBitTimingConfig	Configure the DCAN bit-time values.	DCANBitTimeConfig	API name is changed.
DCANReset	Reset the DCAN module	DCANReset	No change.
DCANAutoReTransmitControl	The API enables/disables the auto retransmission of DCAN messages.	DCANAutoReTransmissionEnable	The API is updated. Macros are discarded and TRUE/FALSE values are used to enable/disable.
DCANTestModeControl	This API is used to enable/disable the test modes of DCAN.	DCANTestModeEnable	This API is discarded and functionality is merged with DCANTestModeEnable API.
DCANAutoBusOnControl	This API is used to enable/disable the Auto bus on feature of DCAN.	DCANAutoBusOnEnable	The API is updated. Macros are discarded and TRUE/FALSE values are being used.
DCANParityControl	The API is used to enable/disable the DCAN parity	DCANParityEnable	The API is updated. Macros are discarded and TRUE/FALSE values are used to enable/disable parity.
DCANIntLineEnable	The API is used to enable the interrupt lines	DCANIntrLineEnable	The API is updated. Enable/Disable is done in the same API using TRUE/FALSE values. Additional parameter will define the interrupt line number.
DCANIntLineDisable	The API is used to disable the interrupt lines.	DCANIntrLineEnable	The API is discarded since enable/disable part is done by DCANIntrLineEnable API.
DCANDmaRequestLineEnable	The API is used to configure DMA request for IF register transfer	DCANDmaRequestLineEnable	The API is updated. Enable/Disable is done within the same API using TRUE/FALSE values.
DCANDmaRequestLineDisable	Disable the DMA request for IF register transfer.	DCANDmaRequestLineEnable	This API is discarded since DCANDmaRequestLineEnable will do the work.

DCANTestModesEnable	The API is used to enable the DCAN test modes	DCANTestModeEnable	The API is updated. Enable/disable is done in the same API and parameter is added which shall deduce the test mode to be selected.
DCANTestModesDisable	The API will disable the DCAN test modes	DCANTestModeEnable	This API is discarded since DCANTestModeEnable will disable the test modes by taking parameter as FALSE.
DCANTxPinControl	The API will control the functioning of DCAN TX pin.	NA	This API is discarded since feature is not supported..
DCANRxPinStatusGet	This API will read the status of DCAN RX pin.	NA	This API is discarded since feature is not supported.
DCANIFBusyStatusGet	This API will check the busy status of IF registers	NA	API is discarded since the required bit will be checked inside the necessary APIs

GPMC

The below table lists the DAL file name differences between the versions

StarterWare 02.01.xx	StarterWare 02.00.xx	Reason for change
gpmc.c	gpmc.c	No Change
gpmc.h	gpmc.h	No Change
hw_gpmc.h	hw_gpmc.h	No Change

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.01.xx: Functionality	StarterWare 02.01.xx: Changes
GPMCAddrDataMuxProtocolSelect	GPMCSetAddrDataMuxType	This API sets the Address/Data pin multiplex (protocol) type.	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t** and muxType input in the following enum **gpmcMuxType_t**
GPMCSetDevType	GPMCSetDevType	This API sets the Device type for a particular chip select of the the GPMC module	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t** and devType input in the following enum **gpmcDevType_t**
GPMCDevSizeSelect	GPMCSetDevSize	This API sets the Device size for a particular chip select of the the GPMC module.	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t** and devSize input in the following enum **gpmcDevSize_t**
GPMCDevPageLenSet	GPMCSetDevPageLength	This API sets the device Page Length.	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t** and pageLen input in the following enum **gpmcDevPageLen_t**

GPMCAccessTypeSelect	GPMCSetAccessMode	This API sets the GPMC access mode to Single/Multiple Access for the Read/Write operations of the GPMC.	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t** and accessMode input in the following enum **gpmcAccessMode_t**
GPMCWriteTypeSelect and GPMCReadTypeSelect	GPMCSetAccessType	This API sets the access type to either the Synchronous or Asynchronous for the Read/Write operations of the GPMC.	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t** and accessType input in the following enum **gpmcAccessType_t**
GPMCSyncWrapBurstConfig	GPMCSyncWrapBurstEnable	This API enables/disables the Synchronous Wrap burst capability.	Function name is updated. The API takes csNum as input for identifying the chip select number from enum **gpmcChipSel_t**. It accepts TRUE/FALSE to decide enabling/disabling of the SyncWrapBurst feature.
GPMCBaseAddrSet	GPMCSetChipSelBaseAddr	This API sets the base Address for the selected chip select number.	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t.**
GPMCMaskAddrSet	GPMCSetChipSelMaskAddr	This API sets the Mask Address for the selected chip select number.	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t.**
GPMCCSConfig	GPMCChipSelEnable	This API controls enabling/disabling of a particular chip select number.	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t**and TRUE/FALSE to enable/disable the particular chip select.
GPMCFclkDividerSelect	GPMCSetFclkDivider	This API sets the clock divider for GPMC functional clock which is used as the base for all timing values for read/write accesses.	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t**'. 'The divider value for the functional clock takes any of the input values from the following enum **gpmcFclkDivider_t**
GPMCTimeParaGranularitySelect	GPMCSetTimeParaGranularity	This API sets the Time Granularity scaling factor which sets the latency for all the timing signal values.	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t**. The scale factor which decides the latency for the functional clock takes any of the input values from the following enum **gpmcTimeGranularity_t**
GPMCClkActivationTimeConfig	GPMCSetClkActivationTime	This API sets the Output Clock Activation time for the selected chip select Number.	Function name is updated. The macro value to identify chip select number is grouped under in the following enum **gpmcChipSel_t**. The clk activation values are taken from the following enum **gpmcTimeGranularity_t**

GPMCCSTimingConfig	GPMCChipSelectTimingConfig	This API configures the Timing Parameters for the Chip Select signal.	Function name is updated. The API takes csNum as input for identifying the chip select number from enum **gpmcChipSel_t**. The timing parameters to be written for Chip Select are part of the structure **gpmcChipSelTimingParams_t**. This structure needs to be populated with required timing values to be set instead of the parameterized macro from the previous implementation of the API.
NA	GPMCGetPrefetchFifoPtr	This API returns the PREFETCH engine FIFO pointer pointing to the number of bytes available to be read or free empty space to be written.	New API
NA	GPMCSetBchEccCapability	This API sets the BCH Error Correction Capability to be used.	New API
NA	GPMCSetBchWrapModeVal	This function sets the wrap mode for the ECC BCH algorithm.	New API

ELM

The below table lists the DAL file name differences between the versions

StarterWare 02.01.xx	StarterWare 02.00.xx	Reason for change
elm.c	elm.c	No Change
elm.h	elm.h	No Change
hw_elm.h	hw_elm.h	No Change

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.01.xx: Functionality	StarterWare 02.01.xx: Changes	Notes
ELMModeSet	ELMSetContinuousMode	This API sets the Error location engine to continuous mode.	The continuous mode and Page Mode setting functionality is merged in a single API.	NA
ELMModeSet	ELMSetPageMode	This API sets the Error location engine to Page mode.	The continuous mode and Page Mode setting functionality is merged in a single API.	NA

EDMA

The below table lists the DAL file name differences between the versions

StarterWare 02.00.xx	StarterWare 02.01.xx	Reason for change
edma.c	edma.c	No change
edma.h	edma.h	No change
hw_edma.h	hw_edma.h	No change

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.00.xx: Functionality	StarterWare 02.01.xx : Changes
EDMA3Init	EDMAInit	This API initializes all regions of EDMA module.	API name is changed.
NA	EDMARegionReset	This API resets given region of EDMA module.	New API
EDMA3PeripheralIdGet	EDMAGetPeripheralId	This API returns the peripheral id of the EDMA module.	API name is changed.
NA	EDMAIsMemProtectionSupported	This API checks if the EDMA module supports memory protection.	New API
NA	EDMAIsChMapSupported	This API checks if the EDMA module supports channel mapping.	New API
NA	EDMAGetNumRegions	This API gives number of memory protected or shadow regions the EDMA module supports.	New API
NA	EDMAGetNumQueues	This API gives number of queues or transfer controllers the EDMA module supports.	New API
NA	EDMAGetNumParamSets	This API gives number of Parameter sets the EDMA module supports.	New API
NA	EDMAGetNumIntrCh	This API gives number of interrupt channels the EDMA module supports.	New API
NA	EDMAGetNumQdmaCh	This API gives number of QDMA channels the EDMA module supports.	New API
NA	EDMAGetNumDmaCh	This API gives number of DMA channels the EDMA module supports.	New API
EDMA3EnableChInShadowReg	EDMAShdwRegionChEnable	This API enables DMA/QDMA channel in the the given shadow region.	API name is changed.
EDMA3DisableChInShadowReg	EDMAShdwRegionChDisable	This API disable DMA/QDMA channel in the the given shadow region.	API name is changed.
NA	EDMAIsChInShdwRegion	This API checks if DMA/QDMA channel is assigned to given region.	New API
EDMA3ChannelToParamMap	EDMAChToParamSetMap	This API maps QDMA/DMA channel to the given PaRAM set.	API name is changed.

EDMA3MapChToEvtQ	EDMAChToEvtQueueMap	This API maps QDMA/DMA channel to the given event queue.	API name is changed.
EDMA3SetQdmaTrigWord	EDMASetQdmaTriggerWord	This API sets the Trigger word for the specific QDMA channel in the QCHMAP Register. Default QDMA trigger word is CCNT.	API name is changed.
EDMA3ClrMissEvt	EDMAChMissEvtClear	This API clears the missed event status for channel in given region	API name is changed.
NA	EDMAEvtMissStatus	This API gives status of the missed event for given QDMA/DMA channel.	New API
EDMA3ClrCCErr	EDMAComplCodeErrClear	This API clears the completion code error status.	API name is changed.
NA	EDMAQueueThresholdErrClear	This API clears the threshold error status for given queue.	New API
EDMA3SetEvt	EDMASetEvt	This API manually sets event to initiate transfer on DMA channel in the the given shadow region.	API name is changed.
EDMA3ClrEvt	EDMAEvtClear	This API clears event of DMA channel in the given global or shadow region.	API name is changed.
NA	EDMAEvtStatus	This API gives event status of DMA channel in the given global or shadow region.	API name is changed.
EDMA3EnableDmaEvt EDMA3EnableQdmaEvt	EDMAEvtEnable	This API enables event of channel in the given global or shadow region.	API name is changed.
EDMA3DisableDmaEvt EDMA3DisableQdmaEvt	EDMAEvtDisable	This API disables event of channel in the given global or shadow region.	API name is changed.
EDMA3GetCCErrStatus	EDMAComplCodeErrStatus	This API gives the completion code error status.	API name is changed.
NA	EDMAQueueThresholdErrStatus	This API gives the threshold error status for given queue..	API name is changed.
EDMA3GetIntrStatus EDMA3IntrStatusHighGet EDMA3GetErrIntrStatus EDMA3ErrIntrHighStatusGet EDMA3QdmaGetErrIntrStatus	EDMAIntrStatus	This API gives pending interrupt of a set of channel in the given global or shadow region.	API name is changed.
EDMA3ClrIntr	EDMAIntrClear	This API clear pending interrupt of a channel in the given global or shadow region.	API name is changed.
EDMA3GetPaRAM EDMA3QdmaGetPaRAM	EDMAGetParamEntry	This API gives PaRAM Set of given index.	API name is changed.
EDMA3SetPaRAM EDMA3QdmaSetPaRAM EDMA3QdmaSetPaRAMEntry EDMA3QdmaGetPaRAMEntry	EDMASetParamEntry	This API takes a PaRAM Set as input and copies it onto the given PaRAM Set index.	API name is changed.
NA	EDMAParamReset	This API resets the given PaRAM Set index	New API

NA	EDMAParamDataConfig	This API takes a PaRAM Set as input and copies it onto the given PaRAM Set index.	New API
NA	EDMAParamXferConfig	This API takes a PaRAM Set as input and copies it onto the given PaRAM Set index.	New API
NA	EDMAParamConfig	This API takes a PaRAM Set as input and copies it onto the given PaRAM Set index.	New API
EDMA3RequestChannel	EDMAChConfig	This API configures given channel.	API name is changed.
EDMA3FreeChannel	EDMAChReset	This API resets given channel.	API name is changed.
EDMA3EnableTransfer	EDMATransferStart	Start EDMA transfer on the specified channel.	API name is changed.
EDMA3DisableTransfer	EDMATransferStop	Disable DMA transfer on the specified channel.	API name is changed.

LCDC

The below table lists the DAL file name differences between the versions

StarterWare 02.00.xx	StarterWare 02.01.xx	Reason for change
raster.c	lcdc.c	Starterware the dal file names should be module names. Since raster is not the module name, this file is renamed to "lcdc.c".
raster.h	lcdc.h	Same as above.

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.00.xx: Functionality	StarterWare 02.01.xx : Changes
RasterClkConfig	LCDCClkConfig	This function configures clk divider to generate required frequency of pixel clock and selects the raster mode in lcdc controller.	API is renamed. API name should start with module name which is LCDC not Raster.
RasterIdGet	LCDCGetRevision	This function returns the Reversion Id of LCD controller.	API is renamed.
RasterAutoUnderFlowEnable	LCDCAutoUnderFlowEnable	This function enables auto under flow feature.	API is renamed. The functionality of both autounderflow enable and disable apis are combined into single API. The unified API will accept additional parameter "**enableAutoUnderFlow**", which accepts the following parameters **TRUE** or**FALSE.**
RasterAutoUnderFlowDisable	Same as above	This function disables auto under flow feature.	Same as above

RasterEnable	LCDCRasterEnable	This function enables raster control.	API is renamed. The functionality of both raster enable and disable apis are combined into single API. The unified API will accept additional parameter "**enableRaster**", which accepts the following parameters **TRUE** or **FALSE**.
RasterDisable	Same as above	This function disables raster control.	Same as above.
RasterModeConfig	LCDCRasterDisplayModeConfig	This function will configures LCD to MonoChrome or color mode, TFT or STN mode and palette loading mode.	API is renamed. All the parameters that are grouped into a single structure "**lcdcRasterDisplayAttrCfg_t**" and the pointer holding the address of this structure is passed to the API.
RasterLSBDataOrderSelect	LCDCRasterDataOrderSelect	This function orders the frame buffer data from least to most significant bit bit/nibble/byte/word.	API is renamed. The functionality of the two data order select apis is combined into single API. The unified API will accept additional parameter "**dataOrder**", which accepts the macros from the following enum "**lcdcRasterDataOrder_t**".
RasterMSBDataOrderSelect	Same as above	This function orders the frame buffer data from most to least significant bit bit/nibble/byte/word.	Same as above
RasterIntEnable	LCDCIntrEnable	This API enables selected interrupts in LCD controller.	API is renamed. One of the input parameter "**flag**" is renamed to "**intrMask**" and the macros that this parameter accepts are grouped into enum "**lcdcIntrMask_t**".
RasterIntDisable	LCDCIntrDisable	This API disables selected interrupts in LCD controller.	Same as above.
RasterNibbleModeEnable	LCDCRasterNibbleModeEnable	This function enables nibble mode.	API is renamed. The functionality of both nibble mode enable and disable apis are combined into single API. The unified API will accept additional parameter "**enableNibbleMode**", which accepts the following parameters **TRUE** or **FALSE**.
RasterNibbleModeDisable	Same as above	This function disables nibble mode.	Same as above
RasterFIFODMADelayConfig	LCDCRasterSetDmaFifoDelay	This function configures input FIFO delay.	API is renamed.
RasterHparamConfig	LCDCRasterTimingConfig	This function configuers horizontal timing parametes and number of pixel per line.	API is renamed. The functionality of the Hparam and Vparam configuration parameters are combined into single API. This unified Api groups all the input parameters into a structure "**lcdcRasterTimingCfg_t**" and the address of this sturcutue is passed as a parameter to this API.
RasterVparamConfig	Same as above	This function configuers vertical timing parameters and number of lines per panel.	Same as above

RasterTiming2Configure	LCDCRasterPolarityConfig	This function configures the polartiy of various timing parameters of LCD controller.	API is renamed. ACBias configurations have been removed from this API as new api is defined for only acbias cofiguration. All the input parameters passed are grouped into a structure "**lcdcRasterPolarityCfg_t**" and the address of this structure is passed as a parameter to this API.
RasterDMAConfig	LCDCDmaConfig	This function configures DMA present inside LCD controller.	API is renamed. All the input parameters are grouped into a structure "**lcdcDmaCfg_t**" and the address of this sturucture is passed as a parameter to this API.
RasterByteSwapEnable	LCDCByteSwapEnable	This function enables byte swap with in a half word of the dma transfer.	API is renamed. The functionality of both byte swap enable and disable apis are combined into single API. The unified API will accept additional parameter "**enableByteSwap**", which accepts the following parameters **TRUE** or **FALSE**.
RasterByteSwapDisable	Same as above.	This function disables byte swap with in a half word of the dma transfer.	Same as above.
RasterIntStatus	LCDCIntrStatus	This function returns status of the specified interrupts.	API is renamed. Old API is accepting a parameter "flag" which selects the status of the required interrupts, but the new API will return status of the all the interrupts. It is up to user to check the status of the required interrupts.
RasterIntRawStatus	LCDCIntrRawStatus	This function returns raw status of the specified interrupts.	API is renamed. Old API is accepting a parameter "flag" which returns the raw status of the required interrupts, but the new API will return raw status of the all the interrupts. It is up to user to check the status of the required interrupts.
RasterIntRawStatusSet	LCDCIntrTrigger	This function asserts specified interrupts.	API is renamed. One of the input parameter "flag" is renamed to "**intrMask**" and the macros that this parameter accepts are grouped into enum "**lcdcIntrMask_t**".
RasterDMAFBConfig	LCDCDmaFrameBufConfig	This function configures base and ceiling value for Frame buffer.	"API is renamed. This API functionality is not changed except the following parameters are renamed : **base** -> **bufAddr**, **flag** ->**frmBufId**. The macros that the parameter "**frmBufId**" accepts are grouped into the following enum "**lcdcFrmBufId_t**".
RasterClearGetIntStatus	LCDCIntrClear	This function clear status of the selected interrupts.	API is renamed. One of the input parameter "**flag**" is renamed to "**intrMask**" and the macros that this parameter accepts are grouped into enum "**lcdcIntrMask_t**". Old API will return the status along with clearing status, whereas new API clears status and will not return the status.

RasterSubPanelEnable	LCDCRasterSubPanelEnable	This function enables raster subpanel feature	API is renamed. The functionality of both raster sub panel enable and disable apis are combined into single API. The unified API will accept additional parameter "**enableSubPanel**", which accepts the following parameters **TRUE** or **FALSE**.
RasterSubPanelDisable	Same as above	This function disables raster subpanel feature	Same as above
RasterSubPanelConfig	LCDCRasterSubPanelConfig	This function configures raster subpanel feature	API is renamed. No functionality change. Input parameters are renamed as **hols** -> **subPanelPos**, **lppt** ->**linesPerPanelThr**, **dpd** ->**defaultPixelData**. The parameter "**subPanelPos**" can take macros from the enum "**lcdcRasterSubPanelPos_t**".
RasterClocksEnable	LCDCClocksEnable	This function Enables the clock for the DMA submodule,LIDD submodule and for the core(which encompasses the Raster active matrix and Passive matrix).	API is renamed.
RasterSoftWareClkEnable	LCDCRasterClkEnable	This function enables software clock for the raster,which encompasses the raster active matrix and passive matrix logic.	API is renamed. The functionality of both raster clock enable and disable apis are combined into single API. The unified API will accept additional parameter "**enableRasterClk**", which accepts the following parameters **TRUE** or **FALSE**.
RasterSoftWareClkDisable	Same as above	This function disables software clock for the raster,which encompasses the raster active matrix and passive matrix logic.	Same as above
RasterDMASoftWareClkEnable	LCDCDmaClkEnable	This function enables software clock for the DMA submodule.	API is renamed. The functionality of both Dma clock enable and disable apis are combined into single API. The unified API will accept additional parameter "**enableDmaClk**", which accepts the following parameters **TRUE** or **FALSE**.
RasterDMASoftWareClkDisable	Same as above	This function disables software clock for the DMA submodule.	Same as above
RasterSoftWareResetControlEnable	LCDCSoftwareResetEnable	This function does Software Resets for LCD module or DMA submodule or Core which encompasses Raster Active Matrix and Passive Matrix logic based on the "flag" argument passed to this function.	API is renamed. The functionalities of enable and disable apis are combined into single api, by adding one more parameter "**enableSoftwareReset**".The input parameter "**moduleId**" accepts macros from the enum "**lcdcSubModuleId_t**" and the parameter "**enableSoftwareReset**" accepts the macros **TRUE** or **FALSE**.

RasterSoftWareResetControlDisable	Same as above	This function disables Software Resets for LCD module or DMA submodule or Core which encompasses Raster Active Matrix and Passive Matrix logic based on the "flag" argument passed to this function.	Same as above
RasterDmaMasterPrioritySet	LCDCSetDmaMasterPriority	This function sets the priority for the L3 OCP Master Bus.	API is renamed.
RasterStandbyModeConfig	LCDCSetStandbyMode	This API configures the standby mode of LCD controller.	API is renamed. The parameter is renamed as **flag -> standbyMode** and the macros that this parameter accepts are grouped into an enum **lcdcStandbyMode_t**.
RasterIdleModeConfig	LCDCSetIdleMode	This API configures the Idle mode of LCD controller.	API is renamed. The parameter is renamed as **flag -> idleMode** and the macros that this parameter accepts are grouped into an enum **lcdcIdleMode_t**.
RasterContextSave	LCDCContextSave	This API saves the context of LCDC registers.	API is renamed. The structure which holds the context information is renamed as **RASTERCONTEXT -> lcdcContext_t**.
RasterContextRestore	LCDCContextRestore	This API restores the context of LCDC registers.	API is renamed. The structure which holds the context information is renamed as RASTERCONTEXT -> lcdcContext_t.

The below table lists the deprecated APIs and their equivalent APIs in new starterware

Deprecated API	Equivalent API	Reason for deprecating the API
RasterEndOfFrameIntEnable	RasterEndOfFrameIntEnable	This API is valid only for am1808 raster but not for am335x raster.
RasterEndOfFrameIntDisable	RasterEndOfFrameIntDisable	Same as above.

MMCSD

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.00.xx: Functionality	StarterWare 02.01.xx : Changes
HSMMCSDLinesReset	HSMMCSDLinesReset	This API performs the soft reset of the MMC/SD controller lines.	The parameter "**flag**" is renamed as "**resetMask**" and the macros that this parameter accepts have been grouped into an enum "**hsMmcsdResetLineMask_t**".
HSMMCSDSystemConfig	HSMMCSDSystemConfig	This API performs the system configurations of the MMC/SD controller like standby, idle and wake up modes.	Instead of passing a configuration parameter "**config**", all the system configuration parameters have been grouped into a structure "**hsMmcsdSysCfg_t**" and the pointer which holds the address of this struture has been passed as a parameter.
HSMMCSDBusWidthSet	HSMMCSDSetBusWidth	This API configures the MMC/SD controller bus width.	API is renamed. The macros that the parameter "**width**" accepts have been grouped into an enum "**hsMmcsdBusWidth_t**".

HSMMCSDBusVoltSet	HSMMCSDSetBusVolt	This API configures the MMC/SD bus voltage.	API is renamed. The macros that the parameter "**voltage**" accepts have been grouped into an enum "**hsMmcsdBusVolt_t**".
HSMMCSDBusPower	HSMMCSDBusPowerOnCtrl	This API turns MMC/SD bus power on / off.	API is renamed. The macros that the parameter "**pwrCtrl**" accepts have been grouped into an enum "**hsMmcsdPwrCtrl_t**".
HSMMCSDIntClock	HSMMCSDIntClockEnable	This API controls the enable/disable of internal clocks.	API is renamed. The second parameter "**pwr**" is renamed as "**enableIntClk**" and the supported macros are **TRUE** or **FALSE**".
HSMMCSDSupportedVoltSet	HSMMCSDSetSupportedVoltage	This API sets the supported bus voltages.	API is renamed. The parameter "**volt**" is renamed as "**voltMask**" and the macros that this parameter accepts have been grouped into an enum "**hsMmcsdSuppVolt_t**".
HSMMCSDIntrStatusEnable	No Change.	This API enables the controller events to set flags in status register.	The parameter "**flag**" is renamed as "**intrMask**" and the macros that this parameter accepts have been grouped into an enum "**hsMmcsdIntrMask_t**".
HSMMCSDIntrStatusDisable	No Change.	This API disables the controller events to set flags in status register.	The parameter "**flag**" is renamed as "**intrMask**" and the macros that this parameter accepts have been grouped into an enum "**hsMmcsdIntrMask_t**".
HSMMCSDIntrEnable	No Change.	This API enables the controller events to generate a h/w interrupt request.	The parameter "**flag**" is renamed as "**intrMask**" and the macros that this parameter accepts have been grouped into an enum "**hsMmcsdIntrMask_t**".
HSMMCSDIntrStatusGet	HSMMCSDIntrStatus	This API gets the status of the hardware interrupts.	API is renamed. Instead of reading the status of specific interrupts, this will return status of the all the interrupts and it is upto the application to check the status of the specific interrupts. Since it is returning entire status, the second paramter is removed from the API defintion.
---	HSMMCSDIntrDisable	This API disables the controller events to generate a h/w interrupt request.	This API is newly added.
HSMMCSDIntrStatusClear	HSMMCSDIntrClear	This API clears the interrupt status bits of the controller.	API is renamed. The parameter "**flag**" is renamed as "**intrMask**" and the macros that this parameter accepts have been grouped into an enum "**hsMmcsdIntrMask_t**".
HSMMCSDCommandSend	No change	Pass the MMC/SD command to the controller/card.	All the command configuration parameters have been grouped into a strucutre **hsMmcsdCmdObj_t**. Instead of passing all the parameters, the address of strucutre type **hsMmcsdCmdObj_t** which contain all the parameters have been passed to the API.

The below table lists the deprecated APIs and their equivalent APIs in new starterware

Deprecated API	Equivalent API	Reason for deprecating the API
HSMMCSDDataGet	HSMMCSDDataGet	---

CPSW

The below table lists API differences between the versions

StarterWare 02.00.xx: API Name	StarterWare 02.01.xx: API Name	StarterWare 02.00.xx: Functionality	StarterWare 02.01.xx : Changes
---	CPSWReset	This API resets the CPSW.	
---	CPSWCpdmaReset	This API resets the CPDMA module.	
CPSWSlControlExtEnable	CPSWSlCtrlInbandEnable	This API configures the operating mode of the sliver.	
CPSWSlGigModeForceEnable	CPSWSlForceExtGigbitMode	This API configures the CPGMAC of sliver if the input gmii_mt_clk has been stopped by the PHY.	
CPSWSlGigModeForceDisable	----		
CPSWSlTransferModeSet	CPSWSlGigModeEnable	This API configures the transfer speed of the sliver to gigabit or non-gigabit mode.	
---	CPSWSlSetDuplexMode	This API sets the transfer duplex mode of the sliver.	
CPSWSlMACStatusGet	----		
CPSWSlRxMaxLenSet	CPSWSetSlRxMaxLen	This API sets the maximum length for received frame in sliver.	
CPSWSlGMIIEnable	CPSWSlGmiiEnable	This API configures the sliver to enable GMII for gigabit mode.	
----	CPSWSetSlRmiiSpeed	This API configures the RMII input speed of slover to either 10Mbps or 100Mbps operation.	
----	CpswMacOperModeConfig	This API configures the operating speed and duplex mode for MAC of given port.	
CPSWSlRGMIIEnable	---		
----	CPSWSlIsExtSpeedGigabit	This API checks the external speed status of sliver is gigabit or non-gigabyte (10/100 Mbps) for in-band mode of operation.	
----	CPSWSlIsExtFullDuplex	This API check the external duplex status is Half/Full for in-band mode of operation.	
----	CPSWSlIsMacIdle	This API checks if the sliver is in idle state.	
CPSWWrControlRegReset	----		
CPSWWrCoreIntEnable	CPSWWrChIntrEnable	This API enables RX, TX or RX threshold interrupts of a channel. It configures given interrupt core interfaced to processor.	
----	CPSWWrMiscIntrEnable	This API enables miscellaneous interrupts. It configures given interrupt core in wrapper module interfaced to processor.	
CPSWWrCoreIntDisable	CPSWWrchIntrDisable	This API disables RX, TX or RX threshold interrupts of a channel. It configures given interrupt core in wrapper module interfaced to processor.	
	CPSWWrMiscIntrDisable	This API disables miscellaneous interrupts. It configures given interrupt core in wrapper module interfaced to processor.	

CPSWWrCoreIntStatusGet	CPSWWrchIntrStatus	This API gives status given channel interrupt of all channels for given interrupt core in wrapper module interfaced to processor.	
	CPSWWrMiscIntrStatus	This API gives status for miscellaneous interrupts of given interrupt core in wrapper module interfaced to processor.	
CPSWWrRGMIIStatusGet	CPSWGetWrRgmiiInbandSpeed	This API gives speed of CPRGMII in in-band mode for given slave port.	
	CPSWGetWrRgmiiInbandDuplexMode	This API gives duplex mode of CPRGMII in in-band mode for given slave port.	
	CPSWWrRgmiiInbandLinkStatus	This API gives link status of CPRGMII in in-band mode for given slave port.	
	CPSWAleEnable	This API enables/disables packet processing by ALE.	
	CPSWAleClearTable	This API clears the entries of address lookup table.	
	CPSWAleBypassEnable	This API enables/disables the bypassing of the ALE logic for all packets received on slave ports to the host port.	
CPSWALEBypassEnable			
CPSWALEBypassDisable			
CPSWRxFlowControlEnable	CPSWSsRxFlowCtrlEnable	This API enables/disables the receive flow control for a given port in CPSW system.	
CPSWRxFlowControlDisable			
CPSWSoftwareIdleEnable	CPSWSsSwIdleEnable	This API enables/disables the software idle mode, causing the switch fabric to stop forward packets at the next start of packet.	
CPSWSoftwareIdleDisable			
CPSWStatisticsEnable	CPSWSsStatsEnable	This API enables/disables the CPSW statistics for the given port	
CPSWVLANAwareEnable	CPSWSsVlanAwareEnable	This API enables/disables the VLAN aware mode for CPSW.	
CPSWVLANAwareDisable			
CPSWPortSrcAddrSet	CPSWSetPortSrcAddr	This API configures the MAC address of the slave port.	
CPSWStatisticsGet			
CPSWCPDMAReset			
CPSWCPDMACmdIdleEnable	CPSWCpdmaIdleEnable	This API enable/disable the idle mode for CPDMA. When enabled idle, the CPSW stops all the reception and transmission. However, if receiving the current frame will be received completely before going to the idle state. Also, while transmitting, the contents in the fifo will be sent fully.	
CPSWCPDMACmdIdleDisable			
CPSWCPDMATxIntEnable	CPSWCpdmaChIntrEnable	This API enables the transfer interrupt for given channel.	

CPSWCPDMARxIntEnable			
CPSWCPDMATxIntDisable	CPSWCpdmaChIntrDisable	This API disables the transfer interrupt for given channel.	
CPSWCPDMARxIntDisable			
CPSWCPDMATxEnable	CPSWCpdmaXferEnable	This API enables the transfer. Any write to hardware descriptor of a channel will start transfer.	
CPSWCPDMARxEnable			
	CPSWCpdmaXferDisable	This API disables the transfer.	
CPSWCPDMATxHdrDescPtrWrite	CPSWCpdmaWriteXferHdrDescPtr	This API writes the transfer HDP register. If transfer is enabled, write to the transfer HDP will immediately start transfer. The data will be taken from the buffer pointer of the transfer buffer descriptor written to the transfer HDP.	
CPSWCPDMARxHdrDescPtrWrite			
CPSWCPDMAEndOfIntVectorWrite	CPSWCpdmaWriteEoiVector	This API writes the interrupt line number to End Of Interrupt Vector.	
CPSWCPDMATxCPWrite	CPSWCpdmaWriteXferCp	This API writes the the transfer Completion Pointer for a specific channel.	
CPSWCPDMARxCPWrite			
CPSWCPDMAStatusGet			
	CPSWIsCpdmaIdle	This API checks if the CPDMA is in idle state.	
	CPSWGetCpdmaHostErrorCode	Returns the CPDMA Status.	
	CPSWGetCpdmaHostErrorCh	This API returns the channel number on which error occurred.	
CPSWCPDMAConfig			
	CPSWCpdmaTxRateLimitChBusConfig	This API configures the transmit rate limit channel bus of CPDMA module.	
CPSWCPDMATxIntStatRawGet	CPSWCpdmaChIntrRawStatus	This API returns the raw status of given interrupt for all channels.	
CPSWCPDMARxIntStatRawGet			
CPSWCPDMATxIntStatMaskedGet	CPSWCpdmachIntrStatus	This API returns the status of given interrupt for all channels.	
CPSWCPDMARxIntStatMaskedGet			
CPSWContextSave			
CPSWContextRestore			
CPSWHostPortDualMacModeSet	CPSWHostPortDualMacModeEnable	This API configures the host port for Dual MAC mode.	
	CPSWAleVlanAwareEnable	This API configures VLAN Aware in ALE module to determine what to be done if VLAN is not found.	
CPSWALEVLANAwareSet			
CPSWALEVLANAwareClear			
CPSWPortVLANConfig	CPSWPortVlanConfig	This API configures Port VLAN.	

Article Sources and Contributors

Processor SDK RTOS Migration Guide for StarterWare 02.00.xx to 02.01.xx *Source*: http://ap-fpdsp-swapps.dal.design.ti.com/index.php?oldid=220325 *Contributors*: A0850439, X0189513, X0222293