My Project

v1.0.0

Generated by Doxygen 1.10.0

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Active_Bit_Register_TypeDef Struct Reference

Public Attributes

- volatile uint32 **IABR0**
- volatile uint32 **IABR1**
- volatile uint32 **IABR2**
- volatile uint32 **IABR3**
- volatile uint32 **IABR4**
- volatile uint32 **IABR5**
- volatile uint32 **IABR6**
- volatile uint32 **IABR7**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.2 Can_ConfigType Struct Reference

Collaboration diagram for Can_ConfigType:

Public Attributes

- CanConfigSet **CanConfigSet**

The documentation for this struct was generated from the following file:

- Can_Module/Can.h

3.3 CAN_Filter_Bank_TypeDef Struct Reference

Public Attributes

- volatile uint32 **CAN_FiR1**
- volatile uint32 **CAN_FiR2**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.4 CAN_Filter_Config_t Struct Reference

Public Attributes

- uint32 **Filter_ID**
- uint32 **Filter_Mask_ID**
- uint32 **Filter_FIFO_Assignment**
- uint32 **Filter_Bank**
- uint32 **Filter_Mode**
- uint32 **Filter_Scale**

The documentation for this struct was generated from the following file:

- Can_Module/stm32_f103c6_CAN.h

3.5 Can_HwType Struct Reference

Public Attributes

- Can_IdType **CanId**
- Can_HwHandleType **Hoh**
- uint8 **ControllerId**

The documentation for this struct was generated from the following file:

- Can_Module/Can_GeneralTypes.h

3.6 Can_PduType Struct Reference

Public Attributes

- Can_IdType **id**
- uint8 * **sdu**
- PduIdType **swPduHandle**
- uint8 **length**

The documentation for this struct was generated from the following file:

- Can_Module/Can_GeneralTypes.h

3.7 CAN_RX_FIFO_TypeDef Struct Reference

Public Attributes

- volatile uint32 **CAN_RIxR**
- volatile uint32 **CAN_RDTxR**
- volatile uint32 **CAN_RDLxR**
- volatile uint32 **CAN_RDHxR**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.8 CAN_TX_mailbox_TypeDef Struct Reference

Public Attributes

- volatile uint32 **CAN_TIxR**
- volatile uint32 **CAN_TDTxR**
- volatile uint32 **CAN_TDLxR**
- volatile uint32 **CAN_TDHxR**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.9 CanConfigSet Struct Reference

Collaboration diagram for CanConfigSet:

Public Attributes

- CanController **CanController**
- CanHardwareObject **CanHardwareObject** [Max_Num_HOH]

The documentation for this struct was generated from the following file:

- Can_Module/Can.h

3.10 CanController Struct Reference

Collaboration diagram for CanController:

Public Attributes

- uint8 **CanBusoffProcessing**
- boolean **CanControllerActivation**
- uint32 **CanControllerBaseAddress**
- uint8 **CanControllerId**
- uint8 **CanRxProcessing**
- uint8 **CanTxProcessing**
- uint8 **CanWakeupProcessing**
- boolean **CanWakeupSupport**
- CanControllerBaudrateConfig **CanControllerBaudrateConfig** [Max_Num_Baud_Rate]
- CanControllerBaudrateConfig * **CanControllerDefaultBaudrate**

The documentation for this struct was generated from the following file:

- Can_Module/Can.h

3.11 CanControllerBaudrateConfig Struct Reference

Public Attributes

- float32 **CanControllerBaudRate**
- uint16 **CanControllerBaudRateConfigID**
- uint16 **CanControllerPropSeg**
- uint8 **CanControllerSeg1**
- uint8 **CanControllerSeg2**
- uint8 **CanControllerSyncJumpWidth**

The documentation for this struct was generated from the following file:

- Can_Module/Can.h

3.12 CanGeneral Struct Reference

Public Attributes

- boolean **Active**

The documentation for this struct was generated from the following file:

- Can_Module/Can.h

3.13 CanHardwareObject Struct Reference

Collaboration diagram for CanHardwareObject:

Public Attributes

- CanHandleType **CanHandleType**
- uint16 **CanHwObjectCount**
- CanIdType **CanIdType**
- uint8 **CanObjectId**
- CanObjectType **CanObjectType**
- boolean **CanTriggerTransmitEnable**
- CanController * **CanControllerRef**
- CanHwFilter **CanHwFilter**

The documentation for this struct was generated from the following file:

- Can_Module/Can.h

3.14 CanHwFilter Struct Reference

Public Attributes

- uint32 **CanHwFilterCode**
- uint32 **CanHwFilterMask**

The documentation for this struct was generated from the following file:

- Can_Module/Can.h

3.15 Clear_Enable_Register_TypeDef Struct Reference

Public Attributes

- volatile uint32 **ICER0**
- volatile uint32 **ICER1**
- volatile uint32 **ICER2**
- volatile uint32 **ICER3**
- volatile uint32 **ICER4**
- volatile uint32 **ICER5**
- volatile uint32 **ICER6**
- volatile uint32 **ICER7**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.16 Clear_Pending_Register_TypeDef Struct Reference

Public Attributes

- volatile uint32 **ICPR0**
- volatile uint32 **ICPR1**
- volatile uint32 **ICPR2**
- volatile uint32 **ICPR3**
- volatile uint32 **ICPR4**
- volatile uint32 **ICPR5**
- volatile uint32 **ICPR6**
- volatile uint32 **ICPR7**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.17 Dio_ChannelGroupType Struct Reference

Public Attributes

- uint8 **mask**
- uint8 **offset**
- Dio_PortType **port**

The documentation for this struct was generated from the following file:

- DIO_Module/Dio.h

3.18 FIFO_Buf_t Struct Reference

Public Attributes

- unsigned int **counter**
- element_type * **head**
- element_type * **tail**
- element_type * **base**
- unsigned int **length**

The documentation for this struct was generated from the following file:

- OSEK_VDX/Inc/MY_RTOS_FIFO.h

3.19 GPIO_TypeDef Struct Reference

Public Attributes

- volatile uint32 **CRL**
- volatile uint32 **CRH**
- volatile uint32 **IDR**
- volatile uint32 **ODR**
- volatile uint32 **BSRR**
- volatile uint32 **BRR**
- volatile uint32 **LCKR**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.20 Icu_ConfigType Struct Reference

Collaboration diagram for Icu_ConfigType:

Public Attributes

- IcuConfigSet **IcuConfigSet**

The documentation for this struct was generated from the following file:

- ICU_Module/Icu.h

3.21 Icu_DutyCycleType Struct Reference

Public Attributes

- Icu_ValueType **ActiveTime**
- Icu_ValueType **PeriodTime**

The documentation for this struct was generated from the following file:

- ICU_Module/Icu.h

3.22 IcuChannel Struct Reference

Collaboration diagram for IcuChannel:

Public Attributes

- Icu_ChannelType **IcuChannelId**
- Icu_ActivationType **IcuDefaultStartEdge**
- Icu_MeasurementModeType **IcuMeasurementMode**
- boolean **IcuWakeupCapability**
- IcuSignalMeasurement **IcuSignal_Measurement**

The documentation for this struct was generated from the following file:

- ICU_Module/Icu.h

3.23 IcuConfigSet Struct Reference

Collaboration diagram for IcuConfigSet:

Public Attributes

- Icu_ChannelType **IcuMaxChannel**
- IcuChannel **IcuChannel** [IcuMax_Channel]

The documentation for this struct was generated from the following file:

- ICU_Module/Icu.h

3.24 IcuSignalMeasurement Struct Reference

Public Attributes

- Icu_SignalMeasurementPropertyType **IcuSignalMeasurementProperty**

The documentation for this struct was generated from the following file:

- ICU_Module/Icu.h

3.25 Message_Object_Status Struct Reference

Public Attributes

- boolean **Object_Free**
- uint8 **CanObjectId**
- uint8 **mailbox**
- PduIdType **swPduHandle**

The documentation for this struct was generated from the following file:

- Can_Module/Can_GeneralTypes.h

3.26 MessageObject Struct Reference

Public Attributes

- uint32 **ID**
- uint8 **DLC**
- uint8 **SDU** [8]
- uint8 **RTR**
- uint8 **IDE**

The documentation for this struct was generated from the following file:

- Can_Module/Can_GeneralTypes.h

3.27 Mutex_Ref Struct Reference

Collaboration diagram for Mutex_Ref:

Public Attributes

- uint16_t * **Ppayload**
- unsigned int **PayloadSize**
- TaskRefType * **CurrentTUser**
- TaskRefType * **NextTUser**
- char **MutexName** [30]
- uint8_t **priority_Inversion**

The documentation for this struct was generated from the following file:

- OSEK_VDX/Inc/Scheduler.h

3.28 PduInfoType Struct Reference

Public Attributes

- uint8 * **SduDataPtr**
- PduLengthType **SduLength**

The documentation for this struct was generated from the following file:

- Can_Module/ComStack_Types.h

3.29 Port_ConfigType Struct Reference

Collaboration diagram for Port_ConfigType:

Public Attributes

- PortPin **PortPin** [PortNumberOfPortPins]

The documentation for this struct was generated from the following file:

- PORT_Module/Port.h

3.30 PortConfigSet Struct Reference

Collaboration diagram for PortConfigSet:

Public Attributes

- PortContainer **PortContainer**

The documentation for this struct was generated from the following file:

- PORT_Module/Port.h

3.31 PortContainer Struct Reference

Collaboration diagram for PortContainer:

Public Attributes

- PortPin **PortPin** [PortNumberOfPortPins]

The documentation for this struct was generated from the following file:

- PORT_Module/Port.h

3.32 PortPin Struct Reference

Public Attributes

- Port_PinDirectionType **PortPinDirection**
- boolean **PortPinDirectionChangeable**
- Port_PinType **PortPinId**
- Port_PinModeType **PortPinInitialMode**
- uint8 **PortPinLevelValue**
- Port_PinModeType **PortPinMode**
- boolean **PortPinModeChangeable**
- boolean **Pull_UP**
- uint8 **Slew_Rate**
- uint8 **Pin_Driven_Mode**

The documentation for this struct was generated from the following file:

- PORT_Module/Port.h

3.33 Priority_Register_TypeDef Struct Reference

Public Attributes

- volatile uint32 **IPR0**
- volatile uint32 **IPR1**
- volatile uint32 **IPR2**
- volatile uint32 **IPR3**
- volatile uint32 **IPR4**
- volatile uint32 **IPR5**
- volatile uint32 **IPR6**
- volatile uint32 **IPR7**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.34 Pwm_ConfigType Struct Reference

Collaboration diagram for Pwm_ConfigType:

Public Attributes

- PwmChannelConfigSet **Config_Pwm**

The documentation for this struct was generated from the following file:

- PWM_Module/Pwm.h

3.35 PwmChannel Struct Reference

Public Attributes

- Pwm_ChannelClassType **PwmChannelClass**
- Pwm_ChannelType **PwmChannelId**
- uint16 **PwmDutycycleDefault**
- Pwm_OutputStateType **PwmIdleState**
- void(∗ **PwmNotification**)(void)
- Pwm_PeriodType **PwmPeriodDefault**
- Pwm_OutputStateType **PwmPolarity**

The documentation for this struct was generated from the following file:

- PWM_Module/Pwm.h

3.36 PwmChannelConfigSet Struct Reference

Collaboration diagram for PwmChannelConfigSet:

Public Attributes

- PwmChannel **Channel_Config** [Max_Num_CH]

The documentation for this struct was generated from the following file:

- PWM_Module/Pwm.h

3.37 RCC_TypeDef Struct Reference

Public Attributes

- volatile uint32 **CR**
- volatile uint32 **CFGR**
- volatile uint32 **CIR**
- volatile uint32 **APB2RSTR**
- volatile uint32 **APB1RSTR**
- volatile uint32 **AHBENR**
- volatile uint32 **APB2ENR**
- volatile uint32 **APB1ENR**
- volatile uint32 **BDCR**
- volatile uint32 **CSR**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.38 Semaphore_Ref Struct Reference

Collaboration diagram for Semaphore_Ref:

Public Attributes

- unsigned char * **Ppayload**
- TaskRefType * **CurrentTUser**
- TaskRefType * **NextTUser**
- char **SemaphoreName** [30]
- uint8_t **state**

The documentation for this struct was generated from the following file:

- OSEK_VDX/Inc/Scheduler.h

3.39 Set_Enable_Register_TypeDef Struct Reference

Public Attributes

- volatile uint32 **ISER0**
- volatile uint32 **ISER1**
- volatile uint32 **ISER2**
- volatile uint32 **ISER3**
- volatile uint32 **ISER4**
- volatile uint32 **ISER5**
- volatile uint32 **ISER6**
- volatile uint32 **ISER7**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.40 Set_Pending_Register_TypeDef Struct Reference

Public Attributes

- volatile uint32 **ISPR0**
- volatile uint32 **ISPR1**
- volatile uint32 **ISPR2**
- volatile uint32 **ISPR3**
- volatile uint32 **ISPR4**
- volatile uint32 **ISPR5**
- volatile uint32 **ISPR6**
- volatile uint32 **ISPR7**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.41 Std_TransformerError Struct Reference

Public Attributes

- Std_TransformerErrorCode **errorCode**
- Std_TransformerClass **transformerClass**

The documentation for this struct was generated from the following file:

- Common/Std_Types.h

3.42 Std_VersionInfoType Struct Reference

Public Attributes

- uint16 **vendorID**
- uint16 **moduleID**
- uint8 **sw_major_version**
- uint8 **sw_minor_version**
- uint8 **sw_patch_version**

The documentation for this struct was generated from the following file:

- Common/Std_Types.h

3.43 System_Conctrol Struct Reference

Collaboration diagram for System_Conctrol:

Public Attributes

- TaskRefType * **OSTasks** [100]
- unsigned int **_S_MSP_Task**
- unsigned int **_E_MSP_Task**
- unsigned int **PSP_Task_Locator**
- unsigned int **NoOfActiveTasks**
- TaskRefType * **CurrentTask**
- TaskRefType * **NextTask**
- OSmode **OSmodeID**
- OS_level **Call_Leve**

The documentation for this struct was generated from the following file:

- OSEK_VDX/Inc/Type.h

3.44 TaskRefType Struct Reference

Public Attributes

- uint32_t **Stack_Size**
- uint8_t **priority**
- void(* **P_TaskEntry**)(void)
- uint32_t **_S_PSP_Task**
- uint32_t **_E_PSP_Task**
- uint32_t * **Current_PSP**
- int8_t **Task_Name** [30]
- TaskStateType **TaskState**
- Task_Type **TaskType**

- Auto_Start **AutoStart**
- Tasks_Scheduler_Type **TaskSchedlerType**
- struct {
  enum { **disable** , **enable** }
  enum TaskRefType:: { ... } **Blocking**
  uint32_t **Ticks_Count**
  } **Timing_Waiting**

- uint8_t **MultipleActivation**
- struct {
  EventMaskType **Public_Mask**
  EventMaskType **Private_Mask**
  } **Events**

The documentation for this struct was generated from the following file:

- OSEK_VDX/Inc/Type.h

3.45 Time_Measurement Struct Reference

Public Attributes

- uint16 **Time_Low**
- uint16 **Time_High**

The documentation for this struct was generated from the following file:

- ICU_Module/Icu.c

3.46 TIMx_TypeDef Struct Reference

Public Attributes

- volatile uint32 **CR1**
- volatile uint32 **CR2**
- volatile uint32 **SMCR**
- volatile uint32 **DIER**
- volatile uint32 **SR**
- volatile uint32 **EGR**
- volatile uint32 **CCMR1**
- volatile uint32 **CCMR2**
- volatile uint32 **CCER**
- volatile uint32 **CNT**
- volatile uint32 **PSC**
- volatile uint32 **ARR**
- uint32 **RESERVED0**
- volatile uint32 **CCR1**
- volatile uint32 **CCR2**
- volatile uint32 **CCR3**
- volatile uint32 **CCR4**
- uint32 **RESERVED1**
- volatile uint32 **DCR**
- volatile uint32 **DMAR**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

3.47 USART_config_t Struct Reference

Public Attributes

- uint8 **MODE**
- uint8 **NUM_DATA_BIT**
- uint8 **NUM_STOP_BIT**
- uint32 **BAUDRATE**
- uint8 **PARITY**
- uint8 **HWFLOWCTL**
- uint8 **IRQ_EN**
- void(∗ **P_IRQ_CALL**)(void)

The documentation for this struct was generated from the following file:

- Complex_Drivers/Inc/stm32_f103c6_USART.h

3.48 USART_TypeDef Struct Reference

Public Attributes

- volatile uint32 **SR**
- volatile uint32 **DR**
- volatile uint32 **BRR**
- volatile uint32 **CR1**
- volatile uint32 **CR2**
- volatile uint32 **CR3**
- volatile uint32 **GTPR**

The documentation for this struct was generated from the following file:

- Common/stm32f103x6.h

Chapter 4

File Documentation

4.1 Can.h

```
00001 /******************************************************************************
00002 *
00003 * @Module      :     CAN
00004 * @File Name   :     Can.h
00005 * @Description :    This specification specifies the functionality, API and the configuration of the
      AUTOSAR
00006                     Basic Software module CAN Driver.
00007 * Author       :    Salama Mohamed
00008 ******************************************************************************/
00009 /******************************************************************************
00010 * Project               : Graduation_Project_2024
00011 * Platform              : STM32F103C8
00012 * Autosar Version       : 4.8.0
00013 * SW Version            : 1.0.0
00014 ******************************************************************************/
00015 #ifndef CAN_H_
00016 #define CAN_H_
00017 /******************************************************************************
00018                             Source File Version Informations
00019 ******************************************************************************/
00020 #define CAN_VERSION_ID                          11
00021 #define CAN_AR_RELEASE_MAJOR_VERSION            4
00022 #define CAN_AR_RELEASE_MINOR_VERSION            8
00023 #define CAN_AR_RELEASE_PATCH_VERSION            0
00024 #define CAN_SW_RELEASE_MAJOR_VERSION            1
00025 #define CAN_SW_RELEASE_MINOR_VERSION            0
00026 #define CAN_SW_RELEASE_PATCH_VERSION            0
00027 #define VENDOR_ID                               100
00028 /******************************************************************************
00029                             Includes
00030 ******************************************************************************/
00031 #include "Det.h"
00032 #include "Can_Cfg.h"
00033 #include "Std_Types.h"
00034 // AUTOSAR checking Std_Version
00035 #if ((STD_TYPES_AR_RELEASE_MAJOR_VERSION != CAN_AR_RELEASE_MAJOR_VERSION)\
00036 ||  (STD_TYPES_AR_RELEASE_MINOR_VERSION  != CAN_AR_RELEASE_MINOR_VERSION)\
00037 ||  (STD_TYPES_AR_RELEASE_PATCH_VERSION  != CAN_AR_RELEASE_PATCH_VERSION))
00038 #error "The Autosar version of Std_Types.h does not match the CAN version"
00039 #endif
00040 #include "ComStack_Types.h"
00041 // AUTOSAR checking ComStack_Types.h
00042 #if ((COMSTACK_TYPES_AR_RELEASE_MAJOR_VERSION != CAN_AR_RELEASE_MAJOR_VERSION)\
00043 ||  (COMSTACK_TYPES_AR_RELEASE_MINOR_VERSION  != CAN_AR_RELEASE_MINOR_VERSION)\
00044 ||  (COMSTACK_TYPES_AR_RELEASE_PATCH_VERSION  != CAN_AR_RELEASE_PATCH_VERSION))
00045 #error "The Autosar version of Std_Types.h does not match the CAN version"
00046 #endif
00047 #include "Can_GeneralTypes.h"
00048 // AUTOSAR checking Can_GeneralTypes.h
00049 #if ((CAN_GENERALTYPES_AR_RELEASE_MAJOR_VERSION != CAN_AR_RELEASE_MAJOR_VERSION)\
00050 ||  (CAN_GENERALTYPES_AR_RELEASE_MINOR_VERSION  != CAN_AR_RELEASE_MINOR_VERSION)\
00051 ||  (CAN_GENERALTYPES_AR_RELEASE_PATCH_VERSION  != CAN_AR_RELEASE_PATCH_VERSION))
00052 #error "The Autosar version of Std_Types.h does not match the CAN version"
00053 #endif
00054 /******************************************************************************
00055                              Development Errors
00056  ******************************************************************************/
00057 //API Service called with wrong parameter
```

```
00058 #define CAN_E_PARAM_POINTER                     (uint8)0x01
00059 //API Service called with wrong parameter
00060 #define CAN_E_PARAM_HANDLE                      (uint8)0x02
00061 //API Service called with wrong parameter
00062 #define CAN_E_PARAM_DATA_LENGTH                 (uint8)0x03
00063 //API Service called with wrong parameter
00064 #define CAN_E_PARAM_CONTROLLER                  (uint8)0x04
00065 //API Service used without initialization
00066 #define CAN_E_UNINIT                            (uint8)0x05
00067 //Invalid transition for the current mode
00068 #define CAN_E_TRANSITION                        (uint8)0x06
00069 //Parameter Baudrate has an invalid value
00070 #define CAN_E_PARAM_BAUDRATE                    (uint8)0x07
00071 //Invalid configuration set selection
00072 #define CAN_E_INIT_FAILED                       (uint8)0x08
00073 //API service called with invalid PDU ID
00074 #define CAN_E_PARAM_LPDU                        (uint8)0x09
00075 /*******************************************************************************
00076                             APIS ID
00077  *******************************************************************************/
00078 #define Can_Init_ID                             (uint8)0x00
00079 #define Can_GetVersionInfo_ID                   (uint8)0x07
00080 #define Can_DeInit_ID                           (uint8)0x10
00081 #define Can_SetBaudrate_ID                      (uint8)0x0f
00082 #define Can_SetControllerMode_ID                (uint8)0x03
00083 #define Can_DisableControllerInterrupts_ID      (uint8)0x04
00084 #define Can_EnableControllerInterrupts_ID       (uint8)0x05
00085 #define Can_CheckWakeup_ID                      (uint8)0x0b
00086 #define Can_GetControllerErrorState_ID          (uint8)0x0b
00087 #define Can_GetControllerMode_ID                (uint8)0x12
00088 #define Can_Write_ID                            (uint8)0x06
00089 /*******************************************************************************
00090                             Type definitions
00091  *******************************************************************************/
00092 //one CAN controller in HW
00093 typedef enum
00094 {
00095     //All the CANIDs are of type extended only (29 bit).
00096     EXTENDED,
00097     //The type of CANIDs can be both Standard or Extended.
00098     MIXED,
00099     //All the CANIDs are of type standard only (11bit).
00100     STANDARD
00101 }CanIdType;
00102 typedef enum  {
00103     BASIC,
00104     FULL
00105 }CanHandleType;
00106 typedef enum
00107 {
00108     //Receive HOH
00109     RECEIVE,
00110     //Transmit HOH
00111     TRANSMIT
00112 }CanObjectType;
00113 /*
00114  The Can module has a very simple state machine, with the two states CAN_UNINIT
00115  and CAN_READY. Figure 7.1 shows the state machine.
00116  */
00117 typedef enum {
00118     CAN_UNINIT,
00119     CAN_READY
00120 }CanDriverStateType;
00121 /*******************************************************************************************
00122 Name            :               CanControllerBaudrateConfig
00123 Parent Container:               CanController
00124 Description     :               This container contains bit timing related configuration parameters of
00125                                 the CAN controller(s).
00126 Type            :               Container
00127 *******************************************************************************************/
00128 typedef struct
00129 {
00130     /*
00131      Specifies the baudrate of the controller in kbps.
00132      */
00133     float32 CanControllerBaudRate;
00134     /*
00135      This ID is used by SetBaudrate API and uniquely identifies a specific baud
00136      rate configuration within a controller configuration.
00137      */
00138     uint16 CanControllerBaudRateConfigID;
00139     /*
00140      Specifies propagation delay in time quantas
00141      */
00142     uint16 CanControllerPropSeg;
00143     /*
00144      Specifies phase segment 1 in time quantas.
```

```
00145      */
00146     uint8 CanControllerSeg1;
00147     /*
00148      Specifies phase segment 2 in time quantas.
00149      */
00150     uint8 CanControllerSeg2;
00151     /*
00152      Specifies the synchronization jump width for the controller in time quantas.
00153      */
00154     uint8 CanControllerSyncJumpWidth;
00155 }CanControllerBaudrateConfig;
00156 /*******************************************************************************
00157 Name            :               CanHwFilter
00158 Parent Container:               CanHardwareObject
00159 Description     :               This container is only valid for HRHs and contains the configuration
      (parameters)
00160                                 of one hardware filter.
00161 Type            :               Container
00162 *******************************************************************************/
00163 typedef struct
00164 {
00165     /*
00166      Specifies (together with the filter mask) the identifiers range that passes
00167      the hardware filter.
00168      */
00169     uint32 CanHwFilterCode;
00170     /*
00171      Describes a mask for hardware-based filtering of CAN identifiers.
00172      The mask shall be build by filling with leading 0. In case of CanIdType
00173      EXTENDED or MIXED a 29 bit mask shall be build. In case of CanIdType
00174      STANDARD a 11 bit mask shall be build
00175      */
00176     uint32 CanHwFilterMask;
00177 }CanHwFilter;
00178 /*******************************************************************************
00179 Name            :               CanController
00180 Parent Container:               CanConfigSet
00181 Description     :               This container contains the configuration parameters of the CAN
00182                                 controller(s).
00183 Type:                           Container
00184 *******************************************************************************/
00185 typedef struct
00186 {
00187     /*
00188      Enables / disables API Can_MainFunction_BusOff() for handling busoff
00189      events in polling mode.
00190      */
00191     uint8 CanBusoffProcessing;
00192     /*
00193      Defines if a CAN controller is used in the configuration.
00194      */
00195     boolean CanControllerActivation;
00196     /*
00197      Specifies the CAN controller base address.
00198      */
00199     uint32 CanControllerBaseAddress;
00200     /*
00201      This parameter provides the controller ID which is unique in a given CAN
00202      Driver. The value for this parameter starts with 0 and continue without any gaps.
00203      */
00204     uint8 CanControllerId;
00205     /*
00206      Enables / disables API Can_MainFunction_Read() for handling PDU
00207      reception events in polling mode.
00208      */
00209     uint8 CanRxProcessing;
00210     /*
00211      Enables / disables API Can_MainFunction_Write() for handling PDU
00212      transmission events in polling mode.
00213      */
00214     uint8 CanTxProcessing;
00215     /*
00216      Enables / disables API Can_MainFunction_Wakeup() for handling wakeup
00217      events in polling mode.
00218      */
00219     uint8 CanWakeupProcessing;
00220     /*
00221      CAN driver support for wakeup over CAN Bus
00222      */
00223     boolean CanWakeupSupport;
00224     /*
00225      This container contains bit timing related configuration parameters of
00226      the CAN controller(s).
00227      */
00228     CanControllerBaudrateConfig CanControllerBaudrateConfig[Max_Num_Baud_Rate];
00229     /*
00230      Reference to baudrate configuration container configured for the Can
```

```
00231      Controller
00232      */
00233     CanControllerBaudrateConfig* CanControllerDefaultBaudrate;
00234 }CanController;
00235 /*******************************************************************************
00236 Name            :               CanHardwareObject
00237 Parent Container:               CanConfigSet
00238 Description     :               This container contains the configuration (parameters) of CAN
00239                                 Hardware Objects.
00240 Type:                           Container
00241 *******************************************************************************/
00242 typedef struct
00243 {
00244     /*
00245      Specifies the type (Full-CAN or Basic-CAN) of a hardware object.
00246      */
00247     CanHandleType CanHandleType;
00248     /*
00249      Number of hardware objects used to implement one HOH. In case of a HRH this
00250      parameter defines the number of elements in the hardware FIFO or the number
00251      of shadow buffers, in case of a HTH it defines the number of hardware objects
00252      used for multiplexed transmission or for a hardware FIFO used by a FullCAN HTH.
00253      */
00254     uint16 CanHwObjectCount;
00255     /*
00256      Specifies whether the IdValue is of type standard identifier, extended
00257      identifier or mixed mode.
00258      */
00259     CanIdType CanIdType;
00260     /*
00261      Holds the handle ID of HRH or HTH. The value of this parameter is unique in a
00262      given CAN Driver, and it should start with 0 and continue without any gaps.
00263      The HRH and HTH Ids share a common ID range.Example: HRH0-0, HRH1-1, HTH0-2, HTH1-3
00264      */
00265     uint8 CanObjectId;
00266     /*
00267      Specifies if the HardwareObject is used as Transmit or as Receive object
00268      */
00269     CanObjectType CanObjectType;
00270     /*
00271      This parameter defines if or if not Can supports the trigger-transmit API for this handle.
00272      */
00273     boolean CanTriggerTransmitEnable;
00274     /*
00275      Reference to CAN Controller to which the HOH is associated to.
00276      */
00277     CanController* CanControllerRef;
00278     /*
00279      This container is only valid for HRHs and contains the configuration
00280      (parameters) of one hardware filter.
00281      */
00282     CanHwFilter  CanHwFilter ;
00283 }CanHardwareObject;
00284 /*******************************************************************************
00285 Name            :               CanConfigSet
00286 Parent Container:               Can
00287 Description     :               This container contains the configuration parameters and sub
      containers of the
00288                                 AUTOSAR Can module.
00289 Type:                           Container
00290 *******************************************************************************/
00291 typedef struct
00292 {
00293     /*
00294      This container contains the configuration parameters of the CAN
00295      controller(s).
00296      */
00297     CanController CanController;
00298     /*
00299      This container contains the configuration (parameters) of CAN
00300      Hardware Objects.
00301      */
00302     CanHardwareObject CanHardwareObject[Max_Num_HOH];
00303 }CanConfigSet;
00304 /*******************************************************************************
00305 Name            :               CanGeneral
00306 Parent Container:               Can
00307 Description     :               This container contains the parameters related each CAN Driver Unit.
00308 Type:                           Container
00309 *******************************************************************************/
00310 typedef struct
00311 {
00312     boolean Active;
00313 }CanGeneral;
00314 /*******************************************************************************
00315 Name            :               Can
00316 Parent Container:               CAN
```

```
00317 Description     :               This container holds the configuration of a single CAN Driver
00318 Type:                           Structure
00319 **********************************************************************************/
00320 typedef struct
00321 {
00322     //CanGeneral CanGeneral;
00323     CanConfigSet CanConfigSet;
00324 }Can_ConfigType;
00325 extern Can_ConfigType Can;
00326 /**********************************************************************************
00327                                  APIS
00328 **********************************************************************************/
00329 /*********************************************************************************
00330  * Service name          : Can_Init
00331  * Service ID[hex]       : 0x00
00332  * Sync/Async            : Synchronous
00333  * Reentrancy            : Non Reentrant
00334  * Parameters (in)       : Config
00335                           Pointer to driver configuration.
00336  * Parameters (inout)    : None
00337  * Parameters (out)      : None
00338  * Return value          : None
00339  * Description           : This function initializes the module.
00340  *********************************************************************************/
00341 void Can_Init (const Can_ConfigType* Config);
00342 /*********************************************************************************
00343  * Service ID[hex]       : 0x07
00344  * Service name          : Can_GetVersionInfo
00345  * Sync/Async            : Synchronous
00346  * Reentrancy            : Reentrant
00347  * Parameters (in)       : None
00348  * Parameters (inout)    : None
00349  * Parameters (out)      : versioninfo
00350                           Pointer to where to store the version information of this
00351                           module.
00352  * Return value          : None
00353  * Description           : This function returns the version information of this
00354                           module
00355 **********************************************************************************/
00356 void Can_GetVersionInfo (Std_VersionInfoType* versioninfo);
00357 /*********************************************************************************
00358  * Service ID [hex]      : 0x10
00359  *  Service name         : Can_DeInit
00360  *  Sync/Async           : Synchronous
00361  *  Reentrancy           : Non-Reentrant
00362  *  Parameters (in)      : None
00363  *  Parameters (inout)   : None
00364  *  Parameters (out)     : None
00365  *  Return value         : None
00366  *  Description          : This function de-initializes the module.
00367 **********************************************************************************/
00368 void Can_DeInit (void);
00369 /*********************************************************************************
00370  *  Service ID[hex]      : 0x0f
00371  *  Function name        : Can_SetBaudrate
00372  *  Sync/Async           : Synchronous
00373  *  Reentrancy           : Reentrant for different Controllers.
00374  *                         Non reentrant for the same Controller.
00375  *  Parameters (in)      : None
00376  *  Parameters (inout)   : None
00377  *  Parameters (out)     : None
00378  *  Return value         : None
00379  *  Description          : This service shall set the baud rate configuration
00380                           of the CAN controller. Depending on necessary baud rate
00381                           modifications the controller might have to reset.
00382 **********************************************************************************/
00383 Std_ReturnType Can_SetBaudrate (uint8 Controller,uint16 BaudRateConfigID);
00384 /*********************************************************************************
00385  * Service ID[hex]       : 0x12
00386  * Service name          : Can_GetControllerMode
00387  * Sync/Async            : Synchronous
00388  * Reentrancy            : Non Reentrant
00389  * Parameters (in)       : Controller
00390  *                         CAN controller for which the status shall be changed
00391  * Parameters (out)      : ControllerModePtr
00392                           Pointer to a memory location,
00393                           where the current mode of the CAN controller will be stored.
00394  * Parameters (inout)    :  None
00395  * Return value          : Std_ReturnType
00396                           E_OK (request accepted)
00397                           E_NOT_OK (request not accepted)
00398  * Description           : This service reports about the current status of the requested CAN
      controller.
00399 **********************************************************************************/
00400 Std_ReturnType Can_SetControllerMode (uint8 Controller,Can_ControllerStateType Transition);
00401 /*********************************************************************************
00402  * Service ID[hex]       : 0x04
```

```
00403  * Service name         : Can_DisableControllerInterrupts
00404  * Sync/Async           : Synchronous
00405  * Reentrancy           : Reentrant
00406  * Parameters (in)      : Controller
00407                           CAN controller for which interrupts shall be disabled.
00408  * Parameters (inout)   : None
00409  * Parameters (out)     : None
00410  * Return value         : None
00411  * Description          : This function disables all interrupts for this CAN controller
00412 *******************************************************************************/
00413 void Can_DisableControllerInterrupts (uint8 Controller);
00414 /*******************************************************************************
00415  * Service ID[hex]      : 0x05
00416  * Service name         : Can_EnableControllerInterrupts
00417  * Sync/Async           : Synchronous
00418  * Reentrancy           : Reentrant
00419  * Parameters (in)      : Controller
00420                           CAN controller for which interrupts shall be re-enabled.
00421  * Parameters (inout)   : None
00422  * Parameters (out)     : None
00423  * Return value         : None
00424  * Description          : This function enables all allowed interrupts
00425 *******************************************************************************/
00426 void Can_EnableControllerInterrupts (uint8 Controller);
00427 /*******************************************************************************
00428  * Service ID[hex]      : 0x0b
00429  * Service name         : Can_CheckWakeup
00430  * Sync/Async           : Synchronous
00431  * Reentrancy           : Non Reentrant
00432  * Parameters (in)      : Controller
00433                           Controller to be checked for a wakeup.
00434  * Parameters (inout)   : None
00435  * Parameters (out)     : None
00436  * Return value         : Std_ReturnType
00437                           E_OK (API call has been accepted)
00438                           E_NOT_OK (API call has not been accepted)
00439  * Description          : This function checks if a wakeup has occurred for the given controller.
00440 *******************************************************************************/
00441 Std_ReturnType Can_CheckWakeup (uint8 Controller);
00442 /*******************************************************************************
00443  * Service ID[hex]      : 0x11
00444  * Service name         : Can_GetControllerErrorState
00445  * Sync/Async           : Synchronous
00446  * Reentrancy           : Non Reentrant
00447  * Parameters (in)      : Controller
00448  *                        CAN controller for which the status shall be changed
00449  * Parameters (out)     : ErrorStatePtr
00450  *                        Pointer to a memory location,
00451  *                        where the error state of the CAN controller will be stored.
00452  * Parameters (inout)   :  None
00453  * Return value         : Std_ReturnType
00454                           E_OK (request accepted)
00455                           E_NOT_OK (request not accepted)
00456  * Description:  This service obtains the error state of the CAN controller
00457 *******************************************************************************/
00458 Std_ReturnType Can_GetControllerErrorState (uint8 ControllerId,Can_ErrorStateType* ErrorStatePtr);
00459 /*******************************************************************************
00460  * Service ID[hex]      : 0x12
00461  * Service name         : Can_GetControllerMode
00462  * Sync/Async           : Synchronous
00463  * Reentrancy           : Non Reentrant
00464  * Parameters (in)      : Controller
00465                           CAN controller for which the status shall be changed
00466  * Parameters (out)     : ControllerModePtr
00467                           Pointer to a memory location,
00468                           where the current mode of the CAN controller will be stored.
00469  * Parameters (inout)   : None
00470  * Return value         : Std_ReturnType
00471                           E_OK (request accepted)
00472                           E_NOT_OK (request not accepted)
00473  * Description          : This service reports about the current status of the requested CAN
     controller.
00474 *******************************************************************************/
00475 Std_ReturnType Can_GetControllerMode (uint8 Controller,Can_ControllerStateType* ControllerModePtr);
00476 /*******************************************************************************
00477  * Service ID[hex]      : 0x30
00478  * Service name         : Can_GetControllerRxErrorCounter
00479  * Sync/Async           : Synchronous
00480  * Reentrancy           : Non Reentrant for the same ControllerId
00481  * Parameters (in)      : Controller
00482                           CAN controller, whose current Rx error counter shall be acquired.
00483  * Parameters (inout)   : None
00484  * Parameters (out)     : RxErrorCounterPtr
00485                           Pointer to a memory location, where the current Rx error
00486                           counter of the CAN controller will be stored.
00487  * Return value         : Std_ReturnType
00488                           E_OK (Rx error counter available)
```

```
00489                           E_NOT_OK (Wrong ControllerId, or Rx error counter not available)
00490  * Description          : Returns the Rx error counter for a CAN controller. This value might not be
    available
00491                           for all CAN controllers, in which case E_NOT_OK would be returned.
00492 ************************************************************************************/
00493 Std_ReturnType Can_GetControllerRxErrorCounter (uint8 ControllerId,uint8* RxErrorCounterPtr);
00494 /************************************************************************************
00495  * Service ID[hex]      : 0x31
00496  * Service name         : Can_GetControllerTxErrorCounter
00497  * Sync/Async           : Synchronous
00498  * Reentrancy           : Non Reentrant for the same ControllerId
00499  * Parameters (in)      : Controller
00500                           CAN controller, whose current Tx error counter shall be acquired.
00501  * Parameters (inout)   : None
00502  * Parameters (out)     : TxErrorCounterPtr
00503                           Pointer to a memory location, where the current Tx error
00504                           counter of the CAN controller will be stored.
00505  * Return value         : Std_ReturnType
00506                           E_OK (Tx error counter available)
00507                           E_NOT_OK (Wrong ControllerId, or Tx error counter not available)
00508  * Description          : Returns the Tx error counter for a CAN controller. This value might not be
    available
00509                           for all CAN controllers, in which case E_NOT_OK would be returned.
00510 ************************************************************************************/
00511 Std_ReturnType Can_GetControllerTxErrorCounter (uint8 ControllerId,uint8* TxErrorCounterPtr);
00512 /************************************************************************************
00513  * Service ID[hex]      : 0x06
00514  * Service name         : Can_Write
00515  * Sync/Async           : Synchronous
00516  * Reentrancy           : Reentrant (thread-safe)
00517  * Parameters (in)      : Hth
00518                           Information which HW-transmit handle shall be used for
00519                           transmit. Implicitly this is also the information about
00520                           the controller to use because the Hth numbers are unique inside one hardware
    unit.
00521  *                     : PduInfo
00522                           Pointer to SDU user memory, DLC and Identifier.
00523  * Parameters (inout)   : None
00524  * Parameters (out)     : None
00525  * Return value         : Std_ReturnType
00526                           E_OK (Write command has been accepted)
00527                           E_NOT_OK (development error occurred)
00528                           CAN_BUSY (No TX hardware buffer available or pre-emptive call
00529                           of Can_Write that can't be implemented re-entrant)
00530  * Description          : This function is called by CanIf to pass a CAN message to CanDrv for
    transmission.
00531 ************************************************************************************/
00532 Std_ReturnType Can_Write (Can_HwHandleType Hth,const Can_PduType* PduInfo);
00533 /************************************************************************************
00534  * Service ID[hex]: 0x08
00535  * Service name   : Can_MainFunction_Read
00536  * Mode           : Supervisor Mode (Privileged Mode)
00537  * Description    : This function performs the polling of RX indications when
00538                     CAN_RX_PROCESSING is set to POLLING.
00539  ************************************************************************************/
00540 void Can_MainFunction_Read (void);
00541 /************************************************************************
00542  ************************************CallBack****************************
00543  ************************************************************************/
00544 void MCAL_CAN_Mailbox_0_Empty_Callback(void);
00545 void MCAL_CAN_Mailbox_1_Empty_Callback(void);
00546 void MCAL_CAN_Mailbox_2_Empty_Callback(void);
00547 void CanIf_TxConfirmation(PduIdType CanTxPduId);
00548 void CanIf_RxIndication (const Can_HwType* Mailbox,const PduInfoType* PduInfoPtr);
00549 //void CanIf_RxIndication(Can_HwHandleType Hrh, Can_IdType CanId, uint8 CanDlc, const uint8*
    CanSduPtr);
00550 #define CanIf_ControllerModeType    Can_ControllerStateType
00551 void CanIf_ControllerModeIndication(uint8 Controller, CanIf_ControllerModeType ControllerMode);
00552 #endif /* CAN_H_ */
```

4.2 Can_Cfg.h

```
00001 /************************************************************************************
00002 * @Module      :     CAN
00003 * @File Name   :     Can_Cfg.h
00004 * @Description :     the Pre-compile configuration of the AUTOSAR Basic Software module CAN Driver.
00005 * Author       :     Salama Mohamed
00006 ************************************************************************************/
00007 /************************************************************************************
00008 * Project               : Graduation_Project_2024
00009 * Platform              : STM32F103C8
00010 * Autosar Version       : 4.8.0
00011 * SW Version            : 1.0.0
```

```
00012 ******************************************************************************/
00013 #ifndef CAN_CFG_H_
00014 #define CAN_CFG_H_
00015 /******************************************************************************
00016                              Source File Version Informations
00017 ******************************************************************************/
00018 #define CAN_VERSION_ID                                  40
00019 #define CAN_CFG_AR_RELEASE_MAJOR_VERSION                4
00020 #define CAN_CFG_AR_RELEASE_MINOR_VERSION                8
00021 #define CAN_CFG_AR_RELEASE_PATCH_VERSION                0
00022 #define CAN_CFG_SW_RELEASE_MAJOR_VERSION                1
00023 #define CAN_CFG_SW_RELEASE_MINOR_VERSION                0
00024 #define CAN_CFG_SW_RELEASE_PATCH_VERSION                0
00025 #define VENDOR_ID                                       100
00026 /******************************************************************************
00027                              Includes
00028 ******************************************************************************/
00029 #include "Std_Types.h"
00030 /******************************************************************************
00031             Pre-compile configuration parameters of the PORT driver.
00032 ******************************************************************************/
00033 #define CanDevErrorDetect                               STD_ON
00034 #define CONTROLLER_ZERO                                 0U
00035 #define Max_Num_HOH                                     ((uint8)4)
00036 #define Max_Num_Baud_Rate                               ((uint8)2)
00037 #define BaudRateConfigID_0                              ((uint16)0)
00038 #define BaudRateConfigID_1                              ((uint16)1)
00039 #endif /* CAN_CFG_H_ */
```

4.3 Can_GeneralTypes.h

```
00001 /******************************************************************************
00002 * @Module               : CAN
00003 * @File Name            : Can_GeneralTypes.h
00004 * @Description          : This specification specifies the functionality, API and the configuration of
      the AUTOSAR
00005                           Basic Software module CAN Driver.
00006 * Author               : Salama Mohamed
00007 ******************************************************************************/
00008
00009 /******************************************************************************
00010 * Project              : Graduation_Project_2024
00011 * Platform             : STM32F103C8
00012 * Autosar Version      : 4.8.0
00013 * SW Version           : 1.0.0
00014 ******************************************************************************/
00015 #ifndef CAN_GENERALTYPES_H_
00016 #define CAN_GENERALTYPES_H_
00017 /******************************************************************************
00018                              Source File Version Informations
00019 ******************************************************************************/
00020 #define CAN_VERSION_ID                                          11
00021 #define CAN_GENERALTYPES_AR_RELEASE_MAJOR_VERSION               4
00022 #define CAN_GENERALTYPES_AR_RELEASE_MINOR_VERSION               8
00023 #define CAN_GENERALTYPES_AR_RELEASE_PATCH_VERSION               0
00024 #define CAN_GENERALTYPES_SW_RELEASE_MAJOR_VERSION               1
00025 #define CAN_GENERALTYPES_SW_RELEASE_MINOR_VERSION               0
00026 #define CAN_GENERALTYPES_SW_RELEASE_PATCH_VERSION               0
00027 /******************************************************************************
00028                              Includes
00029 ******************************************************************************/
00030 #include "ComStack_Types.h"
00031 // AUTOSAR checking ComStack_Types.h
00032 #if ((CAN_GENERALTYPES_AR_RELEASE_MAJOR_VERSION != COMSTACK_TYPES_AR_RELEASE_MAJOR_VERSION)\
00033 ||  (CAN_GENERALTYPES_AR_RELEASE_MINOR_VERSION  != COMSTACK_TYPES_AR_RELEASE_MINOR_VERSION)\
00034 ||  (CAN_GENERALTYPES_AR_RELEASE_PATCH_VERSION  != COMSTACK_TYPES_AR_RELEASE_PATCH_VERSION))
00035 #error "The Autosar version of Std_Types.h does not match the Can_GeneralTypes version"
00036 #endif
00037 /******************************************************************************
00038                               Type definitions
00039  ******************************************************************************/
00040 #define POLLING     0U
00041 #define INTERRUPT   1U
00042 #define Transmi_mailbox_0       (uint8)0
00043 #define Transmi_mailbox_1       (uint8)1
00044 #define Transmi_mailbox_2       (uint8)2
00045 #define Transmi_mailbox_Full    (uint8)3
00046 #define CAN_RTR_Data_Frame                              ((uint8)0)
00047 #define CAN_RTR_Remote_Frame                            ((uint8)1)
00048 #define CAN_IDE_Standard                                ((uint8)0)
00049 #define CAN_IDE_Extended                                ((uint8)1)
00050 /*
00051  Represents the Identifier of an L-PDU. The two most significant bits specify the frame
```

```
00052  type:
00053  00 CAN message with Standard CAN ID
00054  01 CAN FD frame with Standard CAN ID
00055  10 CAN message with Extended CAN ID
00056  11 CAN FD frame with Extended CAN ID
00057  */
00058 typedef uint32 Can_IdType;
00059 /*
00060  This type unites PduId (swPduHandle), SduLength (length), SduData (sdu), and Can
00061  Id (id) for any CAN L-SDU.
00062  */
00063 typedef struct
00064 {
00065     Can_IdType id;
00066     uint8* sdu;
00067     PduIdType swPduHandle;
00068     uint8 length;
00069 }Can_PduType;
00070 /*
00071   Represents the hardware object handles of a CAN hardware unit. For CAN hardware
00072   units with more than 255 HW objects use extended range.
00073  */
00074 typedef uint8 Can_HwHandleType;
00075 /*
00076  This type defines a data structure which clearly provides an Hardware Object Handle
00077  including its corresponding CAN Controller and therefore CanDrv as well as the specific CanId.
00078  */
00079 typedef struct
00080 {
00081     //Standard/Extended CAN ID of CAN L-PDU
00082     Can_IdType CanId;
00083     //ID of the corresponding Hardware Object Range
00084     Can_HwHandleType Hoh;
00085     //ControllerId provided by CanIf clearly identify the corresponding controller
00086     uint8 ControllerId;
00087 }Can_HwType;
00088 /**********************************************************************************
00089                                  Extension to Std_ReturnType
00090 **********************************************************************************/
00091 // transmit request could not be processed because no transmit object was available
00092 #define CAN_BUSY        ((Std_ReturnType)0x02U)
00093 /**********************************************************************************
00094                                Can_ErrorStateType
00095                                "Error states of a CAN controller."
00096 **********************************************************************************/
00097 typedef enum
00098 {
00099     // The CAN controller takes fully part in communication.
00100     CAN_ERRORSTATE_ACTIVE,
00101     //The CAN controller takes part in communication, but does not send active error frames.
00102     CAN_ERRORSTATE_PASSIVE,
00103     //The CAN controller does not take part in communication.
00104     CAN_ERRORSTATE_BUSOFF
00105 }Can_ErrorStateType;
00106 /**********************************************************************************
00107                                  Can_ControllerStateType
00108 **********************************************************************************/
00109 typedef enum
00110 {
00111     //CAN controller state UNINIT.
00112     CAN_CS_UNINIT,
00113     //CAN controller state STARTED.
00114     CAN_CS_STARTED,
00115     // CAN controller state STOPPED.
00116     CAN_CS_STOPPED,
00117     // CAN controller state SLEEP.
00118     CAN_CS_SLEEP
00119 }Can_ControllerStateType;
00120 /**********************************************************************************
00121                                  Can_ErrorType
00122                                  "The enumeration represents a superset of CAN
00123                                  Error Types which typical CAN HW is
00124                                  able to report. That means not all CAN HW will be able
00125                                  to support the complete set."
00126 **********************************************************************************/
00127 typedef enum
00128 {
00129      //A 0 was transmitted and a 1 was read back
00130     CAN_ERROR_BIT_MONITORING1=1,
00131     // A 1 was transmitted and a 0 was read back
00132     CAN_ERROR_BIT_MONITORING0,
00133     //The HW reports a CAN bit error but cannot report distinguish between CAN_ERROR_BIT_MONITORING1
      and CAN_ERROR_BIT_MONITORING0
00134     CAN_ERROR_BIT,
00135     // Acknowledgement check failed
00136     CAN_ERROR_CHECK_ACK_FAILED,
00137     // Acknowledgement delimiter check failed
```

```
00138     CAN_ERROR_ACK_DELIMITER,
00139     // The sender lost in arbitration.
00140     CAN_ERROR_ARBITRATION_LOST,
00141     //CAN overload detected via an overload frame. Indicates that the receive buffers of a receiver
      are full.
00142     CAN_ERROR_OVERLOAD,
00143     // Violations of the fixed frame format
00144     CAN_ERROR_CHECK_FORM_FAILED,
00145     //tuffing bits not as expected
00146     CAN_ERROR_CHECK_STUFFING_FAILED,
00147     // CRC failed
00148     CAN_ERROR_CHECK_CRC_FAILED,
00149     //Bus lock (Bus is stuck to dominant level)
00150     CAN_ERROR_BUS_LOCK
00151 }Can_ErrorType;
00152 //buffer for data RX or TX
00153 typedef struct{
00154     uint32 ID;
00155     uint8 DLC;
00156     uint8 SDU [8];
00157     uint8 RTR;
00158     uint8 IDE;
00159 }MessageObject;
00160 //state and information for hardware object handle
00161 typedef struct{
00162    boolean Object_Free;
00163    uint8 CanObjectId;
00164    uint8 mailbox;
00165    PduIdType swPduHandle;
00166 }Message_Object_Status;
00167 Can_HwType RX_Message;
00168 #endif /* CAN_GENERALTYPES_H_ */
```

4.4 ComStack_Types.h

```
00001 /**********************************************************************************
00002 * @Module               : COM_Stack
00003 * @File Name            : ComStack_Types.h
00004 * @Description          : This document specifies the AUTOSAR communication stack type header file. It
00005                           contains all types that are used across several modules of the communication
      stack
00006                           of the basic software and all types of all basic software modules that are
      platform and
00007                           compiler independent.
00008 * Author               : Salama Mohamed
00009 **********************************************************************************/
00010 /**********************************************************************************
00011 * Project              : Graduation_Project_2024
00012 * Platform             : STM32F103C8
00013 * Autosar Version      : 4.8.0
00014 * SW Version           : 1.0.0
00015 **********************************************************************************/
00016 #ifndef COMSTACK_TYPES_H_
00017 #define COMSTACK_TYPES_H_
00018 /**********************************************************************************
00019                              Source File Version Informations
00020 **********************************************************************************/
00021 #define COMSTACK_TYPES_VERSION_ID                          50
00022 #define COMSTACK_TYPES_AR_RELEASE_MAJOR_VERSION            4
00023 #define COMSTACK_TYPES_AR_RELEASE_MINOR_VERSION            8
00024 #define COMSTACK_TYPES_AR_RELEASE_PATCH_VERSION            0
00025 #define COMSTACK_TYPES_SW_RELEASE_MAJOR_VERSION            1
00026 #define COMSTACK_TYPES_SW_RELEASE_MINOR_VERSION            0
00027 #define CCOMSTACK_TYPES_SW_RELEASE_PATCH_VERSION           0
00028 /**********************************************************************************
00029                              Includes
00030 **********************************************************************************/
00031 #include "Std_Types.h"
00032 // AUTOSAR checking Std_Version
00033 #if ((STD_TYPES_AR_RELEASE_MAJOR_VERSION != COMSTACK_TYPES_AR_RELEASE_MAJOR_VERSION)\
00034 ||  (STD_TYPES_AR_RELEASE_MINOR_VERSION  != COMSTACK_TYPES_AR_RELEASE_MINOR_VERSION)\
00035 ||  (STD_TYPES_AR_RELEASE_PATCH_VERSION  != COMSTACK_TYPES_AR_RELEASE_PATCH_VERSION))
00036 #error "The Autosar version of Std_Types.h does not match the COM version"
00037 #endif
00038 /**********************************************************************************
00039                               Type definitions
00040  **********************************************************************************/
00041 /*
00042  The size of this global type depends on the maximum number of PDUs
00043  used within one software module.
00044  */
00045 /*
00046  [SWS_Comtype_00006] Variables of this type serve as a unique identifier of a PDU
```

```
00047  within a software module or a set thereof, and also for interaction of two software
00048  modules where the PduId of the corresponding target module is being used for
00049  referencing.
00050  */
00051 typedef uint8   PduIdType;
00052 /*
00053  The size of this global type depends on the maximum length of PDUs
00054  to be sent by an ECU.
00055  */
00056 typedef uint8  PduLengthType;
00057 /*
00058  Variables of this type shall be used to store the basic information about a PDU of any
00059  type, namely a pointer variable pointing to its SDU (payload), a pointer to Meta Data of
00060  the PDU, and the corresponding length of the SDU in bytes.
00061  */
00062 typedef struct
00063 {
00064     /*
00065      Pointer to the SDU (i.e. payload data) of the PDU. The type of this pointer
00066      depends on the memory model being used at compile time.
00067      */
00068     uint8*                          SduDataPtr;
00069     //Length of the SDU in bytes.
00070     PduLengthType                   SduLength;
00071 }PduInfoType;
00072 #endif /* COMSTACK_TYPES_H_ */
```

4.5 stm32_f103c6_CAN.h

```
00001 /*****************************************************
00002  *  File Name    : stm32_f103c6_CAN.h
00003  *  Created on   : 6/9/2023
00004  *  Author       : Salama mohamed
00005  *****************************************************/
00006 #ifndef INC_STM32_F103C6_CAN_H_
00007 #define INC_STM32_F103C6_CAN_H_
00008 /*
00009  ****************************************************************
00010  *************************Include File **********************
00011  ****************************************************************
00012  */
00013 #include "stm32f103x6.h"
00014 /*****************************************************************/
00015 typedef struct
00016 {
00017     uint32 Filter_ID;                   // Specifies the filter identification number
00018                                           // this parameter must be set from range 0X00000000 to
      0XFFFFFFFF in 32 bit scale
00019                                           // in 16 bit scale must contain 2 id from range 0X0000 to
      0XFFFF (0X073F085F) ID1=0X073F ID2=0X085F
00020
00021     uint32 Filter_Mask_ID;              // Specifies the filter mask number or identification number,
00022                                           // this parameter must be set from range 0X00000000 to
      0XFFFFFFFF in 32 bit scale
00023                                           // in 16 bit scale must contain 2 Mask from range 0X0000 to
      0XFFFF (0X073F085F) mask1=0X073F mask2=0X085F
00024
00025     uint32 Filter_FIFO_Assignment;      // Specifies the FIFO which will be assigned to the filter.
00026                                           // this parameter must be set based on  @ ref
      CAN_Filter_FIFO_Assignment_Define
00027
00028     uint32 Filter_Bank;                 // Specifies the filter bank which will be initialized from 0
      to 13
00029                                           // this parameter must be set based on  @ ref
      CAN_Filter_Bank_Define
00030
00031     uint32 Filter_Mode;                 // Specifies the filter mode to be initialized.
00032                                           // this parameter must be set based on  @ ref
      CAN_Filter_Mode_Define
00033
00034     uint32 Filter_Scale;                // Specifies the filter scale.
00035                                           // this parameter must be set based on  @ ref
      CAN_Filter_Scale_Define
00036 } CAN_Filter_Config_t;
00037 //-----------------------------
00038 //Macros Configuration References
00039 //-----------------------------
00040 //@ ref CAN_Filter_FIFO_Assignment_Define
00041 #define CAN_Filter_FIFO_Assignment_FIFO0            ((uint8)0)
00042 #define CAN_Filter_FIFO_Assignment_FIFO1            ((uint8)1)
00043 //@ ref CAN_Filter_Bank_Define
00044 #define CAN_Filter_Bank_0                           ((uint8)0)
00045 #define CAN_Filter_Bank_1                           ((uint8)1)
```

```
00046 #define CAN_Filter_Bank_2                          ((uint8)2)
00047 #define CAN_Filter_Bank_3                          ((uint8)3)
00048 #define CAN_Filter_Bank_4                          ((uint8)4)
00049 #define CAN_Filter_Bank_5                          ((uint8)5)
00050 #define CAN_Filter_Bank_6                          ((uint8)6)
00051 #define CAN_Filter_Bank_7                          ((uint8)7)
00052 #define CAN_Filter_Bank_8                          ((uint8)8)
00053 #define CAN_Filter_Bank_9                          ((uint8)9)
00054 #define CAN_Filter_Bank_10                         ((uint8)10)
00055 #define CAN_Filter_Bank_11                         ((uint8)11)
00056 #define CAN_Filter_Bank_12                         ((uint8)12)
00057 #define CAN_Filter_Bank_13                         ((uint8)13)
00058 //@ ref CAN_Filter_Mode_Define
00059 #define CAN_Filter_Mode_Mask                       ((uint8)0)
00060 #define CAN_Filter_Mode_List                       ((uint8)1)
00061 //@ ref CAN_Filter_Scale_Define
00062 #define CAN_Filter_Scale_16                        ((uint8)0)
00063 #define CAN_Filter_Scale_32                        ((uint8)1)
00064 /*******************************************************************************
00065 * ******************************APIS*********************************
00066 * ******************************************************************************/
00067 void MCAL_CAN_Config_Filter(CAN_Filter_Config_t* Filter_Config);
00068 #endif /* INC_STM32_F103C6_CAN_H_ */
```

4.6 CanIf.h

```
00001 /*
00002  * CanIf.h
00003  *
00004  *  Created on: Oct 5, 2023
00005  *      Author: ELBOSTAN
00006  */
00007
00008 #ifndef CANIF_H_
00009 #define CANIF_H_
00010
00011
00012
00013 #endif /* CANIF_H_ */
```

4.7 CanIf_Cbk.h

```
00001 /*****************************************************************
00002  * File name  : CanIf_cbk.h
00003  * Created on : 5/10/2023
00004  * Author     : Salama Mohamed
00005  *****************************************************************/
00006 #ifndef CANIF_CBK_H_
00007 #define CANIF_CBK_H_
00008
00009 #include "Can.h"
00010 void CanIf_RxIndication (const Can_HwType* Mailbox,const PduInfoType* PduInfoPtr);
00011 #endif /* CANIF_CBK_H_ */
```

4.8 CanIf_Types.h

```
00001 /*
00002  * CanIf_Types.h
00003  *
00004  *  Created on: Oct 5, 2023
00005  *      Author: ELBOSTAN
00006  */
00007
00008 #ifndef CANIF_TYPES_H_
00009 #define CANIF_TYPES_H_
00010
00011
00012
00013 #endif /* CANIF_TYPES_H_ */
```

4.9 Det.h

```
00001 /******************************************************************************
00002 * @Module      :    Default Error Tracer
00003 * @File Name   :    Det.h
00004 * @Description :    This specification describes the API of the Default Error Tracer.
00005                     All detected development and runtime errors in the Basic Software
00006                     are reported to this module.
00007 * Author      :    Salama Mohamed
00008 *******************************************************************************/
00009 /******************************************************************************
00010 * Project               : Graduation_Project_2024
00011 * Platform              : STM32F103C8
00012 * Autosar Version       : 4.8.0
00013 * SW Version            : 1.0.0
00014 *******************************************************************************/
00015 #ifndef DET_H_
00016 #define DET_H_
00017 /******************************************************************************
00018                             Source File Version Informations
00019 *******************************************************************************/
00020 #define DET_VERSION_ID                          17
00021 #define DET_AR_RELEASE_MAJOR_VERSION            4
00022 #define DET_AR_RELEASE_MINOR_VERSION            8
00023 #define DET_AR_RELEASE_PATCH_VERSION            0
00024 #define DET_SW_RELEASE_MAJOR_VERSION            1
00025 #define DET_SW_RELEASE_MINOR_VERSION            0
00026 #define DET_SW_RELEASE_PATCH_VERSION            0
00027 /******************************************************************************
00028                             Includes
00029 *******************************************************************************/
00030 #include "Std_Types.h"
00031 // AUTOSAR checking Std_Version
00032 #if ((STD_TYPES_AR_RELEASE_MAJOR_VERSION != DET_AR_RELEASE_MAJOR_VERSION)\
00033 ||  (STD_TYPES_AR_RELEASE_MINOR_VERSION  != DET_AR_RELEASE_MINOR_VERSION)\
00034 ||  (STD_TYPES_AR_RELEASE_PATCH_VERSION  != DET_AR_RELEASE_PATCH_VERSION))
00035 #error "The Autosar version of Std_Types.h does not match the DET version"
00036 #endif
00037 /******************************************************************************
00038                               APIS
00039 *******************************************************************************/
00040 /*****************************************************************************
00041 * Service ID [hex]   :   0x01
00042 * Service Name       :   Det_ReportError
00043 * Sync/Async         :   Synchronous
00044 * Reentrancy         :   reentrant
00045 * Parameters (in)    :   ModuleId (Module ID of calling module)
00046                           InstanceId (The identifier of the index based instance of a module)
00047                           ApiId (ID of API service in which error is detected)
00048                           ErrorId (ID of detected development error)
00049 * Parameters (inout):   None
00050 * Parameters (out)   :   None
00051 * Return value       :   Std_ReturnType (never returns a value, but has a return type for compatibility
      with services and hooks)
00052 * Description        :   Service to report development errors.
00053 *****************************************************************************/
00054 Std_ReturnType Det_ReportError ( uint16 ModuleId , uint8 InstanceId , uint8 ApiId , uint8 ErrorId ) ;
00055
00056 #endif /* DET_H_ */
```

4.10 Platform_Types.h

```
00001 /******************************************************************************
00002 * @Module                : Common
00003 * @File Name             : Platform_Types.h
00004 * @Description           : This document specifies the AUTOSAR platform types header file.
00005                             It contains all plat form dependent types and symbols.
00006                             Those types must be abstracted in order to be come platform and compiler
      independent
00007 * Author                :Salama Mohamed
00008 *******************************************************************************/
00009 /******************************************************************************
00010 * Project               : Graduation_Project_2024
00011 * Platform              : STM32F103C8
00012 * Autosar Version       : 4.8.0
00013 * SW Version            : 1.0.0
00014 *******************************************************************************/
00015 #ifndef PLATFORM_TYPES_H_
00016 #define PLATFORM_TYPES_H_
00017 /******************************************************************************
00018                             Source File Version Informations
00019 *******************************************************************************/
00020 #define PLATFORM_VERSION_ID                       48
```

```
00021 #define PLATFORM_AR_RELEASE_MAJOR_VERSION          4
00022 #define PLATFORM_AR_RELEASE_MINOR_VERSION          8
00023 #define PLATFORM_AR_RELEASE_PATCH_VERSION          0
00024 #define PLATFORM_SW_RELEASE_MAJOR_VERSION          1
00025 #define PLATFORM_SW_RELEASE_MINOR_VERSION          0
00026 #define PLATFORM_SW_RELEASE_PATCH_VERSION          0
00027 /*******************************************************************************
00028                             Symbol and Module Data Types
00029 *******************************************************************************/
00030 //This standard AUTOSAR type shall only be used together with the definitions TRUE and FALSE.
00031 typedef unsigned char boolean ;
00032 #ifndef FALSE
00033 #define FALSE       (0u)
00034 #endif
00035 #ifndef TRUE
00036 #define TRUE        (1u)
00037 #endif
00038 //Indicating a 8 bit processor
00039 #define CPU_TYPE_8      8U
00040 //Indicating a 16 bit processor
00041 #define CPU_TYPE_16     16U
00042 //Indicating a 32 bit processor
00043 #define CPU_TYPE_32     32U
00044 //Indicating a 64 bit processor
00045 #define CPU_TYPE_64     64U
00046 //The most significant bit is the first bit of the bit sequence ( Big endian bit ordering)
00047 #define MSB_FIRST       0U
00048 //The least significant bit is the first bit of the bit sequence. (Little endian bit ordering )
00049 #define LSB_FIRST       1U
00050 //Within uint16, the high byte is located before the low byte. ( Big endian byte  ordering)
00051 #define HIGH_BYTE_FIRST     0U
00052 //Within uint16, the low byte is located before the high byte. (Little endian byte  ordering )
00053 #define LOW_BYTE_FIRST      1U
00054 /*
00055 This symbol shall be defined as #define having one of the values CPU_TYPE_8, CPU_TYPE_16,
00056 CPU_TYPE_32 or CPU_TYPE_64 according to the platform.
00057 */
00058 #define CPU_TYPE        CPU_TYPE_32
00059 /*
00060 This symbol shall be defined as #define having one of the values MSB_FIRST or LSB_FIRST
00061 according to the platform
00062 */
00063 #define CPU_BIT_ORDER   LSB_FIRST
00064 /*
00065 This symbol shall be defined as #define having one of the values HIGH_BYTE_FIRST or LOW_
00066 BYTE_FIRST according to the platform.
00067 */
00068 #define CPU_BYTE_ORDER  LOW_BYTE_FIRST
00069 #if (CPU_TYPE==CPU_TYPE_8)
00070     /*
00071     This standard AUTOSAR type shall be of 8 bit unsigned.
00072     UINT8_MAX 255
00073     UINT8_MIN 0
00074     */
00075     typedef unsigned char         uint8;
00076     /*
00077     This standard AUTOSAR type shall be of 8 bit signed.
00078     SINT8_MAX 127
00079     SINT8_MIN -128
00080     */
00081     typedef signed char           sint8;
00082     /*
00083     This standard AUTOSAR type shall be of 16 bit unsigned..
00084     SINT16_MAX 65535
00085     SINT16_MIN 0
00086     */
00087     typedef unsigned short        uint16;
00088     /*
00089     This standard AUTOSAR type shall be of 16 bit signed.
00090     SINT16_MAX 32767
00091     SINT16_MIN -32768
00092     */
00093     typedef signed short          sint16;
00094     /*
00095     This standard AUTOSAR type shall be 32 bit unsigned.
00096     SINT32_MAX 4294967295
00097     SINT32_MIN 0
00098     */
00099     typedef unsigned long         uint32;
00100     /*
00101     This standard AUTOSAR type shall be 32 bit signed.
00102     SINT32_MAX 2147483647
00103     SINT32_MIN -2147483648
00104     */
00105     typedef signed long           sint32;
00106     /*
00107     This standard AUTOSAR type shall be 64 bit unsigned.
```

```
00108     SINT64_MAX 18446744073709551615
00109     SINT64_MIN 0
00110     */
00111     typedef unsigned long long    uint64;
00112     /*
00113     This standard AUTOSAR type shall be 64 bit signed.
00114     SINT64_MAX 9223372036854775807
00115     SINT64_MIN -9223372036854775808
00116     */
00117     typedef signed long long      sint64;
00118     /*
00119     This standard AUTOSAR type shall follow the 32-bit binary interchange format according to IEEE
00120     754-2008 with encoding parameters specified in chapter 3.6, table 3.5, column "binary32".
00121     FLOAT32_MAX 3.40282347e+38
00122     FLOAT32_MIN 1.17549435e-38
00123     */
00124     typedef float                 float32;
00125     /*
00126     This standard AUTOSAR type shall be of 8 bit signed.
00127     FLOAT64_MAX 2.2204460492503131e-16
00128     FLOAT64_MIN 2.2250738585072014e-308
00129     */
00130     typedef double                float64;
00131     /*
00132     This standard AUTOSAR type shall be a void pointer
00133     Note: This type shall be used for buffers that contain data returned to the caller.
00134     */
00135     #define  VoidPtr void*
00136     /*
00137     This standard AUTOSAR type shall be a void pointer to const.
00138     Note: This type shall be used for buffers that are passed to the callee.
00139     */
00140     #define  ConstVoidPtr const void*
00141 #endif
00142 #if (CPU_TYPE==CPU_TYPE_32)
00143     /*
00144     This standard AUTOSAR type shall be of 8 bit unsigned.
00145     UINT8_MAX 255
00146     UINT8_MIN 0
00147     */
00148     typedef unsigned char         uint8;
00149     /*
00150     This standard AUTOSAR type shall be of 8 bit signed.
00151     SINT8_MAX 127
00152     SINT8_MIN -128
00153     */
00154     typedef signed char           sint8;
00155     /*
00156     This standard AUTOSAR type shall be of 16 bit unsigned..
00157     SINT16_MAX 65535
00158     SINT16_MIN 0
00159     */
00160     typedef unsigned short        uint16;
00161     /*
00162     This standard AUTOSAR type shall be of 16 bit signed.
00163     SINT16_MAX 32767
00164     SINT16_MIN -32768
00165     */
00166     typedef signed short          sint16;
00167     /*
00168     This standard AUTOSAR type shall be 32 bit unsigned.
00169     SINT32_MAX 4294967295
00170     SINT32_MIN 0
00171     */
00172     typedef unsigned long         uint32;
00173     /*
00174     This standard AUTOSAR type shall be 32 bit signed.
00175     SINT32_MAX 2147483647
00176     SINT32_MIN -2147483648
00177     */
00178     typedef signed long           sint32;
00179     /*
00180     This standard AUTOSAR type shall be 64 bit unsigned.
00181     SINT64_MAX 18446744073709551615
00182     SINT64_MIN 0
00183     */
00184     typedef unsigned long long    uint64;
00185     /*
00186     This standard AUTOSAR type shall be 64 bit signed.
00187     SINT64_MAX 9223372036854775807
00188     SINT64_MIN -9223372036854775808
00189     */
00190     typedef signed long long      sint64;
00191     /*
00192     This standard AUTOSAR type shall follow the 32-bit binary interchange format according to IEEE
00193     754-2008 with encoding parameters specified in chapter 3.6, table 3.5, column "binary32".
00194     FLOAT32_MAX 3.40282347e+38
```

```
00195     FLOAT32_MIN 1.17549435e-38
00196     */
00197     typedef float                 float32;
00198     /*
00199     This standard AUTOSAR type shall be of 8 bit signed.
00200     FLOAT64_MAX 2.2204460492503131e-16
00201     FLOAT64_MIN 2.2250738585072014e-308
00202     */
00203     typedef double                float64;
00204     /*
00205     This standard AUTOSAR type shall be a void pointer
00206     Note: This type shall be used for buffers that contain data returned to the caller.
00207     */
00208     #define  VoidPtr void*
00209     /*
00210     This standard AUTOSAR type shall be a void pointer to const.
00211     Note: This type shall be used for buffers that are passed to the callee.
00212     */
00213     #define  ConstVoidPtr const void*
00214 #endif
00215
00216 #endif /* PLATFORM_TYPES_H_ */
```

4.11 Std_Types.h

```
00001 /*******************************************************************************
00002 * @Module       : Common
00003 * @File Name    : StandardTypes.h
00004 * @Description  : This document specifies the AUTOSAR standard types header file. It contains all
00005                   types that are used across several modules of the basic software and that are
00006                   platform and compiler independent.
00007 * Author        :Salama Mohamed
00008 *******************************************************************************/
00009 /*******************************************************************************
00010 * Project               : Graduation_Project_2024
00011 * Platform              : STM32F103C8
00012 * Autosar Version       : 4.8.0
00013 * SW Version            : 1.0.0
00014 *******************************************************************************/
00015 #ifndef STD_TYPES_H
00016 #define STD_TYPES_H
00017 /*******************************************************************************
00018                             Source File Version Informations
00019 *******************************************************************************/
00020 #define STD_VERSION_ID                                  49
00021 #define STD_TYPES_AR_RELEASE_MAJOR_VERSION              4
00022 #define STD_TYPES_AR_RELEASE_MINOR_VERSION              8
00023 #define STD_TYPES_AR_RELEASE_PATCH_VERSION              0
00024 #define STD_TYPES_SW_RELEASE_MAJOR_VERSION              1
00025 #define STD_TYPES_SW_RELEASE_MINOR_VERSION              0
00026 #define STD_TYPES_SW_RELEASE_PATCH_VERSION              0
00027 /*******************************************************************************
00028                             Includes
00029 *******************************************************************************/
00030 #include "Platform_Types.h"
00031 /*******************************************************************************
00032                             Symbol and Module Data Types
00033 *******************************************************************************/
00034 //This type can be used as standard API return type which is shared between the RTE and the BSW
      modules
00035 typedef uint8 Std_ReturnType ;
00036 //Because E_OK is already defined within OSEK, the symbol E_OK has to be shared. To avoid name clashes
      and redefinition problems,
00037 #ifndef STATUSTYPEDEFINED
00038 #define STATUSTYPEDEFINED
00039 #define E_OK        ((Std_ReturnType)0x00U)
00040 #endif
00041 #define E_NOT_OK    ((Std_ReturnType)0x01U)
00042 //This type shall be used to request the version of a BSW module using the <Module name>_Get
      VersionInfo() function.
00043 typedef struct
00044 {
00045     uint16 vendorID ;
00046     uint16 moduleID ;
00047     uint8 sw_major_version ;
00048     uint8 sw_minor_version ;
00049     uint8 sw_patch_version ;
00050 }Std_VersionInfoType;
00051 //The type of the Std_TransformerError
00052 typedef uint8 Std_TransformerErrorCode ;
00053 //Std_TransformerClass is an enumeration where each element represents a transformer class
00054 typedef uint8 Std_TransformerClass ;
00055 //Std_TransformerError represents a transformer error in the context of a certain transformer chain
```

```
00056 typedef struct
00057 {
00058     Std_TransformerErrorCode errorCode ;
00059     Std_TransformerClass transformerClass ;
00060 }Std_TransformerError;
00061 // Physical state 5V or 3.3V
00062 #define STD_HIGH 0x01U
00063 // Physical state 0V
00064 #define STD_LOW 0x00U
00065 // Logical state active
00066 #define STD_ACTIVE 0x01U
00067 // Logical state idle
00068 #define STD_IDLE 0x00U
00069 #define STD_ON 0x01U
00070 #define STD_OFF 0x00U
00071 //The implementation shall provide the NULL_PTR define with a void pointer to zero definition.
00072 #define NULL_PTR ((void *)0)
00073 #endif /* STD_TYPES_H */
```

4.12 stm32f103x6.h

```
00001 /******************************************************************************
00002 * @Module      :     Can_Module
00003 * @File Name   :     stm32f103x6.h
00004 * @Description :    Addresses of the can controller
00005 * Author       :    Salama Mohamed
00006 ******************************************************************************/
00007 /******************************************************************************
00008 * Project               : CAN Module
00009 * Platform              : stm32f103c8
00010 * Autosar Version       : 4.8.0
00011 * SW Version            : 1.0.0
00012 ******************************************************************************/
00013 #ifndef INC_STM32F103X6_H_
00014 #define INC_STM32F103X6_H_
00015 #include "Std_Types.h"
00016 //-----------------------------
00017 //Base addresses for Memories
00018 //-----------------------------
00019 #define PERIPHERALS_BAES                              0X40000000UL
00020 #define NVIC_BASE                                     0xE000E100UL
00021 #define NVIC_ISER0                                    *(volatile uint32*)(NVIC_BASE+0X00)
00022 #define NVIC_ISER1                                    *(volatile uint32*)(NVIC_BASE+0X04)
00023 #define NVIC_ICER0                                    *(volatile uint32*)(NVIC_BASE+0X80)
00024 #define NVIC_ICER1                                    *(volatile uint32*)(NVIC_BASE+0X84)
00025 //=============================================================================
00026 //-----------------------------
00027 //Base addresses for BUS AHB Peripherals
00028 //-----------------------------
00029 //RCC
00030 #define RCC_BASE                              0X40021000UL
00031 //-----------------------------
00032 //Base addresses for BUS APB1 Peripherals
00033 //-----------------------------
00034 //CAN
00035 #define CAN_BASE                                0X40006400UL
00036 #define CAN_MCR                               *(volatile uint32*)(CAN_BASE+0X00)
00037 #define CAN_MSR                               *(volatile uint32*)(CAN_BASE+0X04)
00038 #define CAN_TSR                               *(volatile uint32*)(CAN_BASE+0X08)
00039 #define CAN_RF0R                               *(volatile uint32*)(CAN_BASE+0X0C)
00040 #define CAN_RF1R                               *(volatile uint32*)(CAN_BASE+0X10)
00041 #define CAN_IER                               *(volatile uint32*)(CAN_BASE+0X14)
00042 #define CAN_ESR                               *(volatile uint32*)(CAN_BASE+0X18)
00043 #define CAN_BTR                               *(volatile uint32*)(CAN_BASE+0X1C)
00044 #define CAN_TX_mailbox0_Base                   (CAN_BASE+0X180)
00045 #define CAN_TX_mailbox1_Base                   (CAN_BASE+0X190)
00046 #define CAN_TX_mailbox2_Base                   (CAN_BASE+0X1A0)
00047 #define CAN_TX_mailbox0_Base                   (CAN_BASE+0X180)
00048 #define CAN_TX_mailbox1_Base                   (CAN_BASE+0X190)
00049 #define CAN_RX_FIFO0_Base                      (CAN_BASE+0X1B0)
00050 #define CAN_RX_FIFO1_Base                      (CAN_BASE+0X1C0)
00051 #define CAN_FMR                               *(volatile uint32*)(CAN_BASE+0X200)
00052 #define CAN_FM1R                               *(volatile uint32*)(CAN_BASE+0X204)
00053 #define CAN_FS1R                               *(volatile uint32*)(CAN_BASE+0X20C)
00054 #define CAN_FFA1R                               *(volatile uint32*)(CAN_BASE+0X214)
00055 #define CAN_FA1R                               *(volatile uint32*)(CAN_BASE+0X21C)
00056 #define CAN_Filter_Bank_0_Base                    (CAN_BASE+0X240)
00057 #define CAN_Filter_Bank_1_Base                    (CAN_BASE+0X248)
00058 #define CAN_Filter_Bank_2_Base                    (CAN_BASE+0X250)
00059 #define CAN_Filter_Bank_3_Base                    (CAN_BASE+0X258)
00060 #define CAN_Filter_Bank_4_Base                    (CAN_BASE+0X260)
00061 #define CAN_Filter_Bank_5_Base                    (CAN_BASE+0X268)
00062 #define CAN_Filter_Bank_6_Base                    (CAN_BASE+0X270)
```

```
00063 #define CAN_Filter_Bank_7_Base               (CAN_BASE+0X278)
00064 #define CAN_Filter_Bank_8_Base               (CAN_BASE+0X280)
00065 #define CAN_Filter_Bank_9_Base               (CAN_BASE+0X288)
00066 #define CAN_Filter_Bank_10_Base               (CAN_BASE+0X290)
00067 #define CAN_Filter_Bank_11_Base               (CAN_BASE+0X298)
00068 #define CAN_Filter_Bank_12_Base               (CAN_BASE+0X2A0)
00069 #define CAN_Filter_Bank_13_Base               (CAN_BASE+0X2A8)
00070 //-----------------------------
00071 //Base addresses for BUS APB2 Peripherals
00072 //-----------------------------
00073 //GPIO
00074 //A,B Fully included in LQFP48 package
00075 #define GPIOA_BASE                            0X40010800UL
00076 #define GPIOB_BASE                            0X40010C00UL
00077 //C,D Partial included in LQFP48 package
00078 #define GPIOC_BASE                            0X40011000UL
00079 #define GPIOD_BASE                            0X40011400UL
00080 // E Not included in LQFP48 package
00081 #define GPIOE_BASE                            0X40011800UL
00082 //TIMER2
00083 #define TIM2_BASE                             0x40000000UL
00084 #define TIM3_BASE                             0x40000400UL
00085 #define TIM4_BASE                             0x40000800UL
00086 // USART1
00087 #define USART1_BAES                           0X40013800UL
00088 // USART2
00089 #define USART2_BAES                           0X40004400UL
00090 // USART3
00091 #define USART3_BAES                           0X40004800UL
00092 //NVIC
00093 #define NVIC_ISER_BASE                           0xE000E100UL
00094 #define NVIC_ICER_BASE                           0xE000E180UL
00095 #define NVIC_ISPR_BASE                           0xE000E200UL
00096 #define NVIC_ICPR_BASE                           0xE000E280UL
00097 #define NVIC_IABR_BASE                           0xE000E300UL
00098 #define NVIC_IPR_BASE                            0xE000E400UL
00099 //================================================================================
00100
00101 //-*-*-*-*-*-*-*-*-*-*-*-
00102 //Peripheral register:
00103 //-*-*-*-*-*-*-*-*-*-*-*
00104 //Peripheral register:GPIO
00105 typedef struct
00106 {
00107     volatile uint32 CRL     ;
00108     volatile uint32 CRH     ;
00109     volatile uint32 IDR     ;
00110     volatile uint32 ODR     ;
00111     volatile uint32 BSRR        ;
00112     volatile uint32 BRR     ;
00113     volatile uint32 LCKR        ;
00114 }GPIO_TypeDef;
00115 //Peripheral register:RCC
00116 typedef struct
00117 {
00118     volatile uint32 CR      ;
00119     volatile uint32 CFGR        ;
00120     volatile uint32 CIR     ;
00121     volatile uint32 APB2RSTR    ;
00122     volatile uint32 APB1RSTR    ;
00123     volatile uint32 AHBENR  ;
00124     volatile uint32 APB2ENR ;
00125     volatile uint32 APB1ENR ;
00126     volatile uint32 BDCR        ;
00127     volatile uint32 CSR     ;
00128 }RCC_TypeDef;
00129 //Peripheral register:TIMER
00130 typedef struct
00131 {
00132     volatile uint32 CR1     ;
00133     volatile uint32 CR2     ;
00134     volatile uint32 SMCR        ;
00135     volatile uint32 DIER        ;
00136     volatile uint32 SR      ;
00137     volatile uint32 EGR     ;
00138     volatile uint32 CCMR1       ;//Output Compare mode and input Compare mode
00139     volatile uint32 CCMR2       ;//input Compare mode and Output Compare mode
00140     volatile uint32 CCER        ;
00141     volatile uint32 CNT     ;
00142     volatile uint32 PSC     ;
00143     volatile uint32 ARR     ;
00144     uint32 RESERVED0          ;
00145     volatile uint32 CCR1        ;
00146     volatile uint32 CCR2        ;
00147     volatile uint32 CCR3        ;
00148     volatile uint32 CCR4        ;
00149     uint32 RESERVED1            ;
```

```
00150     volatile uint32 DCR     ;
00151     volatile uint32 DMAR        ;
00152 }TIMx_TypeDef;
00153 typedef struct
00154 {
00155     volatile uint32 CAN_TIxR        ;
00156     volatile uint32 CAN_TDTxR       ;
00157     volatile uint32 CAN_TDLxR       ;
00158     volatile uint32 CAN_TDHxR       ;
00159 }CAN_TX_mailbox_TypeDef;
00160
00161 typedef struct
00162 {
00163     volatile uint32 CAN_RIxR        ;
00164     volatile uint32 CAN_RDTxR       ;
00165     volatile uint32 CAN_RDLxR       ;
00166     volatile uint32 CAN_RDHxR       ;
00167 }CAN_RX_FIFO_TypeDef;
00168
00169 typedef struct
00170 {
00171     volatile uint32 CAN_FiR1        ;
00172     volatile uint32 CAN_FiR2        ;
00173 }CAN_Filter_Bank_TypeDef;
00174 //NVIC
00175 typedef struct
00176 {
00177     volatile uint32 ISER0       ;
00178     volatile uint32 ISER1       ;
00179     volatile uint32 ISER2       ;
00180     volatile uint32 ISER3       ;
00181     volatile uint32 ISER4       ;
00182     volatile uint32 ISER5       ;
00183     volatile uint32 ISER6       ;
00184     volatile uint32 ISER7       ;
00185 }Set_Enable_Register_TypeDef;
00186 typedef struct
00187 {
00188     volatile uint32 ICER0       ;
00189     volatile uint32 ICER1       ;
00190     volatile uint32 ICER2       ;
00191     volatile uint32 ICER3       ;
00192     volatile uint32 ICER4       ;
00193     volatile uint32 ICER5       ;
00194     volatile uint32 ICER6       ;
00195     volatile uint32 ICER7       ;
00196 }Clear_Enable_Register_TypeDef;
00197 typedef struct
00198 {
00199     volatile uint32 ISPR0       ;
00200     volatile uint32 ISPR1       ;
00201     volatile uint32 ISPR2       ;
00202     volatile uint32 ISPR3       ;
00203     volatile uint32 ISPR4       ;
00204     volatile uint32 ISPR5       ;
00205     volatile uint32 ISPR6       ;
00206     volatile uint32 ISPR7       ;
00207 }Set_Pending_Register_TypeDef;
00208 typedef struct
00209 {
00210     volatile uint32 ICPR0       ;
00211     volatile uint32 ICPR1       ;
00212     volatile uint32 ICPR2       ;
00213     volatile uint32 ICPR3       ;
00214     volatile uint32 ICPR4       ;
00215     volatile uint32 ICPR5       ;
00216     volatile uint32 ICPR6       ;
00217     volatile uint32 ICPR7       ;
00218 }Clear_Pending_Register_TypeDef;
00219 typedef struct
00220 {
00221     volatile uint32 IABR0       ;
00222     volatile uint32 IABR1       ;
00223     volatile uint32 IABR2       ;
00224     volatile uint32 IABR3       ;
00225     volatile uint32 IABR4       ;
00226     volatile uint32 IABR5       ;
00227     volatile uint32 IABR6       ;
00228     volatile uint32 IABR7       ;
00229 }Active_Bit_Register_TypeDef;
00230 typedef struct
00231 {
00232     volatile uint32 IPR0        ;
00233     volatile uint32 IPR1        ;
00234     volatile uint32 IPR2        ;
00235     volatile uint32 IPR3        ;
00236     volatile uint32 IPR4        ;
```

```
00237     volatile uint32 IPR5        ;
00238     volatile uint32 IPR6        ;
00239     volatile uint32 IPR7        ;
00240 }Priority_Register_TypeDef;
00241 typedef struct
00242 {
00243     volatile uint32 SR;
00244     volatile uint32 DR;
00245     volatile uint32 BRR;
00246     volatile uint32 CR1;
00247     volatile uint32 CR2;
00248     volatile uint32 CR3;
00249     volatile uint32 GTPR;
00250 }USART_TypeDef;
00251 //=============================================================================================
00252 //-*-*-*-*-*-*-*-*-*-*-*-
00253 //Peripheral Instants:
00254 //-*-*-*-*-*-*-*-*-*-*-*
00255 #define GPIOA       ((GPIO_TypeDef*)GPIOA_BASE)
00256 #define GPIOB       ((GPIO_TypeDef*)GPIOB_BASE)
00257 #define GPIOC       ((GPIO_TypeDef*)GPIOC_BASE)
00258 #define GPIOD       ((GPIO_TypeDef*)GPIOD_BASE)
00259 #define GPIOE       ((GPIO_TypeDef*)GPIOE_BASE)
00260 #define RCC         ((RCC_TypeDef*)RCC_BASE)
00261 #define TIM2        ((TIMx_TypeDef*)TIM2_BASE)
00262 #define TIM3        ((TIMx_TypeDef*)TIM3_BASE)
00263 #define TIM4        ((TIMx_TypeDef*)TIM4_BASE)
00264 #define USART1      ((USART_TypeDef*)USART1_BAES)
00265 #define USART2      ((USART_TypeDef*)USART2_BAES)
00266 #define USART3      ((USART_TypeDef*)USART3_BAES)
00267 //CAN_TX_mailbox
00268 #define CAN_TX_mailbox_0        ((CAN_TX_mailbox_TypeDef*)CAN_TX_mailbox0_Base)
00269 #define CAN_TX_mailbox_1        ((CAN_TX_mailbox_TypeDef*)CAN_TX_mailbox1_Base)
00270 #define CAN_TX_mailbox_2        ((CAN_TX_mailbox_TypeDef*)CAN_TX_mailbox2_Base)
00271 //CAN_RX_FIFO
00272 #define CAN_RX_FIFO_0           ((CAN_RX_FIFO_TypeDef*)CAN_RX_FIFO0_Base)
00273 #define CAN_RX_FIFO_1           ((CAN_RX_FIFO_TypeDef*)CAN_RX_FIFO1_Base)
00274 //Filter bank
00275 #define CAN_FBank_0             ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_0_Base)
00276 #define CAN_FBank_1             ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_1_Base)
00277 #define CAN_FBank_2             ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_2_Base)
00278 #define CAN_FBank_3             ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_3_Base)
00279 #define CAN_FBank_4             ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_4_Base)
00280 #define CAN_FBank_5             ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_5_Base)
00281 #define CAN_FBank_6             ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_6_Base)
00282 #define CAN_FBank_7             ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_7_Base)
00283 #define CAN_FBank_8             ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_8_Base)
00284 #define CAN_FBank_9             ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_9_Base)
00285 #define CAN_FBank_10            ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_10_Base)
00286 #define CAN_FBank_11            ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_11_Base)
00287 #define CAN_FBank_12            ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_12_Base)
00288 #define CAN_FBank_13            ((CAN_Filter_Bank_TypeDef*)CAN_Filter_Bank_13_Base)
00289 //NVIC
00290 #define Set_Enable_Registe      ((Set_Enable_Register_TypeDef*)NVIC_ISER_BASE)
00291 #define Clear_Enable_Register   ((Clear_Enable_Register_TypeDef*)NVIC_ICER_BASE)
00292 #define Set_Pending_Register    ((Set_Pending_Register_TypeDef*)NVIC_ISPR_BASE)
00293 #define Clear_Pending_Register  ((Clear_Pending_Register_TypeDef*)NVIC_ICPR_BASE)
00294 #define Active_Bit_Register     ((Active_Bit_Register_TypeDef*)NVIC_IABR_BASE)
00295 #define Set_Priority_Register   ((Priority_Register_TypeDef*)NVIC_IPR_BASE)
00296 //=============================================================================================
00297 //-*-*-*-*-*-*-*-*-*-*-*-
00298 //clock enable Macros:
00299 //-*-*-*-*-*-*-*-*-*-*-*
00300 #define RCC_GPIOA_CLK_Enable()  (RCC->APB2ENR |=1<<2)
00301 #define RCC_GPIOB_CLK_Enable()  (RCC->APB2ENR |=1<<3)
00302 #define RCC_GPIOC_CLK_Enable()  (RCC->APB2ENR |=1<<4)
00303 #define RCC_GPIOD_CLK_Enable()  (RCC->APB2ENR |=1<<5)
00304 #define RCC_GPIOE_CLK_Enable()  (RCC->APB2ENR |=1<<6)
00305
00306 #define RCC_GPIOA_CLK_Disable() (RCC->APB2RSTR |=1<<2)
00307 #define RCC_GPIOB_CLK_Disable() (RCC->APB2RSTR |=1<<3)
00308 #define RCC_GPIOC_CLK_Disable() (RCC->APB2RSTR |=1<<4)
00309 #define RCC_GPIOD_CLK_Disable() (RCC->APB2RSTR |=1<<5)
00310 #define RCC_GPIOE_CLK_Disable() (RCC->APB2RSTR |=1<<6)
00311 //TIMER2
00312 #define RCC_TIM2_CLK_Enable()   (RCC->APB1ENR |=1<<0)
00313 #define RCC_TIM3_CLK_Enable()   (RCC->APB1ENR |=1<<1)
00314 #define RCC_TIM2_CLK_Disable()  (RCC->APB1RSTR |=1<<0)
00315 #define RCC_TIM3_CLK_Disable()  (RCC->APB1RSTR |=1<<1)
00316 //CAN
00317 #define RCC_CAN_CLK_Enable()    (RCC->APB1ENR |=1<<25)
00318 #define RCC_CAN_CLK_Disable()   (RCC->APB1RSTR |=1<<25)
00319
00320 // USART
00321 #define RCC_USART1_CLK_Enable() (RCC->APB2ENR |=1<<14)
00322 #define RCC_USART2_CLK_Enable() (RCC->APB1ENR |=1<<17)
00323 #define RCC_USART3_CLK_Enable() (RCC->APB1ENR |=1<<18)
```

```
00324 #define RCC_TIM4_CLK_Enable()   (RCC->APB1ENR |=1«2)
00325 #define RCC_USART1_CLK_Disable()    (RCC->APB2RSTR |=1«14)
00326 #define RCC_USART2_CLK_Disable()    (RCC->APB1RSTR |=1«17)
00327 #define RCC_USART3_CLK_Disable()    (RCC->APB1RSTR |=1«18)
00328
00329 //=============================================================================================
00330  /*************************************************************
00331  *                          IVT
00332  *************************************************************/
00333 #define CAN_TX_IRQ              (uint32)19
00334 #define CAN_RX0_IRQ             (uint32)20
00335 #define CAN_RX1_IRQ             (uint32)21
00336 #define CAN_SCE_IRQ             (uint32)22
00337 #define TIM2_ER_IRQ             (uint32)28//TIM2 global interrupt
00338 #define TIM3_ER_IRQ             (uint32)29//TIM3 global interrupt
00339 #define USART1_IRQ              (uint32)37
00340 #define USART2_IRQ              (uint32)38
00341 #define USART3_IRQ              (uint32)39
00342 //-*-*-*-*-*-*-*-*-*-*-*-*-
00343 //NVIC Enable Macros:
00344 //-*-*-*-*-*-*-*-*-*-*-*-*
00345 #define NVIC_IRQ19_CAN_TX_Enable()                          (NVIC_ISER0 |=1«CAN_TX_IRQ)
00346 #define NVIC_IRQ20_CAN_RX0_Enable()                         (NVIC_ISER0 |=1«CAN_RX0_IRQ)
00347 #define NVIC_IRQ21_CAN_RX1_Enable()                         (NVIC_ISER0 |=1«CAN_RX1_IRQ)
00348 #define NVIC_IRQ22_CAN_SCE_Enable()                         (NVIC_ISER0 |=1«CAN_SCE_IRQ)
00349 #define NVIC_IRQ28_TIM2_Enable()                            (NVIC_ISER0 |=1«TIM2_ER_IRQ)
00350 #define NVIC_IRQ29_TIM3_Enable()                            (NVIC_ISER0 |=1«TIM3_ER_IRQ)
00351 #define NVIC_IRQ37_USART1_Enable()                          (NVIC_ISER1 |=1« ( USART1_IRQ - 32) )
00352 #define NVIC_IRQ38_USART2_Enable()                          (NVIC_ISER1 |=1« ( USART2_IRQ - 32) )
00353 #define NVIC_IRQ39_USART3_Enable()                          (NVIC_ISER1 |=1« ( USART3_IRQ - 32) )
00354 //-*-*-*-*-*-*-*-*-*-*-*-*-
00355 //NVIC Disable Macros:
00356 //-*-*-*-*-*-*-*-*-*-*-*-*
00357 #define NVIC_IRQ19_CAN_TX_Disable()                         (NVIC_ICER0 |=1«CAN_TX_IRQ)
00358 #define NVIC_IRQ20_CAN_RX0_Disable()                        (NVIC_ICER0 |=1«CAN_RX0_IRQ)
00359 #define NVIC_IRQ21_CAN_RX1_Disable()                        (NVIC_ICER0 |=1«CAN_RX1_IRQ)
00360 #define NVIC_IRQ22_CAN_SCE_Disable()                        (NVIC_ICER0 |=1«CAN_SCE_IRQ)
00361 #define NVIC_IRQ28_TIM2_Disable()                           (NVIC_ICER0 |=1«TIM2_ER_IRQ)
00362 #define NVIC_IRQ28_TIM3_Disable()                           (NVIC_ICER0 |=1«TIM3_ER_IRQ)
00363 #define NVIC_IRQ37_USART1_Disable()                         (NVIC_ICER1 |=1« ( USART1_IRQ - 32) )
00364 #define NVIC_IRQ38_USART2_Disable()                         (NVIC_ICER1 |=1« ( USART2_IRQ - 32) )
00365 #define NVIC_IRQ39_USART3_Disable()                         (NVIC_ICER1 |=1« ( USART3_IRQ - 32) )
00366 #endif /* INC_STM32F103X6_H_ */
```

4.13 Bluetooth_SWC.h

```
00001 /*************************************************
00002  * SWC         : Bluetooth_SWC.h
00003  * Created on : 16/3/2024
00004  * Author     : Salama Mohamed
00005  * CDD_SWC
00006  *************************************************/
00007 #ifndef HC_05_BLUETOOTH_H_
00008 #define HC_05_BLUETOOTH_H_
00009 #include"stm32_f103c6_USART.h"
00010
00011 void Bluetooth_Init(void);
00012 void Bluetooth_TX_Byte(uint8* PTXBuffer) ;
00013 void Bluetooth_TX_Str(uint8* PTXBuffer) ;
00014 void Bluetooth_RX_Byte(uint8* PRXBuffer) ;
00015 void Bluetooth_RX_Str(uint8* PRXBuffer) ;
00016 void ISR_Bluetooth (void) ;
00017 void USART1_IRQ_Call (void);
00018
00019 #endif /* HC_05_BLUETOOTH_H_ */
```

4.14 Cortex_M3_NVIC.h

```
00001 /*************************************************
00002  * SWC         : Cortex_M3_NVIC.h
00003  * Created on : 10/9/2023
00004  * Author     : Salama Mohamed
00005  *************************************************/
00006 #ifndef INC_CORTEX_M3_NVIC_H_
00007 #define INC_CORTEX_M3_NVIC_H_
00008
00009 #include "stm32f103x6.h"
00010
00011 typedef enum
```

```
00012 {
00013     IRQ_DISABLE,
00014     IRQ_ENABLE
00015 }IRQ_STATE;
00016
00017 void MCAL_NVIC_Enable_IRQ(uint8 IRQ_Position);
00018 void MCAL_NVIC_Disable_IRQ(uint8 IRQ_Position);
00019 IRQ_STATE MCAL_NVIC_Get_State_IRQ(uint8 IRQ_Position);
00020 void MCAL_Set_Pending_IRQ(uint8 IRQ_Position);
00021 void MCAL_Clear_Pending_IRQ(uint8 IRQ_Position);
00022 void MCAL_Set_Priority_IRQ(uint8 IRQ_Position,uint8 Priority);
00023 #endif /* INC_CORTEX_M3_NVIC_H_ */
```

4.15 delay.h

```
00001 /*
00002  * delay.h
00003  *
00004  *  Created on: 20/2/2023
00005  *  Author: Salama mohamed
00006  */
00007
00008 #ifndef INC_DELAY_H_
00009 #define INC_DELAY_H_
00010
00011 #include <stm32f103x6.h>
00012
00013
00014 void delay_ms(int ms);
00015 void delay_us(int us);
00016
00017 #endif /* INC_DELAY_H_ */
```

4.16 RC_Car.h

```
00001 /*
00002  * RC_Car.h
00003  *
00004  *  Created on: Mar 31, 2024
00005  *      Author: ELBOSTAN
00006  */
00007
00008 #ifndef RC_CAR_H_
00009 #define RC_CAR_H_
00010
00011 #include "Std_Types.h"
00012
00013 void Set_Speed(uint8 Speed);
00014 void Forward_Dir(void);
00015 void Back_Dir(void);
00016 void Right_Dir(void);
00017 void Left_Dir(void);
00018 void Stop_Car(void);
00019
00020 #endif /* RC_CAR_H_ */
```

4.17 stm32_f103c6_RCC.h

```
00001 /*
00002  * stm32_f103c6_RCC.h
00003  *  Created on: //
00004  *  Author: Salama mohamed
00005  */
00006
00007 #ifndef INC_STM32_F103C6_RCC_H_
00008 #define INC_STM32_F103C6_RCC_H_
00009
00010 #include "stm32f103x6.h"
00011
00012 #define  HSI_oscillator_clock       (uint32)8000000
00013 #define  HSE_oscillator_clock       (uint32)16000000
00014
00015 uint32 MCAL_Get_SYSCLC_FREQ(void);
00016 uint32 MCAL_Get_HCLC_FREQ(void)  ;
00017 uint32 MCAL_Get_PCLC1_FREQ(void) ;
00018 uint32 MCAL_Get_PCLC2_FREQ(void) ;
00019
00020 #endif /* INC_STM32_F103C6_RCC_H_ */
```

4.18 stm32_f103c6_USART.h

```
00001 /*
00002  * stm32_f103c6_USART.h
00003  *
00004  *  Created on: //
00005  *  Author: Salama mohamed
00006  */
00007
00008 #ifndef INC_STM32_F103C6_USART_H_
00009 #define INC_STM32_F103C6_USART_H_
00010
00011 //Include
00012 #include "stm32f103x6.h"
00013 #include "stm32_f103c6_RCC.h"
00014
00015
00016
00017
00018 //-----------------------------
00019 //User type definitions (structures)
00020 //-----------------------------
00021
00022 typedef struct
00023 {
00024     uint8 MODE           ;      //specified the USART (TX/RX) to be configured .
00025                                   //this parameter must be set based on  @ ref  USART_MODE_Define
00026
00027     uint8 NUM_DATA_BIT  ;    //specified the number of data bit (8or 9 ) .
00028                                   //this parameter must be set based on  @ ref USART_NUM_DATA_BIT_Define
00029
00030     uint8 NUM_STOP_BIT  ;    //specified the number of stop bit ( 0.5 or 1 or 1.5 or 2 ) .
00031                                   //this parameter must be set based on  @ ref
      USART_NUM_STOP_BIT_Define
00032
00033     uint32 BAUDRATE     ;    //specified the baudrate  .
00034                                   //this parameter must be set based on  @ ref  USART_BAUDRATE_Define
00035
00036     uint8 PARITY            ;    //specified the parity disable or enable (odd or even ) .
00037                                   //this parameter must be set based on  @ ref  USART_PARITY_Define
00038
00039     uint8 HWFLOWCTL     ;    //specified the hardware flow control mode (disable or enable )  .
00040                                   //this parameter must be set based on  @ ref  USART_HWFLOWCTL_Define
00041
00042     uint8 IRQ_EN        ;    //specified the interrupt (disable or enable )  .
00043                                   //this parameter must be set based on  @ ref  USART_IRQ_ENABLE_Define
00044
00045     void (*P_IRQ_CALL)(void);     // set the function which will be call at IRQ happen
00046
00047 }USART_config_t;
00048
00049 enum polling_mechanism
00050 {
00051     Enable ,
00052     Disable
00053 };
00054
00055 //=======================================================================
00056
00057 //-----------------------------
00058 //Macros Configuration References
00059 //-----------------------------
00060
00061 // @ ref  USART_MODE_Define
00062 /*
00063 Bit 2 RE: Receiver enable
00064 This bit enables the receiver. It is set and cleared by software.
00065 0: Receiver is disabled
00066 1: Receiver is enabled and begins searching for a start bit
00067
00068 Bit 3 TE: Transmitter enable
00069 This bit enables the transmitter. It is set and cleared by software.
00070 0: Transmitter is disabled
00071 1: Transmitter is enabled
00072  */
00073 #define USART_MODE_TX           (uint32)(1<<2)
00074 #define USART_MODE_RX           (uint32)(1<<3)
00075 #define USART_MODE_TX_RX        (uint32)(1<<2 | 1<<3 )
00076
00077
00078 // @ ref USART_NUM_DATA_BIT_Define
00079 /*
00080  Bit 12 M: Word length
00081 This bit determines the word length. It is set or cleared by software.
00082 0: 1 Start bit, 8 Data bits, n Stop bit
00083 1: 1 Start bit, 9 Data bits, n Stop bit
00084  */
```

```
00085 #define USART_NUM_DATA_BIT_8        (uint32)(0)
00086 #define USART_NUM_DATA_BIT_9        (uint32)(1<<12)
00087
00088 //@ ref  USART_NUM_STOP_BIT_Define
00089 /*
00090  Bits 13:12 STOP: STOP bits
00091 These bits are used for programming the stop bits.
00092 00: 1 Stop bit
00093 01: 0.5 Stop bit
00094 10: 2 Stop bits
00095 11: 1.5 Stop bit
00096  */
00097
00098 #define USART_NUM_STOP_BIT_0_5      (uint32)(1<<12)
00099 #define USART_NUM_STOP_BIT_1        (uint32)(0)
00100 #define USART_NUM_STOP_BIT_1_5      (uint32)(3<<12)
00101 #define USART_NUM_STOP_BIT_2        (uint32)(2<<12)
00102
00103 //  @ ref  USART_BAUDRATE_Define
00104 #define USART_BAUDRATE_2400             2400
00105 #define USART_BAUDRATE_9600             9600
00106 #define USART_BAUDRATE_19200            19200
00107 #define USART_BAUDRATE_57600            57600
00108 #define USART_BAUDRATE_115200           115200
00109 #define USART_BAUDRATE_230400           230400
00110 #define USART_BAUDRATE_460800           460800
00111 #define USART_BAUDRATE_921600           921600
00112 #define USART_BAUDRATE_2250000          2250000
00113 #define USART_BAUDRATE_4500000          4500000
00114
00115
00116 //  @ ref  USART_PARITY_Define
00117 /*
00118  Bit 10 PCE: Parity control enable
00119  0: Parity control disabled
00120  1: Parity control enabled
00121
00122  Bit 9 PS: Parity selection
00123  0: Even parity
00124  1: Odd parity
00125  */
00126 #define USART_PARITY_Disable            (uint32)(0)
00127 #define USART_PARITY_ODD                (uint32)( 1<<10 | 1<<9 )
00128 #define USART_PARITY_EVEN               (uint32)( 1<<10 )
00129
00130
00131 // @ ref  USART_HWFLOWCTL_Define
00132 /*
00133  Bit 9 CTSE: CTS enable
00134  0: CTS hardware flow control disabled
00135  1: CTS mode enabled
00136
00137  Bit 8 RTSE: RTS enable
00138  0: RTS hardware flow control disabled
00139  1: RTS interrupt enabled
00140  */
00141 #define USART_HWFLOWCTL_Disable             (uint32)(0)
00142 #define USART_HWFLOWCTL_RTS                 (uint32)( 1<<8 )
00143 #define USART_HWFLOWCTL_CTS                 (uint32)( 1<<9 )
00144 #define USART_HWFLOWCTL_RTS_CTS             (uint32)( 1<<8 | 1<<9 )
00145
00146
00147 // @ ref  USART_IRQ_ENABLE_Define
00148 #define USART_IRQ_ENABLE_Disable            (uint32)(0)
00149 #define USART_IRQ_ENABLE_TXE                (uint32)( 1<<7 )//Transmit data register empty
00150 #define USART_IRQ_ENABLE_TC                 (uint32)( 1<<6 )//Transmission complete
00151 #define USART_IRQ_ENABLE_RXNE               (uint32)( 1<<5 )//Received data ready to be read
00152 #define USART_IRQ_ENABLE_PE                 (uint32)( 1<<8 )//Parity error
00153
00154 // Baudrate calculation
00155 #define USARTDIV( PCLK , BAUD)              ( uint32 )( PCLK / ( 16 * BAUD ) )
00156 #define USARTDIV_MUL100( PCLK , BAUD)       ( uint32 )( ( 25 * PCLK ) / ( 4 * BAUD ) )
00157 #define Mantissa_MUL100( PCLK , BAUD)       ( uint32 )( USARTDIV( PCLK , BAUD) * 100 )
00158 #define Mantissa( PCLK , BAUD)              ( uint32 )( USARTDIV( PCLK , BAUD) )
00159 #define DIV_Fraction( PCLK , BAUD)          ( uint32 )(( USARTDIV_MUL100( PCLK , BAUD) -
      Mantissa_MUL100( PCLK , BAUD) ) * 100 )
00160 #define USART_BRR_Reg( PCLK , BAUD)         ( (Mantissa( PCLK , BAUD)) << 4 ) | ( ( DIV_Fraction( PCLK
      , BAUD) ) &0XF )
00161 //=============================================================================
00162
00163 /*
00164 * ===============================================
00165 * APIs Supported by "MCAL GPIO DRIVER"
00166 * ===============================================
00167 */
00168 void MCAL_USART_Init( USART_TypeDef* USARTx , USART_config_t* USART_Config );
00169 void MCAL_USART_DeInit( USART_TypeDef* USARTx );
```

```
00170 void MCAL_USART_Set_Pin( USART_TypeDef* USARTx );
00171 void MCAL_USART_ReceiveData( USART_TypeDef* USARTx , uint16* PRXBuffer , enum polling_mechanism
      polling_EN);
00172 void MCAL_USART_SendData( USART_TypeDef* USARTx , uint16* PTXBuffer , enum polling_mechanism
      polling_EN);
00173 void MCAL_USART_WAIT_TC( USART_TypeDef* USARTx );
00174 void MCAL_USART_Send_String( USART_TypeDef* USARTx , uint8* PTXBuffer , enum polling_mechanism
      polling_EN);
00175 void MCAL_USART_Receive_String( USART_TypeDef* USARTx , uint8* PRXBuffer , enum polling_mechanism
      polling_EN);
00176 //===============================================================================
00177
00178 #endif /* INC_STM32_F103C6_USART_H_ */
```

4.19 Dio.h

```
00001 /*******************************************************************************
00002 * @Module      :     Dio
00003 * @File Name   :     Dio.h
00004 * @Description :     This specification specifies the functionality, API and the configuration of the
      AUTOSAR
00005                       Basic Software module Dio Driver.
00006 * Author      :     Salama Mohamed
00007 *******************************************************************************/
00008 /*******************************************************************************
00009 * Project               : Graduation_Project_2024
00010 * Platform              : STM32F103C8
00011 * Autosar Version       : 4.8.0
00012 * SW Version            : 1.0.0
00013 *******************************************************************************/
00014 #ifndef DIO_H_
00015 #define DIO_H_
00016 /*******************************************************************************
00017                             Source File Version Informations
00018 *******************************************************************************/
00019 #define DIO_VERSION_ID                          20
00020 #define DIO_AR_RELEASE_MAJOR_VERSION            4
00021 #define DIO_AR_RELEASE_MINOR_VERSION            8
00022 #define DIO_AR_RELEASE_PATCH_VERSION            0
00023 #define DIO_SW_RELEASE_MAJOR_VERSION            1
00024 #define DIO_SW_RELEASE_MINOR_VERSION            0
00025 #define DIO_SW_RELEASE_PATCH_VERSION            0
00026 #define VENDOR_ID                               100
00027 /*******************************************************************************
00028                             Includes
00029 *******************************************************************************/
00030 #include "Dio_Cfg.h"
00031 #include "Det.h"
00032 #include "Std_Types.h"
00033 // AUTOSAR checking Std_Version
00034 #if ((STD_TYPES_AR_RELEASE_MAJOR_VERSION != DIO_AR_RELEASE_MAJOR_VERSION)\
00035 ||  (STD_TYPES_AR_RELEASE_MINOR_VERSION  != DIO_AR_RELEASE_MINOR_VERSION)\
00036 ||  (STD_TYPES_AR_RELEASE_PATCH_VERSION  != DIO_AR_RELEASE_PATCH_VERSION))
00037 #error "The Autosar version of Std_Types.h does not match the DIO version"
00038 #endif
00039 /*******************************************************************************
00040                       API Service Id Macros
00041 *******************************************************************************/
00042 //Service ID for Dio_ReadChannel
00043 #define Dio_ReadChannel_ID                      (uint8)0x00
00044 //Service ID for Dio_WriteChannel
00045 #define Dio_WriteChannel_ID                     (uint8)0x01
00046 //Service ID for Dio_ReadPort
00047 #define Dio_ReadPort_ID                         (uint8)0x02
00048 //Service ID for Dio_WritePort
00049 #define Dio_WritePort_ID                        (uint8)0x03
00050 //Service ID for Dio_ReadChannelGroup
00051 #define Dio_ReadChannelGroup_ID                 (uint8)0x04
00052 //Service ID for Dio_WriteChannelGroup
00053 #define Dio_WriteChannelGroup_ID                (uint8)0x05
00054 //Service ID for Dio_FlipChannel
00055 #define Dio_FlipChannel_ID                      (uint8)0x11
00056 //Service ID for Dio_GetVersionInfo
00057 #define Dio_GetVersionInfo_ID                   (uint8)0x12
00058 //Service ID for Dio_MaskedWritePort
00059 #define Dio_MaskedWritePort_ID                  (uint8)0x13
00060 /*******************************************************************************
00061                       DET Error Codes
00062 *******************************************************************************/
00063 //Invalid channel requested
00064 #define DIO_E_PARAM_INVALID_CHANNEL_ID                  (uint8)0x0A
00065 //Invalid port requested
00066 #define DIO_E_PARAM_INVALID_PORT_ID                     (uint8)0x14
```

```
00067 //Invalid channel group requested
00068 #define DIO_E_PARAM_INVALID_GROUP                       (uint8)0x1F
00069 //API service called with a NULL pointer
00070 #define DIO_E_PARAM_POINTER                             (uint8)0x20
00071 /*********************************************************************
00072                             Type definitions
00073 *********************************************************************/
00074 // Numeric ID of a DIO channel.
00075 typedef uint16 Dio_ChannelType;
00076 //Numeric ID of a DIO port.
00077 typedef uint8 Dio_PortType;
00078 /*
00079  Type for the definition of a channel group, which consists of several adjoining
00080  channels within a port.
00081  */
00082 typedef struct
00083 {
00084     // This element mask which defines the positions of the channel group.
00085     uint8 mask;
00086     // This element shall be the position of the Channel Group on the port, counted from the LSB.
00087     uint8 offset;
00088     // This shall be the port on which the Channel group is defined.
00089     Dio_PortType port;
00090 }Dio_ChannelGroupType;
00091 //These are the possible levels a DIO channel can have (input or output).Range(STD_LOW or STD_HIGH)
00092 typedef uint8 Dio_LevelType;
00093 /*
00094  If the µC owns ports of different port widths (e.g. 4, 8,16...Bit) Dio_Port
00095  LevelType inherits the size of the largest port
00096  */
00097 typedef uint16 Dio_PortLevelType;
00098 /*********************************************************************
00099                                 APIS
00100 *********************************************************************/
00101 /*********************************************************************
00102 * Service ID [hex]  :   0x00
00103 * Service Name      :   Dio_ReadChannel
00104 * Sync/Async        :   Synchronous
00105 * Reentrancy        :   Reentrant
00106 * Parameters (in)   :   ChannelId (ID of DIO channel).
00107 * Parameters (inout):   None
00108 * Parameters (out)  :   None
00109 * Return value      :   Dio_Level Type
00110                         STD_HIGH ( The physical level of the corresponding Pin is STD_HIGH)
00111                         STD_LOW (The physical level of the corresponding Pin is STD_LOW)
00112 * Description       :   Returns the value of the specified DIO channel.
00113 *********************************************************************/
00114 Dio_LevelType Dio_ReadChannel (Dio_ChannelType ChannelId);
00115 /*********************************************************************
00116 * Service ID [hex]  :   0x01
00117 * Service Name      :   Dio_WriteChannel
00118 * Sync/Async        :   Synchronous
00119 * Reentrancy        :   Reentrant
00120 * Parameters (in)   :   ChannelId (ID of DIO channel).
00121 * Parameters (in)   :   Level (Value to be written).
00122 * Parameters (inout):   None
00123 * Parameters (out)  :   None
00124 * Return value      :   None
00125 * Description       :   Service to set a level of a channel.
00126 *********************************************************************/
00127 void Dio_WriteChannel (Dio_ChannelType ChannelId,Dio_LevelType Level);
00128 /*********************************************************************
00129 * Service ID [hex]  :   0x02
00130 * Service Name      :   Dio_ReadPort
00131 * Sync/Async        :   Synchronous
00132 * Reentrancy        :   Reentrant
00133 * Parameters (in)   :   PortId (ID of DIO Port).
00134 * Parameters (inout):   None
00135 * Parameters (out)  :   None
00136 * Return value      :   Dio_PortLevelType (Level of all channels of that port)
00137 * Description       :   Returns the level of all channels of that port.
00138 *********************************************************************/
00139 Dio_PortLevelType Dio_ReadPort (Dio_PortType PortId);
00140 /*********************************************************************
00141 * Service ID [hex]  :   0x03
00142 * Service Name      :   Dio_WritePort
00143 * Sync/Async        :   Synchronous
00144 * Reentrancy        :   Reentrant
00145 * Parameters (in)   :   PortId (ID of DIO Port).
00146 * Parameters (in)   :   Level (Value to be written).
00147 * Parameters (inout):   None
00148 * Parameters (out)  :   None
00149 * Return value      :   None
00150 * Description       :   Service to set a value of the port.
00151 *********************************************************************/
00152 void Dio_WritePort (Dio_PortType PortId,Dio_PortLevelType Level);
00153 /*********************************************************************
```

```
00154 * Service ID [hex]  :   0x12
00155 * Service Name      :   Dio_GetVersionInfo
00156 * Sync/Async        :   Synchronous
00157 * Reentrancy        :   Reentrant
00158 * Parameters (in)   :   None
00159 * Parameters (inout):   None
00160 * Parameters (out)  :   VersionInfo
00161                          Pointer to where to store the version information of this module.
00162 * Return value      :   None
00163 * Description       :   Service to get the version information of this module.
00164 ************************************************************************/
00165 void Dio_GetVersionInfo (Std_VersionInfoType* VersionInfo);
00166 #endif /* DIO_H_ */
```

4.20 Dio_Cfg.h

```
00001 /******************************************************************************
00002 * @Module      :     Dio
00003 * @File Name   :     Dio_Cfg.h
00004 * @Description :     the Pre-compile configuration of the AUTOSAR Basic Software module Dio Driver.
00005 * Author      :     Salama Mohamed
00006 ******************************************************************************/
00007 /*******************************************************************************
00008 * Project               : Graduation_Project_2024
00009 * Platform              : STM32F103C8
00010 * Autosar Version       : 4.8.0
00011 * SW Version            : 1.0.0
00012 *******************************************************************************/
00013 #ifndef DIO_CFG_H_
00014 #define DIO_CFG_H_
00015 /*****************************************************************************
00016                 General DIO module configuration parameters.
00017 *****************************************************************************/
00018 //Switches the development error detection and notification on or off.
00019 #define DioDevErrorDetect           TRUE
00020 //Adds / removes the service Dio_FlipChannel() from the code.
00021 #define DioFlipChannelApi           FALSE
00022 //Adds / removes the service Dio_MaskedWritePort() from the code.
00023 #define DioMaskedWritePortApi       FALSE
00024 //Adds / removes the service Dio_ GetVersionInfo() from the code.
00025 #define DioVersionInfoApi           TRUE
00026 /*****************************************************************************
00027                 Configuration of individual DIO ports.
00028 *****************************************************************************/
00029 //Numeric identifier of the DIO port.
00030 #define DioPort_A               (Dio_PortType)0
00031 #define DioPort_B               (Dio_PortType)1
00032 #define DioPort_C               (Dio_PortType)2
00033 //Channel Id of the DIO channel.
00034 #define channel_0               (Dio_ChannelType)0
00035 #define channel_1               (Dio_ChannelType)1
00036 #define channel_2               (Dio_ChannelType)2
00037 #define channel_3               (Dio_ChannelType)3
00038 #define channel_4               (Dio_ChannelType)4
00039 #define channel_5               (Dio_ChannelType)5
00040 #define channel_6               (Dio_ChannelType)6
00041 #define channel_7               (Dio_ChannelType)7
00042 #define channel_8               (Dio_ChannelType)8
00043 #define channel_9               (Dio_ChannelType)9
00044 #define channel_10              (Dio_ChannelType)10
00045 #define channel_11              (Dio_ChannelType)11
00046 #define channel_12              (Dio_ChannelType)12
00047 #define channel_13              (Dio_ChannelType)13
00048 #define channel_14              (Dio_ChannelType)14
00049 #define channel_15              (Dio_ChannelType)15
00050 #define channel_16              (Dio_ChannelType)16
00051 #define channel_17              (Dio_ChannelType)17
00052 #define channel_18              (Dio_ChannelType)18
00053 #define channel_19              (Dio_ChannelType)19
00054 #define channel_20              (Dio_ChannelType)20
00055 #define channel_21              (Dio_ChannelType)21
00056 #define channel_22              (Dio_ChannelType)22
00057 #define channel_23              (Dio_ChannelType)23
00058 #define channel_24              (Dio_ChannelType)24
00059 #define channel_25              (Dio_ChannelType)25
00060 #define channel_26              (Dio_ChannelType)26
00061 #define channel_27              (Dio_ChannelType)27
00062 #define channel_28              (Dio_ChannelType)28
00063 #define channel_29              (Dio_ChannelType)29
00064 #define channel_30              (Dio_ChannelType)30
00065 #define channel_31              (Dio_ChannelType)31
00066 #define channel_32              (Dio_ChannelType)32
00067 #define channel_45              (Dio_ChannelType)45
```

```
00068 #define channel_46              (Dio_ChannelType)46
00069 #define channel_47              (Dio_ChannelType)47
00070 //Number of channel
00071 #define MAX_Number_Channel      (Dio_ChannelType)47
00072 #define DIO_INSTANCE_ZERO           0
00073 #endif /* DIO_CFG_H_ */
```

4.21 Icu.h

```
00001 /*******************************************************************************
00002 * @Module      :     ICU
00003 * @File Name   :     Icu.h
00004 * @Description :    This specification specifies the functionality, API and the configuration of the
00005                      AUTOSAR Basic Software module ICU driver.
00006 * Author       :    Salama Mohamed
00007 *******************************************************************************/
00008 /*******************************************************************************
00009 * Project               : Graduation_Project_2024
00010 * Platform              : STM32F103C8
00011 * Autosar Version       : 4.8.0
00012 * SW Version            : 1.0.0
00013 *******************************************************************************/
00014 #ifndef ICU_H_
00015 #define ICU_H_
00016 /*******************************************************************************
00017                             Includes
00018 *******************************************************************************/
00019 #include "Std_Types.h"
00020 #include "Det.h"
00021 #include "Icu_Cfg.h"
00022 /*******************************************************************************
00023                      API Service Id Macros
00024 *******************************************************************************/
00025 #define Icu_Init_ID                                 (uint8)0x00
00026 #define Icu_DeInit_ID                               (uint8)0x01
00027 #define Icu_SetMode_ID                              (uint8)0x02
00028 #define Icu_SetActivationCondition_ID               (uint8)0x05
00029 #define Icu_DisableNotification_ID                  (uint8)0x06
00030 #define Icu_EnableNotification_ID                   (uint8)0x07
00031 #define Icu_GetInputState_ID                        (uint8)0x08
00032 #define Icu_StartTimestamp_ID                       (uint8)0x09
00033 #define Icu_StopTimestamp_ID                        (uint8)0x0a
00034 #define Icu_GetTimestampIndex_ID                    (uint8)0x0b
00035 #define Icu_StartSignalMeasurement_ID               (uint8)0x13
00036 #define Icu_StopSignalMeasurement_ID                (uint8)0x14
00037 #define Icu_GetTimeElapsed_ID                       (uint8)0x10
00038 #define Icu_GetDutyCycleValues_ID                   (uint8)0x11
00039 /*******************************************************************************
00040                      DET Error Codes
00041 *******************************************************************************/
00042 //API IS called with invalid pointer.
00043 #define ICU_E_PARAM_POINTER                         (uint8)0x0a
00044 //API service used with an invalid channel identifier or channel was not configured for the
      functionality of the calling API.
00045 #define ICU_E_PARAM_CHANNEL                         (uint8)0x0b
00046 //API service used with an invalid or not feasible activation.
00047 #define ICU_E_PARAM_ACTIVATION                      (uint8)0x0c
00048 //Init function failed.
00049 #define ICU_E_INIT_FAILED                           (uint8)0x0d
00050 //API service used with an invalid buffer size.
00051 #define ICU_E_PARAM_BUFFER_SIZE                     (uint8)0x0e
00052 //API service Icu_SetMode used with an invalid mode.
00053 #define ICU_E_PARAM_MODE                            (uint8)0x0f
00054 //API service used without module initialization
00055 #define ICU_E_UNINIT                                (uint8)0x14
00056 //API service Icu_SetMode is called while a running operation.
00057 #define ICU_E_BUSY_OPERATION                        (uint8)0x16
00058 //API Icu_Init service is called and when the ICU driver and the Hardware are already initialized.
00059 #define ICU_E_ALREADY_INITIALIZED                   (uint8)0x17
00060 //API Icu_StartTimeStamp is called and the parameter Notify Interval is invalid
00061 #define ICU_E_PARAM_NOTIFY_INTERVAL                 (uint8)0x18
00062 //API Icu_GetVersionInfo is called and the parameter version info is is invalid
00063 #define ICU_E_PARAM_VINFO                           (uint8)0x19
00064 //API service Icu_StopTimestamp called on a channel which was not started or already stopped
00065 #define ICU_E_NOT_STARTED                           (uint8)0x15
00066 /*******************************************************************************
00067                              Type definitions
00068 *******************************************************************************/
00069 #define Icu_NOT_INITIALIZED 0U
00070 #define Icu_INITIALIZED     1U
00071 // Allow enabling / disabling of all interrupts which are not required for the ECU wakeup.
00072 typedef enum
00073 {
```

```
00074     // Normal operation, all used interrupts are enabled according to the notification requests.
00075     ICU_MODE_NORMAL,
00076     /*
00077      Reduced power operation. In sleep mode only those
00078      notifications are available which are configured as wakeup capable.
00079      */
00080     ICU_MODE_SLEEP
00081 }Icu_ModeType;
00082 /*
00083  Numeric identifier of an ICU channel.
00084  Comment:
00085  This is implementation specific but not all values may be valid within the type.
00086  This type shall be chosen in order to have the most efficient implementation on
00087  a specific microcontroller platform.
00088  */
00089 typedef uint8 Icu_ChannelType;
00090 // Input state of an ICU channel
00091 typedef enum
00092 {
00093     // An activation edge has been detected
00094     ICU_ACTIVE,
00095     /*
00096      No activation edge has been detected since the last call of Icu_
00097      GetInputState() or Icu_Init().
00098      */
00099     ICU_IDLE
00100 }Icu_InputStateType;
00101 // Definition of the type of activation of an ICU channel.
00102 typedef enum
00103 {
00104     //An appropriate action shall be executed when a rising edge occurs on the ICU input signal.
00105     ICU_RISING_EDGE,
00106     // An appropriate action shall be executed when a falling edge occurs on the ICU input signal.
00107     ICU_FALLING_EDGE,
00108     // An appropriate action shall be executed when either a rising or falling edge occur on the ICU
      input signal.
00109     ICU_BOTH_EDGES
00110 }Icu_ActivationType;
00111 // Width of the buffer for timestamp ticks and measured elapsed timeticks.(0 ... <width of the timer
      register>)
00112 typedef uint16 Icu_ValueType;
00113 //Type which shall contain the values, needed for calculating duty cycles.
00114 typedef struct
00115 {
00116     // This shall be the coherent active-time measured on a channel
00117     Icu_ValueType ActiveTime;
00118     // This shall be the coherent period-time measured on a channel
00119     Icu_ValueType PeriodTime;
00120 }Icu_DutyCycleType;
00121 // Definition of the measurement mode type
00122 typedef enum
00123 {
00124     // Mode for detecting edges
00125     ICU_MODE_SIGNAL_EDGE_DETECT,
00126     // Mode for measuring different times between various configurable edges
00127     ICU_MODE_SIGNAL_MEASUREMENT,
00128     // Mode for capturing timer values on configurable edges
00129     ICU_MODE_TIMESTAMP,
00130     // Mode for counting edges on configurable edges
00131     ICU_MODE_EDGE_COUNTER
00132 }Icu_MeasurementModeType;
00133 // Definition of the measurement property type
00134 typedef enum
00135 {
00136     // The channel is configured for reading the elapsed Signal Low Time
00137     ICU_LOW_TIME,
00138     // The channel is configured for reading the elapsed Signal High Time
00139     ICU_HIGH_TIME,
00140     // The channel is configured for reading the elapsed Signal Period Time
00141     ICU_PERIOD_TIME,
00142     //The channel is configured to read values which are needed for calculating the duty cycle
      (coherent Active and Period Time
00143     ICU_DUTY_CYCLE
00144 }Icu_SignalMeasurementPropertyType;
00145 // Definition of the timestamp measurement property type
00146 typedef enum
00147 {
00148     // The buffer will just be filled once
00149     ICU_LINEAR_BUFFER,
00150     // After reaching the end of the buffer, the driver restarts at the beginning of the buffer
00151     ICU_CIRCULAR_BUFFER
00152 }Icu_TimestampBufferType;
00153 /*******************************************************************************************
00154 Name            :               IcuSignalMeasurement
00155 Parent Container:               IcuChannel
00156 Description     :               This container contains the configuration (parameters) in case
00157                                 the measurement mode is "IcuSignalMeasurement"
```

```
00158 Type            :               Container
00159 *******************************************************************************/
00160 typedef struct
00161 {
00162     /*
00163      Configures the property that could be measured in case the mode is "IcuSignal
00164      Measurement". This property can not be changed during runtime.
00165      */
00166     Icu_SignalMeasurementPropertyType IcuSignalMeasurementProperty;
00167 }IcuSignalMeasurement;
00168 /*******************************************************************************
00169 Name            :               IcuChannel
00170 Parent Container:               IcuConfigSet
00171 Description     :               Configuration of an individual ICU channel.
00172 Type            :               Container
00173 *******************************************************************************/
00174 typedef struct
00175 {
00176     //Channel Id of the ICU channel.
00177     Icu_ChannelType IcuChannelId;
00178     /*
00179      Configures the default-activation-edge which shall be used for this channel if
00180      there was no activation-edge configured by the call of service Icu_SetActivation Condition().
00181      */
00182     Icu_ActivationType IcuDefaultStartEdge;
00183     //Configures the measurement mode of this channel.
00184     Icu_MeasurementModeType IcuMeasurementMode;
00185     //Information about the wakeup-capability of this channel.
00186     boolean IcuWakeupCapability;
00187     //Information about Signal Measurement
00188     IcuSignalMeasurement IcuSignal_Measurement;
00189 }IcuChannel;
00190 /*******************************************************************************
00191 Name            :               IcuConfigSet
00192 Parent Container:               ICU
00193 Description     :               This container contains the configuration parameters and
00194                                 sub containers of the AUTOSAR ICU module.
00195 Type            :               Container
00196 *******************************************************************************/
00197 typedef struct
00198 {
00199     //This parameter contains the number of Channels configured
00200     Icu_ChannelType IcuMaxChannel;
00201     //Configuration of an individual ICU channel.
00202     IcuChannel IcuChannel[IcuMax_Channel];
00203 }IcuConfigSet;
00204
00205 /*
00206   This type contains initialization data.
00207   Comment:
00208   Hardware and implementation dependent structure. The contents of the
00209   initialization data structure are microcontroller specific.
00210  */
00211 typedef struct
00212 {
00213     IcuConfigSet IcuConfigSet;
00214 }Icu_ConfigType;
00215 /*******************************************************************************
00216                                 APIS
00217 *******************************************************************************/
00218 //callback notification
00219 void Icu_SignalNotification_Channel_0(void);
00220 void Icu_SignalNotification_Channel_1(void);
00221 void Icu_SignalNotification_Channel_2(void);
00222 void Icu_SignalNotification_Channel_3(void);
00223 /*******************************************************************************
00224 * Service ID [hex]   :   0x00
00225 * Service Name       :   Icu_Init
00226 * Sync/Async         :   Synchronous
00227 * Reentrancy         :   Non Reentrant
00228 * Parameters (in)    :   ConfigPtr
00229                          Pointer to a selected configuration structure
00230 * Parameters (inout):   None
00231 * Parameters (out)   :   None
00232 * Return value       :   None
00233 * Description        :   This function initializes the driver.
00234 *******************************************************************************/
00235 void Icu_Init (const Icu_ConfigType* ConfigPtr);
00236 /*******************************************************************************
00237 * Service ID [hex]   :   0x01
00238 * Service Name       :   Icu_DeInit
00239 * Sync/Async         :   Synchronous
00240 * Reentrancy         :   Non Reentrant
00241 * Parameters (in)    :   None
00242 * Parameters (inout):   None
00243 * Parameters (out)   :   None
00244 * Return value       :   None
```

```
00245 * Description       :   This function de-initializes the ICU module.
00246 ************************************************************************/
00247 void Icu_DeInit (void);
00248 /************************************************************************
00249 * Service ID [hex]  :   0x02
00250 * Service Name      :   Icu_SetMode
00251 * Sync/Async        :   Synchronous
00252 * Reentrancy        :   Non Reentrant
00253 * Parameters (in)   :   Mode
00254                         ICU_MODE_NORMAL or ICU_MODE_SLEEP
00255 * Parameters (inout):   None
00256 * Parameters (out)  :   None
00257 * Return value      :   None
00258 * Description       :   This function sets the ICU mode.
00259 ************************************************************************/
00260 void Icu_SetMode (Icu_ModeType Mode);
00261 /************************************************************************
00262 * Service ID [hex]  :   0x05
00263 * Service Name      :   Icu_SetActivationCondition
00264 * Sync/Async        :   Synchronous
00265 * Reentrancy        :   Reentrant
00266 * Parameters (in)   :   Channel-->Numeric identifier of the ICU channel
00267                         Activation-->Type of activation
00268 * Parameters (inout):   None
00269 * Parameters (out)  :   None
00270 * Return value      :   None
00271 * Description       :   This function sets the activation-edge for the given channel.
00272 ************************************************************************/
00273 void Icu_SetActivationCondition (Icu_ChannelType Channel,Icu_ActivationType Activation);
00274 /************************************************************************
00275 * Service ID [hex]  :   0x06
00276 * Service Name      :   Icu_DisableNotification
00277 * Sync/Async        :   Synchronous
00278 * Reentrancy        :   Reentrant
00279 * Parameters (in)   :   Channel
00280                         Numeric identifier of the ICU channe
00281 * Parameters (inout):   None
00282 * Parameters (out)  :   None
00283 * Return value      :   None
00284 * Description       :   This function disables the notification of a channel.
00285 ************************************************************************/
00286 void Icu_DisableNotification (Icu_ChannelType Channel);
00287 /************************************************************************
00288 * Service ID [hex]  :   0x07
00289 * Service Name      :   Icu_EnableNotification
00290 * Sync/Async        :   Synchronous
00291 * Reentrancy        :   Reentrant
00292 * Parameters (in)   :   Channel
00293                         Numeric identifier of the ICU channe
00294 * Parameters (inout):   None
00295 * Parameters (out)  :   None
00296 * Return value      :   None
00297 * Description       :   This function enables the notification on the given channel.
00298 ************************************************************************/
00299 void Icu_EnableNotification (Icu_ChannelType Channel);
00300 /************************************************************************
00301 * Service ID [hex]  :   0x02
00302 * Service Name      :   Icu_GetInputState
00303 * Sync/Async        :   Synchronous
00304 * Reentrancy        :   Reentrant
00305 * Parameters (in)   :   Channel
00306                         Numeric identifier of the ICU channel
00307 * Parameters (inout):   None
00308 * Parameters (out)  :   None
00309 * Return value      :   Icu_Input StateType
00310                         ICU_ACTIVE or ICU_IDLE
00311 * Description       :    This function returns the status of the ICU input.
00312 ************************************************************************/
00313 Icu_InputStateType Icu_GetInputState (Icu_ChannelType Channel);
00314 /************************************************************************
00315 * Service ID [hex]  :   0x13
00316 * Service Name      :   Icu_StartSignalMeasurement
00317 * Sync/Async        :   Synchronous
00318 * Reentrancy        :   Reentrant
00319 * Parameters (in)   :   Channel
00320                         Numeric identifier of the ICU channel
00321 * Parameters (inout):   None
00322 * Parameters (out)  :   None
00323 * Return value      :   None
00324 * Description       :   This function starts the measurement of signals.
00325 ************************************************************************/
00326 void Icu_StartSignalMeasurement (Icu_ChannelType Channel);
00327 /************************************************************************
00328 * Service ID [hex]  :   0x14
00329 * Service Name      :   Icu_StopSignalMeasurement
00330 * Sync/Async        :   Synchronous
00331 * Reentrancy        :   Reentrant
```

```
00332 * Parameters (in)    :   Channel
00333                         Numeric identifier of the ICU channel
00334 * Parameters (inout):   None
00335 * Parameters (out)   :   None
00336 * Return value       :   None
00337 * Description        :   This function stops the measurement of signals of the given channel.
00338 ************************************************************************/
00339 void Icu_StopSignalMeasurement (Icu_ChannelType Channel);
00340 /************************************************************************
00341 * Service ID [hex]   :   0x10
00342 * Service Name       :   Icu_GetTimeElapsed
00343 * Sync/Async         :   Synchronous
00344 * Reentrancy         :   Reentrant
00345 * Parameters (in)    :   Channel
00346                         Numeric identifier of the ICU channel
00347 * Parameters (inout):   None
00348 * Parameters (out)   :   None
00349 * Return value       :   see Description
00350 * Description        :   This function reads the elapsed Signal Low Time for the given channel.
00351 ************************************************************************/
00352 Icu_ValueType Icu_GetTimeElapsed (Icu_ChannelType Channel);
00353 /************************************************************************
00354 * Service ID [hex]   :   0x11
00355 * Service Name       :   Icu_GetDutyCycleValues
00356 * Sync/Async         :   Synchronous
00357 * Reentrancy         :   Reentrant
00358 * Parameters (in)    :   Channel
00359                         Numeric identifier of the ICU channel
00360 * Parameters (inout):   None
00361 * Parameters (out)   :   DutyCycleValues
00362                         Pointer to a buffer where the results
00363 * Return value       :   None
00364 * Description        :   This function stops the measurement of signals of the given channel.
00365 ************************************************************************/
00366 void Icu_GetDutyCycleValues (Icu_ChannelType Channel,Icu_DutyCycleType* DutyCycleValues);
00367
00368 #endif /* ICU_H_ */
```

4.22 Icu_Cfg.h

```
00001 /******************************************************************************
00002 * @Module      :     ICU
00003 * @File Name   :     Icu._Cfgh
00004 * @Description :    the Pre-compile configuration of the AUTOSAR Basic Software module PWM Driver.
00005 * Author       :    Salama Mohamed
00006 ******************************************************************************/
00007 /******************************************************************************
00008 * Project               : Graduation_Project_2024
00009 * Platform              : STM32F103C8
00010 * Autosar Version       : 4.8.0
00011 * SW Version            : 1.0.0
00012 ******************************************************************************/
00013 #ifndef ICU_CFG_H_
00014 #define ICU_CFG_H_
00015 /******************************************************************************
00016                              Includes
00017 ******************************************************************************/
00018 #include "Std_Types.h"
00019 /******************************************************************************
00020            Pre-compile configuration parameters of the Icu driver.
00021 ******************************************************************************/
00022 #define IcuMax_Channel                  (uint8)4
00023 #define Icu_Channel_0                   (uint8)0
00024 #define Icu_Channel_1                   (uint8)1
00025 #define Icu_Channel_2                   (uint8)2
00026 #define Icu_Channel_3                   (uint8)3
00027 //Switches the development error detection and notification on or off.
00028 #define IcuDevErrorDetect               TRUE
00029 //Switch for enabling Wakeup source reporting.
00030 #define IcuReportWakeupSource           FALSE
00031 //Adds / removes the service Icu_DeInit() from the code.
00032 #define IcuDeInitApi                    TRUE
00033 //Adds / removes the service Icu_DisableWakeup() from the code.
00034 #define IcuDisableWakeupApi             FALSE
00035 //Adds / removes all services related to the edge counting functionality from the code.
00036 #define IcuEdgeCountApi                 FALSE
00037 //Adds / removes the services related to the edge detection functionality, from the code:
00038 #define IcuEdgeDetectApi                TRUE
00039 //Adds / removes the service Icu_EnableWakeup() from the code.
00040 #define IcuEnableWakeupApi              FALSE
00041 //Adds / removes the service Icu_GetDutyCycleValues() from the code.
00042 #define IcuGetDutyCycleValuesApi        TRUE
00043 //Adds / removes the service Icu_GetInputState() from the code.
```

```
00044 #define IcuGetInputStateApi             TRUE
00045 //Adds / removes the service Icu_GetTimeElapsed() from the code.
00046 #define IcuGetTimeElapsedApi            TRUE
00047 //Adds / removes the service Icu_GetVersionInfo() from the code
00048 #define IcuGetVersionInfoApi            FALSE
00049 //Adds / removes the service Icu_SetMode() from the code.
00050 #define IcuSetModeApi                   TRUE
00051 //Adds / removes the services Icu_StartSignalMeasurement() and Icu_StopSignal Measurement() from the
      code.
00052 #define IcuSignalMeasurementApi         TRUE
00053 //Adds / removes all services related to the timestamping functionality from the code.
00054 #define IcuTimestampApi                 TRUE
00055 //Adds / removes the service Icu_CheckWakeup() from the code.
00056 #define IcuWakeupFunctionalityApi       FALSE
00057 #endif /* ICU_CFG_H_ */
```

4.23 Cortex_Mx_Porting.h

```
00001 /************************************************
00002  * File Name   : Cortex_Mx_Porting.h
00003  *  Created on : 20/11/2023
00004  *  Author     : Salama mohamed
00005  ************************************************/
00006 #ifndef INC_CORTEX_MX_PORTING_H_
00007 #define INC_CORTEX_MX_PORTING_H_
00008
00009 #include "core_cm3.h"
00010
00011 extern int _estack ;
00012 extern int _eheap  ;
00013 #define MainStackSize   3072
00014
00015 #define OS_SET_PSP(TOP_Add)     __asm("MOV R0,%[in] \t\n MSR PSP,R0"  : :[in] "r" (TOP_Add))
00016
00017 #define OS_GET_PSP(TOP_Add)     __asm("MRS R0,PSP \t\n MOV %[OUT],R0" :[OUT] "=r" (TOP_Add))
00018
00019 #define OS_GET_IRQ_Flag(Flag)     __asm("MRS R0,IPSR \t\n MOV %[OUT],R0" :[OUT] "=r" (Flag))
00020
00021 #define OS_Switch_SP_PSP        __asm("MRS R0,CONTROL \t\n ORR R0,R0,#0x2 \t\n MSR CONTROL,R0")
00022
00023 #define OS_Switch_SP_MSP        __asm("MRS R0,CONTROL \t\n AND R0,R0,#0x5 \t\n MSR CONTROL,R0")
00024
00025 #define CPU_Access_Level_Unprivileged()     { __asm("MRS R3,CONTROL"); \
00026                                               __asm("ORR R3,R3,#0x01"); \
00027                                               __asm("MSR CONTROL,R3"); \
00028                                              }
00029
00030 #define CPU_Access_Level_Privileged()        {__asm("MRS R3,CONTROL"); \
00031                                               __asm("LSR R3,R3,#0x01"); \
00032                                               __asm("LSL R3,R3,#0x01"); \
00033                                               __asm("MSR CONTROL,R3"); \
00034                                              }
00035
00036 void HW_init();
00037 void trigger_OS_PendSV();
00038 void Start_Ticker();
00039 void HardFault_Handler(void);
00040
00041 #endif /* INC_CORTEX_MX_PORTING_H_ */
```

4.24 Event.h

```
00001 /****************************************
00002  * File Name   : Event.h
00003  * Created on  : 28/11/2023
00004  * Author     : Salama mohamed
00005  ****************************************/
00006 #ifndef INC_EVENT_H_
00007 #define INC_EVENT_H_
00008
00009 #include "Type.h"
00010
00011 StatusType SetEvent(TaskRefType* TaskName,EventMaskType Mask);
00012 StatusType ClearEvent(EventMaskType Mask);
00013 StatusType GetEvent(TaskRefType* TaskName,EventMaskType* Event);
00014 StatusType WaitEvent(EventMaskType Mask);
00015 #endif /* INC_EVENT_H_ */
```

4.25 MY_RTOS_FIFO.h

```
00001 /*****************************************
00002  * File Name   : MY_RTOS_FIFO.h
00003  * Created on : 27/11/2023
00004  * Author     : Salama mohamed
00005  *****************************************/
00006 #ifndef INC_MY_RTOS_FIFO_H_
00007 #define INC_MY_RTOS_FIFO_H_
00008
00009 #include "stdio.h"
00010 #include"stdint.h"
00011 #include "Scheduler.h"
00012 /*customer can select element type */
00013 #define element_type TaskRefType*
00014 typedef struct{
00015     unsigned int counter;
00016     element_type* head;
00017     element_type* tail;
00018     element_type* base;
00019     unsigned int  length;
00020 }FIFO_Buf_t;
00021
00022 typedef enum{
00023     FIFO_NO_ERROR,
00024     FIFO_FULL,
00025     FIFO_EMPTY,
00026     FIFO_NULL,
00027     FIFO_NOT_FULL
00028
00029 }Buffer_status;
00030
00031 /*APIs*/
00032
00033 Buffer_status FIFO_init (FIFO_Buf_t* fifo,element_type* buff , unsigned int length);
00034 Buffer_status FIFO_enqueue (FIFO_Buf_t* fifo,element_type item);
00035 Buffer_status FIFO_dequeue (FIFO_Buf_t* fifo,element_type* item);
00036 Buffer_status FIFO_is_full (FIFO_Buf_t* fifo);
00037 void FIFO_print (FIFO_Buf_t* fifo);
00038
00039 #endif /* INC_MY_RTOS_FIFO_H_ */
```

4.26 Scheduler.h

```
00001 /*********************************************
00002  * File Name   : Scheduler.H
00003  *  Created on : 20/11/2023
00004  *  Author     : Salama mohamed
00005  *********************************************/
00006 #ifndef INC_SCHEDULER_H_
00007 #define INC_SCHEDULER_H_
00008 #include "Type.h"
00009
00010 #include "Cortex_Mx_Porting.h"
00011
00012 typedef struct
00013 {
00014     uint16_t* Ppayload                  ;
00015     unsigned int   PayloadSize              ;
00016     TaskRefType*        CurrentTUser            ;//Not Entered by the user
00017     TaskRefType*        NextTUser               ;//Not Entered by the user
00018     char           MutexName[30]            ;
00019     uint8_t         priority_Inversion      ;//Not Entered by the user
00020 } Mutex_Ref;
00021
00022
00023 typedef struct
00024 {
00025     unsigned char* Ppayload                 ;
00026     TaskRefType*        CurrentTUser            ;//Not Entered by the user
00027     TaskRefType*        NextTUser               ;//Not Entered by the user
00028     char           SemaphoreName[30]        ;
00029     uint8_t         state                   ;//Not Entered by the user
00030 } Semaphore_Ref;
00031
00032
00033
00034 /*
00035  * ************************************************************
00036  *                       APIS
00037  ************************************************************
00038  */
00039
```

```
00040 MY_RTOS_ErrorID MYRTOS_Init();
00041 void MYRTOS_CreateTask(TaskRefType* Tref);
00042 void MYRTOS_ActivateTask (TaskRefType* Tref);
00043 void MYRTOS_TerminateTask (TaskRefType* Tref);
00044 void MYRTOS_STARTOS();
00045 void MYRTOS_TaskWait(unsigned int Num_Tick,TaskRefType* Tref);
00046 MY_RTOS_ErrorID MYRTOS_AcquireMutex(Mutex_Ref* Mref , TaskRefType* Tref);
00047 void MYRTOS_ReleaseMutex(Mutex_Ref* Mref , TaskRefType* Tref);
00048 MY_RTOS_ErrorID MYRTOS_AcquireSemaphore(Semaphore_Ref* Sref ,TaskRefType* Tref);
00049 void MYRTOS_ReleaseSemaphore(Semaphore_Ref* Sref);
00050
00051 #endif /* INC_SCHEDULER_H_ */
```

4.27 Task.h

```
00001 /**********************************************
00002  * File Name    : Task.h
00003  *  Created on : 20/11/2023
00004  *  Author     : Salama mohamed
00005  **********************************************/
00006 #ifndef INC_TASK_H_
00007 #define INC_TASK_H_
00008
00009 #include "Type.h"
00010
00011 typedef uint8_t TaskType ;
00012
00013 #define MaxTaskId   20
00014
00015
00016
00017 StatusType TerminateTask(void);
00018 StatusType ActivateTask(TaskRefType* TaskName);
00019 StatusType ChainTask(TaskRefType* TaskName);
00020 StatusType Schedule(void);
00021 StatusType GetTaskId(TaskRefType* TaskName);
00022 StatusType GetTaskState(TaskRefType TaskName,TaskStateType State);
00023 #endif /* INC_TASK_H_ */
```

4.28 Task_Config.h

```
00001 /**********************************************
00002  * File Name    : Task_Config.h
00003  *  Created on : 27/11/2023
00004  *  Author     : Salama mohamed
00005  **********************************************/
00006 #ifndef INC_TASK_CONFIG_H_
00007 #define INC_TASK_CONFIG_H_
00008
00009 #include "Type.h"
00010 #include "Scheduler.h"
00011 #include "Task.h"
00012 #include "Task_Config.h"
00013 #include "Event.h"
00014
00015 //To refer to a task the constructional element should be used to declare the task before references
      to it
00016 #define DeclareTask(TaskName)   TaskRefType TaskName
00017 //
00018 #define DefineTask(_TaskName, _Stack_Size,_AutoStart,_priority,_TaskSchedlerType,_TaskType) \
00019     { \
00020         _TaskName.P_TaskEntry = _TaskName##_Entry; \
00021         _TaskName.Stack_Size = _Stack_Size; \
00022         _TaskName.AutoStart = _AutoStart; \
00023         strcpy(_TaskName.Task_Name,#_TaskName); \
00024         _TaskName.priority = _priority; \
00025         _TaskName.TaskSchedlerType = _TaskSchedlerType; \
00026         MYRTOS_CreateTask(&_TaskName);\
00027         _TaskName.TaskType=_TaskType;\
00028     }
00029 //
00030 #define TASK(TaskName) void TaskName##_Entry(void)
00031 /*
00032  Various requirements of the application software for the system and various capabilities of a
00033  specific system
00034  */
00035 //only basic tasks, limited to one activation request per task and one task per priority
00036 #define BCC1    0
```

```
00037 //like BCC1, plus more than one task per priority possible and multiple requesting of task activation
      allowed
00038 #define BCC2    1
00039 //like BCC1, plus extended tasks
00040 #define ECC1    2
00041 //like ECC1, plus more than one task per priority possible and multiple requesting of task activation
      allowed for basic tasks
00042 #define ECC2    3
00043 #define Conformance_Classe ECC1
00044 void Os_Init(void);
00045 #endif /* INC_TASK_CONFIG_H_ */
```

4.29 Type.h

```
00001 /***********************************************
00002  * File Name   : Type.h
00003  *  Created on : 23/11/2023
00004  *  Author     : Salama mohamed
00005  ***********************************************/
00006 #ifndef INC_TYPE_H_
00007 #define INC_TYPE_H_
00008
00009 #include <stdint.h>
00010 #include <stdlib.h>
00011 #include "Std_Types.h"
00012
00013
00014 typedef enum
00015 {
00016     OSsuspend,
00017     OsRunning
00018 }OSmode;
00019
00020 typedef enum
00021 {
00022     // no error
00023     //E_OK,
00024     // the task has already been activated and multiple requests are not allowed
00025     E_STATE,
00026     //the task identifier TaskName is invalid
00027     E_ID,
00028     //a call at the interrupt level
00029     E_CALLEVEL,
00030     // the task still occupies resources
00031     E_RESOURCE,
00032     // the referenced task is not an Extended Task
00033     E_ACCESS
00034 }StatusType;
00035
00036 typedef enum {
00037     SVC_Activatetask,
00038     SVC_terminateTask,
00039     SVC_TaskWaitingTime,
00040     SVC_AquireMutex,
00041     SVC_ReleaseMutex
00042 }SVC_ID;
00043 //Each ET has a definite number of events - 8 or less
00044 typedef uint8_t EventMaskType;
00045
00046 typedef enum
00047 {
00048     NO_ERROR,
00049     Ready_Queue_Init_Error,
00050     Task_exceeded_StackSize,
00051     SO_Table_Sort_Error,
00052     MutexisReacedToMaxNumberOfUsers
00053 }MY_RTOS_ErrorID;
00054
00055 typedef enum
00056 {
00057     Suspend,
00058     Running,
00059     Waiting,
00060     Ready
00061 }TaskStateType;
00062
00063 typedef enum
00064 {
00065     Basic_Task,
00066     Extended_Task
00067 }Task_Type;
00068
00069 typedef enum
```

```
00070 {
00071   NONE_PREEMPTIVE,
00072   FULL_PREEMPTIVE
00073 }Tasks_Scheduler_Type;
00074
00075 typedef enum
00076 {
00077     Task_Suspend,
00078     Task_Start
00079 }Auto_Start ;
00080
00081 typedef struct
00082 {
00083     uint32_t Stack_Size                         ;
00084     uint8_t priority                            ;
00085     void (*P_TaskEntry)(void)                   ; //pointer to Tack C Function
00086     uint32_t _S_PSP_Task                        ;//Not Entered by the user
00087     uint32_t _E_PSP_Task                        ;//Not Entered by the user
00088     uint32_t* Current_PSP                       ;//Not Entered by the user
00089     int8_t Task_Name[30]                        ;
00090     TaskStateType TaskState                     ;//Not Entered by the user
00091     Task_Type   TaskType                        ;
00092     Auto_Start AutoStart                        ;
00093     Tasks_Scheduler_Type  TaskSchedlerType      ;
00094     struct
00095     {
00096         enum
00097         {
00098             disable,
00099             enable
00100         }Blocking;
00101         uint32_t Ticks_Count ;
00102     }Timing_Waiting;
00103     uint8_t MultipleActivation;
00104     struct
00105     {
00106         EventMaskType Public_Mask;
00107         EventMaskType Private_Mask;
00108     }Events;
00109 }TaskRefType;
00110
00111 typedef enum
00112 {
00113     TASK_LEVEL,
00114     ISR_CAT1,
00115     ISR_CAT2,
00116     HOOK
00117 }OS_level;
00118
00119 struct System_Conctrol
00120 {
00121     TaskRefType* OSTasks[100]        ; //Scheduler Table
00122     unsigned int _S_MSP_Task        ;
00123     unsigned int _E_MSP_Task        ;
00124     unsigned int PSP_Task_Locator   ;
00125     unsigned int NoOfActiveTasks    ;
00126     TaskRefType* CurrentTask         ;
00127     TaskRefType* NextTask            ;
00128     OSmode OSmodeID                 ;
00129     OS_level Call_Leve              ;
00130 };
00131
00132 void OSEK_SVC(SVC_ID ID);
00133
00134 #endif /* INC_TYPE_H_ */
```

4.30 Port.h

```
00001 /******************************************************************************
00002 * @Module      :     Port
00003 * @File Name   :     Port.h
00004 * @Description :    This specification specifies the functionality, API and the configuration of the
      AUTOSAR
00005                     Basic Software module PORT Driver.
00006 * Author       :    Salama Mohamed
00007 ******************************************************************************/
00008 /******************************************************************************
00009 * Project               : Graduation_Project_2024
00010 * Platform              : STM32F103C8
00011 * Autosar Version       : 4.8.0
00012 * SW Version            : 1.0.0
00013 ******************************************************************************/
00014 #ifndef PORT_H_
```

```
00015 #define PORT_H_
00016 /*******************************************************************************
00017                                 Source File Version Informations
00018 *******************************************************************************/
00019 #define PORT_VERSION_ID                         40
00020 #define PORT_AR_RELEASE_MAJOR_VERSION           4
00021 #define PORT_AR_RELEASE_MINOR_VERSION           8
00022 #define PORT_AR_RELEASE_PATCH_VERSION           0
00023 #define PORT_SW_RELEASE_MAJOR_VERSION           1
00024 #define PORT_SW_RELEASE_MINOR_VERSION           0
00025 #define PORT_SW_RELEASE_PATCH_VERSION           0
00026 #define VENDOR_ID                               100
00027 /*******************************************************************************
00028                                 Includes
00029 *******************************************************************************/
00030 #include "Port_Cfg.h"
00031 #include "Det.h"
00032 #include "Std_Types.h"
00033 // AUTOSAR checking Std_Version
00034 #if ((STD_TYPES_AR_RELEASE_MAJOR_VERSION != PORT_AR_RELEASE_MAJOR_VERSION)\
00035 ||  (STD_TYPES_AR_RELEASE_MINOR_VERSION  != PORT_AR_RELEASE_MINOR_VERSION)\
00036 ||  (STD_TYPES_AR_RELEASE_PATCH_VERSION  != PORT_AR_RELEASE_PATCH_VERSION))
00037 #error "The Autosar version of Std_Types.h does not match the Port version"
00038 #endif
00039 /*******************************************************************************
00040                         API Service Id Macros
00041  *******************************************************************************/
00042 // Service ID for Port_Init
00043 #define PORT_INIT_ID                        (uint8)0x00
00044 // Service ID for Port_SetPinDirection
00045 #define PORT_SET_PIN_DIR_ID                 (uint8)0x01
00046 // Service ID for Port_RefreshPortDirection
00047 #define PORT_REFRESH_PORT_DIR_ID            (uint8)0x02
00048 // Service ID for Port_GetVersionInfo
00049 #define PORT_GET_VERSION_INFO_ID            (uint8)0x03
00050 // Service ID for Port_SetPinMode
00051 #define PORT_SET_PIN_MODE_ID                (uint8)0x04
00052 /*******************************************************************************
00053                         DET Error Codes
00054  *******************************************************************************/
00055 // Invalid Port Pin ID requested
00056 #define PORT_E_PARAM_PIN                    (uint8)0x0A
00057 // Port Pin not configured as changeable
00058 #define PORT_E_DIRECTION_UNCHANGEABLE       (uint8)0x0B
00059 // API Port_Init service called with wrong parameter
00060 #define PORT_E_PARAM_CONFIG                 (uint8)0x0C
00061 // API Port_SetPinMode service called when Port Pin Mode passed not valid
00062 #define PORT_E_PARAM_INVALID_MODE           (uint8)0x0D
00063 // API Port_SetPinMode service called when mode is unchangeable
00064 #define PORT_E_MODE_UNCHANGEABLE            (uint8)0x0E
00065 // API service called without module initialization
00066 #define PORT_E_UNINIT                       (uint8)0x0F
00067 // APIs called with a Null Pointer
00068 #define PORT_E_PARAM_POINTER                (uint8)0x10
00069 /*******************************************************************************
00070                             Type definitions
00071 *******************************************************************************/
00072 #define PORT_NOT_INITIALIZED    0U
00073 #define PORT_INITIALIZED        1U
00074 /*
00075  Shall cover all available port pins.
00076  The type should be chosen for the
00077  specific MCU platform (best performance).
00078 */
00079 typedef  uint8  Port_PinType   ;
00080 //Possible directions of a port pin.
00081 typedef enum
00082 {
00083     //Sets port pin as input.
00084     PORT_PIN_IN,
00085     //Sets port pin as output.
00086     PORT_PIN_OUT
00087 } Port_PinDirectionType ;
00088 /*
00089  As several port pin modes shall
00090  be configurable on one pin, the
00091  range shall be determined by the
00092  implementation.
00093  */
00094 typedef enum
00095 {
00096     PORT_PIN_MODE_ADC,
00097     PORT_PIN_MODE_CAN,
00098     PORT_PIN_MODE_DIO,
00099     PORT_PIN_MODE_PWM
00100 }Port_PinModeType;
00101 /*******************************************************************************
```

```
00102 Name            :               PortPin
00103 Parent Container:               PortContainer
00104 Description     :               Configuration of the individual port pins.
00105 Type            :               Container
00106 *******************************************************************************/
00107 typedef struct
00108 {
00109     /*The initial direction of the pin (IN or OUT). If the direction is not changeable,
00110      the value configured here is fixed.
00111      The direction must match the pin mode. E.g. a pin used for an ADC must be configured
00112      to be an in port.
00113     */
00114     Port_PinDirectionType PortPinDirection ;
00115     /*
00116      Parameter to indicate if the direction is changeable on a port pin during runtime. true:
00117      Port Pin direction changeable enabled. false: Port Pin direction changeable disabled.
00118     */
00119     boolean PortPinDirectionChangeable;
00120     /*
00121      Pin Id of the port pin. This value will be assigned to the symbolic name
00122      derived from the port pin container short name.
00123     */
00124     Port_PinType PortPinId;
00125     // Port pin mode from mode list for use with Port_Init() function
00126     Port_PinModeType PortPinInitialMode;
00127     // Port Pin Level value from Port pin list.
00128     uint8 PortPinLevelValue;
00129     // Port pin mode from mode list. Note that more than one mode is allowed by default.
00130     Port_PinModeType PortPinMode ;
00131     /*
00132      Parameter to indicate if the mode is changeable on a port pin during runtime. True:
00133      Port Pin mode changeable allowed. False: Port Pin mode changeable not permitted.
00134     */
00135     boolean PortPinModeChangeable;
00136     // Activation of internal pull-ups
00137     boolean Pull_UP;
00138     // Slew rate contro
00139     uint8 Slew_Rate;
00140     // Pin driven mode (push-pull / open drain).
00141     uint8 Pin_Driven_Mode;
00142 } PortPin;
00143 /*******************************************************************************
00144 Name            :               PortContainer
00145 Parent Container:               PortConfigSet
00146 Description     :               Container collecting the PortPins.
00147 Type            :               Container
00148 *******************************************************************************/
00149 typedef struct
00150 {
00151     PortPin PortPin[PortNumberOfPortPins];
00152 } PortContainer;
00153 /*******************************************************************************
00154 Name            :               PortConfigSet
00155 Parent Container:               Port
00156 Description     :               This container contains the configuration parameters and sub
00157                                 containers of the AUTOSAR Port module
00158 Type            :               Container
00159 *******************************************************************************/
00160 typedef struct
00161 {
00162     PortContainer PortContainer;
00163 }PortConfigSet;
00164 /*******************************************************************************
00165 Name            :               Port
00166 Parent Container:               PORT
00167 Description     :               Configuration of the Port module.
00168 Type            :               Structure
00169 *******************************************************************************/
00170 typedef struct
00171 {
00172     PortPin PortPin[PortNumberOfPortPins];
00173 }Port_ConfigType;
00174 extern const Port_ConfigType Port;
00175 /*******************************************************************************
00176                               APIS
00177 *******************************************************************************/
00178 /*******************************************************************************
00179 * Service ID [hex]   :   0x00
00180 * Service Name       :   Port_Init
00181 * Sync/Async         :   Synchronous
00182 * Reentrancy         :   Non reentrant
00183 * Parameters (in)    :   ConfigPtr (Pointer to configuration set).
00184 * Parameters (inout):   None
00185 * Parameters (out)   :   None
00186 * Return value       :   None
00187 * Description        :   Initializes the Port Driver module.
00188 *******************************************************************************/
```

```
00189 void Port_Init( const Port_ConfigType* ConfigPtr ) ;
00190 /************************************************************************
00191 * Service ID [hex]  :   0x01
00192 * Service Name      :   Port_SetPinDirection
00193 * Sync/Async        :   Synchronous
00194 * Reentrancy        :   Reentrant
00195 * Parameters (in)   :   Pin (Port Pin ID number)
00196                         Direction (Port Pin Direction)
00197 * Parameters (inout):   None
00198 * Parameters (out)  :   None
00199 * Return value      :   None
00200 * Description       :   Sets the port pin direction
00201 ************************************************************************/
00202 void Port_SetPinDirection ( Port_PinType Pin , Port_PinDirectionType Direction ) ;
00203 /************************************************************************
00204 * Service ID [hex]  :   0x02
00205 * Service Name      :   Port_RefreshPortDirection
00206 * Sync/Async        :   Synchronous
00207 * Reentrancy        :   Non Reentrant
00208 * Parameters (in)   :   None
00209 * Parameters (inout):   None
00210 * Parameters (out)  :   None
00211 * Return value      :   None
00212 * Description       :   Refreshes port direction
00213 ************************************************************************/
00214 void Port_RefreshPortDirection (void);
00215 /************************************************************************
00216 * Service ID [hex]  :   0x03
00217 * Service Name      :   Port_GetVersionInfo
00218 * Sync/Async        :   Synchronous
00219 * Reentrancy        :   Reentrant
00220 * Parameters (in)   :   None
00221 * Parameters (inout):   None
00222 * Parameters (out)  :   versioninfo (Pointer to where to store the version information of this
      module).
00223 * Return value      :   None
00224 * Description       :   Returns the version information of this module.
00225 ************************************************************************/
00226 void Port_GetVersionInfo ( Std_VersionInfoType* versioninfo ) ;
00227 /************************************************************************
00228 * Service ID [hex]  :   0x04
00229 * Service Name      :   Port_SetPinMode
00230 * Sync/Async        :   Synchronous
00231 * Reentrancy        :   Reentrant
00232 * Parameters (in)   :   Pin (Port Pin ID number)
00233                         Mode (New Port Pin mode to be set on port pin).
00234 * Parameters (inout):   None
00235 * Parameters (out)  :   None
00236 * Return value      :   None
00237 * Description       :   Mode New Port Pin mode to be set on port pin.
00238 ************************************************************************/
00239 void Port_SetPinMode ( Port_PinType Pin , Port_PinModeType Mode ) ;
00240 #endif /* PORT_H_ */
```

4.31 Port_Cfg.h

```
00001 /******************************************************************************
00002 * @Module      :     Port
00003 * @File Name   :     Port_Cfg.h
00004 * @Description :     the Pre-compile configuration of the AUTOSAR Basic Software module PORT Driver.
00005 * Author       :     Salama Mohamed
00006 ******************************************************************************/
00007 /*****************************************************************************
00008 * Project               : Graduation_Project_2024
00009 * Platform              : STM32F103C8
00010 * Autosar Version       : 4.8.0
00011 * SW Version            : 1.0.0
00012 *****************************************************************************/
00013 #ifndef PORT_CFG_H_
00014 #define PORT_CFG_H_
00015 /*****************************************************************************
00016                            Source File Version Informations
00017 *****************************************************************************/
00018 #define PORT_VERSION_ID                         40
00019 #define PORT_CFG_AR_RELEASE_MAJOR_VERSION       4
00020 #define PORT_CFG_AR_RELEASE_MINOR_VERSION       8
00021 #define PORT_CFG_AR_RELEASE_PATCH_VERSION       0
00022 #define PORT_CFG_SW_RELEASE_MAJOR_VERSION       1
00023 #define PORT_CFG_SW_RELEASE_MINOR_VERSION       0
00024 #define PORT_CFG_SW_RELEASE_PATCH_VERSION       0
00025 #define VENDOR_ID                               100
00026 /*****************************************************************************
00027            Module wide configuration parameters of the PORT driver.
```

```
00028 ********************************************************************************/
00029 //The number of specified PortPins in this PortContainer.
00030 #define PortNumberOfPortPins    32
00031 /*
00032  Switches the development error detection and notification on or off.
00033  true: detection and notification is enabled.
00034  false: detection and notification is disabled.
00035 */
00036 #define PortDevErrorDetect          TRUE
00037 /*
00038  Pre-processor switch to enable / disable the use of the function Port_SetPinDirection().
00039 */
00040 #define PortSetPinDirectionApi      TRUE
00041 /*
00042  Pre-processor switch to enable / disable the use of the function Port_SetPinMode().
00043 */
00044 #define PortSetPinModeApi           TRUE
00045 /*
00046  Pre-processor switch to enable / disable the API to read out the modules version information.
00047 */
00048 #define PortVersionInfoApi          TRUE
00049 #define GPIO_Pin_Driven_Mode_PP     0
00050 #define GPIO_Pin_Driven_Mode_OD     1
00051 #define GPIO_Slew_Rate_10M          1
00052 #define GPIO_Slew_Rate_2M           2
00053 #define GPIO_Slew_Rate_50M          3
00054 #define PORT_INSTANCE_ZERO          0
00055 /*
00056  [SWS_Port_00207] dThese symbolic names for the individual port pins (e.g. PORT_
00057  A_PIN_0) shall be defined in the configuration tool
00058  */
00059 #define  PORT_A_PIN_0       ((Port_PinType)1)
00060 #define  PORT_A_PIN_1       ((Port_PinType)2)
00061 #define  PORT_A_PIN_2       ((Port_PinType)3)
00062 #define  PORT_A_PIN_3       ((Port_PinType)4)
00063 #define  PORT_A_PIN_4       ((Port_PinType)5)
00064 #define  PORT_A_PIN_5       ((Port_PinType)6)
00065 #define  PORT_A_PIN_6       ((Port_PinType)7)
00066 #define  PORT_A_PIN_7       ((Port_PinType)8)
00067 #define  PORT_A_PIN_8       ((Port_PinType)9)
00068 #define  PORT_A_PIN_9       ((Port_PinType)10)
00069 #define  PORT_A_PIN_10      ((Port_PinType)11)
00070 #define  PORT_A_PIN_11      ((Port_PinType)12)
00071 #define  PORT_A_PIN_12      ((Port_PinType)13)
00072 #define  PORT_A_PIN_13      ((Port_PinType)14)
00073 #define  PORT_A_PIN_14      ((Port_PinType)15)
00074 #define  PORT_A_PIN_15      ((Port_PinType)16)
00075 #define  PORT_B_PIN_0       ((Port_PinType)17)
00076 #define  PORT_B_PIN_1       ((Port_PinType)18)
00077 #define  PORT_B_PIN_2       ((Port_PinType)19)
00078 #define  PORT_B_PIN_3       ((Port_PinType)20)
00079 #define  PORT_B_PIN_4       ((Port_PinType)21)
00080 #define  PORT_B_PIN_5       ((Port_PinType)22)
00081 #define  PORT_B_PIN_6       ((Port_PinType)23)
00082 #define  PORT_B_PIN_7       ((Port_PinType)24)
00083 #define  PORT_B_PIN_8       ((Port_PinType)25)
00084 #define  PORT_B_PIN_9       ((Port_PinType)26)
00085 #define  PORT_B_PIN_10      ((Port_PinType)27)
00086 #define  PORT_B_PIN_11      ((Port_PinType)28)
00087 #define  PORT_B_PIN_12      ((Port_PinType)29)
00088 #define  PORT_B_PIN_13      ((Port_PinType)30)
00089 #define  PORT_B_PIN_14      ((Port_PinType)31)
00090 #define  PORT_B_PIN_15      ((Port_PinType)32)
00091 #define  PORT_C_PIN_13      ((Port_PinType)46)
00092 #define  PORT_C_PIN_14      ((Port_PinType)47)
00093 #define  PORT_C_PIN_15      ((Port_PinType)48)
00094 #endif /* PORT_CFG_H_ */
```

4.32 Pwm.h

```
00001 /******************************************************************************
00002 * @Module      :     PWM
00003 * @File Name   :     Pwm.h
00004 * @Description :    This specification specifies the functionality, API and the configuration of the
00005                      AUTOSAR Basic Software module PWM driver.
00006 * Author       :    Salama Mohamed
00007 ******************************************************************************/
00008 /*******************************************************************************
00009 * Project               : Graduation_Project_2024
00010 * Platform              : STM32F103C8
00011 * Autosar Version       : 4.8.0
00012 * SW Version            : 1.0.0
00013 ********************************************************************************/
```

```
00014 #ifndef PWM_H_
00015 #define PWM_H_
00016 /*******************************************************************************
00017                             Source File Version Informations
00018 *******************************************************************************/
00019 #define PWM_VERSION_ID                          37
00020 #define VENDOR_ID                               100
00021 #define PWM_AR_RELEASE_MAJOR_VERSION            4
00022 #define PWM_AR_RELEASE_MINOR_VERSION            8
00023 #define PWM_AR_RELEASE_PATCH_VERSION            0
00024 #define PWM_SW_RELEASE_MAJOR_VERSION            1
00025 #define PWM_SW_RELEASE_MINOR_VERSION            0
00026 #define PWM_SW_RELEASE_PATCH_VERSION            0
00027 /*******************************************************************************
00028                             Includes
00029 *******************************************************************************/
00030 #include "Det.h"
00031 #include "Pwm_Cfg.h"
00032 #include "Std_Types.h"
00033 // AUTOSAR checking Std_Version
00034 #if ((STD_TYPES_AR_RELEASE_MAJOR_VERSION != PWM_AR_RELEASE_MAJOR_VERSION)\
00035 ||  (STD_TYPES_AR_RELEASE_MINOR_VERSION  != PWM_AR_RELEASE_MINOR_VERSION)\
00036 ||  (STD_TYPES_AR_RELEASE_PATCH_VERSION  != PWM_AR_RELEASE_PATCH_VERSION))
00037 #error "The Autosar version of Std_Types.h does not match the Port version"
00038 #endif
00039 /*******************************************************************************
00040                       API Service Id Macros
00041 *******************************************************************************/
00042 #define PWM_INIT_ID                                 (uint8)0x00
00043 #define PWM_DEINIT_ID                               (uint8)0x01
00044 #define PWM_SETDUTYCYCLE_ID                         (uint8)0x02
00045 #define PWM_SETPERIODANDDUTY_ID                     (uint8)0x03
00046 #define PWM_SETOUTPUTTOIDLE_ID                      (uint8)0x04
00047 #define PWM_GETOUTPUTSTATE_ID                       (uint8)0x05
00048 #define PWM_DISABLENOTIFICATION_ID                  (uint8)0x06
00049 #define PWM_ENABLENOTIFICATION_ID                   (uint8)0x07
00050 #define PWM_GETVERSIONINFO_ID                       (uint8)0x08
00051 #define PWM_SETPOWERSTATE_ID                        (uint8)0x09
00052 #define PWM_GETCURRENTPOWERSTATE_ID                 (uint8)0x0A
00053 /*******************************************************************************
00054                       DET Error Codes
00055 *******************************************************************************/
00056 //API Pwm_Init service called with wrong parameter
00057 #define PWM_E_INIT_FAILED                           (uint8)0x10
00058 //API service used without module initialization
00059 #define PWM_E_UNINIT                                (uint8)0x11
00060 //API service used with an invalid channel Identifier
00061 #define PWM_E_PARAM_CHANNEL                         (uint8)0x12
00062 //Usage of unauthorized PWM service on PWM channel configured a fixed period
00063 #define PWM_E_PERIOD_UNCHANGEABLE                   (uint8)0x13
00064 //API Pwm_Init service called while the PWM driver has already been initialised
00065 #define PWM_E_ALREADY_INITIALIZED                   (uint8)0x14
00066 //API Pwm_GetVersionInfo is called with a NULL parameter.
00067 #define PWM_E_PARAM_POINTER                         (uint8)0x15
00068 //API Pwm_SetPowerState is called while the PWM module is still in use.
00069 #define PWM_E_NOT_DISENGAGED                        (uint8)0x16
00070 //The requested power state is not supported by the PWM module.
00071 #define PWM_E_POWER_STATE_NOT_SUPPORTED             (uint8)0x17
00072 //The requested power state is not reachable from the current one
00073 #define PWM_E_TRANSITION_NOT_POSSIBLE               (uint8)0x18
00074 //API Pwm_SetPowerState has been called without having called the API Pwm_PreparePowerState before
00075 #define PWM_E_PERIPHERAL_NOT_PREPARED               (uint8)0x19
00076 /*******************************************************************************
00077                              Type definitions
00078 *******************************************************************************/
00079 #define PWM_NOT_INITIALIZED 0U
00080 #define PWM_INITIALIZED     1U
00081 #define Max_Num_CH  4
00082 //Numeric identifier of a PWM channel.
00083 typedef uint8 Pwm_ChannelType;
00084 // Definition of the period of a PWM channel.
00085 typedef float32 Pwm_PeriodType;
00086 //Output state of a PWM channel.
00087 typedef enum
00088 {
00089     //The PWM channel is in high state
00090     PWM_HIGH,
00091     //The PWM channel is in low state
00092     PWM_LOW
00093 }Pwm_OutputStateType;
00094 //Definition of the type of edge notification of a PWM channel.
00095 typedef enum
00096 {
00097     // Notification will be called when a rising edge occurs on the PWM output signal.
00098     PWM_RISING_EDGE,
00099     // Notification will be called when a falling edge occurs on the PWM output signal.
00100     PWM_FALLING_EDGE,
```

```
00101     //Notification will be called when either a rising edge or falling edge occur on the PWM output
      signal.
00102     PWM_BOTH_EDGES
00103 }Pwm_EdgeNotificationType;
00104 // Defines the class of a PWM channel
00105 typedef enum
00106 {
00107     // The PWM channel has a variable period. The duty cycle and the period can be changed.
00108     PWM_VARIABLE_PERIOD,
00109     // The PWM channel has a fixed period. Only the duty cycle can be changed.
00110     PWM_FIXED_PERIOD,
00111 }Pwm_ChannelClassType;
00112 //Result of the requests related to power state transitions.
00113 typedef enum
00114 {
00115     // Power state change executed.
00116     PWM_SERVICE_ACCEPTED,
00117     //PWM Module not initialized
00118     PWM_NOT_INIT,
00119     //Wrong API call sequence
00120     PWM_SEQUENCE_ERROR,
00121     // The HW module has a failure which prevents it to enter the required power state.
00122     PWM_HW_FAILURE,
00123     // PWM Module does not support the requested power state.
00124     PWM_POWER_STATE_NOT_SUPP,
00125     //PWM Module cannot transition directly from the current power state to the requested power state
      or the HW peripheral is still busy.
00126     PWM_TRANS_NOT_POSSIBLE
00127 }Pwm_PowerStateRequestResultType;
00128 // Power state currently active or set as target power state.
00129 typedef enum
00130 {
00131     // Full Power
00132     PWM_FULL_POWER,
00133     // Half Power
00134     PWM_HALF_POWER
00135 }Pwm_PowerStateType;
00136 /*******************************************************************************************
00137 Name            :               PwmChannel
00138 Parent Container:               PwmChannelConfigSet
00139 Description     :               Configuration of an individual PWM channel.
00140 Type            :               Container
00141 *******************************************************************************************/
00142 typedef struct
00143 {
00144     //Only the duty cycle can be changed or Duty Cycle and period can be changed.
00145     Pwm_ChannelClassType PwmChannelClass;
00146     //Channel Id of the PWM channel. This value will be assigned to the symbolic name derived of the
      PwmChannel container short name
00147     Pwm_ChannelType PwmChannelId;
00148     //Value of duty cycle used for Initialization 0 represents 0% 0x8000 represents 100%
00149     uint16 PwmDutycycleDefault;
00150     //The parameter PWM_IDLE_STATE represents the output state of the PWM after the signal is stopped
00151     Pwm_OutputStateType PwmIdleState;
00152     //Definition of the Callback function.Default value "NULL"
00153     void(*PwmNotification)(void);
00154     //Value of period used for Initialization(in seconds).
00155     Pwm_PeriodType PwmPeriodDefault;
00156     //Defines the starting polarity of each PWM channel.
00157     Pwm_OutputStateType PwmPolarity;
00158
00159 }PwmChannel;
00160 /*******************************************************************************************
00161 Name            :               PwmChannelConfigSet
00162 Parent Container:               PWM
00163 Description     :               This container contains the configuration parameters and
00164                                 sub containers of the AUTOSAR Pwm module.
00165 Type            :               Container
00166 *******************************************************************************************/
00167 typedef struct
00168 {
00169     PwmChannel Channel_Config[Max_Num_CH];
00170 }PwmChannelConfigSet;
00171 /*******************************************************************************************
00172 Name            :               Pwm_ConfigType
00173 Parent Container:               None
00174 Description     :               This is the type of data structure containing the initialization data
      for the PWM driver.
00175 Type            :               Structure
00176 *******************************************************************************************/
00177 typedef struct
00178 {
00179     PwmChannelConfigSet Config_Pwm;
00180 }Pwm_ConfigType;
00181 extern Pwm_ConfigType PWM_Config;
00182 /*************************************************************************
00183                               APIS
```

```
00184 ******************************************************************************/
00185 /*******************************************************************************
00186  * Service name          : Pwm_Init
00187  * Service ID[hex]       : 0x00
00188  * Sync/Async            : Synchronous
00189  * Reentrancy            : Non Reentrant
00190  * Parameters (in)       : ConfigPtr
00191                            Pointer to configuration set
00192  * Parameters (inout)    : None
00193  * Parameters (out)      : None
00194  * Return value          : None
00195  * Description           : Service for PWM initialization
00196  *******************************************************************************/
00197 void Pwm_Init (const Pwm_ConfigType* ConfigPtr);
00198 /*******************************************************************************
00199  * Service name          : Pwm_DeInit
00200  * Service ID[hex]       : 0x01
00201  * Sync/Async            : Synchronous
00202  * Reentrancy            : Non Reentrant
00203  * Parameters (in)       : None
00204  * Parameters (inout)    : None
00205  * Parameters (out)      : None
00206  * Return value          : None
00207  * Description           : Service for PWM De-Initialization.
00208  *******************************************************************************/
00209 void Pwm_DeInit (void);
00210 /*******************************************************************************
00211  * Service name          : Pwm_SetDutyCycle
00212  * Service ID[hex]       : 0x02
00213  * Sync/Async            : Asynchronous
00214  * Reentrancy            : Reentrant for different channel numbers
00215  * Parameters (in)       : ChannelNumber
00216                            Numeric identifier of the PWM
00217  *Parameters (in)        : DutyCycle
00218                            Min=0x0000 Max=0x8000
00219  *
00220  * Parameters (inout)    : None
00221  * Parameters (out)      : None
00222  * Return value          : None
00223  * Description           : Service sets the duty cycle of the PWM channel.
00224  *******************************************************************************/
00225 void Pwm_SetDutyCycle (Pwm_ChannelType ChannelNumber,uint16 DutyCycle);
00226 /*******************************************************************************
00227  * Service name          : Pwm_DeInit
00228  * Service ID[hex]       : 0x03
00229  * Sync/Async            : Asynchronous
00230  * Reentrancy            : Reentrant for different channel numbers
00231  * Parameters (in)       : ChannelNumber
00232                            Numeric identifier of the PWM
00233  * Parameters (in)       : Period
00234                            Period of the PWM signal
00235  * Parameters (in)       : DutyCycle
00236                            Min=0x0000 Max=0x8000
00237  * Parameters (inout)    : None
00238  * Parameters (out)      : None
00239  * Return value          : None
00240  * Description           : Service sets the period and the duty cycle of a PWM channel
00241  *******************************************************************************/
00242 void Pwm_SetPeriodAndDuty (Pwm_ChannelType ChannelNumber,Pwm_PeriodType Period,uint16 DutyCycle);
00243 /*******************************************************************************
00244  * Service name          : Pwm_SetOutputToIdle
00245  * Service ID[hex]       : 0x04
00246  * Sync/Async            : Asynchronous
00247  * Reentrancy            : Reentrant for different channel numbers
00248  * Parameters (in)       : ChannelNumber
00249                            Numeric identifier of the PWM
00250  * Parameters (inout)    : None
00251  * Parameters (out)      : None
00252  * Return value          : None
00253  * Description           : Service sets the PWM output to the configured Idle state
00254  *******************************************************************************/
00255 void Pwm_SetOutputToIdle (Pwm_ChannelType ChannelNumber);
00256 /*******************************************************************************
00257  * Service name          : Pwm_GetOutputState
00258  * Service ID[hex]       : 0x05
00259  * Sync/Async            : Synchronous
00260  * Reentrancy            : Reentrant for different channel numbers
00261  * Parameters (in)       : ChannelNumber
00262                            Numeric identifier of the PWM
00263  * Parameters (inout)    : None
00264  * Parameters (out)      : None
00265  * Return value          : Pwm_OutputStateType
00266                            PWM_HIGH The PWM output state is high
00267                            PWM_LOW The PWM output state is low
00268  * Description           : Service to read the internal state of the PWM output signal
00269  *******************************************************************************/
00270 Pwm_OutputStateType Pwm_GetOutputState (Pwm_ChannelType ChannelNumber);
```

```
00271 /************************************************************************************
00272  * Service name         : Pwm_DisableNotification
00273  * Service ID[hex]      : 0x06
00274  * Sync/Async           : Asynchronous
00275  * Reentrancy           : Reentrant for different channel numbers
00276  * Parameters (in)      : ChannelNumber
00277                           Numeric identifier of the PWM
00278  * Parameters (inout)   : None
00279  * Parameters (out)     : None
00280  * Return value         : None
00281  * Description          : Service to disable the PWM signal edge notification
00282  ************************************************************************************/
00283 void Pwm_DisableNotification (Pwm_ChannelType ChannelNumber);
00284 /************************************************************************************
00285  * Service name         : Pwm_EnableNotification
00286  * Service ID[hex]      : 0x07
00287  * Sync/Async           : Asynchronous
00288  * Reentrancy           : Reentrant for different channel numbers
00289  * Parameters (in)      : ChannelNumber
00290                           Numeric identifier of the PWM
00291  * Parameters (in)      : Notification
00292                           Type of notification PWM_RISING_EDGE or PWM_FALLING_EDGE or PWM_BOTH_EDGES
00293  * Parameters (inout)   : None
00294  * Parameters (out)     : None
00295  * Return value         : None
00296  * Description          : Service to enable the PWM signal edge notification according to notification
      parameter.
00297  ************************************************************************************/
00298 void Pwm_EnableNotification (Pwm_ChannelType ChannelNumber,Pwm_EdgeNotificationType Notification);
00299 #endif /* PWM_H_ */
```

4.33 Pwm_Cfg.h

```
00001 /************************************************************************************
00002 * @Module      :     PWM
00003 * @File Name   :     Pwm_Cfg.h
00004 * @Description :     the Pre-compile configuration of the AUTOSAR Basic Software module PWM Driver.
00005 * Author       :     Salama Mohamed
00006 ************************************************************************************/
00007 /************************************************************************************
00008 * Project               : Graduation_Project_2024
00009 * Platform              : STM32F103C8
00010 * Autosar Version       : 4.8.0
00011 * SW Version            : 1.0.0
00012 ************************************************************************************/
00013 #ifndef PWM_CFG_H_
00014 #define PWM_CFG_H_
00015 /************************************************************************************
00016                             Source File Version Informations
00017 ************************************************************************************/
00018 #define PWM_VERSION_ID                                  37
00019 #define VENDOR_ID                                       100
00020 #define PWM_CFG_AR_RELEASE_MAJOR_VERSION                4
00021 #define PWM_CFG_AR_RELEASE_MINOR_VERSION                8
00022 #define PWM_CFG_AR_RELEASE_PATCH_VERSION                0
00023 #define PWM_CFG_SW_RELEASE_MAJOR_VERSION                1
00024 #define PWM_CFG_SW_RELEASE_MINOR_VERSION                0
00025 #define PWM_CFG_SW_RELEASE_PATCH_VERSION                0
00026 /************************************************************************************
00027                             Includes
00028 ************************************************************************************/
00029 #include "Std_Types.h"
00030 /************************************************************************************
00031            Pre-compile configuration parameters of the PWM driver.
00032 ************************************************************************************/
00033 //Switches the development error detection and notification on or off.
00034 #define PwmDevErrorDetect               TRUE
00035 //Switch for enabling the update of the duty cycle parameter at the end of the current period.
00036 #define PwmDutycycleUpdatedEndperiod    TRUE
00037 //Specifies the InstanceId of this module instance
00038 #define PwmIndex_Zero                   (uint8)0
00039 //Switch to indicate that the notifications are supported
00040 #define PwmNotificationSupported        TRUE
00041 //Switch for enabling the update of the period parameter at the end of the current period.
00042 #define PwmPeriodUpdatedEndperiod       TRUE
00043 //Channel Id of the PWM channel. This value will be assigned to the symbolic name derived of the
      PwmChannel container short name.
00044 #define PwmChannelId_0                  (uint8)0
00045 #define PwmChannelId_1                  (uint8)1
00046 #define PwmChannelId_2                  (uint8)2
00047 #define PwmChannelId_3                  (uint8)3
00048 //PwmDutycycleDefault
00049 #define Default_Dutycycle               (uint16)0X4000
```

```
00050 //PwmPeriodDefault
00051 #define Default_Period                  (float32)0.02
00052 //Adds / removes the service Pwm_DeInit() from the code.
00053 #define PwmDeInitApi                    TRUE
00054 //Adds / removes the service Pwm_GetOutputState() from the code.
00055 #define PwmGetOutputState               TRUE
00056 //Adds / removes the service Pwm_SetDutyCycle() from the code.
00057 #define PwmSetDutyCycle                 TRUE
00058 //Adds / removes the service Pwm_SetOutputToIdle() from the code.
00059 #define PwmSetOutputToIdle              TRUE
00060 //Adds / removes the service Pwm_SetPeriodAndDuty() from the code.
00061 #define PwmSetPeriodAndDuty             TRUE
00062 //Switch to indicate that the Pwm_ GetVersionInfo is supported
00063 #define PwmVersionInfoApi               TRUE
00064 #endif /* PWM_CFG_H_ */
```

4.34 Src/Ecum.c File Reference

Initializes and de-initializes the OS, the SchM and the BswM as well as some basic software driver modules.

```
#include "Ecum.h"
```

Include dependency graph for Ecum.c:

4.35 Src/main.c File Reference

This main file initializes an AUTOSAR project, performing initialization for all stacks, including CAN stack and other necessary modules like OS kernel. It ensures that all components are properly initialized before starting the main application.

```
#include "stm32f103x6.h"
#include "Can.h"
#include "Port.h"
#include "Pwm.h"
#include "Dio.h"
#include "stm32_f103c6_CAN.h"
#include "Icu.h"
#include "stm32_f103c6_USART.h"
#include "Bluetooth_SWC.h"
#include "Rte_Type.h"
#include "delay.h"
#include "Ecum.h"
```

Include dependency graph for main.c:

Functions

- void Can_Filter (void)

  This function configures a CAN filter to accept or reject incoming messages based on defined criteria.
- int main (void)

  The main function initializes the ECU Manager (EcuM) module, which is responsible for managing the startup and shutdown behavior of the electronic control unit (ECU). After initialization, the function enters an infinite loop, where it remains indefinitely, allowing the ECU to perform its designated tasks.

4.35.1 Detailed Description

This main file initializes an AUTOSAR project, performing initialization for all stacks, including CAN stack and other necessary modules like OS kernel. It ensures that all components are properly initialized before starting the main application.

Author

Salama Mohamed salamamohamedabdelaal@gmail.com

Version

0.1

Date

2024-04-30

Copyright

4.35.2 Function Documentation

4.35.2.1 Can_Filter()

```
void Can_Filter (
            void  )
```

This function configures a CAN filter to accept or reject incoming messages based on defined criteria.

Returns

None

4.35.2.2 main()

```
int main (
            void  )
```

The main function initializes the ECU Manager (EcuM) module, which is responsible for managing the startup and shutdown behavior of the electronic control unit (ECU). After initialization, the function enters an infinite loop, where it remains indefinitely, allowing the ECU to perform its designated tasks.

Returns

int