Auto Sense

AUTHOR
Version

Table of Contents

Topic Index

Topics

Here is a list of all topics with brief descriptions:

Data Structure Index

Data Structures

Here are the data structures with brief descriptions:

File Index

File List

Here is a list of all files with brief descriptions:

Topic Documentation

ACC Configuration

Macros

- #define **RED_RANGE**   0
  Red range definition for ACC.

- #define **BLUE_RANGE**   20
  Blue range definition for ACC.

- #define **GREEN_RANGE**   50
  Green range definition for ACC.

Detailed Description

Macro Definition Documentation

#define BLUE_RANGE   20

Blue range definition for ACC.

Define the blue range for the ACC system. This range may represent a specific condition or state.

#define GREEN_RANGE   50

Green range definition for ACC.

Define the green range for the ACC system. This range may represent a specific condition or state.

#define RED_RANGE   0

Red range definition for ACC.

Define the red range for the ACC system. This range may represent a specific condition or state.

Accelerated Adaptive Cruise Control (AACC) Module

Functions

- void **AACC_vSetSpeedLimit** (**ST_DCM_cfg_t** *rightdcm, **ST_DCM_cfg_t** *leftdcm, **uint8_t** copy_u8SpeedLimit)
  Set speed limit for the AACC.

- void **AACC_vControlingCar** (**ST_DCM_cfg_t** *rightdcm, **ST_DCM_cfg_t** *leftdcm, **uint32_t** copy_u32CurrentDistance)
  Control the car using AACC based on the current distance.

- void **AACC_vStopAcc** (**ST_DCM_cfg_t** *rightdcm, **ST_DCM_cfg_t** *leftdcm)
  Stop the acceleration in the AACC.

- **uint8_t AACC_vChangeAccSpeedLimit** (**uint8_t** copy_u8SpeedAction)
  Change the AACC acceleration speed limit.

Detailed Description

Function Documentation

uint8_t AACC_vChangeAccSpeedLimit (uint8_t *copy_u8SpeedAction*)

Change the AACC acceleration speed limit.

Parameters

copy_u8SpeedAction	Action to determine the speed limit change.

Returns

New speed limit after the change.

void AACC_vControlingCar (ST_DCM_cfg_t * *rightdcm*, ST_DCM_cfg_t * *leftdcm*, uint32_t *copy_u32CurrentDistance*)

Control the car using AACC based on the current distance.

Parameters

rightdcm	Pointer to the configuration structure for the right DC motor.
leftdcm	Pointer to the configuration structure for the left DC motor.
copy_u32CurrentDistance	Current distance from the front obstacle.

void AACC_vSetSpeedLimit (ST_DCM_cfg_t * *rightdcm*, ST_DCM_cfg_t * *leftdcm*, uint8_t *copy_u8SpeedLimit*)

Set speed limit for the AACC.

Parameters

rightdcm	Pointer to the configuration structure for the right DC motor.
leftdcm	Pointer to the configuration structure for the left DC motor.
copy_u8SpeedLimit	Speed limit to be set.

void AACC_vStopAcc (ST_DCM_cfg_t * *rightdcm*, ST_DCM_cfg_t * *leftdcm*)

Stop the acceleration in the AACC.

Parameters

rightdcm	Pointer to the configuration structure for the right DC motor.
leftdcm	Pointer to the configuration structure for the left DC motor.

Automatic Emergency Brake (AEB) Configuration

Macros

- #define **STOP_SPEED** 1
  Speed value to indicate stopping in the AEB system.

- #define **DANGEROUS_ZONE** 10
  Distance threshold for a dangerous zone in the AEB system.

Enumerations

- enum **EN_AAEB_zones_t** { **AAEB_SAFE_ZONE** = 0, **AAEB_DANGEROUS_ZONE** }
  Enumeration representing different zones for AEB action.

Detailed Description

Macro Definition Documentation

#define DANGEROUS_ZONE 10

Distance threshold for a dangerous zone in the AEB system.

Define the distance at which the AEB system considers a zone as dangerous.

#define STOP_SPEED   1

Speed value to indicate stopping in the AEB system.

Define the speed at which the AEB system considers stopping.

Enumeration Type Documentation

enum EN_AAEB_zones_t

Enumeration representing different zones for AEB action.

Enumerator:

AAEB_SAFE_ZONE	Safe zone where no AEB action is required.
AAEB_DANGEROUS_ZONE	Dangerous zone where AEB action is needed.

Automatic Emergency Brake (AEB) Interface

Functions

- void **AAEB_vIsReady** (void)
  Check if the AEB system is ready.

- **EN_AAEB_zones_t AAEB_uddtCheckForObstacles** (**ST_DCM_cfg_t** *PS_uddtRightDcm, **ST_DCM_cfg_t** *PS_uddtLeftDcm, **uint32_t** copy_u32CurrentDistance)
  Check for obstacles and determine the AEB action.

Detailed Description

Function Documentation

EN_AAEB_zones_t AAEB_uddtCheckForObstacles (ST_DCM_cfg_t * *PS_uddtRightDcm*, ST_DCM_cfg_t * *PS_uddtLeftDcm*, uint32_t *copy_u32CurrentDistance*)

Check for obstacles and determine the AEB action.

This function checks for obstacles and determines the appropriate action to be taken by the AEB system.

Parameters

PS_uddtRightDcm	Pointer to the configuration structure for the right DC motor.

PS_uddtLeftDcm	Pointer to the configuration structure for the left DC motor.
copy_u32Current Distance	Current distance from the front obstacle.

Returns

Enumeration representing the AEB action zones.

void AAEB_vIsReady (void)

Check if the AEB system is ready.

This function checks the readiness of the AEB system. It may perform any necessary initialization checks.

Body Control Module (BCM) Configuration

Enumerations

- enum **EN_ABCM_carStates_t** { **ABCM_CAR_STANDBY** = 0, **ABCM_CAR_ON**, **ABCM_CAR_NCC_ACTIVE**, **ABCM_CAR_ACC_SET**, **ABCM_CAR_ACC_ACTIVE**, **ABCM_CAR_GET_FAULT**, **ABCM_CAR_NCC_OFF**, **ABCM_CAR_ACC_OFF**, **ABCM_CAR_IDLE**, **ABCM_UPDATE_FIRMWARE**, **ABCM_CHANGE_SPEED_LIMIT** }
  Enumeration representing different states of the car in BCM.
- enum **EN_ABCM_faultCodes_t** { **ABCM_FAULT_CAR_IS_ALREADY_ON** = 1, **ABCM_FAULT_NCC_IS_ALREADY_ACTIVE**, **ABCM_FAULT_ACC_IS_ALREADY_ACTIVE**, **ABCM_FAULT_CAR_IS_ALREADY_OFF**, **ABCM_FAULT_ACC_IS_ALREADY_OFF**, **ABCM_FAULT_NCC_IS_ALREADY_OFF**, **ABCM_FAULT_SPEED_RANGE_INVALID**, **ABCM_FAULT_ACC_NOR_NCC_IS_WORKING**, **ABCM_NO_FIRMWARE** }
  Enumeration representing different fault codes in BCM.

Detailed Description

Enumeration Type Documentation

enum EN_ABCM_carStates_t

Enumeration representing different states of the car in BCM.

Enumerator:

ABCM_CAR_STANDBY	Car in standby state.
ABCM_CAR_ON	Car turned on.
ABCM_CAR_NCC_ACTIVE	Car with NCC (Non-Collision Control) active.
ABCM_CAR_AC	Car with ACC (Adaptive Cruise Control) speed set.

C_SET	
ABCM_CAR_ACC_ACTIVE	Car with ACC (Adaptive Cruise Control) active.
ABCM_CAR_GET_FAULT	Car checking for faults.
ABCM_CAR_NCC_OFF	Car with NCC (Non-Collision Control) turned off.
ABCM_CAR_ACC_OFF	Car with ACC (Adaptive Cruise Control) turned off.
ABCM_CAR_IDLE	Car in idle state.
ABCM_UPDATE_FIRMWARE	Car updating firmware.
ABCM_CHANGE_SPEED_LIMIT	Car changing speed limit.

enum EN_ABCM_faultCodes_t

Enumeration representing different fault codes in BCM.

Enumerator:

ABCM_FAULT_CAR_IS_ALREADY_ON	
ABCM_FAULT_NCC_IS_ALREADY_ACTIVE	
ABCM_FAULT_ACC_IS_ALREADY_ACTIVE	
ABCM_FAULT_CAR_IS_ALREADY_OFF	
ABCM_FAULT_ACC_IS_ALREADY_OFF	
ABCM_FAULT_NCC_IS_ALREADY_OFF	
ABCM_FAULT_SPEED_RANGE_INVALID	
ABCM_FAULT_ACC_NOR_NCC_IS_WORKING	
ABCM_NO_FIRMWARE	

Body Control Module (BCM) Interface

Functions

- void **ABCM_vSysInit** (void)
  Initialize the Body Control Module (BCM) system.

- void **ABCM_vSysMangment** (void)
  Manage the Body Control Module (BCM) system.

- void **ABCM_vThreadMode** (void)
  Execute the Body Control Module (BCM) in thread mode.

- **EN_ABCM_carStates_t ABCM_uddtDetermineCarState** (**uint8_t** copy_u8Action)
  Determine the car state based on the given action.

- **EN_ABCM_carStates_t ABCM_uddtFaultDetection** (**EN_ABCM_faultCodes_t** copy_uddtFaultCode)
  Detect faults in the Body Control Module (BCM) system.

Detailed Description

Function Documentation

EN_ABCM_carStates_t ABCM_uddtDetermineCarState (uint8_t *copy_u8Action*)

Determine the car state based on the given action.

This function determines the car state based on the provided action.

Parameters

copy_u8Action	Action to be considered for determining the car state.

Returns

Enumeration representing the determined car state.

EN_ABCM_carStates_t ABCM_uddtFaultDetection (EN_ABCM_faultCodes_t *copy_uddtFaultCode*)

Detect faults in the Body Control Module (BCM) system.

This function detects faults in the BCM system based on the provided fault code.

Parameters

copy_uddtFaultCode	Fault code to be used for fault detection.

Returns

Enumeration representing the detected car state after fault detection.

void ABCM_vSysInit (void)

Initialize the Body Control Module (BCM) system.

This function initializes the necessary components and modules for the BCM system.

void ABCM_vSysMangment (void)

Manage the Body Control Module (BCM) system.

This function manages the overall operation of the BCM system. It includes handling different states and modes of the car.

void ABCM_vThreadMode (void)

Execute the Body Control Module (BCM) in thread mode.

This function represents the thread mode execution of the BCM system. It may perform tasks related to the thread-based functionality of the BCM.

Lane Keep Assistant (LKA) Configuration

Enumerations

- enum **EN_ALKA_systeamState_t** { **ALKA_IN_LANE** = 0, **ALKA_OUT_LEFT_LANE**, **ALKA_OUT_RIGHT_LANE**, **ALKA_OUT_BOTH_LANE**, **ALKA_PTR_NULL**, **ALKA_POS_SET** }
  Enumeration representing different system states for Lane Keep Assistant (LKA).

Detailed Description

Enumeration Type Documentation

enum EN_ALKA_systeamState_t

Enumeration representing different system states for Lane Keep Assistant (LKA).

Enumerator:

ALKA_IN_LANE	Car is in the lane.
ALKA_OUT_LEFT_LANE	Car is out of the left lane.
ALKA_OUT_RIGHT_LANE	Car is out of the right lane.

ALKA_OUT_BOTH_LANE	Car is out of both lanes.
ALKA_PTR_NULL	Pointer is null (used for error handling).
ALKA_POS_SET	Lane position is set.

Lane Keep Assistant (LKA) Interface

Functions

- **EN_ALKA_systeamState_t ALKA_uddtGetLanePosition** (**ST_HIR_cfg_t** *PS_uddtRightIr, **ST_HIR_cfg_t** *PS_uddtLeftIr)
  Get the lane position using IR sensors.

- **EN_ALKA_systeamState_t ALKA_uddtSetCarInLanes** (**ST_HIR_cfg_t** *PS_uddtRightIr, **ST_HIR_cfg_t** *PS_uddtLeftIr, **ST_DCM_cfg_t** *PS_uddtRightDcm, **ST_DCM_cfg_t** *PS_uddtLeftDcm)
  Set the car in lanes based on IR sensor readings and DC motor configurations.

- **EN_ALKA_systeamState_t ALKA_forward** (**ST_HIR_cfg_t** *PS_uddtRightIr, **ST_HIR_cfg_t** *PS_uddtLeftIr, **ST_DCM_cfg_t** *PS_uddtRightDcm, **ST_DCM_cfg_t** *PS_uddtLeftDcm)
  Move the car forward based on IR sensor readings and DC motor configurations.

- **EN_ALKA_systeamState_t ALKA_right** (**ST_HIR_cfg_t** *PS_uddtRightIr, **ST_HIR_cfg_t** *PS_uddtLeftIr, **ST_DCM_cfg_t** *PS_uddtRightDcm, **ST_DCM_cfg_t** *PS_uddtLeftDcm)
  Move the car to the right based on IR sensor readings and DC motor configurations.

- **EN_ALKA_systeamState_t ALKA_left** (**ST_HIR_cfg_t** *PS_uddtRightIr, **ST_HIR_cfg_t** *PS_uddtLeftIr, **ST_DCM_cfg_t** *PS_uddtRightDcm, **ST_DCM_cfg_t** *PS_uddtLeftDcm)
  Move the car to the left based on IR sensor readings and DC motor configurations.

Detailed Description

Function Documentation

EN_ALKA_systeamState_t ALKA_forward (ST_HIR_cfg_t * *PS_uddtRightIr*, ST_HIR_cfg_t * *PS_uddtLeftIr*, ST_DCM_cfg_t * *PS_uddtRightDcm*, ST_DCM_cfg_t * *PS_uddtLeftDcm*)

Move the car forward based on IR sensor readings and DC motor configurations.

This function moves the car forward based on IR sensor readings and DC motor configurations.

Parameters

PS_uddtRightIr	Pointer to the configuration structure for the right IR sensor.
PS_uddtLeftIr	Pointer to the configuration structure for the left IR sensor.
PS_uddtRightDcm	Pointer to the configuration structure for the right DC motor.
PS_uddtLeftDcm	Pointer to the configuration structure for the left DC motor.

Returns

Enumeration representing the system state after moving the car forward.

EN_ALKA_systeamState_t ALKA_left (ST_HIR_cfg_t * *PS_uddtRightIr*, ST_HIR_cfg_t * *PS_uddtLeftIr*, ST_DCM_cfg_t * *PS_uddtRightDcm*, ST_DCM_cfg_t * *PS_uddtLeftDcm*)

Move the car to the left based on IR sensor readings and DC motor configurations.

This function moves the car to the left based on IR sensor readings and DC motor configurations.

Parameters

PS_uddtRightIr	Pointer to the configuration structure for the right IR sensor.
PS_uddtLeftIr	Pointer to the configuration structure for the left IR sensor.
PS_uddtRightDcm	Pointer to the configuration structure for the right DC motor.
PS_uddtLeftDcm	Pointer to the configuration structure for the left DC motor.

Returns

Enumeration representing the system state after moving the car to the left.

EN_ALKA_systeamState_t ALKA_right (ST_HIR_cfg_t * *PS_uddtRightIr*, ST_HIR_cfg_t * *PS_uddtLeftIr*, ST_DCM_cfg_t * *PS_uddtRightDcm*, ST_DCM_cfg_t * *PS_uddtLeftDcm*)

Move the car to the right based on IR sensor readings and DC motor configurations.

This function moves the car to the right based on IR sensor readings and DC motor configurations.

Parameters

PS_uddtRightIr	Pointer to the configuration structure for the right IR sensor.
PS_uddtLeftIr	Pointer to the configuration structure for the left IR sensor.
PS_uddtRightDcm	Pointer to the configuration structure for the right DC motor.
PS_uddtLeftDcm	Pointer to the configuration structure for the left DC motor.

Returns

Enumeration representing the system state after moving the car to the right.

EN_ALKA_systeamState_t ALKA_uddtGetLanePosition (ST_HIR_cfg_t * *PS_uddtRightIr*, ST_HIR_cfg_t * *PS_uddtLeftIr*)

Get the lane position using IR sensors.

This function retrieves the lane position based on IR sensor readings.

Parameters

PS_uddtRightIr	Pointer to the configuration structure for the right IR sensor.
PS_uddtLeftIr	Pointer to the configuration structure for the left IR sensor.

Returns

Enumeration representing the system state based on lane position.

EN_ALKA_systeamState_t ALKA_uddtSetCarInLanes (ST_HIR_cfg_t * *PS_uddtRightIr*, ST_HIR_cfg_t * *PS_uddtLeftIr*, ST_DCM_cfg_t * *PS_uddtRightDcm*, ST_DCM_cfg_t * *PS_uddtLeftDcm*)

Set the car in lanes based on IR sensor readings and DC motor configurations.

This function sets the car within lanes based on IR sensor readings and DC motor configurations.

Parameters

PS_uddtRightIr	Pointer to the configuration structure for the right IR sensor.
PS_uddtLeftIr	Pointer to the configuration structure for the left IR sensor.
PS_uddtRightDcm	Pointer to the configuration structure for the right DC motor.
PS_uddtLeftDcm	Pointer to the configuration structure for the left DC motor.

Returns

Enumeration representing the system state after setting the car in lanes.

Normal Cruise Control (NCC) Configuration

Macros

- #define **NCC_INCREMENT_SPEED** +
  Macro to represent the speed increment operation in NCC.

- #define **NCC_DECREMENT_SPEED** -
  Macro to represent the speed decrement operation in NCC.

Detailed Description

Macro Definition Documentation

#define NCC_DECREMENT_SPEED -

Macro to represent the speed decrement operation in NCC.

This macro defines the operation used for decrementing the speed in the NCC system.

#define NCC_INCREMENT_SPEED +

Macro to represent the speed increment operation in NCC.

This macro defines the operation used for incrementing the speed in the NCC system.

Normal Cruise Control (NCC) Interface

Macros

- #define **STOP_SPEED** 1
  Speed value to indicate stopping in the NCC system.

Functions

- void **ANCC_vStartNcc** (**ST_DCM_cfg_t** *leftdcm, **ST_DCM_cfg_t** *rightdcm, **uint8_t** copy_u8Speed)
  Start the Normal Cruise Control (NCC) system.

- void **ANCC_vStopNcc** (**ST_DCM_cfg_t** *leftdcm, **ST_DCM_cfg_t** *rightdcm)
  Stop the Normal Cruise Control (NCC) system.

- **uint8_t ANCC_vChangeNccSpeedLimit** (**uint8_t** copy_u8SpeedAction)
  Change the Normal Cruise Control (NCC) speed limit.

Detailed Description

Macro Definition Documentation

#define STOP_SPEED 1

Speed value to indicate stopping in the NCC system.

Define the speed at which the NCC system considers stopping.

Function Documentation

uint8_t ANCC_vChangeNccSpeedLimit (uint8_t *copy_u8SpeedAction*)

Change the Normal Cruise Control (NCC) speed limit.

Parameters

copy_u8SpeedAction	Action to determine the speed limit change.

Returns

New speed limit after the change.

void ANCC_vStartNcc (ST_DCM_cfg_t * *leftdcm*, ST_DCM_cfg_t * *rightdcm*, uint8_t *copy_u8Speed*)

Start the Normal Cruise Control (NCC) system.

Parameters

leftdcm	Pointer to the configuration structure for the left DC motor.
rightdcm	Pointer to the configuration structure for the right DC motor.
copy_u8Speed	Speed to be set for the NCC system.

void ANCC_vStopNcc (ST_DCM_cfg_t * *leftdcm*, ST_DCM_cfg_t * *rightdcm*)

Stop the Normal Cruise Control (NCC) system.

Parameters

leftdcm	Pointer to the configuration structure for the left DC motor.
rightdcm	Pointer to the configuration structure for the right DC motor.

User Dashboard Interface (UDI) Interface

Functions

- void **AUDI_vInitInterface** (**ST_MUART_RegistersMap_t** *PS_USARTx, **ST_MUSART_cfg_t** const *PS_uddtUartCfg, void(*ptr)(void))
  Initialize the User Dashboard Interface (UDI).
- void **AUDI_vStandByDashboard** (**ST_MUART_RegistersMap_t** *PS_USARTx)
  Put the User Dashboard Interface (UDI) in standby mode.
- void **AUDI_vOnDashboard** (**ST_MUART_RegistersMap_t** *PS_USARTx)
  Turn on the User Dashboard Interface (UDI).

Detailed Description

Function Documentation

void AUDI_vInitInterface (ST_MUART_RegistersMap_t * *PS_USARTx*, ST_MUSART_cfg_t const * *PS_uddtUartCfg*, void(*)(void) *ptr*)

Initialize the User Dashboard Interface (UDI).

Parameters

PS_USARTx	Pointer to the USART registers map.
PS_uddtUartCfg	Pointer to the USART configuration structure.
ptr	Pointer to the callback function.

void AUDI_vOnDashboard (ST_MUART_RegistersMap_t * *PS_USARTx*)

Turn on the User Dashboard Interface (UDI).

Parameters

PS_USARTx	Pointer to the USART registers map.

void AUDI_vStandByDashboard (ST_MUART_RegistersMap_t * *PS_USARTx*)

Put the User Dashboard Interface (UDI) in standby mode.

Parameters

PS_USARTx	Pointer to the USART registers map.

Data Structure Documentation

ST_DCM_cfg_t Struct Reference

```
#include <dcm_config.h>
```

Data Fields

- **ST_MGPIOx_RegistersMap_t * DCM_5vPort**
- **EN_MGPIO_pinOptions_t DCM_5vPin**
- **ST_MGPIOx_RegistersMap_t * DCM_gndPort**
- **EN_MGPIO_pinOptions_t DCM_gndPin**
- **EN_DCM_states_t DCM_intialState**
- **EN_DCM_direction_t DCM_defaultDirection**

Field Documentation

EN_MGPIO_pinOptions_t DCM_5vPin

ST_MGPIOx_RegistersMap_t* DCM_5vPort

EN_DCM_direction_t DCM_defaultDirection

EN_MGPIO_pinOptions_t DCM_gndPin

ST_MGPIOx_RegistersMap_t* DCM_gndPort

EN_DCM_states_t DCM_intialState

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/HAL/dcm/**dcm_config.h**

ST_HIR_cfg_t Struct Reference

Configuration structure for the Human Interface Receiver (HIR) module.
`#include <IR_config.h>`

Data Fields

- **ST_MGPIOx_RegistersMap_t * HIR_port**
- **EN_MGPIO_pinOptions_t HIR_pin**

Detailed Description

Configuration structure for the Human Interface Receiver (HIR) module.

This structure holds the configuration parameters for the HIR module.

Field Documentation

EN_MGPIO_pinOptions_t HIR_pin

Pin option for the HIR module.

ST_MGPIOx_RegistersMap_t* HIR_port

Pointer to the GPIO port for the HIR module.

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/HAL/ir/**IR_config.h**

ST_MEXTI_RegistersMap_t Struct Reference

```
#include <exti_private.h>
```

Data Fields

- **vuint32_t MEXTI_IMR**
- **vuint32_t MEXTI_EMR**
- **vuint32_t MEXTI_RTSR**
- **vuint32_t MEXTI_FTSR**
- **vuint32_t MEXTI_SWIER**
- **vuint32_t MEXTI_PR**

Field Documentation

vuint32_t MEXTI_EMR

vuint32_t MEXTI_FTSR

vuint32_t MEXTI_IMR

vuint32_t MEXTI_PR

vuint32_t MEXTI_RTSR

vuint32_t MEXTI_SWIER

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/exti/**exti_private.h**

ST_MGPIO_altPinCfg_t Struct Reference

```
#include <gpio_config.h>
```

Data Fields

- **ST_MGPIOx_RegistersMap_t * PS_GPIOx**
- **EN_MGPIO_pinOptions_t copy_uddtPinNum**
- **EN_MGPIO_altfnOptions_t Copy_uddtAltFun**
- **EN_MGPIO_outputResistorOptions_t copy_uddtOutputResistor**
- **EN_MGPIO_outputSpeedOptions_t copy_uddtOutputSpeed**
- **EN_MGPIO_pushPullOptions_t copy_uddtPullState**

Field Documentation

EN_MGPIO_altfnOptions_t Copy_uddtAltFun

EN_MGPIO_outputResistorOptions_t copy_uddtOutputResistor

EN_MGPIO_outputSpeedOptions_t copy_uddtOutputSpeed

EN_MGPIO_pinOptions_t copy_uddtPinNum

EN_MGPIO_pushPullOptions_t copy_uddtPullState

ST_MGPIOx_RegistersMap_t* PS_GPIOx

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/gpio/**gpio_config.h**

ST_MGPIO_pinCfg_t Struct Reference

```
#include <gpio_config.h>
```

Data Fields

- **ST_MGPIOx_RegistersMap_t * PS_GPIOx**
- **EN_MGPIO_pinOptions_t copy_uddtPinNum**
- **EN_MGPIO_pinModeOptions_t copy_uddtPinMode**
- **EN_MGPIO_outputResistorOptions_t copy_uddtOutputResistor**
- **EN_MGPIO_outputSpeedOptions_t copy_uddtOutputSpeed**
- **EN_MGPIO_pinLogicOptions_t copy_uddtPtrRetOfPinLogic**
- **EN_MGPIO_pushPullOptions_t copy_uddtPullState**

Field Documentation

EN_MGPIO_outputResistorOptions_t copy_uddtOutputResistor

EN_MGPIO_outputSpeedOptions_t copy_uddtOutputSpeed

EN_MGPIO_pinModeOptions_t copy_uddtPinMode

EN_MGPIO_pinOptions_t copy_uddtPinNum

EN_MGPIO_pinLogicOptions_t copy_uddtPtrRetOfPinLogic

EN_MGPIO_pushPullOptions_t copy_uddtPullState

ST_MGPIOx_RegistersMap_t* PS_GPIOx

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/gpio/**gpio_config.h**

ST_MGPIOx_RegistersMap_t Struct Reference

```
#include <gpio_private.h>
```

Data Fields

- **vuint32_t MGPIOx_MODER**
- **vuint32_t MGPIOx_OTYPER**
- **vuint32_t MGPIOx_OSPEEDR**
- **vuint32_t MGPIOx_PUPDR**
- **vuint32_t MGPIOx_IDR**
- **vuint32_t MGPIOx_ODR**
- **vuint32_t MGPIOx_BSRR**
- **vuint32_t MGPIOx_LCKR**
- **vuint32_t MGPIOx_AFRL**
- **vuint32_t MGPIOx_AFRH**

Field Documentation

vuint32_t MGPIOx_AFRH

vuint32_t MGPIOx_AFRL

vuint32_t MGPIOx_BSRR

vuint32_t MGPIOx_IDR

vuint32_t MGPIOx_LCKR

vuint32_t MGPIOx_MODER

vuint32_t MGPIOx_ODR

vuint32_t MGPIOx_OSPEEDR

vuint32_t MGPIOx_OTYPER

vuint32_t MGPIOx_PUPDR

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/gpio/**gpio_private.h**

ST_MNVIC_RegistersMap_t Struct Reference

```
#include <nvic_private.h>
```

Data Fields

- **vuint32_t MNVIC_ISERx** [8]
- **vuint32_t MNVIC_RESERVED0** [24]
- **vuint32_t MNVIC_ICERx** [8]
- **vuint32_t MNVIC_RESERVED1** [24]
- **vuint32_t MNVIC_ISPRx** [8]
- **vuint32_t MNVIC_RESERVED2** [24]
- **vuint32_t MNVIC_ICPRx** [8]
- **vuint32_t MNVIC_RESERVED3** [24]
- **vuint32_t MNVIC_IABRx** [8]
- **vuint32_t MNVIC_RESERVED4** [56]
- **vuint8_t MNVIC_IPRx** [240]
- **vuint32_t MNVIC_RESERVED5** [580]
- **vuint32_t MNVIC_STIR**

Field Documentation

vuint32_t MNVIC_IABRx[8]

vuint32_t MNVIC_ICERx[8]

vuint32_t MNVIC_ICPRx[8]

vuint8_t MNVIC_IPRx[240]

vuint32_t MNVIC_ISERx[8]

vuint32_t MNVIC_ISPRx[8]

vuint32_t MNVIC_RESERVED0[24]

vuint32_t MNVIC_RESERVED1[24]

vuint32_t MNVIC_RESERVED2[24]

vuint32_t MNVIC_RESERVED3[24]

vuint32_t MNVIC_RESERVED4[56]

vuint32_t MNVIC_RESERVED5[580]

vuint32_t MNVIC_STIR

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/nvic/**nvic_private.h**

ST_MRCC_RegistersMap_t Struct Reference

```
#include <rcc_private.h>
```

Data Fields

- **vuint32_t RCC_CR_REG**
- **vuint32_t RCC_PLLCFGR_REG**
- **vuint32_t RCC_CFGR_REG**
- **vuint32_t RCC_CIR_REG**
- **vuint32_t RCC_AHB1RSTR_REG**
- **vuint32_t RCC_AHB2RSTR_REG**
- **vuint32_t RESERVED0_REG**
- **vuint32_t RESERVED1_REG**
- **vuint32_t RCC_APB1RSTR_REG**
- **vuint32_t RCC_APB2RSTR_REG**
- **vuint32_t RESERVED2_REG**
- **vuint32_t RESERVED3_REG**
- **vuint32_t RCC_AHB1ENR_REG**
- **vuint32_t RCC_AHB2ENR_REG**
- **vuint32_t Reserved5_REG**
- **vuint32_t Reserved6_REG**
- **vuint32_t RCC_APB1ENR_REG**
- **vuint32_t RCC_APB2ENR_REG**
- **vuint32_t RESERVED7_REG**
- **vuint32_t RESERVED8_REG**
- **vuint32_t RCC_AHB1LPENR_REG**
- **vuint32_t RCC_AHB2LPENR_REG**
- **vuint32_t RESERVED9_REG**
- **vuint32_t RESERVED10_REG**
- **vuint32_t RCC_APB1LPENR_REG**
- **vuint32_t RCC_APB2LPENR_REG**
- **vuint32_t RESERVED11_REG**
- **vuint32_t RESERVED12_REG**
- **vuint32_t RCC_BDCR_REG**
- **vuint32_t RCC_CSR_REG**
- **vuint32_t RESERVED13_REG**
- **vuint32_t RESERVED14_REG**
- **vuint32_t RCC_SSCGR_REG**
- **vuint32_t RCC_PLLI2SCFGR_REG**
- **vuint32_t RESERVED15_REG**
- **vuint32_t RCC_DCKCFGR_REG**

Field Documentation

vuint32_t RCC_AHB1ENR_REG

vuint32_t RCC_AHB1LPENR_REG

vuint32_t RCC_AHB1RSTR_REG

vuint32_t RCC_AHB2ENR_REG

vuint32_t RCC_AHB2LPENR_REG

vuint32_t RCC_AHB2RSTR_REG

vuint32_t RCC_APB1ENR_REG

vuint32_t RCC_APB1LPENR_REG

vuint32_t RCC_APB1RSTR_REG

vuint32_t RCC_APB2ENR_REG

vuint32_t RCC_APB2LPENR_REG

vuint32_t RCC_APB2RSTR_REG

vuint32_t RCC_BDCR_REG

vuint32_t RCC_CFGR_REG

vuint32_t RCC_CIR_REG

vuint32_t RCC_CR_REG

vuint32_t RCC_CSR_REG

vuint32_t RCC_DCKCFGR_REG

vuint32_t RCC_PLLCFGR_REG

vuint32_t RCC_PLLI2SCFGR_REG

vuint32_t RCC_SSCGR_REG

vuint32_t RESERVED0_REG

vuint32_t RESERVED10_REG

vuint32_t RESERVED11_REG

vuint32_t RESERVED12_REG

vuint32_t RESERVED13_REG

vuint32_t RESERVED14_REG

vuint32_t RESERVED15_REG

vuint32_t RESERVED1_REG

vuint32_t RESERVED2_REG

vuint32_t RESERVED3_REG

vuint32_t Reserved5_REG

vuint32_t Reserved6_REG

vuint32_t RESERVED7_REG

vuint32_t RESERVED8_REG

vuint32_t RESERVED9_REG

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/rcc/**rcc_private.h**

ST_MSTK_RegistersMap_t Struct Reference

```
#include <systick_private.h>
```

Data Fields

- **vuint32_t MSTK_STK_CTRL**
- **vuint32_t MSTK_STK_LOAD**
- **vuint32_t MSTK_STK_VAL**
- **vuint32_t MSTK_STK_CALIB**

Field Documentation

vuint32_t MSTK_STK_CALIB

vuint32_t MSTK_STK_CTRL

vuint32_t MSTK_STK_LOAD

vuint32_t MSTK_STK_VAL

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/systick/**systick_private.h**

ST_MSYSCFG_RegistersMap_t Struct Reference

```
#include <exti_private.h>
```

Data Fields

- **vuint32_t MSYSCFG_MEMRMP**
- **vuint32_t MSYSCFG_PMC**
- **vuint32_t MSYSCFG_EXTICR1**
- **vuint32_t MSYSCFG_EXTICR2**
- **vuint32_t MSYSCFG_EXTICR3**
- **vuint32_t MSYSCFG_EXTICR4**
- **vuint32_t MSYSCFG_CMPCR**

Field Documentation

vuint32_t MSYSCFG_CMPCR

vuint32_t MSYSCFG_EXTICR1

vuint32_t MSYSCFG_EXTICR2

vuint32_t MSYSCFG_EXTICR3

vuint32_t MSYSCFG_EXTICR4

vuint32_t MSYSCFG_MEMRMP

vuint32_t MSYSCFG_PMC

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/exti/**exti_private.h**

ST_MTMRx_RegistersMap_t Struct Reference

```
#include <tmr_private.h>
```

Data Fields

- **vuint32_t MTMRx_CR1**
- **vuint32_t MTMRx_CR2**
- **vuint32_t MTMRx_SMCR**
- **vuint32_t MTMRx_DIER**
- **vuint32_t MTMRx_SR**
- **vuint32_t MTMRx_EGR**
- **vuint32_t MTMRx_CCMR1**
- **vuint32_t MTMRx_CCMR2**
- **vuint32_t MTMRx_CCER**
- **vuint32_t MTMRx_CNT**
- **vuint32_t MTMRx_PSC**
- **vuint32_t MTMRx_ARR**
- **vuint32_t MTMRx_RESERVED_1**
- **vuint32_t MTMRx_CCR1**
- **vuint32_t MTMRx_CCR2**
- **vuint32_t MTMRx_CCR3**
- **vuint32_t MTMRx_CCR4**
- **vuint32_t MTMRx_RESERVED_2**
- **vuint32_t MTMRx_DCR**
- **vuint32_t MTMRx_DMAR**
- **vuint32_t MTMRx_OR**

Field Documentation

vuint32_t MTMRx_ARR

vuint32_t MTMRx_CCER

vuint32_t MTMRx_CCMR1

vuint32_t MTMRx_CCMR2

vuint32_t MTMRx_CCR1

vuint32_t MTMRx_CCR2

vuint32_t MTMRx_CCR3

vuint32_t MTMRx_CCR4

vuint32_t MTMRx_CNT

vuint32_t MTMRx_CR1

vuint32_t MTMRx_CR2

vuint32_t MTMRx_DCR

vuint32_t MTMRx_DIER

vuint32_t MTMRx_DMAR

vuint32_t MTMRx_EGR

vuint32_t MTMRx_OR

vuint32_t MTMRx_PSC

vuint32_t MTMRx_RESERVED_1

vuint32_t MTMRx_RESERVED_2

vuint32_t MTMRx_SMCR

vuint32_t MTMRx_SR

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/tmr/**tmr_private.h**

ST_MUART_RegistersMap_t Struct Reference

```
#include <usart_private.h>
```

Data Fields

- **vuint32_t MUSART_SR**
- **vuint32_t MUSART_DR**
- **vuint32_t MUSART_BRR**
- **vuint32_t MUSART_CR1**
- **vuint32_t MUSART_CR2**
- **vuint32_t MUSART_CR3**
- **vuint32_t MUSART_GTPR**

Field Documentation

vuint32_t MUSART_BRR

vuint32_t MUSART_CR1

vuint32_t MUSART_CR2

vuint32_t MUSART_CR3

vuint32_t MUSART_DR

vuint32_t MUSART_GTPR

vuint32_t MUSART_SR

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/usart/**usart_private.h**

ST_MUSART_cfg_t Struct Reference

Structure for USART configuration.

```
#include <usart_config.h>
```

Data Fields

- **EN_MUSART_transferControl_t copy_uddtTransferDirection**
- **EN_MUSART_samplingModeOptions_t copy_uddtSamplingModeOption**
- **EN_MUSART_baudRateOptions_t copy_uddtBuadRateOption**
- **EN_MUSART_dataSizeOptions_t copy_uddtDataSizeOption**
- **EN_MUSART_parityControlOption_t copy_uddtParityControl**
- **EN_MUSART_paritySelectionOption_t copy_uddtParitySelection**
- **EN_MUSART_stopBitOption_t copy_uddtStopBitSelection**
- **uint8_t copy_HardwareFlowControl**
- **ST_MUSART_clockInit_t copy_uddtUartClockInit**

Detailed Description

Structure for USART configuration.

Field Documentation

uint8_t copy_HardwareFlowControl

Hardware flow control.

EN_MUSART_baudRateOptions_t copy_uddtBuadRateOption

Baud rate option.

EN_MUSART_dataSizeOptions_t copy_uddtDataSizeOption

Data size option.

EN_MUSART_parityControlOption_t copy_uddtParityControl

Parity control option.

EN_MUSART_paritySelectionOption_t copy_uddtParitySelection

Parity selection option.

EN_MUSART_samplingModeOptions_t copy_uddtSamplingModeOption

Sampling mode option.

EN_MUSART_stopBitOption_t copy_uddtStopBitSelection

Stop bit option.

EN_MUSART_transferControl_t copy_uddtTransferDirection

Transfer direction.

ST_MUSART_clockInit_t copy_uddtUartClockInit

USART clock initialization.

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/usart/**usart_config.h**

ST_MUSART_clockInit_t Struct Reference

Structure for USART clock initialization.
`#include <usart_config.h>`

Data Fields

- **uint8_t clockOutput**
- **uint8_t clockPolarity**
- **uint8_t clockPhase**
- **uint8_t lastBitClockPulse**

Detailed Description

Structure for USART clock initialization.

Field Documentation

uint8_t clockOutput

Clock output.

uint8_t clockPhase

Clock phase.

uint8_t clockPolarity

Clock polarity.

uint8_t lastBitClockPulse

Last bit clock pulse.

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

- D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/usart/**usart_config.h**

File Documentation

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/APPLICATION/Adaptive_Cruise_Control/acc_config.h File Reference

Configuration file for Adaptive Cruise Control (ACC) in the application.

Macros

- #define **RED_RANGE**   0
  Red range definition for ACC.

- #define **BLUE_RANGE**   20
  Blue range definition for ACC.

- #define **GREEN_RANGE**   50
  Green range definition for ACC.

Detailed Description

Configuration file for Adaptive Cruise Control (ACC) in the application.

acc_config.h

Go to the documentation of this file.

```
/**************************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : Adaptive_Cruise_Control
// File          : acc_config.h
// Date          : Oct 23, 2023
// GitHub        : https://github.com/sherifkhadr
/**************************************************************************/
#ifndef APPLICATION_ADAPTIVE_CRUISE_CONTROL_ACC_CONFIG_H_
#define APPLICATION_ADAPTIVE_CRUISE_CONTROL_ACC_CONFIG_H_

#define RED_RANGE   0

#define BLUE_RANGE  20

#define GREEN_RANGE 50

// End of ACC_Configuration group
#endif /* APPLICATION_ADAPTIVE_CRUISE_CONTROL_ACC_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/APPLICATION/Adaptive_Cruise_Control/acc_interface.h File Reference

Interfacing file for Adaptive Cruise Control (ACC) in the application.

```
#include "../../HAL/dcm/dcm_interface.h"
#include "../../HAL/hc05/hc05_interface.h"
#include "acc_config.h"
```

Functions

- void **AACC_vSetSpeedLimit** (**ST_DCM_cfg_t** *rightdcm, **ST_DCM_cfg_t** *leftdcm, **uint8_t** copy_u8SpeedLimit)
  Set speed limit for the AACC.

- void **AACC_vControlingCar** (**ST_DCM_cfg_t** *rightdcm, **ST_DCM_cfg_t** *leftdcm, **uint32_t** copy_u32CurrentDistance)
  Control the car using AACC based on the current distance.

- void **AACC_vStopAcc** (**ST_DCM_cfg_t** *rightdcm, **ST_DCM_cfg_t** *leftdcm)
  Stop the acceleration in the AACC.

- **uint8_t AACC_vChangeAccSpeedLimit** (**uint8_t** copy_u8SpeedAction)
  Change the AACC acceleration speed limit.

Detailed Description

Interfacing file for Adaptive Cruise Control (ACC) in the application.

acc_interface.h

Go to the documentation of this file.

```
/*************************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : Adaptive_Cruise_Control
// File          : acc_interface.h
// Date          : Oct 23, 2023
// GitHub        : https://github.com/sherifkhadr
/*************************************************************************/
#ifndef ACC_INTERFACE_H_
#define ACC_INTERFACE_H_

#include "../../HAL/dcm/dcm_interface.h"
#include "../../HAL/hc05/hc05_interface.h"

#include "acc_config.h"

void AACC_vSetSpeedLimit(ST_DCM_cfg_t *rightdcm, ST_DCM_cfg_t *leftdcm, uint8_t
copy_u8SpeedLimit);

void AACC_vControlingCar(ST_DCM_cfg_t *rightdcm, ST_DCM_cfg_t *leftdcm, uint32_t
copy_u32CurrentDistance);

void AACC_vStopAcc(ST_DCM_cfg_t *rightdcm, ST_DCM_cfg_t *leftdcm);

uint8_t AACC_vChangeAccSpeedLimit(uint8_t copy_u8SpeedAction);

 // End of ACC_INTERFACE_H_ group61
#endif
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/APPLICATION/Automatic_Emergency_Brake/aeb_config.h File Reference

Configuration file for Automatic Emergency Brake (AEB) application.

Macros

- #define **STOP_SPEED**   1
  Speed value to indicate stopping in the AEB system.

- #define **DANGEROUS_ZONE**   10
  Distance threshold for a dangerous zone in the AEB system.

Enumerations

- enum **EN_AAEB_zones_t** { **AAEB_SAFE_ZONE** = 0, **AAEB_DANGEROUS_ZONE** }
  Enumeration representing different zones for AEB action.

Detailed Description

Configuration file for Automatic Emergency Brake (AEB) application.

aeb_config.h

Go to the documentation of this file.

```c
/***********************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : Adaptive_Cruise_Control
// File           : aeb_config.h
// Date           : Oct 26, 2023
// GitHub         : https://github.com/sherifkhadr
/***********************************************************************/
#ifndef APPLICATION_AUTOMATIC_EMERGENCY_BRAKE_AEB_CONFIG_H_
#define APPLICATION_AUTOMATIC_EMERGENCY_BRAKE_AEB_CONFIG_H_

#define STOP_SPEED 1

#define DANGEROUS_ZONE 10

typedef enum
{
    AAEB_SAFE_ZONE = 0,
    AAEB_DANGEROUS_ZONE
} EN_AAEB_zones_t;

// End of AEB_Configuration group
#endif /* APPLICATION_AUTOMATIC_EMERGENCY_BRAKE_AEB_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/APPLICATION/Automatic_Emergency_Brake/aeb_interface.h File Reference

Interface for Automatic Emergency Brake (AEB) application.

```
#include "../../HAL/dcm/dcm_interface.h"
#include "../../HAL/hc05/hc05_interface.h"
#include "aeb_config.h"
```

Functions

- void **AAEB_vIsReady** (void)
  Check if the AEB system is ready.

- **EN_AAEB_zones_t AAEB_uddtCheckForObstacles** (**ST_DCM_cfg_t** *PS_uddtRightDcm, **ST_DCM_cfg_t** *PS_uddtLeftDcm, **uint32_t** copy_u32CurrentDistance)
  Check for obstacles and determine the AEB action.

Detailed Description

Interface for Automatic Emergency Brake (AEB) application.

aeb_interface.h

Go to the documentation of this file.

```c
/*************************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : Adaptive_Cruise_Control
// File          : aeb_interface.h
// Date          : Oct 26, 2023
// GitHub        : https://github.com/sherifkhadr
/*************************************************************************/
#ifndef APPLICATION_AUTOMATIC_EMERGENCY_BRAKE_AEB_INTERFACE_H_
#define APPLICATION_AUTOMATIC_EMERGENCY_BRAKE_AEB_INTERFACE_H_

#include "../../HAL/dcm/dcm_interface.h"
#include "../../HAL/hc05/hc05_interface.h"
#include "aeb_config.h"

void AAEB_vIsReady(void);

EN_AAEB_zones_t AAEB_uddtCheckForObstacles(ST_DCM_cfg_t *PS_uddtRightDcm,
                                           ST_DCM_cfg_t *PS_uddtLeftDcm,
                                           uint32_t copy_u32CurrentDistance);

// End of AEB_Interface group
#endif /* APPLICATION_AUTOMATIC_EMERGENCY_BRAKE_AEB_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/APPLICATION/Body_Control_Module/bcm_config.h File Reference

Configuration file for Body Control Module (BCM) in the application.

Enumerations

- enum **EN_ABCM_carStates_t** { **ABCM_CAR_STANDBY** = 0, **ABCM_CAR_ON**, **ABCM_CAR_NCC_ACTIVE**, **ABCM_CAR_ACC_SET**, **ABCM_CAR_ACC_ACTIVE**, **ABCM_CAR_GET_FAULT**, **ABCM_CAR_NCC_OFF**, **ABCM_CAR_ACC_OFF**, **ABCM_CAR_IDLE**, **ABCM_UPDATE_FIRMWARE**, **ABCM_CHANGE_SPEED_LIMIT** }
  Enumeration representing different states of the car in BCM.
- enum **EN_ABCM_faultCodes_t** { **ABCM_FAULT_CAR_IS_ALREADY_ON** = 1, **ABCM_FAULT_NCC_IS_ALREADY_ACTIVE**, **ABCM_FAULT_ACC_IS_ALREADY_ACTIVE**, **ABCM_FAULT_CAR_IS_ALREADY_OFF**, **ABCM_FAULT_ACC_IS_ALREADY_OFF**, **ABCM_FAULT_NCC_IS_ALREADY_OFF**, **ABCM_FAULT_SPEED_RANGE_INVALID**, **ABCM_FAULT_ACC_NOR_NCC_IS_WORKING**, **ABCM_NO_FIRMWARE** }
  Enumeration representing different fault codes in BCM.

Detailed Description

Configuration file for Body Control Module (BCM) in the application.

bcm_config.h

Go to the documentation of this file.

```c
/*************************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : Body_Control_Module
// File          : bcm_config.h
// Date          : Oct 18, 2023
// GitHub        : https://github.com/sherifkhadr
/*************************************************************************/
#ifndef APPLICATION_BODY_CONTROL_MODULE_BCM_CONFIG_H_
#define APPLICATION_BODY_CONTROL_MODULE_BCM_CONFIG_H_

typedef enum
{
    ABCM_CAR_STANDBY = 0,
    ABCM_CAR_ON,
    ABCM_CAR_NCC_ACTIVE,
    ABCM_CAR_ACC_SET,
    ABCM_CAR_ACC_ACTIVE,
    ABCM_CAR_GET_FAULT,
    ABCM_CAR_NCC_OFF,
    ABCM_CAR_ACC_OFF,
    ABCM_CAR_IDLE,
    ABCM_UPDATE_FIRMWARE,
    ABCM_CHANGE_SPEED_LIMIT
} EN_ABCM_carStates_t;

typedef enum
{
    ABCM_FAULT_CAR_IS_ALREADY_ON = 1,
    ABCM_FAULT_NCC_IS_ALREADY_ACTIVE,
    ABCM_FAULT_ACC_IS_ALREADY_ACTIVE,
    ABCM_FAULT_CAR_IS_ALREADY_OFF,
    ABCM_FAULT_ACC_IS_ALREADY_OFF,
    ABCM_FAULT_NCC_IS_ALREADY_OFF,
    ABCM_FAULT_SPEED_RANGE_INVALID,
    ABCM_FAULT_ACC_NOR_NCC_IS_WORKING,
    ABCM_NO_FIRMWARE
} EN_ABCM_faultCodes_t;

// End of BCM_Configuration group
#endif /* APPLICATION_BODY_CONTROL_MODULE_BCM_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/APPLICATION/Body_Control_Module/bcm_interface.h File Reference

Interface for Body Control Module (BCM) in the application.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "bcm_config.h"
#include "../../COMMON/std_types.h"
#include "../../MCAL/rcc/rcc_interface.h"
#include "../../MCAL/gpio/gpio_interface.h"
#include "../../MCAL/tmr/tmr_interface.h"
#include "../../MCAL/nvic/nvic_interface.h"
#include "../../HAL/ir/IR_interface.h"
#include "../../HAL/ultrasonic/ult_interface.h"
#include "../../MCAL/exti/exti_interface.h"
#include "../User_Dashboard_Interface/udi_interface.h"
#include "../Normal_Cruise_Control/ncc_interface.h"
#include "../Adaptive_Cruise_Control/acc_interface.h"
#include "../Automatic_Emergency_Brake/aeb_interface.h"
#include "../Lane_Keep_Assistant/LKA_interface.h"
```

Functions

- void **ABCM_vSysInit** (void)
  Initialize the Body Control Module (BCM) system.

- void **ABCM_vSysMangment** (void)
  Manage the Body Control Module (BCM) system.

- void **ABCM_vThreadMode** (void)
  Execute the Body Control Module (BCM) in thread mode.

- **EN_ABCM_carStates_t ABCM_uddtDetermineCarState** (**uint8_t** copy_u8Action)
  Determine the car state based on the given action.

- **EN_ABCM_carStates_t ABCM_uddtFaultDetection** (**EN_ABCM_faultCodes_t** copy_uddtFaultCode)
  Detect faults in the Body Control Module (BCM) system.

Detailed Description

Interface for Body Control Module (BCM) in the application.

bcm_interface.h

Go to the documentation of this file.

```
/**************************************************************************/
// Author          : Sherif Ashraf Khadr
// Project         : Body_Control_Module
// File            : bcm_interface.h
// Date            : Oct 17, 2023
// GitHub          : https://github.com/sherifkhadr
/**************************************************************************/
#ifndef APPLICATION_BODY_CONTROL_MODULE_BCM_INTERFACE_H_
#define APPLICATION_BODY_CONTROL_MODULE_BCM_INTERFACE_H_

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "bcm_config.h"
#include "../../COMMON/std_types.h"
#include "../../MCAL/rcc/rcc_interface.h"
#include "../../MCAL/gpio/gpio_interface.h"
#include "../../MCAL/tmr/tmr_interface.h"
#include "../../MCAL/nvic/nvic_interface.h"
#include "../../HAL/ir/IR_interface.h"
#include "../../HAL/ultrasonic/ult_interface.h"
#include "../../MCAL/exti/exti_interface.h"
#include "../User_Dashboard_Interface/udi_interface.h"
#include "../Normal_Cruise_Control/ncc_interface.h"
#include "../Adaptive_Cruise_Control/acc_interface.h"
#include "../Automatic_Emergency_Brake/aeb_interface.h"
#include "../Lane_Keep_Assistant/LKA_interface.h"

void ABCM_vSysInit(void);

void ABCM_vSysMangment(void);

void ABCM_vThreadMode(void);

EN_ABCM_carStates_t ABCM_uddtDetermineCarState(uint8_t copy_u8Action);

EN_ABCM_carStates_t ABCM_uddtFaultDetection(EN_ABCM_faultCodes_t 
copy_uddtFaultCode);

// End of BCM_Interface group
#endif /* APPLICATION_BODY_CONTROL_MODULE_BCM_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/APPLICATION/Lane_Keep_Assistant/LKA_config.h File Reference

Configuration file for Lane Keep Assistant (LKA) application.

Enumerations

- enum **EN_ALKA_systeamState_t** { **ALKA_IN_LANE** = 0, **ALKA_OUT_LEFT_LANE**, **ALKA_OUT_RIGHT_LANE**, **ALKA_OUT_BOTH_LANE**, **ALKA_PTR_NULL**, **ALKA_POS_SET** }
  Enumeration representing different system states for Lane Keep Assistant (LKA).

Detailed Description

Configuration file for Lane Keep Assistant (LKA) application.

LKA_config.h

Go to the documentation of this file.

```
/**************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : Adaptive_Cruise_Control
// File           : LKA_config.h
// Date           : Nov 7, 2023
// GitHub         : https://github.com/sherifkhadr
/**************************************************************************/
#ifndef APPLICATION_LANE_KEEP_ASSISTANT_LKA_CONFIG_H_
#define APPLICATION_LANE_KEEP_ASSISTANT_LKA_CONFIG_H_

typedef enum
{
    ALKA_IN_LANE = 0,
    ALKA_OUT_LEFT_LANE,
    ALKA_OUT_RIGHT_LANE,
    ALKA_OUT_BOTH_LANE,
    ALKA_PTR_NULL,
    ALKA_POS_SET
} EN_ALKA_systeamState_t;

 // End of LKA_Configuration group
#endif /* APPLICATION_LANE_KEEP_ASSISTANT_LKA_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/APPLICATION/Lane_Keep_Assistant/LKA_interface.h File Reference

Interface for Lane Keep Assistant (LKA) in the application.

```
#include "../../../Inc/COMMON/std_types.h"
#include "../../../Inc/HAL/ir/IR_interface.h"
#include "../../../Inc/HAL/dcm/dcm_interface.h"
#include "LKA_config.h"
```

Functions

- **EN_ALKA_systeamState_t ALKA_uddtGetLanePosition** (**ST_HIR_cfg_t** *PS_uddtRightIr, **ST_HIR_cfg_t** *PS_uddtLeftIr)
  Get the lane position using IR sensors.

- **EN_ALKA_systeamState_t ALKA_uddtSetCarInLanes** (**ST_HIR_cfg_t** *PS_uddtRightIr, **ST_HIR_cfg_t** *PS_uddtLeftIr, **ST_DCM_cfg_t** *PS_uddtRightDcm, **ST_DCM_cfg_t** *PS_uddtLeftDcm)
  Set the car in lanes based on IR sensor readings and DC motor configurations.

- **EN_ALKA_systeamState_t ALKA_forward** (**ST_HIR_cfg_t** *PS_uddtRightIr, **ST_HIR_cfg_t** *PS_uddtLeftIr, **ST_DCM_cfg_t** *PS_uddtRightDcm, **ST_DCM_cfg_t** *PS_uddtLeftDcm)
  Move the car forward based on IR sensor readings and DC motor configurations.

- **EN_ALKA_systeamState_t ALKA_right** (**ST_HIR_cfg_t** *PS_uddtRightIr, **ST_HIR_cfg_t** *PS_uddtLeftIr, **ST_DCM_cfg_t** *PS_uddtRightDcm, **ST_DCM_cfg_t** *PS_uddtLeftDcm)
  Move the car to the right based on IR sensor readings and DC motor configurations.

- **EN_ALKA_systeamState_t ALKA_left** (**ST_HIR_cfg_t** *PS_uddtRightIr, **ST_HIR_cfg_t** *PS_uddtLeftIr, **ST_DCM_cfg_t** *PS_uddtRightDcm, **ST_DCM_cfg_t** *PS_uddtLeftDcm)
  Move the car to the left based on IR sensor readings and DC motor configurations.

Detailed Description

Interface for Lane Keep Assistant (LKA) in the application.

LKA_interface.h

Go to the documentation of this file.

```c
/**************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : Adaptive_Cruise_Control
// File           : LKA_interface.h
// Date           : Nov 7, 2023
// GitHub         : https://github.com/sherifkhadr
/**************************************************************************/
#ifndef APPLICATION_LANE_KEEP_ASSISTANT_LKA_INTERFACE_H_
#define APPLICATION_LANE_KEEP_ASSISTANT_LKA_INTERFACE_H_

#include "../../../Inc/COMMON/std_types.h"
#include "../../../Inc/HAL/ir/IR_interface.h"
#include "../../../Inc/HAL/dcm/dcm_interface.h"
#include "LKA_config.h"

EN_ALKA_systeamState_t ALKA_uddtGetLanePosition(ST_HIR_cfg_t *PS_uddtRightIr, ST_HIR_cfg_t *PS_uddtLeftIr);

EN_ALKA_systeamState_t ALKA_uddtSetCarInLanes(ST_HIR_cfg_t *PS_uddtRightIr, ST_HIR_cfg_t *PS_uddtLeftIr,
                                              ST_DCM_cfg_t *PS_uddtRightDcm, ST_DCM_cfg_t *PS_uddtLeftDcm);

EN_ALKA_systeamState_t ALKA_forward(ST_HIR_cfg_t *PS_uddtRightIr, ST_HIR_cfg_t *PS_uddtLeftIr,
                                    ST_DCM_cfg_t *PS_uddtRightDcm, ST_DCM_cfg_t *PS_uddtLeftDcm);

EN_ALKA_systeamState_t ALKA_right(ST_HIR_cfg_t *PS_uddtRightIr, ST_HIR_cfg_t *PS_uddtLeftIr,
                                  ST_DCM_cfg_t *PS_uddtRightDcm, ST_DCM_cfg_t *PS_uddtLeftDcm);

EN_ALKA_systeamState_t ALKA_left(ST_HIR_cfg_t *PS_uddtRightIr, ST_HIR_cfg_t *PS_uddtLeftIr,
                                 ST_DCM_cfg_t *PS_uddtRightDcm, ST_DCM_cfg_t *PS_uddtLeftDcm);

 // End of LKA_Interface group
#endif /* APPLICATION_LANE_KEEP_ASSISTANT_LKA_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/APPLICATION/Normal_Cruise_Control/ncc_config.h File Reference

Configuration file for Normal Cruise Control (NCC) in the Adaptive Cruise Control project.

Macros

- #define **NCC_INCREMENT_SPEED** +
  Macro to represent the speed increment operation in NCC.

- #define **NCC_DECREMENT_SPEED** -
  Macro to represent the speed decrement operation in NCC.

Detailed Description

Configuration file for Normal Cruise Control (NCC) in the Adaptive Cruise Control project.

Author

Sherif Ashraf Khadr

Date

Oct 18, 2023

See also

`https://github.com/sherifkhadr`

ncc_config.h

Go to the documentation of this file.

```

#ifndef APPLICATION_NORMAL_CRUISE_CONTROL_NCC_CONFIG_H_
#define APPLICATION_NORMAL_CRUISE_CONTROL_NCC_CONFIG_H_

#define NCC_INCREMENT_SPEED +

#define NCC_DECREMENT_SPEED -

 // End of NCC_Configuration group
#endif /* APPLICATION_NORMAL_CRUISE_CONTROL_NCC_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/APPLICATION/Normal_Cruise_Control/ncc_interface.h File Reference

Interface for Normal Cruise Control (NCC) in the Adaptive Cruise Control project.

```
#include "../../HAL/dcm/dcm_interface.h"
#include "../../HAL/hc05/hc05_interface.h"
#include "ncc_config.h"
```

Macros

- #define **STOP_SPEED**   1
  Speed value to indicate stopping in the NCC system.

Functions

- void **ANCC_vStartNcc** (**ST_DCM_cfg_t** *leftdcm, **ST_DCM_cfg_t** *rightdcm, **uint8_t** copy_u8Speed)
  Start the Normal Cruise Control (NCC) system.

- void **ANCC_vStopNcc** (**ST_DCM_cfg_t** *leftdcm, **ST_DCM_cfg_t** *rightdcm)
  Stop the Normal Cruise Control (NCC) system.

- **uint8_t** **ANCC_vChangeNccSpeedLimit** (**uint8_t** copy_u8SpeedAction)
  Change the Normal Cruise Control (NCC) speed limit.

Detailed Description

Interface for Normal Cruise Control (NCC) in the Adaptive Cruise Control project.

Author

Sherif Ashraf Khadr

Date

Oct 18, 2023

See also

`https://github.com/sherifkhadr`

ncc_interface.h

Go to the documentation of this file.

```

#ifndef APPLICATION_NORMAL_CRUISE_CONTROL_NCC_INTERFACE_H_
#define APPLICATION_NORMAL_CRUISE_CONTROL_NCC_INTERFACE_H_

#include "../../HAL/dcm/dcm_interface.h"
#include "../../HAL/hc05/hc05_interface.h"
#include "ncc_config.h"

#define STOP_SPEED 1

void ANCC_vStartNcc(ST_DCM_cfg_t *leftdcm, ST_DCM_cfg_t *rightdcm, uint8_t copy_u8Speed);

void ANCC_vStopNcc(ST_DCM_cfg_t *leftdcm, ST_DCM_cfg_t *rightdcm);

uint8_t ANCC_vChangeNccSpeedLimit(uint8_t copy_u8SpeedAction);

 // End of NCC_Interface group
#endif /* APPLICATION_NORMAL_CRUISE_CONTROL_NCC_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/APPLICATION/User_Dashboard_Interface/udi_interface.h File Reference

Interface for User Dashboard in the Adaptive Cruise Control project.
`#include "../../HAL/hc05/hc05_interface.h"`

Functions

- void **AUDI_vInitInterface** (**ST_MUART_RegistersMap_t** *PS_USARTx, **ST_MUSART_cfg_t** const *PS_uddtUartCfg, void(*ptr)(void))
  Initialize the User Dashboard Interface (UDI).

- void **AUDI_vStandByDashboard** (**ST_MUART_RegistersMap_t** *PS_USARTx)
  Put the User Dashboard Interface (UDI) in standby mode.

- void **AUDI_vOnDashboard** (**ST_MUART_RegistersMap_t** *PS_USARTx)
  Turn on the User Dashboard Interface (UDI).

Detailed Description

Interface for User Dashboard in the Adaptive Cruise Control project.

Author

Sherif Ashraf Khadr

Date

Oct 18, 2023

See also

`https://github.com/sherifkhadr`

udi_interface.h

Go to the documentation of this file.

```
#ifndef APPLICATION_USER_DASHBOARD_INTERFACE_UDI_INTERFACE_H_
#define APPLICATION_USER_DASHBOARD_INTERFACE_UDI_INTERFACE_H_

#include "../../HAL/hc05/hc05_interface.h"

void AUDI_vInitInterface(ST_MUART_RegistersMap_t *PS_USARTx, ST_MUSART_cfg_t const
*PS_uddtUartCfg, void (*ptr)(void));

void AUDI_vStandByDashboard(ST_MUART_RegistersMap_t *PS_USARTx);

void AUDI_vOnDashboard(ST_MUART_RegistersMap_t *PS_USARTx);

 // End of UDI_Interface group
#endif /* APPLICATION_USER_DASHBOARD_INTERFACE_UDI_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/COMMON/bit_math.h File Reference

Macros

- #define **SET_BIT**(REG, BIT_NUMBER) (REG |= (1 << BIT_NUMBER))
- #define **GET_BIT**(REG, BIT_NUMBER) ((REG >> BIT_NUMBER) & 1)
- #define **CLR_BIT**(REG, BIT_NUMBER) (REG &= (~(1 << BIT_NUMBER)))
- #define **TOG_BIT**(REG, BIT_NUMBER) (REG ^= (1 << BIT_NUMBER))
- #define **SET_BITS**(REG, MSK) (REG) |= (MSK)
- #define **CLR_BITS**(REG, MSK) (REG) &= ~(MSK)
- #define **TOG_BITS**(REG, MSK) (REG) ^= (MSK)
- #define **SET_ALL_BITS**(REG) (REG) = (0xFFFFFFFF)
- #define **CLR_ALL_BITS**(REG) (REG) = (0x00000000)
- #define **TOG_ALL_BITS**(REG) (REG) ^= (0xFFFFFFFF)

Macro Definition Documentation

#define CLR_ALL_BITS(REG) (REG) = (0x00000000)

#define CLR_BIT(REG, BIT_NUMBER) (REG &= (~(1 << BIT_NUMBER)))

#define CLR_BITS(REG, MSK) (REG) &= ~(MSK)

#define GET_BIT(REG, BIT_NUMBER) ((REG >> BIT_NUMBER) & 1)

#define SET_ALL_BITS(REG) (REG) = (0xFFFFFFFF)

#define SET_BIT(REG, BIT_NUMBER) (REG |= (1 << BIT_NUMBER))

#define SET_BITS(REG, MSK) (REG) |= (MSK)

#define TOG_ALL_BITS(REG) (REG) ^= (0xFFFFFFFF)

#define TOG_BIT(REG, BIT_NUMBER) (REG ^= (1 << BIT_NUMBER))

#define TOG_BITS(REG, MSK) (REG) ^= (MSK)

bit_math.h

Go to the documentation of this file.

```c
/**************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : STM32F401xC_Drivers
// File           : main.c
// Date           : Sep 8, 2023
// GitHub         : https://github.com/sherifkhadr
/**************************************************************************/

#ifndef COMMON_BIT_MATH_H_
#define COMMON_BIT_MATH_H_

#define SET_BIT(REG, BIT_NUMBER) (REG |= (1 << BIT_NUMBER))
#define GET_BIT(REG, BIT_NUMBER) ((REG >> BIT_NUMBER) & 1)
#define CLR_BIT(REG, BIT_NUMBER) (REG &= (~(1 << BIT_NUMBER)))
#define TOG_BIT(REG, BIT_NUMBER) (REG ^= (1 << BIT_NUMBER))


 #define SET_BITS(REG,MSK)          (REG) |=  (MSK)
 #define CLR_BITS(REG,MSK)          (REG) &= ~(MSK)
 #define TOG_BITS(REG,MSK)          (REG) ^=  (MSK)

 #define SET_ALL_BITS(REG)          (REG) =  (0xFFFFFFFF)
 #define CLR_ALL_BITS(REG)          (REG) =  (0x00000000)
 #define TOG_ALL_BITS(REG)          (REG) ^= (0xFFFFFFFF)

#endif /* COMMON_BIT_MATH_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/COMMON/std_types.h File Reference

Macros

- #define **TRUE**   1
- #define **FALSE**   0
- #define **STR_NULL**   '\0'
- #define **PTR_NULL**   (void*)0

Typedefs

- typedef unsigned char **uint8_t**
- typedef unsigned short int **uint16_t**
- typedef unsigned long int **uint32_t**
- typedef signed char **sint8_t**
- typedef signed short int **sint16_t**
- typedef signed long int **sint32_t**
- typedef float **float32_t**
- typedef double **float64_t**
- typedef long double **float96_t**
- typedef volatile unsigned char **vuint8_t**
- typedef volatile unsigned short int **vuint16_t**
- typedef volatile unsigned long int **vuint32_t**
- typedef volatile signed char **vsint8_t**
- typedef volatile signed short int **vsint16_t**
- typedef volatile signed long int **vsint32_t**
- typedef volatile float **vfloat32_t**
- typedef volatile double **vfloat64_t**
- typedef volatile long double **vfloat96_t**

Macro Definition Documentation

#define FALSE   0

#define PTR_NULL   (void*)0

#define STR_NULL   '\0'

#define TRUE   1

Typedef Documentation

typedef float float32_t

typedef double float64_t

typedef long double float96_t

typedef signed short int sint16_t

typedef signed long int sint32_t

typedef signed char sint8_t

typedef unsigned short int uint16_t

typedef unsigned long int uint32_t

typedef unsigned char uint8_t

typedef volatile float vfloat32_t

typedef volatile double vfloat64_t

typedef volatile long double vfloat96_t

typedef volatile signed short int vsint16_t

typedef volatile signed long int vsint32_t

typedef volatile signed char vsint8_t

typedef volatile unsigned short int vuint16_t

typedef volatile unsigned long int vuint32_t

typedef volatile unsigned char vuint8_t

std_types.h

Go to the documentation of this file.

```c
/*************************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : STM32F401xC_Drivers
// File          : main.c
// Date          : Sep 8, 2023
// GitHub        : https://github.com/sherifkhadr
/*************************************************************************/

#ifndef COMMON_STD_TYPES_H_
#define COMMON_STD_TYPES_H_

typedef unsigned char                        uint8_t     ;
typedef unsigned short int                   uint16_t    ;
typedef unsigned long  int                   uint32_t    ;
typedef signed   char                        sint8_t     ;
typedef signed   short int                   sint16_t    ;
typedef signed   long  int                   sint32_t    ;
typedef float                                float32_t   ;
typedef double                               float64_t   ;
typedef long double                          float96_t   ;


typedef volatile unsigned char               vuint8_t    ;
typedef volatile unsigned short int          vuint16_t   ;
typedef volatile unsigned long  int          vuint32_t   ;
typedef volatile signed   char               vsint8_t    ;
typedef volatile signed   short int          vsint16_t   ;
typedef volatile signed   long  int          vsint32_t   ;
typedef volatile float                       vfloat32_t  ;
typedef volatile double                      vfloat64_t  ;
typedef volatile long double                 vfloat96_t  ;


#ifndef TRUE
#define TRUE   1
#endif


#ifndef FALSE
#define FALSE   0
#endif


#ifndef STR_NULL
#define STR_NULL    '\0'
#endif

#ifndef PTR_NULL
#define PTR_NULL    (void*)0
#endif


#endif /* COMMON_STD_TYPES_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/HAL/dcm/dcm_config.h File Reference

Data Structures

struct ST_DCM_cfg_t**Enumerations**

- enum **EN_DCM_systemState_t** { **DCM_OK** = 0, **DCM_NOK** }
  Enumeration for the system states of the DC motor.
- enum **EN_DCM_states_t** { **DCM_TURN_OFF** = 0, **DCM_TURN_ON** }
  Enumeration for the states of the DC motor (turn on or off).
- enum **EN_DCM_direction_t** { **DCM_DIR_CLOCKWISE** = 0, **DCM_DIR_ANTI_CLOCKWISE** }
  Enumeration for the direction of the DC motor.
- enum **EN_DCM_rotation_t** { **DCM_ROTATE_LEFT** = 0, **DCM_ROTATE_RIGHT** }
  Enumeration for the rotation of the DC motor.

Enumeration Type Documentation

enum EN_DCM_direction_t

Enumeration for the direction of the DC motor.

Enumerator:

DCM_DIR_CLOCKWISE	DC motor rotates in the clockwise direction.
DCM_DIR_ANTI_CLOCKWISE	DC motor rotates in the anti-clockwise direction.

enum EN_DCM_rotation_t

Enumeration for the rotation of the DC motor.

Enumerator:

DCM_ROTATE_LEFT	DC motor rotates to the left.
DCM_ROTATE_RIGHT	DC motor rotates to the right.

enum EN_DCM_states_t

Enumeration for the states of the DC motor (turn on or off).

Enumerator:

DCM_TURN_OFF	DC motor turned off.
DCM_TURN_ON	DC motor turned on.

enum EN_DCM_systemState_t

Enumeration for the system states of the DC motor.

Enumerator:

DCM_OK	DC motor operation successful.
DCM_NOK	DC motor operation failed.

dcm_config.h

Go to the documentation of this file.

```
/**************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : Adaptive_Cruise_Control
// File           : dcm_config.h
// Date           : Oct 17, 2023
// GitHub         : https://github.com/sherifkhadr
/**************************************************************************/
#ifndef HAL_DCM_DCM_CONFIG_H_
#define HAL_DCM_DCM_CONFIG_H_


typedef enum
{
    DCM_OK = 0,
    DCM_NOK
} EN_DCM_systemState_t;

typedef enum
{
    DCM_TURN_OFF = 0,
    DCM_TURN_ON
} EN_DCM_states_t;

typedef enum
{
    DCM_DIR_CLOCKWISE = 0,
    DCM_DIR_ANTI_CLOCKWISE
} EN_DCM_direction_t;

typedef enum
{
    DCM_ROTATE_LEFT = 0,
    DCM_ROTATE_RIGHT
} EN_DCM_rotation_t;

typedef struct
{
    ST_MGPIOx_RegistersMap_t  *DCM_5vPort;
    EN_MGPIO_pinOptions_t     DCM_5vPin;
    ST_MGPIOx_RegistersMap_t  *DCM_gndPort;
    EN_MGPIO_pinOptions_t     DCM_gndPin;
    EN_DCM_states_t       DCM_intialState;
    EN_DCM_direction_t    DCM_defaultDirection;
}ST_DCM_cfg_t;


#endif /* HAL_DCM_DCM_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/HAL/dcm/dcm_interface.h File Reference

Header file for the Direct Current Motor (DCM) module.

```
#include "../../COMMON/std_types.h"
#include "../../COMMON/bit_math.h"
#include "../../MCAL/gpio/gpio_interface.h"
#include "../../MCAL/tmr/tmr_interface.h"
#include "dcm_config.h"
```

Functions

- **EN_DCM_systemState_t HDCM_init** (**ST_DCM_cfg_t** *dcmCfg)
  Initialize the DC motor.
- **EN_DCM_systemState_t HDCM_turnOff** (**ST_DCM_cfg_t** *dcmCfg)
  Turn off the DC motor.
- **EN_DCM_systemState_t HDCM_turnOn** (**ST_DCM_cfg_t** *dcmCfg)
  Turn on the DC motor.
- **EN_DCM_systemState_t HDCM_changeDirection** (**ST_DCM_cfg_t** *dcmCfg, **EN_DCM_direction_t** dcmDirection)
  Change the direction of the DC motor.
- **EN_DCM_systemState_t HDCM_controlSpeed** (**uint8_t** copy_u8Speed)
  Control the speed of the DC motor.

Detailed Description

Header file for the Direct Current Motor (DCM) module.

Function Documentation

EN_DCM_systemState_t HDCM_changeDirection (ST_DCM_cfg_t * *dcmCfg*, EN_DCM_direction_t *dcmDirection*)

Change the direction of the DC motor.

This function changes the direction of the DC motor.

Parameters

dcmCfg	Pointer to the configuration structure of the DC motor.
dcmDirection	The new direction for the DC motor.

Returns

The system state after changing the direction of the DC motor.

- **DCM_OK**: DC motor direction changed successfully.
- **DCM_NOK**: Failed to change the DC motor direction.

EN_DCM_systemState_t HDCM_controlSpeed (uint8_t *copy_u8Speed*)

Control the speed of the DC motor.

This function controls the speed of the DC motor.

Parameters

copy_u8Speed	The desired speed value for the DC motor.

Returns

The system state after controlling the speed of the DC motor.

- **DCM_OK**: DC motor speed controlled successfully.
- **DCM_NOK**: Failed to control the DC motor speed.

EN_DCM_systemState_t HDCM_init (ST_DCM_cfg_t * *dcmCfg*)

Initialize the DC motor.

This function initializes the DC motor based on the provided configuration.

Parameters

dcmCfg	Pointer to the configuration structure of the DC motor.

Returns

The system state after initializing the DC motor.

- **DCM_OK**: DC motor initialization successful.
- **DCM_NOK**: DC motor initialization failed.

EN_DCM_systemState_t HDCM_turnOff (ST_DCM_cfg_t * *dcmCfg*)

Turn off the DC motor.

This function turns off the DC motor.

Parameters

dcmCfg	Pointer to the configuration structure of the DC motor.

Returns

The system state after turning off the DC motor.

- **DCM_OK**: DC motor turned off successfully.
- **DCM_NOK**: Failed to turn off the DC motor.

EN_DCM_systemState_t HDCM_turnOn (ST_DCM_cfg_t * *dcmCfg*)

Turn on the DC motor.

This function turns on the DC motor.

Parameters

dcmCfg	Pointer to the configuration structure of the DC motor.

Returns

The system state after turning on the DC motor.

- **DCM_OK**: DC motor turned on successfully.
- **DCM_NOK**: Failed to turn on the DC motor.

dcm_interface.h

Go to the documentation of this file.

```
#ifndef HAL_DCM_DCM_INTERFACE_H_
#define HAL_DCM_DCM_INTERFACE_H_

#include "../../COMMON/std_types.h"
#include "../../COMMON/bit_math.h"
#include "../../MCAL/gpio/gpio_interface.h"
#include "../../MCAL/tmr/tmr_interface.h"
#include "dcm_config.h"

EN_DCM_systemState_t HDCM_init(ST_DCM_cfg_t *dcmCfg);

EN_DCM_systemState_t HDCM_turnOff(ST_DCM_cfg_t *dcmCfg);

EN_DCM_systemState_t HDCM_turnOn(ST_DCM_cfg_t *dcmCfg);

EN_DCM_systemState_t HDCM_changeDirection(ST_DCM_cfg_t *dcmCfg, EN_DCM_direction_t
dcmDirection);

EN_DCM_systemState_t HDCM_controlSpeed(uint8_t copy_u8Speed);
#endif /* HAL_DCM_DCM_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/HAL/hc05/hc05_config.h File Reference

Enumerations

- enum **EN_HHC05_systeamState_t** { **HHC05_NOK** = 0, **HHC05_OK**, **HHC05_PTR_NULL** }
  Enumeration for the system states of the HHC05 module.

Enumeration Type Documentation

enum EN_HHC05_systeamState_t

Enumeration for the system states of the HHC05 module.

Enumerator:

HHC05_NOK	HHC05 module operation unsuccessful.
HHC05_OK	HHC05 module operation successful.
HHC05_PTR_NULL	Null pointer encountered during the operation.

hc05_config.h

Go to the documentation of this file.

```
/*************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : Adaptive_Cruise_Control
// File           : hc05_config.h
// Date           : Oct 18, 2023
// GitHub         : https://github.com/sherifkhadr
/*************************************************************************/
#ifndef HAL_HC05_HC05_CONFIG_H_
#define HAL_HC05_HC05_CONFIG_H_

typedef enum
{
    HHC05_NOK = 0,
    HHC05_OK,
    HHC05_PTR_NULL
} EN_HHC05_systeamState_t;

#endif /* HAL_HC05_HC05_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/HAL/hc05/hc05_interface.h File Reference

Header file for the Bluetooth Module (HHC05) module.

```
#include "../../MCAL/usart/usart_interface.h"
#include "hc05_config.h"
```

Functions

- **EN_HHC05_systeamState_t HHC05_uddtInit** (**ST_MUART_RegistersMap_t** *PS_USARTx, **ST_MUSART_cfg_t** const *PS_uddtUartCfg)
  Initialize the HHC05 module using the specified UART configuration.

- **EN_HHC05_systeamState_t HHC05_uddtEnable** (**ST_MUART_RegistersMap_t** *PS_USARTx)
  Enable the HHC05 module.

- **EN_HHC05_systeamState_t HHC05_uddtDisable** (**ST_MUART_RegistersMap_t** *PS_USARTx)
  Disable the HHC05 module.

- **EN_HHC05_systeamState_t HHC05_uddtTransmitByte** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** copy_u8ByteToSend)
  Transmit a byte through the HHC05 module.

- **EN_HHC05_systeamState_t HHC05_uddtTransmitString** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** *copy_u8StringToSend)
  Transmit a string through the HHC05 module.

- **EN_HHC05_systeamState_t HHC05_uddtReadDataRegister** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** *copy_u8ByteToReceive)
  Read a byte from the HHC05 module data register.

- **EN_HHC05_systeamState_t HHC05_uddtReceiveByteSynchNonBlocking** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** *copy_u8ByteToReceive)
  Receive a byte synchronously (non-blocking) from the HHC05 module.

- **EN_HHC05_systeamState_t HHC05_uddtReceiveStringSynchNonBlocking** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** *copy_u8ByteToReceive)
  Receive a string synchronously (non-blocking) from the HHC05 module.

- **EN_HHC05_systeamState_t HHC05_uddtReceiveStringAsynchBlocking** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** *copy_u8ByteToReceive)
  Receive a string asynchronously (blocking) from the HHC05 module.

- **EN_HHC05_systeamState_t HHC05_RxIntSetStatus** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** copy_u8Status)
  Set the status of the receive interrupt for the HHC05 module.

- **EN_HHC05_systeamState_t HHC05_uddtSetCallBackUart1** (void(*ptr)(void))
  Set the callback function for UART1 communication with the HHC05 module.

- **EN_HHC05_systeamState_t HHC05_uddtSetCallBackUart2** (void(*ptr)(void))
  Set the callback function for UART2 communication with the HHC05 module.

- **EN_HHC05_systeamState_t HHC05_uddtSetCallBackUart6** (void(*ptr)(void))
  Set the callback function for UART6 communication with the HHC05 module.

Detailed Description

Header file for the Bluetooth Module (HHC05) module.

Function Documentation

EN_HHC05_systeamState_t HHC05_RxIntSetStatus (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t *copy_u8Status*)

Set the status of the receive interrupt for the HHC05 module.

This function sets the status of the receive interrupt for the HHC05 module.

Parameters

PS_USARTx	Pointer to the UART registers map.
copy_u8Status	The status to set for the receive interrupt.

Returns

The system state after setting the interrupt status.

- **HHC05_OK**: Interrupt status setting successful.
- **HHC05_NOK**: Interrupt status setting failed.

EN_HHC05_systeamState_t HHC05_uddtDisable (ST_MUART_RegistersMap_t * *PS_USARTx*)

Disable the HHC05 module.

This function disables the HHC05 module.

Parameters

PS_USARTx	Pointer to the UART registers map.

Returns

The system state after disabling the HHC05 module.

- **HHC05_OK**: HHC05 module disabling successful.
- **HHC05_NOK**: HHC05 module disabling failed.

EN_HHC05_systeamState_t HHC05_uddtEnable (ST_MUART_RegistersMap_t * *PS_USARTx*)

Enable the HHC05 module.

This function enables the HHC05 module.

Parameters

PS_USARTx	Pointer to the UART registers map.

Returns

The system state after enabling the HHC05 module.

- **HHC05_OK**: HHC05 module enabling successful.
- **HHC05_NOK**: HHC05 module enabling failed.

EN_HHC05_systeamState_t HHC05_uddtInit (ST_MUART_RegistersMap_t * *PS_USARTx*, ST_MUSART_cfg_t const * *PS_uddtUartCfg*)

Initialize the HHC05 module using the specified UART configuration.

This function initializes the HHC05 module using the provided UART configuration.

Parameters

PS_USARTx	Pointer to the UART registers map.
PS_uddtUartCfg	Pointer to the UART configuration structure.

Returns

The system state after initializing the HHC05 module.

- **HHC05_OK**: HHC05 module initialization successful.
- **HHC05_NOK**: HHC05 module initialization failed.
- **HHC05_PTR_NULL**: Null pointer encountered during the operation.

EN_HHC05_systeamState_t HHC05_uddtReadDataRegister (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t * *copy_u8ByteToReceive*)

Read a byte from the HHC05 module data register.

This function reads a byte from the HHC05 module data register.

Parameters

PS_USARTx	Pointer to the UART registers map.
copy_u8ByteToReceive	Pointer to store the received byte.

Returns

The system state after reading the byte.

- **HHC05_OK**: Byte reading successful.
- **HHC05_NOK**: Byte reading failed.

EN_HHC05_systeamState_t HHC05_uddtReceiveByteSynchNonBlocking (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t * *copy_u8ByteToReceive*)

Receive a byte synchronously (non-blocking) from the HHC05 module.

This function receives a byte synchronously (non-blocking) from the HHC05 module.

Parameters

PS_USARTx	Pointer to the UART registers map.
copy_u8ByteToReceive	Pointer to store the received byte.

Returns

The system state after receiving the byte.

- **HHC05_OK**: Byte reception successful.

- **HHC05_NOK**: Byte reception failed.

EN_HHC05_systeamState_t HHC05_uddtReceiveStringAsynchBlocking (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t * *copy_u8ByteToReceive*)

Receive a string asynchronously (blocking) from the HHC05 module.

This function receives a string asynchronously (blocking) from the HHC05 module.

Parameters

PS_USARTx	Pointer to the UART registers map.
copy_u8ByteToReceive	Pointer to store the received string.

Returns

The system state after receiving the string.

- **HHC05_OK**: String reception successful.
- **HHC05_NOK**: String reception failed.

EN_HHC05_systeamState_t HHC05_uddtReceiveStringSynchNonBlocking (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t * *copy_u8ByteToReceive*)

Receive a string synchronously (non-blocking) from the HHC05 module.

This function receives a string synchronously (non-blocking) from the HHC05 module.

Parameters

PS_USARTx	Pointer to the UART registers map.
copy_u8ByteToReceive	Pointer to store the received string.

Returns

The system state after receiving the string.

- **HHC05_OK**: String reception successful.
- **HHC05_NOK**: String reception failed.

EN_HHC05_systeamState_t HHC05_uddtSetCallBackUart1 (void(*)(void) *ptr*)

Set the callback function for UART1 communication with the HHC05 module.

This function sets the callback function for UART1 communication with the HHC05 module.

Parameters

ptr	Pointer to the callback function.

Returns

The system state after setting the callback function.

- **HHC05_OK**: Callback function setting successful.
- **HHC05_NOK**: Callback function setting failed.

EN_HHC05_systeamState_t HHC05_uddtSetCallBackUart2 (void(*)(void) *ptr*)

Set the callback function for UART2 communication with the HHC05 module.

This function sets the callback function for UART2 communication with the HHC05 module.

Parameters

ptr	Pointer to the callback function.

Returns

The system state after setting the callback function.

- **HHC05_OK**: Callback function setting successful.
- **HHC05_NOK**: Callback function setting failed.

EN_HHC05_systeamState_t HHC05_uddtSetCallBackUart6 (void(*)(void) *ptr*)

Set the callback function for UART6 communication with the HHC05 module.

This function sets the callback function for UART6 communication with the HHC05 module.

Parameters

ptr	Pointer to the callback function.

Returns

The system state after setting the callback function.

- **HHC05_OK**: Callback function setting successful.
- **HHC05_NOK**: Callback function setting failed.

EN_HHC05_systeamState_t HHC05_uddtTransmitByte (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t *copy_u8ByteToSend*)

Transmit a byte through the HHC05 module.

This function transmits a byte through the HHC05 module.

Parameters

PS_USARTx	Pointer to the UART registers map.
copy_u8ByteToSend	The byte to transmit.

Returns

The system state after transmitting the byte.

- **HHC05_OK**: Byte transmission successful.
- **HHC05_NOK**: Byte transmission failed.

EN_HHC05_systeamState_t HHC05_uddtTransmitString (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t * *copy_u8StringToSend*)

Transmit a string through the HHC05 module.

This function transmits a string through the HHC05 module.

Parameters

PS_USARTx	Pointer to the UART registers map.
copy_u8StringToSend	The string to transmit.

Returns

The system state after transmitting the string.

- **HHC05_OK**: String transmission successful.
- **HHC05_NOK**: String transmission failed.

hc05_interface.h

Go to the documentation of this file.

```c

#ifndef HAL_HC05_HC05_INTERFACE_H_
#define HAL_HC05_HC05_INTERFACE_H_

#include "../../MCAL/usart/usart_interface.h"
#include "hc05_config.h"


EN_HHC05_systeamState_t HHC05_uddtInit(ST_MUART_RegistersMap_t *PS_USARTx, ST_MUSART_cfg_t const *PS_uddtUartCfg);

EN_HHC05_systeamState_t HHC05_uddtEnable(ST_MUART_RegistersMap_t *PS_USARTx);

EN_HHC05_systeamState_t HHC05_uddtDisable(ST_MUART_RegistersMap_t *PS_USARTx);

EN_HHC05_systeamState_t HHC05_uddtTransmitByte(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t copy_u8ByteToSend);

EN_HHC05_systeamState_t HHC05_uddtTransmitString(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t *copy_u8StringToSend);

EN_HHC05_systeamState_t HHC05_uddtReadDataRegister(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t *copy_u8ByteToReceive);

EN_HHC05_systeamState_t HHC05_uddtReceiveByteSynchNonBlocking(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t *copy_u8ByteToReceive);

EN_HHC05_systeamState_t HHC05_uddtReceiveStringSynchNonBlocking(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t *copy_u8ByteToReceive);

EN_HHC05_systeamState_t HHC05_uddtReceiveStringAsynchBlocking(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t *copy_u8ByteToReceive);

EN_HHC05_systeamState_t HHC05_RxIntSetStatus(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t copy_u8Status);

EN_HHC05_systeamState_t HHC05_uddtSetCallBackUart1(void (*ptr)(void));

EN_HHC05_systeamState_t HHC05_uddtSetCallBackUart2(void (*ptr)(void));

EN_HHC05_systeamState_t HHC05_uddtSetCallBackUart6(void (*ptr)(void));



#endif /* HAL_HC05_HC05_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/HAL/ir/IR_config.h File Reference

Data Structures

struct **ST_HIR_cfg_t***Configuration structure for the Human Interface Receiver (HIR) module.*

Enumerations

- enum **EN_HIR_systemState_t** { **HIR_NOK** = 0, **HIR_OK**, **HIR_PTR_NULL** }
  Enumeration for the system state of the Human Interface Receiver (HIR) module.
- enum **EN_HIR_surfaceState_t** { **HIR_ON_WHITE** = 0, **HIR_ON_BLACK** }
  Enumeration for the surface state of the Human Interface Receiver (HIR) module.

Enumeration Type Documentation

enum EN_HIR_surfaceState_t

Enumeration for the surface state of the Human Interface Receiver (HIR) module.

This enumeration defines possible surface states for the HIR module.

Enumerator:

HIR_ON_WHITE	HIR module is on a white surface.
HIR_ON_BLACK	HIR module is on a black surface.

enum EN_HIR_systemState_t

Enumeration for the system state of the Human Interface Receiver (HIR) module.

This enumeration defines possible system states for the HIR module.

Enumerator:

HIR_NOK	HIR module encountered an error.
HIR_OK	HIR module operation successful.
HIR_PTR_NULL	Null pointer encountered during the operation.

IR_config.h

Go to the documentation of this file.

```c
/**************************************************************************/
// Author          : Sherif Ashraf Khadr
// Project         : Adaptive_Cruise_Control
// File            : IR_config.h
// Date            : Nov 7, 2023
// GitHub          : https://github.com/sherifkhadr
/**************************************************************************/
#ifndef HAL_IR_IR_CONFIG_H_
#define HAL_IR_IR_CONFIG_H_

typedef enum
{
    HIR_NOK = 0,
    HIR_OK,
    HIR_PTR_NULL
} EN_HIR_systemState_t;

typedef enum
{
    HIR_ON_WHITE = 0,
    HIR_ON_BLACK
} EN_HIR_surfaceState_t;

typedef struct
{
    ST_MGPIOx_RegistersMap_t *HIR_port;
    EN_MGPIO_pinOptions_t    HIR_pin;
} ST_HIR_cfg_t;

#endif /* HAL_IR_IR_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/HAL/ir/IR_interface.h File Reference

Header file for the Infrared Sensor (HIR) module.

```
#include "../../../Inc/COMMON/std_types.h"
#include "../../../Inc/MCAL/gpio/gpio_interface.h"
#include "IR_config.h"
```

Functions

- **EN_HIR_systemState_t HIR_uddtInit** (**ST_HIR_cfg_t** *PS_uddtIrInstance)
  Initialize the Human Interface Receiver (HIR) module.

- **EN_HIR_systemState_t HIR_uddtReadData** (**ST_HIR_cfg_t** *PS_uddtIrInstance, **EN_HIR_surfaceState_t** *copy_uddtRetOfIrRead)
  Read data from the Human Interface Receiver (HIR) module.

Detailed Description

Header file for the Infrared Sensor (HIR) module.

Function Documentation

EN_HIR_systemState_t HIR_uddtInit (ST_HIR_cfg_t * *PS_uddtIrInstance*)

Initialize the Human Interface Receiver (HIR) module.

This function initializes the HIR module based on the provided configuration.

Parameters

PS_uddtIrInstance	Pointer to the configuration structure for the HIR module.

Returns

The system state after initializing the HIR module.

- **HIR_OK**: HIR module initialization successful.
- **HIR_NOK**: HIR module initialization failed.
- **HIR_PTR_NULL**: Null pointer encountered during the operation.

EN_HIR_systemState_t HIR_uddtReadData (ST_HIR_cfg_t * *PS_uddtIrInstance*, EN_HIR_surfaceState_t * *copy_uddtRetOfIrRead*)

Read data from the Human Interface Receiver (HIR) module.

This function reads data from the HIR module and provides the surface state.

Parameters

PS_uddtIrInstance	Pointer to the configuration structure for the HIR module.
copy_uddtRetOfIrRead	Pointer to store the retrieved surface state.

Returns

The system state after reading data from the HIR module.

- **HIR_OK**: Data reading successful.
- **HIR_NOK**: Data reading failed.
- **HIR_PTR_NULL**: Null pointer encountered during the operation.

IR_interface.h

Go to the documentation of this file.

```

#ifndef HAL_IR_IR_INTERFACE_H_
#define HAL_IR_IR_INTERFACE_H_

#include "../../../Inc/COMMON/std_types.h"
#include "../../../Inc/MCAL/gpio/gpio_interface.h"
#include "IR_config.h"

EN_HIR_systemState_t HIR_uddtInit(ST_HIR_cfg_t *PS_uddtIrInstance);

EN_HIR_systemState_t HIR_uddtReadData(ST_HIR_cfg_t *PS_uddtIrInstance,
EN_HIR_surfaceState_t *copy_uddtRetOfIrRead);
#endif /* HAL_IR_IR_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/HAL/ultrasonic/Delay.h File Reference

```
#include "../../COMMON/std_types.h"
```

Functions

- void **delay_us** (**uint32_t** microseconds)

Function Documentation

void delay_us (uint32_t *microseconds*)

Delay.h

Go to the documentation of this file.

```
/*
 * Delay.h
 *
 *  Created on: Nov 7, 2023
 *      Author: Omar Abouzaid
 */

#ifndef DELAY_H_
#define DELAY_H_

#include "../../COMMON/std_types.h"

void delay_us(uint32_t microseconds);

#endif /* DELAY_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/HAL/ultrasonic/ult_config.h File Reference

Macros

- #define **TRIG_PORT   MGPIOA_PERIPHERAL**
- #define **TRIG_PIN   MGPIO_PIN9**
- #define **ECHO_PORT   MGPIOA_PERIPHERAL**
- #define **ECHO_PIN   MGPIO_PIN10**

Macro Definition Documentation

#define ECHO_PIN   MGPIO_PIN10

#define ECHO_PORT   MGPIOA_PERIPHERAL

#define TRIG_PIN   MGPIO_PIN9

#define TRIG_PORT   MGPIOA_PERIPHERAL

ult_config.h

Go to the documentation of this file.

```c
/*
 * ULTRASONIC_Config.h
 *
 *  Created on: Nov 7, 2023
 *      Author: Omar Abouzaid
 */

#ifndef SERVO_CONFIG_H_
#define SERVO_CONFIG_H_


 /*TRIG PIN CONFIG*/
 #define TRIG_PORT   MGPIOA_PERIPHERAL
 #define TRIG_PIN    MGPIO_PIN9


 /*ECHO PIN CONFIG*/
 #define ECHO_PORT   MGPIOA_PERIPHERAL
 #define ECHO_PIN    MGPIO_PIN10


 #endif
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/HAL/ultrasonic/ult_interface .h File Reference

Functions

- void **UltraSonic_Init** (void)
- void **UltraSonic_Send_Pulse** (void)
  Send a pulse on the Trig pin.
- **uint32_t UltraSonic_Measure_Distance** (void)
  Measure the distance using the Ultrasonic sensor.

Function Documentation

void UltraSonic_Init (void)

uint32_t UltraSonic_Measure_Distance (void)

Measure the distance using the Ultrasonic sensor.

This function measures the distance using the Ultrasonic sensor. It returns the calculated distance value in centimeters.

Returns

The measured distance in centimeters.

void UltraSonic_Send_Pulse (void)

Send a pulse on the Trig pin.

This function triggers the Ultrasonic sensor to send a pulse on its Trig pin. It is used to initiate the distance measurement process.

Returns

No return.

ult_interface.h

Go to the documentation of this file.

```
/*
 * ULTRASONIC_Interface.h
 *
 *  Created on: Nov 7, 2023
 *      Author: Omar Abouzaid
 */

#ifndef ULTRASONIC_INTERFACE_H_
#define ULTRASONIC_INTERFACE_H_

void UltraSonic_Init(void);

void UltraSonic_Send_Pulse(void);

uint32_t UltraSonic_Measure_Distance(void);

#endif /* ULTRASONIC_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/exti/exti_config.h File Reference

Enumerations

- enum **EN_MEXTI_systemState_t** { **MEXTI_OK** = 0, **MEXTI_NOK**, **MEXTI_INVALID_PARAMTER** }
  Enumeration for the system state of EXTI functions.
- enum **EN_MEXTI_triggerOptions_t** { **MEXTI_RISING_EDGE** = 0, **MEXTI_FALLING_EDGE**, **MEXTI_ON_CHANGE** }
- enum **EN_MEXTI_lines_t** { **MEXTI_LINE0** = 0, **MEXTI_LINE1**, **MEXTI_LINE2**, **MEXTI_LINE3**, **MEXTI_LINE4**, **MEXTI_LINE5**, **MEXTI_LINE6**, **MEXTI_LINE7**, **MEXTI_LINE8**, **MEXTI_LINE9**, **MEXTI_LINE10**, **MEXTI_LINE11**, **MEXTI_LINE12**, **MEXTI_LINE13**, **MEXTI_LINE14**, **MEXTI_LINE15**, **MEXTI_LINE16**, **MEXTI_LINE17**, **MEXTI_LINE18**, **MEXTI_LINE19**, **MEXTI_LINE20**, **MEXTI_LINE21**, **MEXTI_LINE22** }
- enum **EN_MEXTI_port_t** { **MEXTI_PORTA** = 0, **MEXTI_PORTB**, **MEXTI_PORTC**, **MEXTI_PORTD**, **MEXTI_PORTE**, **MEXTI_PORTH** }

Enumeration Type Documentation

enum EN_MEXTI_lines_t

Enumerator:

MEXTI_LINE0	
MEXTI_LINE1	
MEXTI_LINE2	
MEXTI_LINE3	
MEXTI_LINE4	
MEXTI_LINE5	
MEXTI_LINE6	
MEXTI_LINE7	
MEXTI_LINE8	
MEXTI_LINE9	
MEXTI_LINE10	
MEXTI_LINE11	
MEXTI_LINE12	
MEXTI_LINE13	
MEXTI_LINE14	
MEXTI_LINE15	
MEXTI_LINE16	
MEXTI_LINE17	
MEXTI_LINE18	
MEXTI_LINE19	
MEXTI_LINE20	
MEXTI_LINE21	
MEXTI_LINE22	

enum EN_MEXTI_port_t

Enumerator:

MEXTI_PORTA	

MEXTI_PORTB	
MEXTI_PORTC	
MEXTI_PORTD	
MEXTI_PORTE	
MEXTI_PORTH	

enum EN_MEXTI_systemState_t

Enumeration for the system state of EXTI functions.

Enumerator:

MEXTI_OK	Operation successful.
MEXTI_NOK	Operation failed.
MEXTI_INVALID_PARAMTER	Invalid parameter detected.

enum EN_MEXTI_triggerOptions_t

Enumerator:

MEXTI_RISING_EDGE	
MEXTI_FALLING_EDGE	
MEXTI_ON_CHANGE	

exti_config.h

Go to the documentation of this file.

```c
/**********************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : STM32F401xC
// File          : exti_config.h
// Date          : Sep 11, 2023
// GitHub        : https://github.com/sherifkhadr
/**********************************************************************/
#ifndef MCAL_EXTI_EXTI_CONFIG_H_
#define MCAL_EXTI_EXTI_CONFIG_H_



typedef enum
{
    MEXTI_OK = 0,
    MEXTI_NOK,
    MEXTI_INVALID_PARAMTER
} EN_MEXTI_systemState_t;


typedef enum
{
    MEXTI_RISING_EDGE = 0,
    MEXTI_FALLING_EDGE,
    MEXTI_ON_CHANGE
}EN_MEXTI_triggerOptions_t;


typedef enum
{
    MEXTI_LINE0 = 0,
    MEXTI_LINE1,
    MEXTI_LINE2,
    MEXTI_LINE3,
    MEXTI_LINE4,
    MEXTI_LINE5,
    MEXTI_LINE6,
    MEXTI_LINE7,
    MEXTI_LINE8,
    MEXTI_LINE9,
    MEXTI_LINE10,
    MEXTI_LINE11,
    MEXTI_LINE12,
    MEXTI_LINE13,
    MEXTI_LINE14,
    MEXTI_LINE15,
    MEXTI_LINE16,
    MEXTI_LINE17,
    MEXTI_LINE18,
    MEXTI_LINE19,
    MEXTI_LINE20,
    MEXTI_LINE21,
    MEXTI_LINE22
}EN_MEXTI_lines_t;


typedef enum
{
    MEXTI_PORTA = 0,
    MEXTI_PORTB,
    MEXTI_PORTC,
    MEXTI_PORTD,
    MEXTI_PORTE,
    MEXTI_PORTH
}EN_MEXTI_port_t;

#endif /* MCAL_EXTI_EXTI_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/exti/exti_interface.h File Reference

Header file for the EXTI (External Interrupt) module interface.

```
#include "../../COMMON/bit_math.h"
#include "../../COMMON/std_types.h"
#include "exti_private.h"
#include "exti_config.h"
```

Functions

- **EN_MEXTI_systemState_t MEXTI_enableEXTI** (**EN_MEXTI_lines_t** copy_uddtLineNum)
  Enable the EXTI line for a specific GPIO pin.
- **EN_MEXTI_systemState_t MEXTI_disableEXTI** (**EN_MEXTI_lines_t** copy_uddtLineNum)
  Disable the EXTI line for a specific GPIO pin.
- **EN_MEXTI_systemState_t MEXTI_softwareInterrupt** (**EN_MEXTI_lines_t** copy_uddtLineNum)
  Generate a software interrupt on the specified EXTI line.
- **EN_MEXTI_systemState_t MEXTI_setTriggerSource** (**EN_MEXTI_lines_t** copy_uddtLineNum, **EN_MEXTI_triggerOptions_t** copy_uddtTriggerOption)
  Set the trigger source for the specified EXTI line.
- **EN_MEXTI_systemState_t MEXTI_setExtiConfig** (**EN_MEXTI_lines_t** copy_uddtLineNum, **EN_MEXTI_port_t** copy_uddtPortNum)
  Set the EXTI configuration for the specified GPIO pin.
- **EN_MEXTI_systemState_t MEXTI_setCallBack** (void(*ptr)(void), **EN_MEXTI_lines_t** copy_uddtLineNum)
  Set the callback function for the specified EXTI line.

Detailed Description

Header file for the EXTI (External Interrupt) module interface.

Function Documentation

EN_MEXTI_systemState_t MEXTI_disableEXTI (EN_MEXTI_lines_t *copy_uddtLineNum*)

Disable the EXTI line for a specific GPIO pin.

This function disables the EXTI line for a specified GPIO pin.

Parameters

copy_uddtLineNum	The EXTI line number to disable. Possible values are: • #EN_MEXTI_LINE0 • #EN_MEXTI_LINE1 • ... • #EN_MEXTI_LINE15

Returns

The system state after disabling the EXTI line.

- #EN_MEXTI_OK: EXTI line disabling successful.
- #EN_MEXTI_NOK: EXTI line disabling failed.
- #EN_MEXTI_INVALID_PARAMTER: Invalid parameter detected during the operation.

EN_MEXTI_systemState_t MEXTI_enableEXTI (EN_MEXTI_lines_t *copy_uddtLineNum*)

Enable the EXTI line for a specific GPIO pin.

This function enables the EXTI line for a specified GPIO pin.

Parameters

copy_uddtLineNum	The EXTI line number to enable. Possible values are: • #EN_MEXTI_LINE0 • #EN_MEXTI_LINE1 • ... • #EN_MEXTI_LINE15

Returns

The system state after enabling the EXTI line.

- #EN_MEXTI_OK: EXTI line enabling successful.
- #EN_MEXTI_NOK: EXTI line enabling failed.
- #EN_MEXTI_INVALID_PARAMTER: Invalid parameter detected during the operation.

EN_MEXTI_systemState_t MEXTI_setCallBack (void(*)(void) *ptr*, EN_MEXTI_lines_t *copy_uddtLineNum*)

Set the callback function for the specified EXTI line.

This function sets the callback function for the specified EXTI line.

Parameters

ptr	Pointer to the callback function.
copy_uddtLineNum	The EXTI line number to set the callback for. Possible values are: • #EN_MEXTI_LINE0 • #EN_MEXTI_LINE1 • ... • #EN_MEXTI_LINE15

Returns

The system state after setting the callback function.

- #EN_MEXTI_OK: Callback function setting successful.
- #EN_MEXTI_NOK: Callback function setting failed.
- #EN_MEXTI_INVALID_PARAMTER: Invalid parameter detected during the operation.
- #EN_MEXTI_PTR_NULL: Null pointer encountered during the operation.

EN_MEXTI_systemState_t MEXTI_setExtiConfig (EN_MEXTI_lines_t *copy_uddtLineNum*, EN_MEXTI_port_t *copy_uddtPortNum*)

Set the EXTI configuration for the specified GPIO pin.

This function sets the EXTI configuration for the specified GPIO pin.

Parameters

copy_uddtLineNum	The EXTI line number to configure. Possible values are: • #EN_MEXTI_LINE0 • #EN_MEXTI_LINE1 • ... • #EN_MEXTI_LINE15
copy_uddtPortNum	The GPIO port number. Possible values are: • #EN_MEXTI_PORTA • #EN_MEXTI_PORTB • #EN_MEXTI_PORTC • ... • #EN_MEXTI_PORTH

Returns

The system state after setting the EXTI configuration.

- #EN_MEXTI_OK: EXTI configuration setting successful.
- #EN_MEXTI_NOK: EXTI configuration setting failed.
- #EN_MEXTI_INVALID_PARAMTER: Invalid parameter detected during the operation.

EN_MEXTI_systemState_t MEXTI_setTriggerSource (EN_MEXTI_lines_t *copy_uddtLineNum*, EN_MEXTI_triggerOptions_t *copy_uddtTriggerOption*)

Set the trigger source for the specified EXTI line.

This function sets the trigger source for the specified EXTI line.

Parameters

copy_uddtLineNum	The EXTI line number to configure. Possible values are: • #EN_MEXTI_LINE0 • #EN_MEXTI_LINE1 • ... • #EN_MEXTI_LINE15
copy_uddtTriggerOption	The trigger source option. Possible values are: • #EN_MEXTI_TRIGGER_RISING_EDGE • #EN_MEXTI_TRIGGER_FALLING_EDGE • #EN_MEXTI_TRIGGER_BOTH_EDGES

Returns

The system state after setting the trigger source.

- #EN_MEXTI_OK: Trigger source setting successful.
- #EN_MEXTI_NOK: Trigger source setting failed.
- #EN_MEXTI_INVALID_PARAMTER: Invalid parameter detected during the operation.

EN_MEXTI_systemState_t MEXTI_softwareInterrupt (EN_MEXTI_lines_t *copy_uddtLineNum*)

Generate a software interrupt on the specified EXTI line.

This function generates a software interrupt on the specified EXTI line.

Parameters

copy_uddtLineNum	The EXTI line number to trigger. Possible values are: • #EN_MEXTI_LINE0 • #EN_MEXTI_LINE1 • ... • #EN_MEXTI_LINE15

Returns

The system state after triggering the software interrupt.

- #EN_MEXTI_OK: Software interrupt triggering successful.
- #EN_MEXTI_NOK: Software interrupt triggering failed.
- #EN_MEXTI_INVALID_PARAMTER: Invalid parameter detected during the operation.

exti_interface.h

Go to the documentation of this file.

```
#ifndef MCAL_EXTI_EXTI_INTERFACE_H_
#define MCAL_EXTI_EXTI_INTERFACE_H_

#include "../../COMMON/bit_math.h"
#include "../../COMMON/std_types.h"
#include "exti_private.h"
#include "exti_config.h"

EN_MEXTI_systemState_t MEXTI_enableEXTI(EN_MEXTI_lines_t copy_uddtLineNum);

EN_MEXTI_systemState_t MEXTI_disableEXTI(EN_MEXTI_lines_t copy_uddtLineNum);

EN_MEXTI_systemState_t MEXTI_softwareInterrupt(EN_MEXTI_lines_t copy_uddtLineNum);

EN_MEXTI_systemState_t MEXTI_setTriggerSource(EN_MEXTI_lines_t copy_uddtLineNum,
EN_MEXTI_triggerOptions_t copy_uddtTriggerOption);

EN_MEXTI_systemState_t MEXTI_setExtiConfig(EN_MEXTI_lines_t copy_uddtLineNum,
EN_MEXTI_port_t copy_uddtPortNum);

EN_MEXTI_systemState_t MEXTI_setCallBack(void (*ptr)(void), EN_MEXTI_lines_t
copy_uddtLineNum);


#endif /* MCAL_EXTI_EXTI_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/exti/exti_private.h File Reference

Data Structures

- struct **ST_MEXTI_RegistersMap_t**struct **ST_MSYSCFG_RegistersMap_t**

Macros

- #define **MEXTI_PERIPHERAL_BASE_ADDR**   (0x40013C00)
- #define **MSYSCFG_PERIPHERAL_BASE_ADDR**   (0x40013800)
- #define **EXTI_CONFG_MASK1**   0xFFFFFFF0UL
- #define **EXTI_CONFG_MASK2**   0xFFFFFF0FUL
- #define **EXTI_CONFG_MASK3**   0xFFFFF0FFUL
- #define **EXTI_CONFG_MASK4**   0xFFFF0FFFUL
- #define **MEXTI_PERIPHERAL**   ((volatile **ST_MEXTI_RegistersMap_t** *)**MEXTI_PERIPHERAL_BASE_ADDR**)
- #define **MSYSCFG_PERIPHERAL**   ((volatile **ST_MSYSCFG_RegistersMap_t** *)**MSYSCFG_PERIPHERAL_BASE_ADDR**)

Macro Definition Documentation

#define EXTI_CONFG_MASK1   0xFFFFFFF0UL

#define EXTI_CONFG_MASK2   0xFFFFFF0FUL

#define EXTI_CONFG_MASK3   0xFFFFF0FFUL

#define EXTI_CONFG_MASK4   0xFFFF0FFFUL

#define MEXTI_PERIPHERAL   ((volatile ST_MEXTI_RegistersMap_t *)MEXTI_PERIPHERAL_BASE_ADDR)

#define MEXTI_PERIPHERAL_BASE_ADDR   (0x40013C00)

#define MSYSCFG_PERIPHERAL   ((volatile ST_MSYSCFG_RegistersMap_t *)MSYSCFG_PERIPHERAL_BASE_ADDR)

#define MSYSCFG_PERIPHERAL_BASE_ADDR   (0x40013800)

exti_private.h

Go to the documentation of this file.

```
/*******************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : STM32F401xC
// File           : exti_private.h
// Date           : Sep 11, 2023
// GitHub         : https://github.com/sherifkhadr
/*******************************************************************************/
#ifndef MCAL_EXTI_PRIVATE_H_
#define MCAL_EXTI_PRIVATE_H_



#define  MEXTI_PERIPHERAL_BASE_ADDR         (0x40013C00)
#define  MSYSCFG_PERIPHERAL_BASE_ADDR       (0x40013800)


#define   EXTI_CONFG_MASK1                0xFFFFFFF0UL
#define   EXTI_CONFG_MASK2                0xFFFFFF0FUL
#define   EXTI_CONFG_MASK3                0xFFFFF0FFUL
#define   EXTI_CONFG_MASK4                0xFFFF0FFFUL

typedef struct
{

    vuint32_t MEXTI_IMR;
    vuint32_t MEXTI_EMR;
    vuint32_t MEXTI_RTSR;
    vuint32_t MEXTI_FTSR;
    vuint32_t MEXTI_SWIER;
    vuint32_t MEXTI_PR;

}ST_MEXTI_RegistersMap_t;


typedef struct
{
    vuint32_t MSYSCFG_MEMRMP;
    vuint32_t MSYSCFG_PMC;
    vuint32_t MSYSCFG_EXTICR1;
    vuint32_t MSYSCFG_EXTICR2;
    vuint32_t MSYSCFG_EXTICR3;
    vuint32_t MSYSCFG_EXTICR4;
    vuint32_t MSYSCFG_CMPCR;
}ST_MSYSCFG_RegistersMap_t;


#define MEXTI_PERIPHERAL    ((volatile ST_MEXTI_RegistersMap_t
*)MEXTI_PERIPHERAL_BASE_ADDR)
#define MSYSCFG_PERIPHERAL  ((volatile ST_MSYSCFG_RegistersMap_t
*)MSYSCFG_PERIPHERAL_BASE_ADDR)


#endif /* MCAL_EXTI_PRIVATE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/gpio/gpio_config.h File Reference

Data Structures

- struct **ST_MGPIO_pinCfg_t**struct **ST_MGPIO_altPinCfg_t**

Macros

- #define **MIN_VAL_OF_U16**   0
- #define **MAX_VAL_OF_U16**   65536

Enumerations

- enum **EN_MGPIO_systemState_t** { **MGPIO_NOK** = 0, **MGPIO_OK**, **MGPIO_INVALID_PARAMTER**, **MGPIO_PTR_NULL** }
- enum **EN_MGPIO_pinOptions_t** { **MGPIO_PIN0** = 0, **MGPIO_PIN1**, **MGPIO_PIN2**, **MGPIO_PIN3**, **MGPIO_PIN4**, **MGPIO_PIN5**, **MGPIO_PIN6**, **MGPIO_PIN7**, **MGPIO_PIN8**, **MGPIO_PIN9**, **MGPIO_PIN10**, **MGPIO_PIN11**, **MGPIO_PIN12**, **MGPIO_PIN13**, **MGPIO_PIN14**, **MGPIO_PIN15** }
- enum **EN_MGPIO_pinModeOptions_t** { **MGPIO_MODE_INPUT** = 0, **MGPIO_MODE_OUTPUT**, **MGPIO_MODE_ALTF**, **MGPIO_MODE_ANALOG** }
- enum **EN_MGPIO_pinLogicOptions_t** { **MGPIO_LOGIC_LOW** = 0, **MGPIO_LOGIC_HIGH** }
- enum **EN_MGPIO_outputSpeedOptions_t** { **MGPIO_OUTPUT_SPEED_LOW** = 0, **MGPIO_OUTPUT_SPEED_MEDIUM**, **MGPIO_OUTPUT_SPEED_HIGH**, **MGPIO_OUTPUT_SPEED_VERY_HIGH** }
- enum **EN_MGPIO_outputResistorOptions_t** { **MGPIO_OUTPUT_RESISTOR_PUSH_PULL** = 0, **MGPIO_OUTPUT_RESISTOR_OPEN_DRAIN** }
- enum **EN_MGPIO_pushPullOptions_t** { **MGPIO_PULL_FLOATING** = 0, **MGPIO_PULL_PULL_UP**, **MGPIO_PULL_PULL_DOWN** }
- enum **EN_MGPIO_altfnOptions_t** { **MGPIO_ALTFN_0** = 0, **MGPIO_ALTFN_1**, **MGPIO_ALTFN_2**, **MGPIO_ALTFN_3**, **MGPIO_ALTFN_4**, **MGPIO_ALTFN_5**, **MGPIO_ALTFN_6**, **MGPIO_ALTFN_7**, **MGPIO_ALTFN_8**, **MGPIO_ALTFN_9**, **MGPIO_ALTFN_10**, **MGPIO_ALTFN_11**, **MGPIO_ALTFN_12**, **MGPIO_ALTFN_13**, **MGPIO_ALTFN_14**, **MGPIO_ALTFN_15** }
- enum **EN_MGPIO_setResetOptions_t** { **MGPIO_PIN_RESET** = 0, **MGPIO_PIN_SET** }

Macro Definition Documentation

#define MAX_VAL_OF_U16   65536

#define MIN_VAL_OF_U16   0

Enumeration Type Documentation

enum EN_MGPIO_altfnOptions_t

Enumerator:

MGPIO_ALTFN_0	

MGPIO_ALTFN_1	
MGPIO_ALTFN_2	
MGPIO_ALTFN_3	
MGPIO_ALTFN_4	
MGPIO_ALTFN_5	
MGPIO_ALTFN_6	
MGPIO_ALTFN_7	
MGPIO_ALTFN_8	
MGPIO_ALTFN_9	
MGPIO_ALTFN_10	
MGPIO_ALTFN_11	
MGPIO_ALTFN_12	
MGPIO_ALTFN_13	
MGPIO_ALTFN_14	
MGPIO_ALTFN_15	

enum EN_MGPIO_outputResistorOptions_t

Enumerator:

MGPIO_OUTPUT_RESISTOR_PUSH_PULL	
MGPIO_OUTPUT_RESISTOR_OPEN_DRAIN	

enum EN_MGPIO_outputSpeedOptions_t

Enumerator:

MGPIO_OUTPUT_SPEED_LOW	
MGPIO_OUTPUT_SPEED_MEDIUM	
MGPIO_OUTPUT_SPEED_HIGH	
MGPIO_OUTPUT_SPEED_VERY_HIGH	

enum EN_MGPIO_pinLogicOptions_t

Enumerator:

MGPIO_LOGIC_LOW	
MGPIO_LOGIC_HIGH	

enum EN_MGPIO_pinModeOptions_t

Enumerator:

MGPIO_MODE_INPUT	
MGPIO_MODE_OUTPUT	
MGPIO_MODE_ALTF	
MGPIO_MODE_ANALOG	

enum EN_MGPIO_pinOptions_t

Enumerator:

MGPIO_PIN0	
MGPIO_PIN1	
MGPIO_PIN2	
MGPIO_PIN3	
MGPIO_PIN4	
MGPIO_PIN5	
MGPIO_PIN6	
MGPIO_PIN7	
MGPIO_PIN8	
MGPIO_PIN9	
MGPIO_PIN10	
MGPIO_PIN11	
MGPIO_PIN12	
MGPIO_PIN13	
MGPIO_PIN14	
MGPIO_PIN15	

enum EN_MGPIO_pushPullOptions_t

Enumerator:

MGPIO_PULL_FLOATING	
MGPIO_PULL_PULL_UP	
MGPIO_PULL_PULL_DOWN	

enum EN_MGPIO_setResetOptions_t

Enumerator:

MGPIO_PIN_RESET	
MGPIO_PIN_SET	

enum EN_MGPIO_systemState_t

Enumerator:

MGPIO_NOK	
MGPIO_OK	
MGPIO_INVALID_PARAMTER	
MGPIO_PTR_NULL	

gpio_config.h

Go to the documentation of this file.

```c
/*************************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : STM32F401xC
// File          : gpio_config.h
// Date          : Sep 10, 2023
// GitHub        : https://github.com/sherifkhadr
/*************************************************************************/
#ifndef MCAL_GPIO_GPIO_CONFIG_H_
#define MCAL_GPIO_GPIO_CONFIG_H_


 #define MIN_VAL_OF_U16              0
 #define MAX_VAL_OF_U16              65536

 typedef enum
 {
     MGPIO_NOK = 0,
     MGPIO_OK,
     MGPIO_INVALID_PARAMTER,
     MGPIO_PTR_NULL
 }EN_MGPIO_systemState_t;


 typedef enum
 {
     MGPIO_PIN0 = 0,
     MGPIO_PIN1,
     MGPIO_PIN2,
     MGPIO_PIN3,
     MGPIO_PIN4,
     MGPIO_PIN5,
     MGPIO_PIN6,
     MGPIO_PIN7,
     MGPIO_PIN8,
     MGPIO_PIN9,
     MGPIO_PIN10,
     MGPIO_PIN11,
     MGPIO_PIN12,
     MGPIO_PIN13,
     MGPIO_PIN14,
     MGPIO_PIN15,

 }EN_MGPIO_pinOptions_t;


 typedef enum
 {
     MGPIO_MODE_INPUT = 0,
     MGPIO_MODE_OUTPUT,
     MGPIO_MODE_ALTF,
     MGPIO_MODE_ANALOG

 }EN_MGPIO_pinModeOptions_t;


 typedef enum
 {
     MGPIO_LOGIC_LOW = 0,
     MGPIO_LOGIC_HIGH
 }EN_MGPIO_pinLogicOptions_t;


 typedef enum
 {
     MGPIO_OUTPUT_SPEED_LOW = 0,
     MGPIO_OUTPUT_SPEED_MEDIUM,
     MGPIO_OUTPUT_SPEED_HIGH,
     MGPIO_OUTPUT_SPEED_VERY_HIGH
 }EN_MGPIO_outputSpeedOptions_t;

 typedef enum
 {
     MGPIO_OUTPUT_RESISTOR_PUSH_PULL = 0,
     MGPIO_OUTPUT_RESISTOR_OPEN_DRAIN
 }EN_MGPIO_outputResistorOptions_t;


 typedef enum
 {
     MGPIO_PULL_FLOATING = 0,
     MGPIO_PULL_PULL_UP,
     MGPIO_PULL_PULL_DOWN
 }EN_MGPIO_pushPullOptions_t;


 typedef enum
 {
     MGPIO_ALTFN_0 = 0,
     MGPIO_ALTFN_1 ,
     MGPIO_ALTFN_2 ,
     MGPIO_ALTFN_3 ,
     MGPIO_ALTFN_4 ,
     MGPIO_ALTFN_5 ,
     MGPIO_ALTFN_6 ,
     MGPIO_ALTFN_7 ,
     MGPIO_ALTFN_8 ,
     MGPIO_ALTFN_9 ,
     MGPIO_ALTFN_10,
     MGPIO_ALTFN_11,
     MGPIO_ALTFN_12,
     MGPIO_ALTFN_13,
     MGPIO_ALTFN_14,
     MGPIO_ALTFN_15
 }EN_MGPIO_altfnOptions_t;


 typedef enum
 {
     MGPIO_PIN_RESET = 0,
     MGPIO_PIN_SET
 }EN_MGPIO_setResetOptions_t;


 typedef struct
 {
     ST_MGPIOx_RegistersMap_t  *PS_GPIOx;
     EN_MGPIO_pinOptions_t copy_uddtPinNum;
     EN_MGPIO_pinModeOptions_t copy_uddtPinMode;
     EN_MGPIO_outputResistorOptions_t copy_uddtOutputResistor;
     EN_MGPIO_outputSpeedOptions_t copy_uddtOutputSpeed;
     EN_MGPIO_pinLogicOptions_t copy_uddtPtrRetOfPinLogic;
     EN_MGPIO_pushPullOptions_t copy_uddtPullState;
 }ST_MGPIO_pinCfg_t;

 typedef struct
 {
     ST_MGPIOx_RegistersMap_t  *PS_GPIOx;
     EN_MGPIO_pinOptions_t copy_uddtPinNum;
     EN_MGPIO_altfnOptions_t Copy_uddtAltFun;
     EN_MGPIO_outputResistorOptions_t copy_uddtOutputResistor;
     EN_MGPIO_outputSpeedOptions_t copy_uddtOutputSpeed;
     EN_MGPIO_pushPullOptions_t copy_uddtPullState;
 }ST_MGPIO_altPinCfg_t;

 #endif /* MCAL_GPIO_GPIO_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/gpio/gpio_interface.h File Reference

Header file for GPIO (General Purpose I/O) module.

```
#include "../../COMMON/bit_math.h"
#include "../../COMMON/std_types.h"
#include "gpio_private.h"
#include "gpio_config.h"
```

Functions

- **EN_MGPIO_systemState_t MGPIO_uddtSetPinMode** (**ST_MGPIOx_RegistersMap_t** *PS_GPIOx, **EN_MGPIO_pinOptions_t** copy_uddtPinNum, **EN_MGPIO_pinModeOptions_t** copy_uddtPinMode)
  Set the mode of a GPIO pin.

- **EN_MGPIO_systemState_t MGPIO_uddtSetOutputMode** (**ST_MGPIOx_RegistersMap_t** *PS_GPIOx, **EN_MGPIO_pinOptions_t** copy_uddtPinNum, **EN_MGPIO_outputResistorOptions_t** copy_uddtOutputResistor)
  Set the output mode of a GPIO pin.

- **EN_MGPIO_systemState_t MGPIO_uddtSetOutputSpeed** (**ST_MGPIOx_RegistersMap_t** *PS_GPIOx, **EN_MGPIO_pinOptions_t** copy_uddtPinNum, **EN_MGPIO_outputSpeedOptions_t** copy_uddtOutputSpeed)
  Set the output speed of a GPIO pin.

- **EN_MGPIO_systemState_t MGPIO_uddtSetPullState** (**ST_MGPIOx_RegistersMap_t** *PS_GPIOx, **EN_MGPIO_pinOptions_t** copy_uddtPinNum, **EN_MGPIO_pushPullOptions_t** copy_uddtPullState)
  Set the pull state of a GPIO pin.

- **EN_MGPIO_systemState_t MGPIO_uddtGetPinVal** (**ST_MGPIOx_RegistersMap_t** *PS_GPIOx, **EN_MGPIO_pinOptions_t** copy_uddtPinNum, **EN_MGPIO_pinLogicOptions_t** *copy_uddtPtrRetOfPinLogic)
  Get the logic level of a GPIO pin.

- **EN_MGPIO_systemState_t MGPIO_uddtSetPinVal** (**ST_MGPIOx_RegistersMap_t** *PS_GPIOx, **EN_MGPIO_pinOptions_t** copy_uddtPinNum, **EN_MGPIO_pinLogicOptions_t** copy_uddtPinLogic)
  Set the logic level of a GPIO pin.

- **EN_MGPIO_systemState_t MGPIO_uddtDirectSetReset** (**ST_MGPIOx_RegistersMap_t** *PS_GPIOx, **EN_MGPIO_pinOptions_t** copy_uddtPinNum, **EN_MGPIO_setResetOptions_t** copy_uddtSetResetState)
  Perform a direct set/reset operation on a GPIO pin.

- **EN_MGPIO_systemState_t MGPIO_uddtSetPortVal** (**ST_MGPIOx_RegistersMap_t** *PS_GPIOx, **uint16_t** copy_u16OutputVal)
  Set the value of an entire GPIO port.

- **EN_MGPIO_systemState_t MGPIO_uddtSetAltFun** (**ST_MGPIOx_RegistersMap_t** *PS_GPIOx, **EN_MGPIO_pinOptions_t** copy_uddtPinNum, **EN_MGPIO_altfnOptions_t** Copy_uddtAltFun)
  Set the alternate function of a GPIO pin.

- **EN_MGPIO_systemState_t MGPIO_uddtInitPin** (**ST_MGPIO_pinCfg_t** *PS_pinInstance)
  Initialize a GPIO pin based on a configuration structure.

- **EN_MGPIO_systemState_t MGPIO_uddtInitAltPin** (**ST_MGPIO_altPinCfg_t** *PS_altPinInstance)
  Initialize an alternate GPIO pin based on a configuration structure.

Detailed Description

Header file for GPIO (General Purpose I/O) module.

Function Documentation

EN_MGPIO_systemState_t MGPIO_uddtDirectSetReset (ST_MGPIOx_RegistersMap_t * *PS_GPIOx*, EN_MGPIO_pinOptions_t *copy_uddtPinNum*, EN_MGPIO_setResetOptions_t *copy_uddtSetResetState*)

Perform a direct set/reset operation on a GPIO pin.

This function performs a direct set/reset operation on a specified GPIO pin.

Parameters

PS_GPIOx	Pointer to the GPIOx registers map.
copy_uddtPinNum	The pin number to configure. Possible values are: • **MGPIO_PIN0** • **MGPIO_PIN1** • ... • **MGPIO_PIN15**
copy_uddtSetReset State	The set/reset option. Possible values are: • **MGPIO_PIN_RESET** • **MGPIO_PIN_SET**

Returns

The system state after the set/reset operation.

- **MGPIO_OK**: Set/reset operation successful.
- **MGPIO_NOK**: Set/reset operation failed.
- **MGPIO_INVALID_PARAMTER**: Invalid parameter detected during the operation.
- **MGPIO_PTR_NULL**: Null pointer encountered during the operation.

EN_MGPIO_systemState_t MGPIO_uddtGetPinVal (ST_MGPIOx_RegistersMap_t * *PS_GPIOx*, EN_MGPIO_pinOptions_t *copy_uddtPinNum*, EN_MGPIO_pinLogicOptions_t * *copy_uddtPtrRetOfPinLogic*)

Get the logic level of a GPIO pin.

This function retrieves the logic level of a specified GPIO pin.

Parameters

PS_GPIOx	Pointer to the GPIOx registers map.
copy_uddtPinNum	The pin number to read. Possible values are: • **MGPIO_PIN0** • **MGPIO_PIN1** • ... • **MGPIO_PIN15**
copy_uddtPtrRetOfPinLogic	Pointer to store the retrieved logic level.

Returns

The system state after getting the pin logic level.

- **MGPIO_OK**: Pin logic level retrieval successful.
- **MGPIO_NOK**: Pin logic level retrieval failed.
- **MGPIO_INVALID_PARAMTER**: Invalid parameter detected during the operation.
- **MGPIO_PTR_NULL**: Null pointer encountered during the operation.

EN_MGPIO_systemState_t MGPIO_uddtInitAltPin (ST_MGPIO_altPinCfg_t * *PS_altPinInstance*)

Initialize an alternate GPIO pin based on a configuration structure.

This function initializes an alternate GPIO pin based on the provided configuration structure.

Parameters

PS_altPinInstance	Pointer to the alternate GPIO pin configuration structure.

Returns

The system state after initializing the alternate GPIO pin.

- **MGPIO_OK**: Alternate GPIO pin initialization successful.
- **MGPIO_NOK**: Alternate GPIO pin initialization failed.
- **MGPIO_INVALID_PARAMTER**: Invalid parameter detected during the operation.
- **MGPIO_PTR_NULL**: Null pointer encountered during the operation.

EN_MGPIO_systemState_t MGPIO_uddtInitPin (ST_MGPIO_pinCfg_t * *PS_pinInstance*)

Initialize a GPIO pin based on a configuration structure.

This function initializes a GPIO pin based on the provided configuration structure.

Parameters

PS_pinInstance	Pointer to the GPIO pin configuration structure.

Returns

The system state after initializing the GPIO pin.

- **MGPIO_OK**: GPIO pin initialization successful.
- **MGPIO_NOK**: GPIO pin initialization failed.
- **MGPIO_INVALID_PARAMTER**: Invalid parameter detected during the operation.
- **MGPIO_PTR_NULL**: Null pointer encountered during the operation.

EN_MGPIO_systemState_t MGPIO_uddtSetAltFun (ST_MGPIOx_RegistersMap_t * *PS_GPIOx*, EN_MGPIO_pinOptions_t *copy_uddtPinNum*, EN_MGPIO_altfnOptions_t *Copy_uddtAltFun*)

Set the alternate function of a GPIO pin.

This function sets the alternate function of a specified GPIO pin.

Parameters

PS_GPIOx	Pointer to the GPIOx registers map.
copy_uddtPinNum	The pin number to configure. Possible values are: • **MGPIO_PIN0** • **MGPIO_PIN1** • ... • **MGPIO_PIN15**
Copy_uddtAltFun	The alternate function option. Possible values are: • **MGPIO_ALTFN_0** • **MGPIO_ALTFN_1** • ... • **MGPIO_ALTFN_15**

Returns

The system state after setting the alternate function.

- **MGPIO_OK**: Alternate function setting successful.
- **MGPIO_NOK**: Alternate function setting failed.
- **MGPIO_INVALID_PARAMTER**: Invalid parameter detected during the operation.
- **MGPIO_PTR_NULL**: Null pointer encountered during the operation.

EN_MGPIO_systemState_t MGPIO_uddtSetOutputMode (ST_MGPIOx_RegistersMap_t * *PS_GPIOx*, EN_MGPIO_pinOptions_t *copy_uddtPinNum*, EN_MGPIO_outputResistorOptions_t *copy_uddtOutputResistor*)

Set the output mode of a GPIO pin.

This function sets the output mode of a specified GPIO pin.

Parameters

PS_GPIOx	Pointer to the GPIOx registers map.
copy_uddtPinNum	The pin number to configure. Possible values are: • **MGPIO_PIN0** • **MGPIO_PIN1** • ... • **MGPIO_PIN15**
copy_uddtOutputResistor	The output resistor option. Possible values are: • **MGPIO_OUTPUT_RESISTOR_PUSH_PULL** • **MGPIO_OUTPUT_RESISTOR_OPEN_DRAIN**

Returns

The system state after setting the pin output mode.

- **MGPIO_OK**: Pin output mode setting successful.
- **MGPIO_NOK**: Pin output mode setting failed.
- **MGPIO_INVALID_PARAMTER**: Invalid parameter detected during the operation.
- **MGPIO_PTR_NULL**: Null pointer encountered during the operation.

EN_MGPIO_systemState_t MGPIO_uddtSetOutputSpeed (ST_MGPIOx_RegistersMap_t * *PS_GPIOx*, EN_MGPIO_pinOptions_t *copy_uddtPinNum*, EN_MGPIO_outputSpeedOptions_t *copy_uddtOutputSpeed*)

Set the output speed of a GPIO pin.

This function sets the output speed of a specified GPIO pin.

Parameters

PS_GPIOx	Pointer to the GPIOx registers map.
copy_uddtPinNum	The pin number to configure. Possible values are: • **MGPIO_PIN0** • **MGPIO_PIN1** • ... • **MGPIO_PIN15**
copy_uddtOutputSpeed	The output speed option. Possible values are: • **MGPIO_OUTPUT_SPEED_LOW** • **MGPIO_OUTPUT_SPEED_MEDIUM** • **MGPIO_OUTPUT_SPEED_HIGH** • **MGPIO_OUTPUT_SPEED_VERY_HIGH**

Returns

The system state after setting the pin output speed.

- **MGPIO_OK**: Pin output speed setting successful.
- **MGPIO_NOK**: Pin output speed setting failed.
- **MGPIO_INVALID_PARAMTER**: Invalid parameter detected during the operation.
- **MGPIO_PTR_NULL**: Null pointer encountered during the operation.

EN_MGPIO_systemState_t MGPIO_uddtSetPinMode (ST_MGPIOx_RegistersMap_t * *PS_GPIOx*, EN_MGPIO_pinOptions_t *copy_uddtPinNum*, EN_MGPIO_pinModeOptions_t *copy_uddtPinMode*)

Set the mode of a GPIO pin.

This function sets the mode of a specified GPIO pin.

Parameters

PS_GPIOx	Pointer to the GPIOx registers map.
copy_uddtPinNum	The pin number to configure. Possible values are: • **MGPIO_PIN0** • **MGPIO_PIN1** • ... • **MGPIO_PIN15**
copy_uddtPinMode	The mode to set for the pin. Possible values are: • **MGPIO_MODE_INPUT** • **MGPIO_MODE_OUTPUT** • **MGPIO_MODE_ALTF** • **MGPIO_MODE_ANALOG**

Returns

The system state after setting the pin mode.

- **MGPIO_OK**: Pin mode setting successful.
- **MGPIO_NOK**: Pin mode setting failed.
- **MGPIO_INVALID_PARAMTER**: Invalid parameter detected during the operation.

- **MGPIO_PTR_NULL**: Null pointer encountered during the operation.

EN_MGPIO_systemState_t MGPIO_uddtSetPinVal (ST_MGPIOx_RegistersMap_t * *PS_GPIOx*, EN_MGPIO_pinOptions_t *copy_uddtPinNum*, EN_MGPIO_pinLogicOptions_t *copy_uddtPinLogic*)

Set the logic level of a GPIO pin.

This function sets the logic level of a specified GPIO pin.

Parameters

PS_GPIOx	Pointer to the GPIOx registers map.
copy_uddtPinNum	The pin number to configure. Possible values are: • **MGPIO_PIN0** • **MGPIO_PIN1** • ... • **MGPIO_PIN15**
copy_uddtPinLogic	The logic level to set for the pin. Possible values are: • **MGPIO_LOGIC_LOW** • **MGPIO_LOGIC_HIGH**

Returns

The system state after setting the pin logic level.

- **MGPIO_OK**: Pin logic level setting successful.
- **MGPIO_NOK**: Pin logic level setting failed.
- **MGPIO_INVALID_PARAMTER**: Invalid parameter detected during the operation.
- **MGPIO_PTR_NULL**: Null pointer encountered during the operation.

EN_MGPIO_systemState_t MGPIO_uddtSetPortVal (ST_MGPIOx_RegistersMap_t * *PS_GPIOx*, uint16_t *copy_u16OutputVal*)

Set the value of an entire GPIO port.

This function sets the value of an entire GPIO port.

Parameters

PS_GPIOx	Pointer to the GPIOx registers map.
copy_u16OutputVal	The value to set for the entire port.

Returns

The system state after setting the port value.

- **MGPIO_OK**: Port value setting successful.
- **MGPIO_NOK**: Port value setting failed.
- **MGPIO_INVALID_PARAMTER**: Invalid parameter detected during the operation.
- **MGPIO_PTR_NULL**: Null pointer encountered during the operation.

EN_MGPIO_systemState_t MGPIO_uddtSetPullState (ST_MGPIOx_RegistersMap_t * *PS_GPIOx*, EN_MGPIO_pinOptions_t *copy_uddtPinNum*, EN_MGPIO_pushPullOptions_t *copy_uddtPullState*)

Set the pull state of a GPIO pin.

This function sets the pull state of a specified GPIO pin.

Parameters

PS_GPIOx	Pointer to the GPIOx registers map.

copy_uddtPinNum	The pin number to configure. Possible values are: • **MGPIO_PIN0** • **MGPIO_PIN1** • ... • **MGPIO_PIN15**
copy_uddtPullState	The pull state option. Possible values are: • **MGPIO_PULL_FLOATING** • **MGPIO_PULL_PULL_UP** • **MGPIO_PULL_PULL_DOWN**

Returns

The system state after setting the pin pull state.

- **MGPIO_OK**: Pin pull state setting successful.
- **MGPIO_NOK**: Pin pull state setting failed.
- **MGPIO_INVALID_PARAMTER**: Invalid parameter detected during the operation.
- **MGPIO_PTR_NULL**: Null pointer encountered during the operation.

gpio_interface.h

Go to the documentation of this file.

```c

#ifndef MCAL_GPIO_GPIO_INTERFACE_H_
#define MCAL_GPIO_GPIO_INTERFACE_H_

#include "../../COMMON/bit_math.h"
#include "../../COMMON/std_types.h"
#include "gpio_private.h"
#include "gpio_config.h"

EN_MGPIO_systemState_t MGPIO_uddtSetPinMode(ST_MGPIOx_RegistersMap_t *PS_GPIOx,
EN_MGPIO_pinOptions_t copy_uddtPinNum, EN_MGPIO_pinModeOptions_t copy_uddtPinMode);

EN_MGPIO_systemState_t MGPIO_uddtSetOutputMode(ST_MGPIOx_RegistersMap_t *PS_GPIOx,
EN_MGPIO_pinOptions_t copy_uddtPinNum, EN_MGPIO_outputResistorOptions_t
copy_uddtOutputResistor);

EN_MGPIO_systemState_t MGPIO_uddtSetOutputSpeed(ST_MGPIOx_RegistersMap_t *PS_GPIOx,
EN_MGPIO_pinOptions_t copy_uddtPinNum, EN_MGPIO_outputSpeedOptions_t
copy_uddtOutputSpeed);

EN_MGPIO_systemState_t MGPIO_uddtSetPullState(ST_MGPIOx_RegistersMap_t *PS_GPIOx,
EN_MGPIO_pinOptions_t copy_uddtPinNum, EN_MGPIO_pushPullOptions_t copy_uddtPullState);

EN_MGPIO_systemState_t MGPIO_uddtGetPinVal(ST_MGPIOx_RegistersMap_t *PS_GPIOx,
EN_MGPIO_pinOptions_t copy_uddtPinNum, EN_MGPIO_pinLogicOptions_t
*copy_uddtPtrRetOfPinLogic);

EN_MGPIO_systemState_t MGPIO_uddtSetPinVal(ST_MGPIOx_RegistersMap_t *PS_GPIOx,
EN_MGPIO_pinOptions_t copy_uddtPinNum, EN_MGPIO_pinLogicOptions_t copy_uddtPinLogic);

EN_MGPIO_systemState_t MGPIO_uddtDirectSetReset(ST_MGPIOx_RegistersMap_t *PS_GPIOx,
EN_MGPIO_pinOptions_t copy_uddtPinNum, EN_MGPIO_setResetOptions_t
copy_uddtSetResetState);

EN_MGPIO_systemState_t MGPIO_uddtSetPortVal(ST_MGPIOx_RegistersMap_t *PS_GPIOx,
uint16_t copy_u16OutputVal);

EN_MGPIO_systemState_t MGPIO_uddtSetAltFun(ST_MGPIOx_RegistersMap_t *PS_GPIOx,
EN_MGPIO_pinOptions_t copy_uddtPinNum, EN_MGPIO_altfnOptions_t Copy_uddtAltFun);

EN_MGPIO_systemState_t MGPIO_uddtInitPin(ST_MGPIO_pinCfg_t *PS_pinInstance);

EN_MGPIO_systemState_t MGPIO_uddtInitAltPin(ST_MGPIO_altPinCfg_t
*PS_altPinInstance);


//EN_MGPIO_systemState_t MGPIO_PinLock      (u8 copy_u8PortName , u8 copy_u8PinNum );
//EN_MGPIO_systemState_t MGPIO_SetPortMode (u8 Copy_u8PortName , u8 Copy_u8Mode);


#endif /* MCAL_GPIO_GPIO_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/gpio/gpio_private.h File Reference

Data Structures

struct ST_MGPIOx_RegistersMap_t**Macros**

- #define **MGPIOA_PERIPHERAL_BASE_ADDR**   (0x40020000)
- #define **MGPIOB_PERIPHERAL_BASE_ADDR**   (0x40020400)
- #define **MGPIOC_PERIPHERAL_BASE_ADDR**   (0x40020800)
- #define **MGPIOD_PERIPHERAL_BASE_ADDR**   (0x40020C00)
- #define **MGPIOE_PERIPHERAL_BASE_ADDR**   (0x40021000)
- #define **MGPIOH_PERIPHERAL_BASE_ADDR**   (0x40021C00)
- #define **MGPIOA_PERIPHERAL**   ((**ST_MGPIOx_RegistersMap_t** *)**MGPIOA_PERIPHERAL_BASE_ADDR**)
- #define **MGPIOB_PERIPHERAL**   ((**ST_MGPIOx_RegistersMap_t** *)**MGPIOB_PERIPHERAL_BASE_ADDR**)

Macro Definition Documentation

#define MGPIOA_PERIPHERAL   ((ST_MGPIOx_RegistersMap_t *)MGPIOA_PERIPHERAL_BASE_ADDR)

#define MGPIOA_PERIPHERAL_BASE_ADDR   (0x40020000)

#define MGPIOB_PERIPHERAL   ((ST_MGPIOx_RegistersMap_t *)MGPIOB_PERIPHERAL_BASE_ADDR)

#define MGPIOB_PERIPHERAL_BASE_ADDR   (0x40020400)

#define MGPIOC_PERIPHERAL_BASE_ADDR   (0x40020800)

#define MGPIOD_PERIPHERAL_BASE_ADDR   (0x40020C00)

#define MGPIOE_PERIPHERAL_BASE_ADDR   (0x40021000)

#define MGPIOH_PERIPHERAL_BASE_ADDR   (0x40021C00)

gpio_private.h

Go to the documentation of this file.

```c
/*************************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : STM32F401xC
// File          : gpio_private.h
// Date          : Sep 10, 2023
// GitHub        : https://github.com/sherifkhadr
/*************************************************************************/
#ifndef MCAL_GPIO_GPIO_PRIVATE_H_
#define MCAL_GPIO_GPIO_PRIVATE_H_


#define  MGPIOA_PERIPHERAL_BASE_ADDR        (0x40020000)
#define  MGPIOB_PERIPHERAL_BASE_ADDR        (0x40020400)
#define  MGPIOC_PERIPHERAL_BASE_ADDR        (0x40020800)
#define  MGPIOD_PERIPHERAL_BASE_ADDR        (0x40020C00)
#define  MGPIOE_PERIPHERAL_BASE_ADDR        (0x40021000)
#define  MGPIOH_PERIPHERAL_BASE_ADDR        (0x40021C00)


typedef struct
{

    vuint32_t   MGPIOx_MODER;
    vuint32_t   MGPIOx_OTYPER;
    vuint32_t   MGPIOx_OSPEEDR;
    vuint32_t   MGPIOx_PUPDR;
    vuint32_t   MGPIOx_IDR;
    vuint32_t   MGPIOx_ODR;
    vuint32_t   MGPIOx_BSRR;
    vuint32_t   MGPIOx_LCKR;
    vuint32_t   MGPIOx_AFRL;
    vuint32_t   MGPIOx_AFRH;


}ST_MGPIOx_RegistersMap_t;




#define MGPIOA_PERIPHERAL (( ST_MGPIOx_RegistersMap_t *)MGPIOA_PERIPHERAL_BASE_ADDR)
#define MGPIOB_PERIPHERAL (( ST_MGPIOx_RegistersMap_t *)MGPIOB_PERIPHERAL_BASE_ADDR)




#endif /* MCAL_GPIO_GPIO_PRIVATE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/nvic/nvic_config.h File Reference

Macros

- #define **POS_OF_FIRST_INT**   0
- #define **POS_OF_LAST_INT**   84
- #define **getIntRegNumber**(IntNumber)   (IntNumber / 32)
- #define **getIntBitNumber**(IntNumber)   (IntNumber % 32)

Enumerations

- enum **EN_MNVIC_systemState_t** { **MNVIC_OK** = 0, **MNVIC_NOK**, **MNVIC_INVALID_PARAMTER** }
  Enumeration for the system state of NVIC functions.
- enum **EN_MNVIC_priorityGrouping_t** { **GP_16_SP_00** = 0, **GP_08_SP_02**, **GP_04_SP_04**, **GP_02_SP_08**, **GP_00_SP_16** }
  Enumeration for the priority grouping options in the NVIC.

Macro Definition Documentation

#define getIntBitNumber(IntNumber)   (IntNumber % 32)

#define getIntRegNumber(IntNumber)   (IntNumber / 32)

#define POS_OF_FIRST_INT   0

#define POS_OF_LAST_INT   84

Enumeration Type Documentation

enum EN_MNVIC_priorityGrouping_t

Enumeration for the priority grouping options in the NVIC.

Enumerator:

GP_16_SP_00	16 priority levels, 0 subpriority levels.
GP_08_SP_02	8 priority levels, 2 subpriority levels.
GP_04_SP_04	4 priority levels, 4 subpriority levels.
GP_02_SP_08	2 priority levels, 8 subpriority levels.
GP_00_SP_16	0 priority levels, 16 subpriority levels.

enum EN_MNVIC_systemState_t

Enumeration for the system state of NVIC functions.

Enumerator:

MNVIC_OK	Operation successful.
MNVIC_NOK	Operation failed.
MNVIC_INVALID_PARAMTER	Invalid parameter detected.

nvic_config.h

Go to the documentation of this file.

```c
/**************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : STM32F401xC
// File           : nvic_config.h
// Date           : Sep 10, 2023
// GitHub         : https://github.com/sherifkhadr
/**************************************************************************/
#ifndef MCAL_NVIC_NVIC_CONFIG_H_
#define MCAL_NVIC_NVIC_CONFIG_H_

#define POS_OF_FIRST_INT            0
#define POS_OF_LAST_INT             84

#define getIntRegNumber(IntNumber)      (IntNumber / 32)
#define getIntBitNumber(IntNumber)      (IntNumber % 32)


typedef enum
{
    MNVIC_OK = 0,
    MNVIC_NOK,
    MNVIC_INVALID_PARAMTER
} EN_MNVIC_systemState_t;

typedef enum
{
    GP_16_SP_00 = 0,
    GP_08_SP_02,
    GP_04_SP_04,
    GP_02_SP_08,
    GP_00_SP_16
} EN_MNVIC_priorityGrouping_t;


#endif /* MCAL_NVIC_NVIC_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/nvic/nvic_interface.h File Reference

Header file for the NVIC (Nested Vectored Interrupt Controller) module interface.

```
#include "../../COMMON/bit_math.h"
#include "../../COMMON/std_types.h"
#include "nvic_private.h"
#include "nvic_config.h"
```

Functions

- **EN_MNVIC_systemState_t MNVIC_enableInterrupt** (**uint8_t** copy_u8IntPos)
  Enable an interrupt in the NVIC.
- **EN_MNVIC_systemState_t MNVIC_disableInterrupt** (**uint8_t** copy_u8IntPos)
  Disable an interrupt in the NVIC.
- **EN_MNVIC_systemState_t MNVIC_enableInterruptPending** (**uint8_t** copy_u8IntPos)
  Enable pending status for an interrupt in the NVIC.
- **EN_MNVIC_systemState_t MNVIC_disableInterruptPending** (**uint8_t** copy_u8IntPos)
  Disable pending status for an interrupt in the NVIC.
- **EN_MNVIC_systemState_t MNVIC_IsInterruptActive** (**uint8_t** copy_u8IntPos, **uint8_t** *ptrOfRetReading)
  Check if an interrupt is active in the NVIC.
- **EN_MNVIC_systemState_t MNVIC_SetInterruptPriority** (**uint8_t** copy_u8IntPos, **EN_MNVIC_priorityGrouping_t** copy_uddtGroupOption, **uint8_t** copy_u8GroupPriority, **uint8_t** copy_u8SubPriority)
  Set the priority of an interrupt in the NVIC.
- **EN_MNVIC_systemState_t MNVIC_SetInterruptGroup** (**EN_MNVIC_priorityGrouping_t** copy_uddtGroupOption)
  Set the priority grouping for the NVIC.

Detailed Description

Header file for the NVIC (Nested Vectored Interrupt Controller) module interface.

Function Documentation

EN_MNVIC_systemState_t MNVIC_disableInterrupt (uint8_t *copy_u8IntPos*)

Disable an interrupt in the NVIC.

This function disables the specified interrupt in the NVIC.

Parameters

copy_u8IntPos	The position of the interrupt to disable.

Returns

The system state after disabling the interrupt.

- MNVIC_OK: Operation successful.
- MNVIC_NOK: Operation failed.
- MNVIC_INVALID_PARAMTER: Invalid parameter detected.

EN_MNVIC_systemState_t MNVIC_disableInterruptPending (uint8_t *copy_u8IntPos*)

Disable pending status for an interrupt in the NVIC.

This function disables the pending status for the specified interrupt in the NVIC.

Parameters

copy_u8IntPos	The position of the interrupt to disable pending status for.

Returns

The system state after disabling pending status for the interrupt.

- MNVIC_OK: Operation successful.
- MNVIC_NOK: Operation failed.
- MNVIC_INVALID_PARAMTER: Invalid parameter detected.

EN_MNVIC_systemState_t MNVIC_enableInterrupt (uint8_t *copy_u8IntPos*)

Enable an interrupt in the NVIC.

This function enables the specified interrupt in the NVIC.

Parameters

copy_u8IntPos	The position of the interrupt to enable.

Returns

The system state after enabling the interrupt.

- MNVIC_OK: Operation successful.
- MNVIC_NOK: Operation failed.
- MNVIC_INVALID_PARAMTER: Invalid parameter detected.

EN_MNVIC_systemState_t MNVIC_enableInterruptPending (uint8_t *copy_u8IntPos*)

Enable pending status for an interrupt in the NVIC.

This function enables the pending status for the specified interrupt in the NVIC.

Parameters

copy_u8IntPos	The position of the interrupt to enable pending status for.

Returns

The system state after enabling pending status for the interrupt.

- MNVIC_OK: Operation successful.
- MNVIC_NOK: Operation failed.
- MNVIC_INVALID_PARAMTER: Invalid parameter detected.

EN_MNVIC_systemState_t MNVIC_IsInterruptActive (uint8_t *copy_u8IntPos*, uint8_t * *ptrOfRetReading*)

Check if an interrupt is active in the NVIC.

This function checks if the specified interrupt is active in the NVIC.

Parameters

copy_u8IntPos	The position of the interrupt to check.
ptrOfRetReading	Pointer to store the result of the interrupt's active status.

Returns

The system state after checking the interrupt's active status.

- MNVIC_OK: Operation successful.
- MNVIC_NOK: Operation failed.
- MNVIC_INVALID_PARAMTER: Invalid parameter detected.

EN_MNVIC_systemState_t MNVIC_SetInterruptGroup (EN_MNVIC_priorityGrouping_t *copy_uddtGroupOption*)

Set the priority grouping for the NVIC.

This function sets the priority grouping for the NVIC.

Parameters

copy_uddtGroupOption	The priority grouping option (GP_16_SP_00, GP_08_SP_02, GP_04_SP_04, GP_02_SP_08, GP_00_SP_16).

Returns

The system state after setting the priority grouping.

- MNVIC_OK: Operation successful.
- MNVIC_NOK: Operation failed.
- MNVIC_INVALID_PARAMTER: Invalid parameter detected.

EN_MNVIC_systemState_t MNVIC_SetInterruptPriority (uint8_t *copy_u8IntPos*, EN_MNVIC_priorityGrouping_t *copy_uddtGroupOption*, uint8_t *copy_u8GroupPriority*, uint8_t *copy_u8SubPriority*)

Set the priority of an interrupt in the NVIC.

This function sets the priority of the specified interrupt in the NVIC.

Parameters

copy_u8IntPos	The position of the interrupt to set the priority for.
copy_uddtGroupOption	The priority grouping option (GP_16_SP_00, GP_08_SP_02, GP_04_SP_04, GP_02_SP_08, GP_00_SP_16).
copy_u8GroupPriority	The group priority value (0 to 15).
copy_u8SubPriority	The subpriority value (0 to 15).

Returns

The system state after setting the interrupt priority.

- MNVIC_OK: Operation successful.
- MNVIC_NOK: Operation failed.
- MNVIC_INVALID_PARAMTER: Invalid parameter detected.

nvic_interface.h

Go to the documentation of this file.

```
#ifndef MCAL_NVIC_NVIC_INTERFACE_H_
#define MCAL_NVIC_NVIC_INTERFACE_H_

#include "../../COMMON/bit_math.h"
#include "../../COMMON/std_types.h"
#include "nvic_private.h"
#include "nvic_config.h"

EN_MNVIC_systemState_t MNVIC_enableInterrupt(uint8_t copy_u8IntPos);

EN_MNVIC_systemState_t MNVIC_disableInterrupt(uint8_t copy_u8IntPos);

EN_MNVIC_systemState_t MNVIC_enableInterruptPending(uint8_t copy_u8IntPos);

EN_MNVIC_systemState_t MNVIC_disableInterruptPending(uint8_t copy_u8IntPos);

EN_MNVIC_systemState_t MNVIC_IsInterruptActive(uint8_t copy_u8IntPos, uint8_t 
*ptrOfRetReading);

EN_MNVIC_systemState_t MNVIC_SetInterruptPriority(uint8_t copy_u8IntPos, 
EN_MNVIC_priorityGrouping_t copy_uddtGroupOption, uint8_t copy_u8GroupPriority, uint8_t 
copy_u8SubPriority);

EN_MNVIC_systemState_t MNVIC_SetInterruptGroup(EN_MNVIC_priorityGrouping_t 
copy_uddtGroupOption);

#endif /* MCAL_NVIC_NVIC_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/nvic/nvic_private.h File Reference

Data Structures

struct ST_MNVIC_RegistersMap_t**Macros**

- #define **MNVIC_PERIPHERAL_BASE_ADDR**   (0xE000E100)
- #define **MNVIC_PERIPHERAL**   ((volatile **ST_MNVIC_RegistersMap_t** *)**MNVIC_PERIPHERAL_BASE_ADDR**)
- #define **SCB_AIRCR**   *((volatile **uint32_t** *)(0xE000ED0C))
- #define **MNVIC_VECTKEY**   (0x05FA0000) /* Key to write to AIRCR register */

Macro Definition Documentation

#define MNVIC_PERIPHERAL   ((volatile ST_MNVIC_RegistersMap_t *)MNVIC_PERIPHERAL_BASE_ADDR)

#define MNVIC_PERIPHERAL_BASE_ADDR   (0xE000E100)

#define MNVIC_VECTKEY   (0x05FA0000) /* Key to write to AIRCR register */

#define SCB_AIRCR   *((volatile uint32_t *)(0xE000ED0C))

nvic_private.h

Go to the documentation of this file.

```
/*************************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : STM32F401xC
// File          : nvic_private.h
// Date          : Sep 10, 2023
// GitHub        : https://github.com/sherifkhadr
/*************************************************************************/
#ifndef MCAL_NVIC_NVIC_PRIVATE_H_
#define MCAL_NVIC_NVIC_PRIVATE_H_


#define  MNVIC_PERIPHERAL_BASE_ADDR     (0xE000E100)


typedef struct
{

    vuint32_t MNVIC_ISERx[8];
    vuint32_t MNVIC_RESERVED0[24];
    vuint32_t MNVIC_ICERx[8];
    vuint32_t MNVIC_RESERVED1[24];
    vuint32_t MNVIC_ISPRx[8];
    vuint32_t MNVIC_RESERVED2[24];
    vuint32_t MNVIC_ICPRx[8];
    vuint32_t MNVIC_RESERVED3[24];
    vuint32_t MNVIC_IABRx[8];
    vuint32_t MNVIC_RESERVED4[56];
    vuint8_t MNVIC_IPRx[240];
    vuint32_t MNVIC_RESERVED5[580];
    vuint32_t MNVIC_STIR;

}ST_MNVIC_RegistersMap_t;


#define MNVIC_PERIPHERAL ((volatile ST_MNVIC_RegistersMap_t
*)MNVIC_PERIPHERAL_BASE_ADDR)

#define SCB_AIRCR        *((volatile uint32_t *)(0xE000ED0C))

#define MNVIC_VECTKEY       (0x05FA0000) /* Key to write to AIRCR register */


#endif /* MCAL_NVIC_NVIC_PRIVATE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/rcc/rcc_config.h File Reference

Macros

- #define **sysClkSelect**   **MRCC_SYS_CLK_HSI**
- #define **pllStatus**   **MRCC_PLL_DISABLE**
- #define **pllSourceOfEntryClk**   **MRCC_PLL_ENTRY_CLK_HSI**
- #define **pllpDivisionFactor**   **MRCC_PLL_DIVISION_FACTOR_2**
- #define **pllmDivisionFactor**   0
- #define **pllnMulFactor**   0
- #define **apbHighSpeedPrescaler**   **MRCC_APB_PRESCALER_SPEED_NOT_DIVIDED**
- #define **apbLowSpeedPrescaler**   **MRCC_APB_PRESCALER_SPEED_NOT_DIVIDED**
- #define **ahbPrescaler**   **MRCC_AHB_PRESCALER_2**

Enumerations

- enum **EN_MRCC_systemState_t** { **MRCC_OK** = 0, **MRCC_NOK**, **MRCC_PTR_NULL**, **MRCC_INVALID_PARAMTER** }
- enum **EN_MRCC_sysClkSelect_t** { **MRCC_SYS_CLK_HSI** = 0, **MRCC_SYS_CLK_HSE_BYPASS**, **MRCC_SYS_CLK_HSE_NOT_BYPASS** = 1, **MRCC_SYS_CLK_PLL**, **MRCC_SYS_CLK_NOT_ALLOWED** }

  Enumeration for the system clock selection.
- enum **EN_MRCC_pllClkSourceEntry_t** { **MRCC_PLL_ENTRY_CLK_HSI** = 0, **MRCC_PLL_ENTRY_CLK_HSE** }

  Enumeration for the PLL entry clock source.
- enum **EN_MRCC_pllDivisionFactor_t** { **MRCC_PLL_DIVISION_FACTOR_2** = 0, **MRCC_PLL_DIVISION_FACTOR_4**, **MRCC_PLL_DIVISION_FACTOR_6**, **MRCC_PLL_DIVISION_FACTOR_8** }

  Enumeration for the PLL division factor.
- enum **EN_MRCC_apbPrescalerSpeed_t** { **MRCC_APB_PRESCALER_SPEED_NOT_DIVIDED** = 0, **MRCC_APB_PRESCALER_SPEED_2** = 4, **MRCC_APB_PRESCALER_SPEED_4**, **MRCC_APB_PRESCALER_SPEED_8**, **MRCC_APB_PRESCALER_SPEED_16** }

  Enumeration for APB (Advanced Peripheral Bus) prescaler speed.
- enum **EN_MRCC_ahbPrescaler_t** { **MRCC_AHB_PRESCALER_NOT_DIVIDED** = 0, **MRCC_AHB_PRESCALER_2** = 8, **MRCC_AHB_PRESCALER_4**, **MRCC_AHB_PRESCALER_8**, **MRCC_AHB_PRESCALER_16**, **MRCC_AHB_PRESCALER_64**, **MRCC_AHB_PRESCALER_128**, **MRCC_AHB_PRESCALER_256**, **MRCC_AHB_PRESCALER_512** }

  Enumeration for AHB (Advanced High-Performance Bus) prescaler.
- enum **EN_MRCC_pllStatus_t** { **MRCC_PLL_DISABLE** = 0, **MRCC_PLL_ENABLE** }

  Enumeration for PLL status.
- enum **EN_MRCC_busOptions_t** { **MRCC_AHP1_BUS** = 0, **MRCC_AHP2_BUS**, **MRCC_APB1_BUS**, **MRCC_APB2_BUS** }

  Enumeration for different buses in MRCC.
- enum **EN_MRCC_peripheralOptions_t** { **MRCC_GPIOA_PERIPHERAL** = 0, **MRCC_GPIOB_PERIPHERAL**, **MRCC_CRC_PERIPHERAL** = 12, **MRCC_DMA1_PERIPHERAL** = 21, **MRCC_DMA2_PERIPHERAL**, **MRCC_OTGFS_PERIPHERAL** = 7, **MRCC_TIM2_PERIPHERAL** = 0, **MRCC_TIM3_PERIPHERAL**, **MRCC_TIM4_PERIPHERAL**, **MRCC_TIM5_PERIPHERAL**, **MRCC_WWDG_PERIPHERAL** = 11, **MRCC_SPI2_PERIPHERAL** = 14, **MRCC_SPI3_PERIPHERAL** = 15, **MRCC_USART2_PERIPHERAL** = 17, **MRCC_I2C1_PERIPHERAL** = 21,

MRCC_I2C2_PERIPHERAL, **MRCC_I2C3_PERIPHERAL**, **MRCC_PWR_PERIPHERAL** = 28, **MRCC_TIM1_PERIPHERAL** = 0, **MRCC_USART1_PERIPHERAL** = 4, **MRCC_USART6_PERIPHERAL**, **MRCC_ADC1_PERIPHERAL** = 8, **MRCC_SDIO_PERIPHERAL** = 11, **MRCC_SPI1_PERIPHERAL**, **MRCC_SPI4_PERIPHERAL**, **MRCC_SYSCFG_PERIPHERAL**, **MRCC_TIM9_PERIPHERAL** = 16, **MRCC_TIM10_PERIPHERAL**, **MRCC_TIM11_PERIPHERAL** }

Enumeration for different peripheral options in MRCC.

Macro Definition Documentation

#define ahbPrescaler   MRCC_AHB_PRESCALER_2

#define apbHighSpeedPrescaler   MRCC_APB_PRESCALER_SPEED_NOT_DIVIDED

#define apbLowSpeedPrescaler   MRCC_APB_PRESCALER_SPEED_NOT_DIVIDED

#define pllmDivisionFactor   0

#define pllnMulFactor   0

#define pllpDivisionFactor   MRCC_PLL_DIVISION_FACTOR_2

#define pllSourceOfEntryClk   MRCC_PLL_ENTRY_CLK_HSI

#define pllStatus   MRCC_PLL_DISABLE

#define sysClkSelect   MRCC_SYS_CLK_HSI

Enumeration Type Documentation

enum EN_MRCC_ahbPrescaler_t

Enumeration for AHB (Advanced High-Performance Bus) prescaler.

Enumerator:

MRCC_AHB_PRESCALER_NOT_DIVIDED	AHB prescaler: Not divided.
MRCC_AHB_PRESCALER_2	AHB prescaler: 2.
MRCC_AHB_PRESCALER_4	AHB prescaler: 4.
MRCC_AHB_PRESCALER_8	AHB prescaler: 8.
MRCC_AHB_PRESCALER_16	AHB prescaler: 16.
MRCC_AHB_PRESCALER_64	AHB prescaler: 64.

MRCC_AHB_PRESCALER_128	AHB prescaler: 128.
MRCC_AHB_PRESCALER_256	AHB prescaler: 256.
MRCC_AHB_PRESCALER_512	AHB prescaler: 512.

enum EN_MRCC_apbPrescalerSpeed_t

Enumeration for APB (Advanced Peripheral Bus) prescaler speed.

Enumerator:

MRCC_APB_PRESCALER_SPEED_NOT_DIVIDED	APB prescaler speed: Not divided.
MRCC_APB_PRESCALER_SPEED_2	APB prescaler speed: 2.
MRCC_APB_PRESCALER_SPEED_4	APB prescaler speed: 4.
MRCC_APB_PRESCALER_SPEED_8	APB prescaler speed: 8.
MRCC_APB_PRESCALER_SPEED_16	APB prescaler speed: 16.

enum EN_MRCC_busOptions_t

Enumeration for different buses in MRCC.

Enumerator:

MRCC_AHP1_BUS	AHP1 bus.
MRCC_AHP2_BUS	AHP2 bus.
MRCC_APB1_BUS	APB1 bus.
MRCC_APB2_BUS	APB2 bus.

enum EN_MRCC_peripheralOptions_t

Enumeration for different peripheral options in MRCC.

Enumerator:

MRCC_GPIOA_PERIPHERAL	
MRCC_GPIOB_P	

ERIPHERAL	
MRCC_CRC_PERIPHERAL	
MRCC_DMA1_PERIPHERAL	
MRCC_DMA2_PERIPHERAL	
MRCC_OTGFS_PERIPHERAL	
MRCC_TIM2_PERIPHERAL	
MRCC_TIM3_PERIPHERAL	
MRCC_TIM4_PERIPHERAL	
MRCC_TIM5_PERIPHERAL	
MRCC_WWDG_PERIPHERAL	
MRCC_SPI2_PERIPHERAL	
MRCC_SPI3_PERIPHERAL	
MRCC_USART2_PERIPHERAL	
MRCC_I2C1_PERIPHERAL	
MRCC_I2C2_PERIPHERAL	
MRCC_I2C3_PERIPHERAL	
MRCC_PWR_PERIPHERAL	
MRCC_TIM1_PERIPHERAL	
MRCC_USART1_PERIPHERAL	
MRCC_USART6_PERIPHERAL	
MRCC_ADC1_PERIPHERAL	
MRCC_SDIO_PERIPHERAL	
MRCC_SPI1_PERIPHERAL	
MRCC_SPI4_PERIPHERAL	
MRCC_SYSCFG_PERIPHERAL	
MRCC_TIM9_PERIPHERAL	
MRCC_TIM10_PERIPHERAL	
MRCC_TIM11_PERIPHERAL	

enum EN_MRCC_pllClkSourceEntry_t

Enumeration for the PLL entry clock source.

Enumerator:

MRCC_PLL_ENTRY_CLK_HSI	PLL entry clock source: HSI.
MRCC_PLL_ENTRY_CLK_HSE	PLL entry clock source: HSE.

enum EN_MRCC_pllDivisionFactor_t

Enumeration for the PLL division factor.

Enumerator:

MRCC_PLL_DIVISION_FACTOR_2	PLL division factor: 2.
MRCC_PLL_DIVISION_FACTOR_4	PLL division factor: 4.
MRCC_PLL_DIVISION_FACTOR_6	PLL division factor: 6.
MRCC_PLL_DIVISION_FACTOR_8	PLL division factor: 8.

enum EN_MRCC_pllStatus_t

Enumeration for PLL status.

Enumerator:

MRCC_PLL_DISABLE	PLL is disabled.
MRCC_PLL_ENABLE	PLL is enabled.

enum EN_MRCC_sysClkSelect_t

Enumeration for the system clock selection.

Enumerator:

MRCC_SYS_CLK_HSI	HSI (High-Speed Internal) oscillator.
MRCC_SYS_CLK_HSE_BYPASS	HSE (High-Speed External) oscillator with bypass.
MRCC_SYS_CLK_HSE_NOT_BYPASS	HSE oscillator without bypass.
MRCC_SYS_CLK	PLL (Phase-Locked Loop).

_PLL	
MRCC_SYS_CLK_NOT_ALLOWED	Not allowed system clock source.

enum EN_MRCC_systemState_t

Enumerator:

MRCC_OK	Operation successful.
MRCC_NOK	Operation failed.
MRCC_PTR_NULL	Null pointer encountered.
MRCC_INVALID_PARAMTER	Invalid parameter detected.

rcc_config.h

Go to the documentation of this file.

```
/*****************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : STM32F401xC_Drivers
// File           : rcc_config.h
// Date           : Sep 8, 2023
// GitHub         : https://github.com/sherifkhadr
/*****************************************************************************/
#ifndef MCAL_RCC_RCC_CONFIG_H_
#define MCAL_RCC_RCC_CONFIG_H_

 #define sysClkSelect                    MRCC_SYS_CLK_HSI
 #define pllStatus                       MRCC_PLL_DISABLE
 #define pllSourceOfEntryClk             MRCC_PLL_ENTRY_CLK_HSI
 #define pllpDivisionFactor              MRCC_PLL_DIVISION_FACTOR_2
 #define pllmDivisionFactor              0
 #define pllnMulFactor                   0
 #define apbHighSpeedPrescaler           MRCC_APB_PRESCALER_SPEED_NOT_DIVIDED
 #define apbLowSpeedPrescaler            MRCC_APB_PRESCALER_SPEED_NOT_DIVIDED
 #define ahbPrescaler                    MRCC_AHB_PRESCALER_2

 typedef enum
 {
     MRCC_OK = 0,
     MRCC_NOK,
     MRCC_PTR_NULL,
     MRCC_INVALID_PARAMTER
 } EN_MRCC_systemState_t;

 typedef enum
 {
     MRCC_SYS_CLK_HSI = 0,
     MRCC_SYS_CLK_HSE_BYPASS,
     MRCC_SYS_CLK_HSE_NOT_BYPASS = 1,
     MRCC_SYS_CLK_PLL,
     MRCC_SYS_CLK_NOT_ALLOWED
 } EN_MRCC_sysClkSelect_t;

 typedef enum
 {
     MRCC_PLL_ENTRY_CLK_HSI = 0,
     MRCC_PLL_ENTRY_CLK_HSE
 } EN_MRCC_pllClkSourceEntry_t;

 typedef enum
 {
     MRCC_PLL_DIVISION_FACTOR_2 = 0,
     MRCC_PLL_DIVISION_FACTOR_4,
     MRCC_PLL_DIVISION_FACTOR_6,
     MRCC_PLL_DIVISION_FACTOR_8
 } EN_MRCC_pllDivisionFactor_t;

 typedef enum
 {
     MRCC_APB_PRESCALER_SPEED_NOT_DIVIDED = 0,
     MRCC_APB_PRESCALER_SPEED_2 = 4,
     MRCC_APB_PRESCALER_SPEED_4,
     MRCC_APB_PRESCALER_SPEED_8,
     MRCC_APB_PRESCALER_SPEED_16
 } EN_MRCC_apbPrescalerSpeed_t;

 typedef enum
 {
     MRCC_AHB_PRESCALER_NOT_DIVIDED = 0,
     MRCC_AHB_PRESCALER_2 = 8,
     MRCC_AHB_PRESCALER_4,
     MRCC_AHB_PRESCALER_8,
     MRCC_AHB_PRESCALER_16,
     MRCC_AHB_PRESCALER_64,
     MRCC_AHB_PRESCALER_128,
     MRCC_AHB_PRESCALER_256,
     MRCC_AHB_PRESCALER_512
 } EN_MRCC_ahbPrescaler_t;
```

```c

typedef enum
{
    MRCC_PLL_DISABLE = 0,
    MRCC_PLL_ENABLE
} EN_MRCC_pllStatus_t;

typedef enum
{
    MRCC_AHP1_BUS = 0,
    MRCC_AHP2_BUS,
    MRCC_APB1_BUS,
    MRCC_APB2_BUS
} EN_MRCC_busOptions_t;

typedef enum
{
    MRCC_GPIOA_PERIPHERAL = 0,
    MRCC_GPIOB_PERIPHERAL,
    MRCC_CRC_PERIPHERAL = 12,
    MRCC_DMA1_PERIPHERAL = 21,
    MRCC_DMA2_PERIPHERAL,
    MRCC_OTGFS_PERIPHERAL = 7,
    MRCC_TIM2_PERIPHERAL = 0,
    MRCC_TIM3_PERIPHERAL,
    MRCC_TIM4_PERIPHERAL,
    MRCC_TIM5_PERIPHERAL,
    MRCC_WWDG_PERIPHERAL = 11,
    MRCC_SPI2_PERIPHERAL = 14,
    MRCC_SPI3_PERIPHERAL = 15,
    MRCC_USART2_PERIPHERAL = 17,
    MRCC_I2C1_PERIPHERAL = 21,
    MRCC_I2C2_PERIPHERAL,
    MRCC_I2C3_PERIPHERAL,
    MRCC_PWR_PERIPHERAL = 28,
    MRCC_TIM1_PERIPHERAL = 0,
    MRCC_USART1_PERIPHERAL = 4,
    MRCC_USART6_PERIPHERAL,
    MRCC_ADC1_PERIPHERAL = 8,
    MRCC_SDIO_PERIPHERAL = 11,
    MRCC_SPI1_PERIPHERAL,
    MRCC_SPI4_PERIPHERAL,
    MRCC_SYSCFG_PERIPHERAL,
    MRCC_TIM9_PERIPHERAL = 16,
    MRCC_TIM10_PERIPHERAL,
    MRCC_TIM11_PERIPHERAL

}EN_MRCC_peripheralOptions_t;


#endif /* MCAL_RCC_RCC_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/rcc/rcc_interface.h File Reference

Header file for RCC (Reset and Clock Control) module.

```
#include "../../COMMON/bit_math.h"
#include "../../COMMON/std_types.h"
#include "rcc_private.h"
#include "rcc_config.h"
```

Functions

- **EN_MRCC_systemState_t MRCC_Init** (void)
  Initialize the MRCC (Reset and Clock Control) module.
- **EN_MRCC_systemState_t MRCC_enablePeripheral** (**EN_MRCC_busOptions_t** busSelection, **EN_MRCC_peripheralOptions_t** PeripheralNumber)
  Enable a specific peripheral on a selected bus.
- **EN_MRCC_systemState_t MRCC_disablePeripheral** (**EN_MRCC_busOptions_t** busSelection, **EN_MRCC_peripheralOptions_t** PeripheralNumber)
  Disable a specific peripheral on a selected bus.
- void **HAL_DeInit** (void)

Detailed Description

Header file for RCC (Reset and Clock Control) module.

Function Documentation

void HAL_DeInit (void)

EN_MRCC_systemState_t MRCC_disablePeripheral (EN_MRCC_busOptions_t *busSelection*, EN_MRCC_peripheralOptions_t *PeripheralNumber*)

Disable a specific peripheral on a selected bus.

This function disables a peripheral on the specified bus.

Parameters

busSelection	The bus on which the peripheral is located. Possible values are: • #EN_MRCC_AHP1_BUS • #EN_MRCC_AHP2_BUS • #EN_MRCC_APB1_BUS • #EN_MRCC_APB2_BUS
PeripheralNumber	The specific peripheral to disable. Refer to the enumeration **EN_MRCC_peripheralOptions_t** for available options.

Returns

The state of peripheral disabling. Possible values are:

- #EN_MRCC_OK: Peripheral disabling successful.
- #EN_MRCC_NOK: Peripheral disabling failed.
- #EN_MRCC_PTR_NULL: Null pointer encountered during the operation.
- #EN_MRCC_INVALID_PARAMTER: Invalid parameter detected during the operation.

EN_MRCC_systemState_t MRCC_enablePeripheral (EN_MRCC_busOptions_t *busSelection*, EN_MRCC_peripheralOptions_t *PeripheralNumber*)

Enable a specific peripheral on a selected bus.

This function enables a peripheral on the specified bus.

Parameters

busSelection	The bus on which the peripheral is located. Possible values are: • #EN_MRCC_AHP1_BUS • #EN_MRCC_AHP2_BUS • #EN_MRCC_APB1_BUS • #EN_MRCC_APB2_BUS
PeripheralNumber	The specific peripheral to enable. Refer to the enumeration **EN_MRCC_peripheralOptions_t** for available options.

Returns

The state of peripheral enabling. Possible values are:

- #EN_MRCC_OK: Peripheral enabling successful.
- #EN_MRCC_NOK: Peripheral enabling failed.
- #EN_MRCC_PTR_NULL: Null pointer encountered during the operation.
- #EN_MRCC_INVALID_PARAMTER: Invalid parameter detected during the operation.

EN_MRCC_systemState_t MRCC_Init (void)

Initialize the MRCC (Reset and Clock Control) module.

This function initializes the MRCC module, configuring the system clocks and other essential settings.

Returns

The system initialization state. Possible values are:

- #EN_MRCC_OK: Initialization successful.
- #EN_MRCC_NOK: Initialization failed.
- #EN_MRCC_PTR_NULL: Null pointer encountered during initialization.
- #EN_MRCC_INVALID_PARAMTER: Invalid parameter detected during initialization.

rcc_interface.h

Go to the documentation of this file.

```
#ifndef MCAL_RCC_RCC_INTERFACE_H_
#define MCAL_RCC_RCC_INTERFACE_H_


#include "../../COMMON/bit_math.h"
#include "../../COMMON/std_types.h"
#include "rcc_private.h"
#include "rcc_config.h"

EN_MRCC_systemState_t MRCC_Init(void);

EN_MRCC_systemState_t MRCC_enablePeripheral(EN_MRCC_busOptions_t busSelection,
EN_MRCC_peripheralOptions_t PeripheralNumber);

EN_MRCC_systemState_t MRCC_disablePeripheral(EN_MRCC_busOptions_t busSelection,
EN_MRCC_peripheralOptions_t PeripheralNumber);

void HAL_DeInit(void);

#endif /* MCAL_RCC_RCC_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/rcc/rcc_private.h File Reference

Data Structures

struct ST_MRCC_RegistersMap_t**Macros**

- #define **RCC_PERIPHERAL_BASE_ADDR**   (0x40023800)
- #define **MRCC_PERIPHERAL**   ((volatile **ST_MRCC_RegistersMap_t** *)**RCC_PERIPHERAL_BASE_ADDR**)
- #define **HSION_BIT**   0
- #define **HSIRDY_BIT**   1
- #define **HSEON_BIT**   16
- #define **HSERDY_BIT**   17
- #define **HSEBYP_BIT**   18
- #define **CSSON_BIT**   19
- #define **PLLON_BIT**   24
- #define **PLLRDY_BIT**   25
- #define **PLLI2SON_BIT**   26
- #define **PLLI2SRDY_BIT**   27
- #define **PLLM0_BIT**   0
- #define **PLLM1_BIT**   1
- #define **PLLM2_BIT**   2
- #define **PLLM3_BIT**   3
- #define **PLLM4_BIT**   4
- #define **PLLM5_BIT**   5
- #define **PLLN0_BIT**   6
- #define **PLLP0_BIT**   16
- #define **PLLSRC_BIT**   22
- #define **SW0_BIT**   0
- #define **SW1_BIT**   1
- #define **SWS0_BIT**   2
- #define **SWS1_BIT**   3
- #define **HPRE0_BIT**   4
- #define **PPRE10_BIT**   10
- #define **PPRE20_BIT**   13
- #define **__HAL_RCC_APB1_FORCE_RESET**()   (**MRCC_PERIPHERAL**->RCC_APB1RSTR_REG = 0xFFFFFFFFU)
- #define **__HAL_RCC_APB1_RELEASE_RESET**()   (**MRCC_PERIPHERAL**->RCC_APB1RSTR_REG = 0x00U)
- #define **__HAL_RCC_APB2_FORCE_RESET**()   (**MRCC_PERIPHERAL**->RCC_APB2RSTR_REG = 0xFFFFFFFFU)
- #define **__HAL_RCC_APB2_RELEASE_RESET**()   (**MRCC_PERIPHERAL**->RCC_APB2RSTR_REG = 0x00U)
- #define **__HAL_RCC_AHB1_FORCE_RESET**()   (**MRCC_PERIPHERAL**->RCC_AHB1RSTR_REG = 0xFFFFFFFFU)
- #define **__HAL_RCC_AHB1_RELEASE_RESET**()   (**MRCC_PERIPHERAL**->RCC_AHB1RSTR_REG = 0x00U)

Macro Definition Documentation

#define __HAL_RCC_AHB1_FORCE_RESET()   (MRCC_PERIPHERAL->RCC_AHB1RSTR_REG = 0xFFFFFFFFU)

#define __HAL_RCC_AHB1_RELEASE_RESET()   (MRCC_PERIPHERAL->RCC_AHB1RSTR_REG = 0x00U)

#define __HAL_RCC_APB1_FORCE_RESET()   (MRCC_PERIPHERAL->RCC_APB1RSTR_REG = 0xFFFFFFFFU)

#define __HAL_RCC_APB1_RELEASE_RESET()   (MRCC_PERIPHERAL->RCC_APB1RSTR_REG = 0x00U)

#define __HAL_RCC_APB2_FORCE_RESET()   (MRCC_PERIPHERAL->RCC_APB2RSTR_REG = 0xFFFFFFFFU)

#define __HAL_RCC_APB2_RELEASE_RESET()   (MRCC_PERIPHERAL->RCC_APB2RSTR_REG = 0x00U)

#define CSSON_BIT   19

#define HPRE0_BIT   4

#define HSEBYP_BIT   18

#define HSEON_BIT   16

#define HSERDY_BIT   17

#define HSION_BIT   0

#define HSIRDY_BIT   1

#define MRCC_PERIPHERAL   ((volatile ST_MRCC_RegistersMap_t *)RCC_PERIPHERAL_BASE_ADDR)

#define PLLI2SON_BIT   26

#define PLLI2SRDY_BIT   27

#define PLLM0_BIT   0

#define PLLM1_BIT   1

#define PLLM2_BIT   2

#define PLLM3_BIT   3

```
#define PLLM4_BIT   4

#define PLLM5_BIT   5

#define PLLN0_BIT   6

#define PLLON_BIT   24

#define PLLP0_BIT   16

#define PLLRDY_BIT   25

#define PLLSRC_BIT   22

#define PPRE10_BIT   10

#define PPRE20_BIT   13

#define RCC_PERIPHERAL_BASE_ADDR   (0x40023800)

#define SW0_BIT   0

#define SW1_BIT   1

#define SWS0_BIT   2

#define SWS1_BIT   3
```

rcc_private.h

Go to the documentation of this file.

```c
/*******************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : STM32F401xC_Drivers
// File           : rcc_private.h
// Date           : Sep 8, 2023
// GitHub         : https://github.com/sherifkhadr
/*******************************************************************************/
#ifndef MCAL_RCC_RCC_PRIVATE_H_
#define MCAL_RCC_RCC_PRIVATE_H_


#define  RCC_PERIPHERAL_BASE_ADDR       (0x40023800)


typedef struct
{
    vuint32_t RCC_CR_REG;
    vuint32_t RCC_PLLCFGR_REG;
    vuint32_t RCC_CFGR_REG;
    vuint32_t RCC_CIR_REG;
    vuint32_t RCC_AHB1RSTR_REG;
    vuint32_t RCC_AHB2RSTR_REG;
    vuint32_t RESERVED0_REG;
    vuint32_t RESERVED1_REG;
    vuint32_t RCC_APB1RSTR_REG;
    vuint32_t RCC_APB2RSTR_REG;
    vuint32_t RESERVED2_REG;
    vuint32_t RESERVED3_REG;
    vuint32_t RCC_AHB1ENR_REG;
    vuint32_t RCC_AHB2ENR_REG;
    vuint32_t Reserved5_REG;
    vuint32_t Reserved6_REG;
    vuint32_t RCC_APB1ENR_REG;
    vuint32_t RCC_APB2ENR_REG;
    vuint32_t RESERVED7_REG;
    vuint32_t RESERVED8_REG;
    vuint32_t RCC_AHB1LPENR_REG;
    vuint32_t RCC_AHB2LPENR_REG;
    vuint32_t RESERVED9_REG;
    vuint32_t RESERVED10_REG;
    vuint32_t RCC_APB1LPENR_REG;
    vuint32_t RCC_APB2LPENR_REG;
    vuint32_t RESERVED11_REG;
    vuint32_t RESERVED12_REG;
    vuint32_t RCC_BDCR_REG;
    vuint32_t RCC_CSR_REG;
    vuint32_t RESERVED13_REG;
    vuint32_t RESERVED14_REG;
    vuint32_t RCC_SSCGR_REG;
    vuint32_t RCC_PLLI2SCFGR_REG;
    vuint32_t RESERVED15_REG;
    vuint32_t RCC_DCKCFGR_REG;
}ST_MRCC_RegistersMap_t;

#define MRCC_PERIPHERAL ((volatile ST_MRCC_RegistersMap_t *)RCC_PERIPHERAL_BASE_ADDR)

/* RCC_CR_REG Bits */

#define HSION_BIT           0
#define HSIRDY_BIT          1
#define HSEON_BIT           16
#define HSERDY_BIT          17
#define HSEBYP_BIT          18
#define CSSON_BIT           19
#define PLLON_BIT           24
#define PLLRDY_BIT          25
#define PLLI2SON_BIT        26
#define PLLI2SRDY_BIT       27

/* RCC_PLLCFGR_REG Bits */

#define PLLM0_BIT   0
```

```c
#define PLLM1_BIT   1
#define PLLM2_BIT   2
#define PLLM3_BIT   3
#define PLLM4_BIT   4
#define PLLM5_BIT   5
#define PLLN0_BIT   6
#define PLLP0_BIT   16
#define PLLSRC_BIT  22

/* RCC_CFGR_REG Bits */

#define SW0_BIT     0
#define SW1_BIT     1
#define SWS0_BIT    2
#define SWS1_BIT    3
#define HPRE0_BIT   4
#define PPRE10_BIT  10
#define PPRE20_BIT  13


#define __HAL_RCC_APB1_FORCE_RESET()      (MRCC_PERIPHERAL->RCC_APB1RSTR_REG = 0xFFFFFFFFU)
#define __HAL_RCC_APB1_RELEASE_RESET()    (MRCC_PERIPHERAL->RCC_APB1RSTR_REG = 0x00U)
#define __HAL_RCC_APB2_FORCE_RESET()      (MRCC_PERIPHERAL->RCC_APB2RSTR_REG = 0xFFFFFFFFU)
#define __HAL_RCC_APB2_RELEASE_RESET()    (MRCC_PERIPHERAL->RCC_APB2RSTR_REG = 0x00U)
#define __HAL_RCC_AHB1_FORCE_RESET()      (MRCC_PERIPHERAL->RCC_AHB1RSTR_REG = 0xFFFFFFFFU)
#define __HAL_RCC_AHB1_RELEASE_RESET()    (MRCC_PERIPHERAL->RCC_AHB1RSTR_REG = 0x00U)
#endif /* MCAL_RCC_RCC_PRIVATE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/systick/systick_config.h File Reference

Macros

- #define **MIN_VAL_OF_U32**   0
- #define **MAX_VAL_OF_U32**   0xFFFFFFFF
- #define **MSTK_IntervalSingle**   0
- #define **MSTK_IntervalPeriodic**   1

Enumerations

- enum **EN_MSTK_systemState_t** { **MSTK_OK** = 0, **MSTK_NOK**, **MSTK_INVALID_PARAMTER** }
  Enumeration for the system state of SysTick functions.
- enum **EN_MSTK_clkSourceOptions_t** { **MSTK_CLK_AHB_8** = 0, **MSTK_CLK_PROCESSOR_AHB** }
  Enumeration for the clock source options in SysTick.
- enum **EN_MSTK_interruptStates_t** { **MSTK_INTERRUPT_ENABLED** = 0, **MSTK_INTERRUPT_DISABLED** }
  Enumeration for the interrupt states in SysTick.

Macro Definition Documentation

#define MAX_VAL_OF_U32   0xFFFFFFFF

#define MIN_VAL_OF_U32   0

#define MSTK_IntervalPeriodic   1

#define MSTK_IntervalSingle   0

Enumeration Type Documentation

enum EN_MSTK_clkSourceOptions_t

Enumeration for the clock source options in SysTick.

Enumerator:

MSTK_CLK_AHB_8	SysTick clock source is AHB/8.
MSTK_CLK_PROCESSOR_AHB	SysTick clock source is the processor clock (AHB).

enum EN_MSTK_interruptStates_t

Enumeration for the interrupt states in SysTick.

Enumerator:

MSTK_INTERRUPT_ENABLED	SysTick interrupt is enabled.
MSTK_INTERRUPT_DISABLED	SysTick interrupt is disabled.

enum EN_MSTK_systemState_t

Enumeration for the system state of SysTick functions.

Enumerator:

MSTK_OK	Operation successful.
MSTK_NOK	Operation failed.
MSTK_INVALID_PARAMTER	Invalid parameter detected.

systick_config.h

Go to the documentation of this file.

```c
/*************************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : STM32F401xC
// File          : systick_config.h
// Date          : Sep 12, 2023
// GitHub        : https://github.com/sherifkhadr
/*************************************************************************/
#ifndef MCAL_SYSTICK_SYSTICK_CONFIG_H_
#define MCAL_SYSTICK_SYSTICK_CONFIG_H_


#define MIN_VAL_OF_U32              0
#define MAX_VAL_OF_U32              0xFFFFFFFF

#define MSTK_IntervalSingle         0
#define MSTK_IntervalPeriodic       1


typedef enum
{
    MSTK_OK = 0,
    MSTK_NOK,
    MSTK_INVALID_PARAMTER
} EN_MSTK_systemState_t;

typedef enum
{
    MSTK_CLK_AHB_8 = 0,
    MSTK_CLK_PROCESSOR_AHB
} EN_MSTK_clkSourceOptions_t;

typedef enum
{
    MSTK_INTERRUPT_ENABLED = 0,
    MSTK_INTERRUPT_DISABLED
} EN_MSTK_interruptStates_t;

#endif /* MCAL_SYSTICK_SYSTICK_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/systick/systick_interface.h File Reference

Header file for the SysTick (System Timer) module interface.

```
#include "../../COMMON/std_types.h"
#include "../../COMMON/bit_math.h"
#include "systick_private.h"
#include "systick_config.h"
```

Functions

- **EN_MSTK_systemState_t MSTK_init** (**EN_MSTK_clkSourceOptions_t** copy_uddtClkSource, **EN_MSTK_interruptStates_t** copy_uddtIntStates)
  Initialize the SysTick timer.
- **EN_MSTK_systemState_t MSTK_setBusyWait** (**uint32_t** copy_u32NumberOfTicks)
  Set a busy-wait delay using the SysTick timer.
- **EN_MSTK_systemState_t MSTK_SetIntervalSingle** (**uint32_t** copy_u32NumberOfTicks, void(*Pf)(void))
  Set a single-shot interval using the SysTick timer.
- **EN_MSTK_systemState_t MSTK_SetIntervalPeriodic** (**uint32_t** copy_u32NumberOfTicks, void(*Pf)(void))
  Set a periodic interval using the SysTick timer.
- **EN_MSTK_systemState_t MSTK_StopInterval** (void)
  Stop the current interval in the SysTick timer.
- **EN_MSTK_systemState_t MSTK_getElapsedTime** (**uint32_t** *copy_u32PtrRetOfElapsedTicks)
  Get the elapsed time since the last SysTick timer initialization.

Detailed Description

Header file for the SysTick (System Timer) module interface.

Function Documentation

EN_MSTK_systemState_t MSTK_getElapsedTime (uint32_t * *copy_u32PtrRetOfElapsedTicks*)

Get the elapsed time since the last SysTick timer initialization.

This function retrieves the elapsed time since the last SysTick timer initialization.

Parameters

copy_u32PtrRetOfElapsedTicks	Pointer to store the result of the elapsed ticks.

Returns

The system state after getting the elapsed time.

- MSTK_OK: Operation successful.
- MSTK_NOK: Operation failed.
- MSTK_INVALID_PARAMTER: Invalid parameter detected.

EN_MSTK_systemState_t MSTK_init (EN_MSTK_clkSourceOptions_t *copy_uddtClkSource*, EN_MSTK_interruptStates_t *copy_uddtIntStates*)

Initialize the SysTick timer.

This function initializes the SysTick timer with the specified clock source and interrupt state.

Parameters

copy_uddtClkSource	The clock source option (MSTK_CLK_AHB_8, MSTK_CLK_PROCESSOR_AHB).
copy_uddtIntStates	The interrupt state option (MSTK_INTERRUPT_ENABLED, MSTK_INTERRUPT_DISABLED).

Returns

The system state after initializing the SysTick timer.

- MSTK_OK: Operation successful.
- MSTK_NOK: Operation failed.
- MSTK_INVALID_PARAMTER: Invalid parameter detected.

EN_MSTK_systemState_t MSTK_setBusyWait (uint32_t *copy_u32NumberOfTicks*)

Set a busy-wait delay using the SysTick timer.

This function sets a busy-wait delay using the SysTick timer for the specified number of ticks.

Parameters

copy_u32NumberOfTicks	The number of ticks for the busy-wait delay.

Returns

The system state after setting the busy-wait delay.

- MSTK_OK: Operation successful.
- MSTK_NOK: Operation failed.
- MSTK_INVALID_PARAMTER: Invalid parameter detected.

EN_MSTK_systemState_t MSTK_SetIntervalPeriodic (uint32_t *copy_u32NumberOfTicks*, void(*)(void) *Pf*)

Set a periodic interval using the SysTick timer.

This function sets a periodic interval using the SysTick timer for the specified number of ticks and associates a callback function.

Parameters

copy_u32NumberOfTicks	The number of ticks for the periodic interval.
Pf	Pointer to the callback function to be executed after each interval elapses.

Returns

The system state after setting the periodic interval.

- MSTK_OK: Operation successful.
- MSTK_NOK: Operation failed.
- MSTK_INVALID_PARAMTER: Invalid parameter detected.

EN_MSTK_systemState_t MSTK_SetIntervalSingle (uint32_t *copy_u32NumberOfTicks*, void(*)(void) *Pf*)

Set a single-shot interval using the SysTick timer.

This function sets a single-shot interval using the SysTick timer for the specified number of ticks and associates a callback function.

Parameters

copy_u32NumberOfTicks	The number of ticks for the single-shot interval.
Pf	Pointer to the callback function to be executed after the interval elapses.

Returns

The system state after setting the single-shot interval.

- MSTK_OK: Operation successful.
- MSTK_NOK: Operation failed.
- MSTK_INVALID_PARAMTER: Invalid parameter detected.

EN_MSTK_systemState_t MSTK_StopInterval (void)

Stop the current interval in the SysTick timer.

This function stops the current interval in the SysTick timer.

Returns

The system state after stopping the interval.

- MSTK_OK: Operation successful.
- MSTK_NOK: Operation failed.
- MSTK_INVALID_PARAMTER: Invalid parameter detected.

systick_interface.h

Go to the documentation of this file.

```c

#ifndef MCAL_SYSTICK_SYSTICK_INTERFACE_H_
#define MCAL_SYSTICK_SYSTICK_INTERFACE_H_

#include "../../COMMON/std_types.h"
#include "../../COMMON/bit_math.h"
#include "systick_private.h"
#include "systick_config.h"

EN_MSTK_systemState_t MSTK_init(EN_MSTK_clkSourceOptions_t copy_uddtClkSource,
EN_MSTK_interruptStates_t copy_uddtIntStates);

EN_MSTK_systemState_t MSTK_setBusyWait(uint32_t copy_u32NumberOfTicks);

EN_MSTK_systemState_t MSTK_SetIntervalSingle(uint32_t copy_u32NumberOfTicks, void
(*Pf)(void));

EN_MSTK_systemState_t MSTK_SetIntervalPeriodic(uint32_t copy_u32NumberOfTicks, void
(*Pf)(void));

EN_MSTK_systemState_t MSTK_StopInterval(void);

EN_MSTK_systemState_t MSTK_getElapsedTime(uint32_t *copy_u32PtrRetOfElapsedTicks);

#endif /* MCAL_SYSTICK_SYSTICK_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/systick/systick_private.h File Reference

Data Structures

struct ST_MSTK_RegistersMap_t**Macros**

- #define **MSTK_PERIPHERAL_BASE_ADDR**   (0xE000E010)
- #define **MSTK_ENABLE_BIT**   0
- #define **MSTK_TICKINT_BIT**   1
- #define **MSTK_CLKSOURCE_BIT**   2
- #define **MSTK_COUNTFLAG_BIT**   16
- #define **MSTK_PERIPHERAL**   ((volatile **ST_MSTK_RegistersMap_t** *)**MSTK_PERIPHERAL_BASE_ADDR**)

Macro Definition Documentation

#define MSTK_CLKSOURCE_BIT   2

#define MSTK_COUNTFLAG_BIT   16

#define MSTK_ENABLE_BIT   0

#define MSTK_PERIPHERAL   ((volatile ST_MSTK_RegistersMap_t *)MSTK_PERIPHERAL_BASE_ADDR)

#define MSTK_PERIPHERAL_BASE_ADDR   (0xE000E010)

#define MSTK_TICKINT_BIT   1

systick_private.h

Go to the documentation of this file.

```
/*************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : STM32F401xC
// File           : systick_private.h
// Date           : Sep 12, 2023
// GitHub         : https://github.com/sherifkhadr
/*************************************************************************/
#ifndef MCAL_SYSTICK_SYSTICK_PRIVATE_H_
#define MCAL_SYSTICK_SYSTICK_PRIVATE_H_

#define  MSTK_PERIPHERAL_BASE_ADDR          (0xE000E010)


typedef struct
{

    vuint32_t MSTK_STK_CTRL;
    vuint32_t MSTK_STK_LOAD;
    vuint32_t MSTK_STK_VAL;
    vuint32_t MSTK_STK_CALIB;

}ST_MSTK_RegistersMap_t;

#define MSTK_ENABLE_BIT          0
#define MSTK_TICKINT_BIT         1
#define MSTK_CLKSOURCE_BIT       2
#define MSTK_COUNTFLAG_BIT       16

#define MSTK_PERIPHERAL     ((volatile ST_MSTK_RegistersMap_t
*)MSTK_PERIPHERAL_BASE_ADDR)


#endif /* MCAL_SYSTICK_SYSTICK_PRIVATE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/tmr/tmr_config.h File Reference

Enumerations

- enum **EN_MTMR_number_t** { **MTMR2** = 0, **MTMR3**, **MTMR4**, **MTMR5** }
  Enumeration for Multi-Function Timer numbers.
- enum **EN_MTMR_channel_t** { **MTMR_CH1** = 1, **MTMR_CH2**, **MTMR_CH3**, **MTMR_CH4** }
  Enumeration for Multi-Function Timer channels.
- enum **EN_MTMR_selectedMode_t** { **MTMR_MODE_FROZEN** = 0, **MTMR_MODE_ACTIVE**, **MTMR_MODE_INACTIVE**, **MTMR_MODE_TOGGLE**, **MTMR_MODE_INACTIVE_FORCE**, **MTMR_MODE_ACTIVE_FORCE**, **MTMR_MODE_PWM_MODE1**, **MTMR_MODE_PWM_MODE2** }
  Enumeration for selected modes of Multi-Function Timer.
- enum **CH_MODE_t** { **OUTPUT**, **IC_T2**, **IC_T1**, **IC_TRC** }
  Enumeration for Multi-Function Timer channel modes.
- enum **EDGE_t** { **RISIN**, **FALLIN**, **RESERVED**, **BOTH** }
  Enumeration for Multi-Function Timer edge options.

Enumeration Type Documentation

enum CH_MODE_t

Enumeration for Multi-Function Timer channel modes.

Enumerator:

OUTPUT	Output mode.
IC_T2	Input capture mode T2.
IC_T1	Input capture mode T1.
IC_TRC	Input capture mode TRC.

enum EDGE_t

Enumeration for Multi-Function Timer edge options.

Enumerator:

RISIN	Rising edge.
FALLIN	Falling edge.

RESERVED	Reserved.
BOTH	Both edges.

enum EN_MTMR_channel_t

Enumeration for Multi-Function Timer channels.

Enumerator:

MTMR_CH1	Multi-Function Timer Channel 1.
MTMR_CH2	Multi-Function Timer Channel 2.
MTMR_CH3	Multi-Function Timer Channel 3.
MTMR_CH4	Multi-Function Timer Channel 4.

enum EN_MTMR_number_t

Enumeration for Multi-Function Timer numbers.

Enumerator:

MTMR2	Multi-Function Timer 2.
MTMR3	Multi-Function Timer 3.
MTMR4	Multi-Function Timer 4.
MTMR5	Multi-Function Timer 5.

enum EN_MTMR_selectedMode_t

Enumeration for selected modes of Multi-Function Timer.

Enumerator:

MTMR_MODE_FROZEN	Frozen mode.
MTMR_MODE_ACTIVE	Active mode.
MTMR_MODE_INACTIVE	Inactive mode.
MTMR_MODE_TOGGLE	Toggle mode.

MTMR_MODE_INACTIVE_FORCE	Inactive force mode.
MTMR_MODE_ACTIVE_FORCE	Active force mode.
MTMR_MODE_PWM_MODE1	PWM mode 1.
MTMR_MODE_PWM_MODE2	PWM mode 2.

tmr_config.h

Go to the documentation of this file.

```
/******************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : Adaptive_Cruise_Control
// File           : tmr_config.h
// Date           : Oct 14, 2023
// GitHub         : https://github.com/sherifkhadr
/******************************************************************************/
#ifndef MCAL_TMR_TMR_CONFIG_H_
#define MCAL_TMR_TMR_CONFIG_H_


typedef enum
{
    MTMR2 = 0,
    MTMR3,
    MTMR4,
    MTMR5
} EN_MTMR_number_t;

typedef enum
{
    MTMR_CH1 = 1,
    MTMR_CH2,
    MTMR_CH3,
    MTMR_CH4
} EN_MTMR_channel_t;

typedef enum
{
    MTMR_MODE_FROZEN = 0,
    MTMR_MODE_ACTIVE,
    MTMR_MODE_INACTIVE,
    MTMR_MODE_TOGGLE,
    MTMR_MODE_INACTIVE_FORCE,
    MTMR_MODE_ACTIVE_FORCE,
    MTMR_MODE_PWM_MODE1,
    MTMR_MODE_PWM_MODE2
} EN_MTMR_selectedMode_t;

typedef enum
{
    OUTPUT,
    IC_T2,
    IC_T1,
    IC_TRC
} CH_MODE_t;

typedef enum
{
    RISIN,
    FALLIN,
    RESERVED,
    BOTH
} EDGE_t;
#endif /* MCAL_TMR_TMR_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/tmr/tmr_interface.h File Reference

Header file for the Multi-Mode Timer (MTMR) module.

```
#include "../../COMMON/bit_math.h"
#include "../../COMMON/std_types.h"
#include "tmr_private.h"
#include "tmr_config.h"
```

Functions

- void **MTMR_vStartTimer** (**EN_MTMR_number_t** copy_uddtTMRNumber)
  Start the specified Multi-Function Timer.

- void **MTMR_vStopTimer** (**EN_MTMR_number_t** copy_uddtTMRNumber)
  Stop the specified Multi-Function Timer.

- void **MTMR_vSetTimerPrescaler** (**EN_MTMR_number_t** copy_uddtTMRNumber, **uint16_t** copy_u16Value)
  Set the prescaler value for the specified Multi-Function Timer.

- void **MTMR_vEnableTimerOPM** (**EN_MTMR_number_t** copy_uddtTMRNumber)
  Enable One-Pulse Mode for the specified Multi-Function Timer.

- void **MTMR_vTimerCountRst** (**EN_MTMR_number_t** copy_uddtTMRNumber)
  Reset the count of the specified Multi-Function Timer.

- void **MTMR_vSetTimerChannelOutput** (**EN_MTMR_number_t** copy_uddtTMRNumber, **EN_MTMR_selectedMode_t** copy_uddtTimerMode, **EN_MTMR_channel_t** copy_uddtChannelNumber)
  Set the output mode for a specific channel of the Multi-Function Timer.

- void **MTMR_vSetTimerChannelInput** (**EN_MTMR_number_t** copy_uddtTMRNumber, **EN_MTMR_channel_t** copy_uddtChannelNumber)
  Set the input mode for a specific channel of the Multi-Function Timer.

- void **MTMR_vSetTimerARR** (**EN_MTMR_number_t** copy_uddtTMRNumber, **uint32_t** copy_u32Value)
  Set the Auto-Reload Register value for the specified Multi-Function Timer.

- void **MTMR_vSetTimerStop** (**EN_MTMR_number_t** copy_uddtTMRNumber)
  Stop the specified Multi-Function Timer.

- void **MTMR_vClearTimerCount** (**EN_MTMR_number_t** copy_uddtTMRNumber)
  Clear the count of the specified Multi-Function Timer.

- void **MTMR_vEnableTimerICUInt** (**EN_MTMR_number_t** copy_uddtTMRNumber)
  Enable the interrupt for the specified Multi-Function Timer input capture.

- void **MTMR_vSetTimerCMPVal** (**EN_MTMR_number_t** copy_uddtTMRNumber, **EN_MTMR_channel_t** copy_uddtChannelNumber, **uint32_t** copy_u32CmpValue)
  Set the compare value for the specified channel of the Multi-Function Timer.

- **uint32_t MTMR_vReadCaptureVal** (**EN_MTMR_number_t** copy_uddtTMRNumber, **EN_MTMR_channel_t** copy_uddtChannelNumber)
  Read the capture value for the specified channel of the Multi-Function Timer.

- void **MTMR3_vCaptureCompareInit** (void)
  Initialize the capture compare functionality for Multi-Function Timer 3.

Detailed Description

Header file for the Multi-Mode Timer (MTMR) module.

Function Documentation

void MTMR3_vCaptureCompareInit (void)

Initialize the capture compare functionality for Multi-Function Timer 3.

This function initializes the capture compare functionality for Multi-Function Timer 3.

Returns

No return.

void MTMR_vClearTimerCount (EN_MTMR_number_t *copy_uddtTMRNumber*)

Clear the count of the specified Multi-Function Timer.

This function clears the count of the specified Multi-Function Timer.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to clear. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**

Returns

No return.

void MTMR_vEnableTimerICUInt (EN_MTMR_number_t *copy_uddtTMRNumber*)

Enable the interrupt for the specified Multi-Function Timer input capture.

This function enables the interrupt for the specified Multi-Function Timer input capture.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to configure. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**

Returns

No return.

void MTMR_vEnableTimerOPM (EN_MTMR_number_t *copy_uddtTMRNumber*)

Enable One-Pulse Mode for the specified Multi-Function Timer.

This function enables One-Pulse Mode for the specified Multi-Function Timer.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to configure. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**

Returns

No return.

uint32_t MTMR_vReadCaptureVal (EN_MTMR_number_t *copy_uddtTMRNumber*, EN_MTMR_channel_t *copy_uddtChannelNumber*)

Read the capture value for the specified channel of the Multi-Function Timer.

This function reads the capture value for the specified channel of the Multi-Function Timer.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to read from. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**
copy_uddtChannelNumber	The channel number to read from. Possible values are: • **MTMR_CH1** • **MTMR_CH2** • **MTMR_CH3** • **MTMR_CH4**

Returns

The captured value from the specified channel.

void MTMR_vSetTimerARR (EN_MTMR_number_t *copy_uddtTMRNumber*, uint32_t *copy_u32Value*)

Set the Auto-Reload Register value for the specified Multi-Function Timer.

This function sets the Auto-Reload Register value for the specified Multi-Function Timer.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to configure. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**
copy_u32Value	The value to set for the Auto-Reload Register.

Returns

No return.

void MTMR_vSetTimerChannelInput (EN_MTMR_number_t *copy_uddtTMRNumber*, EN_MTMR_channel_t *copy_uddtChannelNumber*)

Set the input mode for a specific channel of the Multi-Function Timer.

This function sets the input mode for a specific channel of the Multi-Function Timer.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to configure. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**
copy_uddtChannelNumber	The channel number to configure. Possible values are: • **MTMR_CH1** • **MTMR_CH2** • **MTMR_CH3** • **MTMR_CH4**

Returns

No return.

void MTMR_vSetTimerChannelOutput (EN_MTMR_number_t *copy_uddtTMRNumber*, EN_MTMR_selectedMode_t *copy_uddtTimerMode*, EN_MTMR_channel_t *copy_uddtChannelNumber*)

Set the output mode for a specific channel of the Multi-Function Timer.

This function sets the output mode for a specific channel of the Multi-Function Timer.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to configure. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**
copy_uddtTimerMode	The mode to set for the timer channel. Possible values are: • **MTMR_MODE_FROZEN** • **MTMR_MODE_ACTIVE** • **MTMR_MODE_INACTIVE** • **MTMR_MODE_TOGGLE** • **MTMR_MODE_INACTIVE_FORCE**

	• **MTMR_MODE_ACTIVE_FORCE** • **MTMR_MODE_PWM_MODE1** • **MTMR_MODE_PWM_MODE2**
copy_uddtChannelNumber	The channel number to configure. Possible values are: • **MTMR_CH1** • **MTMR_CH2** • **MTMR_CH3** • **MTMR_CH4**

Returns

No return.

void MTMR_vSetTimerCMPVal (EN_MTMR_number_t *copy_uddtTMRNumber*, EN_MTMR_channel_t *copy_uddtChannelNumber*, uint32_t *copy_u32CmpValue*)

Set the compare value for the specified channel of the Multi-Function Timer.

This function sets the compare value for the specified channel of the Multi-Function Timer.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to configure. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**
copy_uddtChannelNumber	The channel number to configure. Possible values are: • **MTMR_CH1** • **MTMR_CH2** • **MTMR_CH3** • **MTMR_CH4**
copy_u32CmpValue	The compare value to set.

Returns

No return.

void MTMR_vSetTimerPrescaler (EN_MTMR_number_t *copy_uddtTMRNumber*, uint16_t *copy_u16Value*)

Set the prescaler value for the specified Multi-Function Timer.

This function sets the prescaler value for the specified Multi-Function Timer.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to configure. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**
copy_u16Value	The prescaler value to set.

Returns

No return.

void MTMR_vSetTimerStop (EN_MTMR_number_t *copy_uddtTMRNumber*)

Stop the specified Multi-Function Timer.

This function stops the specified Multi-Function Timer.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to stop. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**

Returns

No return.

void MTMR_vStartTimer (EN_MTMR_number_t *copy_uddtTMRNumber*)

Start the specified Multi-Function Timer.

This function starts the specified Multi-Function Timer.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to start. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**

Returns

No return.

void MTMR_vStopTimer (EN_MTMR_number_t *copy_uddtTMRNumber*)

Stop the specified Multi-Function Timer.

This function stops the specified Multi-Function Timer.

Parameters

copy_uddtTMRNumber	The Multi-Function Timer to stop. Possible values are: • **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**

Returns

No return.

void MTMR_vTimerCountRst (EN_MTMR_number_t *copy_uddtTMRNumber*)

Reset the count of the specified Multi-Function Timer.

This function resets the count of the specified Multi-Function Timer.

Parameters

copy_uddtTMRNu	The Multi-Function Timer to reset. Possible values are:

mber	• **MTMR2** • **MTMR3** • **MTMR4** • **MTMR5**

Returns

No return.

tmr_interface.h

Go to the documentation of this file.

```

#ifndef MCAL_TMR_TMR_INTERFACE_H_
#define MCAL_TMR_TMR_INTERFACE_H_

#include "../../COMMON/bit_math.h"
#include "../../COMMON/std_types.h"
#include "tmr_private.h"
#include "tmr_config.h"

void MTMR_vStartTimer(EN_MTMR_number_t copy_uddtTMRNumber);

void MTMR_vStopTimer(EN_MTMR_number_t copy_uddtTMRNumber);

void MTMR_vSetTimerPrescaler(EN_MTMR_number_t copy_uddtTMRNumber, uint16_t
copy_u16Value);

void MTMR_vEnableTimerOPM(EN_MTMR_number_t copy_uddtTMRNumber);

void MTMR_vTimerCountRst(EN_MTMR_number_t copy_uddtTMRNumber);

void MTMR_vSetTimerChannelOutput(EN_MTMR_number_t copy_uddtTMRNumber,
EN_MTMR_selectedMode_t copy_uddtTimerMode, EN_MTMR_channel_t copy_uddtChannelNumber);

void MTMR_vSetTimerChannelInput(EN_MTMR_number_t copy_uddtTMRNumber,
EN_MTMR_channel_t copy_uddtChannelNumber);

void MTMR_vSetTimerARR(EN_MTMR_number_t copy_uddtTMRNumber, uint32_t copy_u32Value);

void MTMR_vSetTimerStop(EN_MTMR_number_t copy_uddtTMRNumber);

void MTMR_vClearTimerCount(EN_MTMR_number_t copy_uddtTMRNumber);

void MTMR_vEnableTimerICUInt(EN_MTMR_number_t copy_uddtTMRNumber);

void MTMR_vSetTimerCMPVal(EN_MTMR_number_t copy_uddtTMRNumber, EN_MTMR_channel_t
copy_uddtChannelNumber, uint32_t copy_u32CmpValue);

uint32_t MTMR_vReadCaptureVal(EN_MTMR_number_t copy_uddtTMRNumber, EN_MTMR_channel_t
copy_uddtChannelNumber);

void MTMR3_vCaptureCompareInit(void);

#endif /* MCAL_TMR_TMR_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/tmr/tmr_private.h File Reference

Data Structures

struct ST_MTMRx_RegistersMap_t**Macros**

- #define **MTMR2_PERIPHERAL_BASE_ADDR**   (0x40000000)
- #define **MTMR3_PERIPHERAL_BASE_ADDR**   (0x40000400)
- #define **MTMR4_PERIPHERAL_BASE_ADDR**   (0x40000800)
- #define **MTMR5_PERIPHERAL_BASE_ADDR**   (0x40000C00)
- #define **CEN_BIT**   0
- #define **OPM_BIT**   3
- #define **CC1S_SHIFT**   0
- #define **OC1M_SHIFT**   4
- #define **CC2S_SHIFT**   8
- #define **OC2M_SHIFT**   12
- #define **CC3S_SHIFT**   0
- #define **OC3M_SHIFT**   4
- #define **CC4S_SHIFT**   8
- #define **OC4M_SHIFT**   12
- #define **CC1IE_BIT**   1
- #define **CC1P_BIT**   1
- #define **CC1NP_BIT**   3
- #define **CC3P_BIT**   9
- #define **CC3NP_BIT**   11
- #define **CC3EN_BIT**   8
- #define **CC1EN_BIT**   0
- #define **MTMR2_PERIPHERAL**   ((volatile **ST_MTMRx_RegistersMap_t** *)**MTMR2_PERIPHERAL_BASE_ADDR**)
- #define **MTMR3_PERIPHERAL**   ((volatile **ST_MTMRx_RegistersMap_t** *)**MTMR3_PERIPHERAL_BASE_ADDR**)
- #define **MTMR4_PERIPHERAL**   ((volatile **ST_MTMRx_RegistersMap_t** *)**MTMR4_PERIPHERAL_BASE_ADDR**)
- #define **MTMR5_PERIPHERAL**   ((volatile **ST_MTMRx_RegistersMap_t** *)**MTMR5_PERIPHERAL_BASE_ADDR**)

Macro Definition Documentation

#define CC1EN_BIT   0

#define CC1IE_BIT   1

#define CC1NP_BIT   3

#define CC1P_BIT   1

#define CC1S_SHIFT   0

#define CC2S_SHIFT   8

#define CC3EN_BIT   8

#define CC3NP_BIT   11

#define CC3P_BIT   9

#define CC3S_SHIFT   0

#define CC4S_SHIFT   8

#define CEN_BIT   0

#define MTMR2_PERIPHERAL   ((volatile ST_MTMRx_RegistersMap_t *)MTMR2_PERIPHERAL_BASE_ADDR)

#define MTMR2_PERIPHERAL_BASE_ADDR   (0x40000000)

#define MTMR3_PERIPHERAL   ((volatile ST_MTMRx_RegistersMap_t *)MTMR3_PERIPHERAL_BASE_ADDR)

#define MTMR3_PERIPHERAL_BASE_ADDR   (0x40000400)

#define MTMR4_PERIPHERAL   ((volatile ST_MTMRx_RegistersMap_t *)MTMR4_PERIPHERAL_BASE_ADDR)

#define MTMR4_PERIPHERAL_BASE_ADDR   (0x40000800)

#define MTMR5_PERIPHERAL   ((volatile ST_MTMRx_RegistersMap_t *)MTMR5_PERIPHERAL_BASE_ADDR)

#define MTMR5_PERIPHERAL_BASE_ADDR   (0x40000C00)

#define OC1M_SHIFT   4

#define OC2M_SHIFT   12

#define OC3M_SHIFT   4

#define OC4M_SHIFT   12

```
#define OPM_BIT   3
```

tmr_private.h

Go to the documentation of this file.

```
/************************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : Adaptive_Cruise_Control
// File          : tmr_private.h
// Date          : Oct 14, 2023
// GitHub        : https://github.com/sherifkhadr
/************************************************************************/
#ifndef MCAL_TMR_TMR_PRIVATE_H_
#define MCAL_TMR_TMR_PRIVATE_H_


#define  MTMR2_PERIPHERAL_BASE_ADDR     (0x40000000)
#define  MTMR3_PERIPHERAL_BASE_ADDR     (0x40000400)
#define  MTMR4_PERIPHERAL_BASE_ADDR     (0x40000800)
#define  MTMR5_PERIPHERAL_BASE_ADDR     (0x40000C00)



typedef struct
{

    vuint32_t   MTMRx_CR1;
    vuint32_t   MTMRx_CR2;
    vuint32_t   MTMRx_SMCR;
    vuint32_t   MTMRx_DIER;
    vuint32_t   MTMRx_SR;
    vuint32_t   MTMRx_EGR;
    vuint32_t   MTMRx_CCMR1;
    vuint32_t   MTMRx_CCMR2;
    vuint32_t   MTMRx_CCER;
    vuint32_t   MTMRx_CNT;
    vuint32_t   MTMRx_PSC;
    vuint32_t   MTMRx_ARR;
    vuint32_t   MTMRx_RESERVED_1;
    vuint32_t   MTMRx_CCR1;
    vuint32_t   MTMRx_CCR2;
    vuint32_t   MTMRx_CCR3;
    vuint32_t   MTMRx_CCR4;
    vuint32_t   MTMRx_RESERVED_2;
    vuint32_t   MTMRx_DCR;
    vuint32_t   MTMRx_DMAR;
    vuint32_t   MTMRx_OR;

}ST_MTMRx_RegistersMap_t;


#define CEN_BIT             0
#define OPM_BIT             3
#define CC1S_SHIFT          0
#define OC1M_SHIFT          4
#define CC2S_SHIFT          8
#define OC2M_SHIFT          12
#define CC3S_SHIFT          0
#define OC3M_SHIFT          4
#define CC4S_SHIFT          8
#define OC4M_SHIFT          12
#define CC1IE_BIT           1
#define CC1P_BIT            1
#define CC1NP_BIT           3
#define CC3P_BIT            9
#define CC3NP_BIT           11
#define CC3EN_BIT           8
#define CC1EN_BIT           0



#define MTMR2_PERIPHERAL ((volatile ST_MTMRx_RegistersMap_t
*)MTMR2_PERIPHERAL_BASE_ADDR)
#define MTMR3_PERIPHERAL ((volatile ST_MTMRx_RegistersMap_t
*)MTMR3_PERIPHERAL_BASE_ADDR)
#define MTMR4_PERIPHERAL ((volatile ST_MTMRx_RegistersMap_t
*)MTMR4_PERIPHERAL_BASE_ADDR)
```

```
#define MTMR5_PERIPHERAL ((volatile ST_MTMRx_RegistersMap_t
*)MTMR5_PERIPHERAL_BASE_ADDR)




#endif /* MCAL_TMR_TMR_PRIVATE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/usart/usart_config.h File Reference

Data Structures

struct **ST_MUSART_clockInit_t***Structure for USART clock initialization.*

struct **ST_MUSART_cfg_t***Structure for USART configuration.*

Macros

- #define **THRESHOLD_VALUE**   50000
- #define **__PCLK__**   8000000UL
- #define **MUSART_ENABLE**   1
- #define **MUSART_DISABLE**   0

Enumerations

- enum **EN_MUSART_systeamState_t** { **MUSART_NOK** = 0, **MUSART_OK**, **MUSART_PTR_NULL** }
  Enumeration for USART system states.
- enum **EN_MUSART_samplingModeOptions_t** { **MUSART_SAMPLING_BY_16** = 0, **MUSART_SAMPLING_BY_8** }
  Enumeration for USART sampling mode options.
- enum **EN_MUSART_baudRateOptions_t** { **MUSART_BUAD_RATE_1200_bps** = 1200, **MUSART_BUAD_RATE_2400_bps** = 2400, **MUSART_BUAD_RATE_9600_bps** = 9600, **MUSART_BUAD_RATE_19200_bps** = 19200, **MUSART_BUAD_RATE_38400_bps** = 38400, **MUSART_BUAD_RATE_57600_bps** = 57600, **MUSART_BUAD_RATE_115200_bps** = 115200, **MUSART_BUAD_RATE_230400_bps** = 230400, **MUSART_BUAD_RATE_460800_bps** = 460800, **MUSART_BUAD_RATE_921600_bps** = 921600, **MUSART_BUAD_RATE_1792000_bps** = 1792000, **MUSART_BUAD_RATE_1843200_bps** = 1843200, **MUSART_BUAD_RATE_3584000_bps** = 3584000, **MUSART_BUAD_RATE_3686400_bps** = 3686400, **MUSART_BUAD_RATE_7168000_bps** = 7168000, **MUSART_BUAD_RATE_7372800_bps** = 7372800, **MUSART_BUAD_RATE_9000000_bps** = 9000000, **MUSART_BUAD_RATE_10500000_bps** = 10500000 }
  Enumeration for USART baud rate options.
- enum **EN_MUSART_transferControl_t** { **MUSART_TX_ONLY** = 0, **MUSART_RX_ONLY**, **MUSART_TX_RX** }
  Enumeration for USART transfer control options.
- enum **EN_MUSART_stopBitOption_t** { **MUSART_ONE_STOP_BIT** = 0, **MUSART_HALF_STOP_BIT**, **MUSART_TWO_STOP_BIT**, **MUSART_ONE_AND_HALF_BIT** }
  Enumeration for USART stop bit options.
- enum **EN_MUSART_parityControlOption_t** { **MUSART_PARITY_DISABLED** = 0, **MUSART_PARITY_ENABLED** }
  Enumeration for USART parity control options.
- enum **EN_MUSART_paritySelectionOption_t** { **MUSART_EVEN_PARITY** = 0, **MUSART_ODD_PARITY** }
  Enumeration for USART parity selection options.
- enum **EN_MUSART_dataSizeOptions_t** { **MUSART_DATA_SIZE_8_BIT** = 0, **MUSART_DATA_SIZE_9_BIT** }
  Enumeration for USART data size options.

Macro Definition Documentation

#define __PCLK__   8000000UL

#define MUSART_DISABLE   0

#define MUSART_ENABLE   1

#define THRESHOLD_VALUE   50000

Enumeration Type Documentation

enum EN_MUSART_baudRateOptions_t

Enumeration for USART baud rate options.

Enumerator:

MUSART_BUAD_RATE_1200_bps	
MUSART_BUAD_RATE_2400_bps	
MUSART_BUAD_RATE_9600_bps	
MUSART_BUAD_RATE_19200_bps	
MUSART_BUAD_RATE_38400_bps	
MUSART_BUAD_RATE_57600_bps	
MUSART_BUAD_RATE_115200_bps	
MUSART_BUAD_RATE_230400_bps	
MUSART_BUAD_RATE_460800_bps	
MUSART_BUAD_RATE_921600_bps	
MUSART_BUAD_RATE_1792000_bps	
MUSART_BUAD_RATE_1843200_bps	
MUSART_BUAD_RATE_3584000_bps	
MUSART_BUAD	

_RATE_3686400_bps	
MUSART_BUAD_RATE_7168000_bps	
MUSART_BUAD_RATE_7372800_bps	
MUSART_BUAD_RATE_9000000_bps	
MUSART_BUAD_RATE_10500000_bps	

enum EN_MUSART_dataSizeOptions_t

Enumeration for USART data size options.

Enumerator:

MUSART_DATA_SIZE_8_BIT	USART data size: 8 bits.
MUSART_DATA_SIZE_9_BIT	USART data size: 9 bits.

enum EN_MUSART_parityControlOption_t

Enumeration for USART parity control options.

Enumerator:

MUSART_PARITY_DISABLED	USART parity control disabled.
MUSART_PARITY_ENABLED	USART parity control enabled.

enum EN_MUSART_paritySelectionOption_t

Enumeration for USART parity selection options.

Enumerator:

MUSART_EVEN_PARITY	USART even parity.
MUSART_ODD_PARITY	USART odd parity.

enum EN_MUSART_samplingModeOptions_t

Enumeration for USART sampling mode options.

Enumerator:

MUSART_SAMPLING_BY_16	USART sampling by 16.
MUSART_SAMPLING_BY_8	USART sampling by 8.

enum EN_MUSART_stopBitOption_t

Enumeration for USART stop bit options.

Enumerator:

MUSART_ONE_STOP_BIT	USART one stop bit.
MUSART_HALF_STOP_BIT	USART half stop bit.
MUSART_TWO_STOP_BIT	USART two stop bits.
MUSART_ONE_AND_HALF_BIT	USART one and a half stop bits.

enum EN_MUSART_systeamState_t

Enumeration for USART system states.

Enumerator:

MUSART_NOK	USART operation unsuccessful.
MUSART_OK	USART operation successful.
MUSART_PTR_NULL	Null pointer encountered during the operation.

enum EN_MUSART_transferControl_t

Enumeration for USART transfer control options.

Enumerator:

MUSART_TX_ONLY	USART transmit only.
MUSART_RX_ONLY	USART receive only.
MUSART_TX_RX	USART transmit and receive.

usart_config.h

Go to the documentation of this file.

```c
/*******************************************************************************/
// Author         : Sherif Ashraf Khadr
// Project        : STM32F401xC
// File           : usart_config.h
// Date           : Sep 19, 2023
// GitHub         : https://github.com/sherifkhadr
/*******************************************************************************/
#ifndef MCAL_USART_USART_CONFIG_H_
#define MCAL_USART_USART_CONFIG_H_

#define THRESHOLD_VALUE     50000
#define __PCLK__            8000000UL

#define MUSART_ENABLE               1
#define MUSART_DISABLE              0

typedef enum
{
    MUSART_NOK = 0,
    MUSART_OK,
    MUSART_PTR_NULL
} EN_MUSART_systeamState_t;

typedef enum
{
    MUSART_SAMPLING_BY_16 = 0,
    MUSART_SAMPLING_BY_8
} EN_MUSART_samplingModeOptions_t;

typedef enum
{
    MUSART_BUAD_RATE_1200_bps = 1200,
    MUSART_BUAD_RATE_2400_bps = 2400,
    MUSART_BUAD_RATE_9600_bps = 9600,
    MUSART_BUAD_RATE_19200_bps = 19200,
    MUSART_BUAD_RATE_38400_bps = 38400,
    MUSART_BUAD_RATE_57600_bps = 57600,
    MUSART_BUAD_RATE_115200_bps = 115200,
    MUSART_BUAD_RATE_230400_bps = 230400,
    MUSART_BUAD_RATE_460800_bps = 460800,
    MUSART_BUAD_RATE_921600_bps = 921600,
    MUSART_BUAD_RATE_1792000_bps = 1792000,
    MUSART_BUAD_RATE_1843200_bps = 1843200,
    MUSART_BUAD_RATE_3584000_bps = 3584000,
    MUSART_BUAD_RATE_3686400_bps = 3686400,
    MUSART_BUAD_RATE_7168000_bps = 7168000,
    MUSART_BUAD_RATE_7372800_bps = 7372800,
    MUSART_BUAD_RATE_9000000_bps = 9000000,
    MUSART_BUAD_RATE_10500000_bps = 10500000,

}EN_MUSART_baudRateOptions_t;

typedef enum
{
    MUSART_TX_ONLY = 0,
    MUSART_RX_ONLY,
    MUSART_TX_RX
} EN_MUSART_transferControl_t;

typedef enum
{
    MUSART_ONE_STOP_BIT = 0,
    MUSART_HALF_STOP_BIT,
    MUSART_TWO_STOP_BIT,
    MUSART_ONE_AND_HALF_BIT
} EN_MUSART_stopBitOption_t;

typedef enum
{
    MUSART_PARITY_DISABLED = 0,
    MUSART_PARITY_ENABLED
} EN_MUSART_parityControlOption_t;
```

```c

typedef enum
{
    MUSART_EVEN_PARITY = 0,
    MUSART_ODD_PARITY
} EN_MUSART_paritySelectionOption_t;

typedef enum
{
    MUSART_DATA_SIZE_8_BIT = 0,
    MUSART_DATA_SIZE_9_BIT
} EN_MUSART_dataSizeOptions_t;

typedef struct
{
    uint8_t clockOutput;
    uint8_t clockPolarity;
    uint8_t clockPhase;
    uint8_t lastBitClockPulse;
} ST_MUSART_clockInit_t;

typedef struct
{
    EN_MUSART_transferControl_t copy_uddtTransferDirection;
    EN_MUSART_samplingModeOptions_t copy_uddtSamplingModeOption;
    EN_MUSART_baudRateOptions_t copy_uddtBuadRateOption;
    EN_MUSART_dataSizeOptions_t copy_uddtDataSizeOption;
    EN_MUSART_parityControlOption_t copy_uddtParityControl;
    EN_MUSART_paritySelectionOption_t copy_uddtParitySelection;
    EN_MUSART_stopBitOption_t copy_uddtStopBitSelection;
    uint8_t copy_HardwareFlowControl;
    ST_MUSART_clockInit_t copy_uddtUartClockInit;
} ST_MUSART_cfg_t;

#endif /* MCAL_USART_USART_CONFIG_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/usart/usart_interface.h File Reference

Header file for the Universal Synchronous/Asynchronous Receiver Transmitter (MUSART) module.

```
#include "../../COMMON/std_types.h"
#include "../../COMMON/bit_math.h"
#include "../../MCAL/gpio/gpio_interface.h"
#include "usart_private.h"
#include "usart_config.h"
```

Functions

- **EN_MUSART_systeamState_t MUSART_uddtInit** (**ST_MUART_RegistersMap_t** *PS_USARTx, **ST_MUSART_cfg_t** const *PS_uddtUartCfg)
  Initialize the USART module with the given configuration.
- **EN_MUSART_systeamState_t MUSART_uddtEnable** (**ST_MUART_RegistersMap_t** *PS_USARTx)
  Enable the USART module.
- **EN_MUSART_systeamState_t MUSART_uddtDisable** (**ST_MUART_RegistersMap_t** *PS_USARTx)
  Disable the USART module.
- **EN_MUSART_systeamState_t MUSART_uddtTransmitByte** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** copy_u8ByteToSend)
  Transmit a byte through the USART module.
- **EN_MUSART_systeamState_t MUSART_uddtTransmitString** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** *copy_u8StringToSend)
  Transmit a string through the USART module.
- **EN_MUSART_systeamState_t MUSART_uddtReadDataRegister** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** *copy_u8ByteToReceive)
  Read data from the USART data register.
- **EN_MUSART_systeamState_t MUSART_uddtClearFlags** (**ST_MUART_RegistersMap_t** *PS_USARTx)
  Clear the USART flags.
- **EN_MUSART_systeamState_t MUSART_uddtReceiveByteSynchNonBlocking** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** *copy_u8ByteToReceive)
  Receive a byte asynchronously in a non-blocking manner.
- **EN_MUSART_systeamState_t MUSART_uddtReceiveStringAsynchBlocking** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** *copy_u8ByteToReceive)
  Receive a string asynchronously in a blocking manner.

- **EN_MUSART_systeamState_t MUSART_uddtReceiveStringSynchNonBlocking** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** *copy_u8ByteToReceive)
  Receive a string asynchronously in a non-blocking manner.

- **EN_MUSART_systeamState_t MUSART_RxIntSetStatus** (**ST_MUART_RegistersMap_t** *PS_USARTx, **uint8_t** copy_u8Status)
  Set the receive interrupt status for the USART module.

- **EN_MUSART_systeamState_t MUSART1_uddtSetCallBack** (void(*ptr)(void))
  Set the callback function for USART1.

- **EN_MUSART_systeamState_t MUSART2_uddtSetCallBack** (void(*ptr)(void))
  Set the callback function for USART2.

- **EN_MUSART_systeamState_t MUSART6_uddtSetCallBack** (void(*ptr)(void))
  Set the callback function for USART6.

Detailed Description

Header file for the Universal Synchronous/Asynchronous Receiver Transmitter (MUSART) module.

Function Documentation

EN_MUSART_systeamState_t MUSART1_uddtSetCallBack (void(*)(void) *ptr*)

Set the callback function for USART1.

This function sets the callback function for USART1.

Parameters

ptr	Pointer to the callback function.

Returns

The system state after setting the callback function.

- #EN_MUSART_OK: Callback setting successful.
- #EN_MUSART_NOK: Callback setting failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART2_uddtSetCallBack (void(*)(void) *ptr*)

Set the callback function for USART2.

This function sets the callback function for USART2.

Parameters

ptr	Pointer to the callback function.

Returns

The system state after setting the callback function.

- #EN_MUSART_OK: Callback setting successful.
- #EN_MUSART_NOK: Callback setting failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART6_uddtSetCallBack (void(*)(void) *ptr*)

Set the callback function for USART6.

This function sets the callback function for USART6.

Parameters

ptr	Pointer to the callback function.

Returns

The system state after setting the callback function.

- #EN_MUSART_OK: Callback setting successful.
- #EN_MUSART_NOK: Callback setting failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART_RxIntSetStatus (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t *copy_u8Status*)

Set the receive interrupt status for the USART module.

This function sets the receive interrupt status for the USART module.

Parameters

PS_USARTx	Pointer to the USARTx registers map.
copy_u8Status	The status to set.

Returns

The system state after setting the receive interrupt status.

- #EN_MUSART_OK: Status setting successful.
- #EN_MUSART_NOK: Status setting failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART_uddtClearFlags (ST_MUART_RegistersMap_t * *PS_USARTx*)

Clear the USART flags.

This function clears the USART flags.

Parameters

PS_USARTx	Pointer to the USARTx registers map.

Returns

The system state after clearing the USART flags.

- #EN_MUSART_OK: Flag clearing successful.
- #EN_MUSART_NOK: Flag clearing failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART_uddtDisable (ST_MUART_RegistersMap_t * *PS_USARTx*)

Disable the USART module.

This function disables the USART module.

Parameters

PS_USARTx	Pointer to the USARTx registers map.

Returns

The system state after disabling the USART module.

- #EN_MUSART_OK: USART disabling successful.
- #EN_MUSART_NOK: USART disabling failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART_uddtEnable (ST_MUART_RegistersMap_t * *PS_USARTx*)

Enable the USART module.

This function enables the USART module.

Parameters

PS_USARTx	Pointer to the USARTx registers map.

Returns

The system state after enabling the USART module.

- #EN_MUSART_OK: USART enabling successful.
- #EN_MUSART_NOK: USART enabling failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART_uddtInit (ST_MUART_RegistersMap_t * *PS_USARTx*, ST_MUSART_cfg_t const * *PS_uddtUartCfg*)

Initialize the USART module with the given configuration.

This function initializes the USART module with the provided configuration.

Parameters

PS_USARTx	Pointer to the USARTx registers map.
PS_uddtUartCfg	Pointer to the USART configuration structure.

Returns

The system state after initializing the USART module.

- #EN_MUSART_OK: Initialization successful.
- #EN_MUSART_NOK: Initialization failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART_uddtReadDataRegister (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t * *copy_u8ByteToReceive*)

Read data from the USART data register.

This function reads data from the USART data register.

Parameters

PS_USARTx	Pointer to the USARTx registers map.
copy_u8ByteToReceive	Pointer to store the received byte.

Returns

The system state after reading the data register.

- #EN_MUSART_OK: Data read successful.
- #EN_MUSART_NOK: Data read failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART_uddtReceiveByteSynchNonBlocking (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t * *copy_u8ByteToReceive*)

Receive a byte asynchronously in a non-blocking manner.

This function receives a byte asynchronously in a non-blocking manner.

Parameters

PS_USARTx	Pointer to the USARTx registers map.
copy_u8ByteToReceive	Pointer to store the received byte.

Returns

The system state after receiving the byte.

- #EN_MUSART_OK: Byte reception successful.
- #EN_MUSART_NOK: Byte reception failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART_uddtReceiveStringAsynchBlocking (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t * *copy_u8ByteToReceive*)

Receive a string asynchronously in a blocking manner.

This function receives a string asynchronously in a blocking manner.

Parameters

PS_USARTx	Pointer to the USARTx registers map.
copy_u8ByteToReceive	Pointer to store the received string.

Returns

The system state after receiving the string.

- #EN_MUSART_OK: String reception successful.
- #EN_MUSART_NOK: String reception failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART_uddtReceiveStringSynchNonBlocking (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t * *copy_u8ByteToReceive*)

Receive a string asynchronously in a non-blocking manner.

This function receives a string asynchronously in a non-blocking manner.

Parameters

PS_USARTx	Pointer to the USARTx registers map.
copy_u8ByteToReceive	Pointer to store the received string.

Returns

The system state after receiving the string.

- #EN_MUSART_OK: String reception successful.
- #EN_MUSART_NOK: String reception failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART_uddtTransmitByte (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t *copy_u8ByteToSend*)

Transmit a byte through the USART module.

This function transmits a byte through the USART module.

Parameters

PS_USARTx	Pointer to the USARTx registers map.
copy_u8ByteToSend	The byte to transmit.

Returns

The system state after transmitting the byte.

- #EN_MUSART_OK: Byte transmission successful.
- #EN_MUSART_NOK: Byte transmission failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

EN_MUSART_systeamState_t MUSART_uddtTransmitString (ST_MUART_RegistersMap_t * *PS_USARTx*, uint8_t * *copy_u8StringToSend*)

Transmit a string through the USART module.

This function transmits a string through the USART module.

Parameters

PS_USARTx	Pointer to the USARTx registers map.
copy_u8StringToSend	Pointer to the string to transmit.

Returns

The system state after transmitting the string.

- #EN_MUSART_OK: String transmission successful.
- #EN_MUSART_NOK: String transmission failed.
- #EN_MUSART_PTR_NULL: Null pointer encountered during the operation.

usart_interface.h

Go to the documentation of this file.

```c

#ifndef MCAL_USART_USART_INTERFACE_H_
#define MCAL_USART_USART_INTERFACE_H_

#include "../../COMMON/std_types.h"
#include "../../COMMON/bit_math.h"
#include "../../MCAL/gpio/gpio_interface.h"
#include "usart_private.h"
#include "usart_config.h"


EN_MUSART_systeamState_t MUSART_uddtInit(ST_MUART_RegistersMap_t *PS_USARTx, ST_MUSART_cfg_t const *PS_uddtUartCfg);

EN_MUSART_systeamState_t MUSART_uddtEnable(ST_MUART_RegistersMap_t *PS_USARTx);

EN_MUSART_systeamState_t MUSART_uddtDisable(ST_MUART_RegistersMap_t *PS_USARTx);

EN_MUSART_systeamState_t MUSART_uddtTransmitByte(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t copy_u8ByteToSend);

EN_MUSART_systeamState_t MUSART_uddtTransmitString(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t *copy_u8StringToSend);

EN_MUSART_systeamState_t MUSART_uddtReadDataRegister(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t *copy_u8ByteToReceive);

EN_MUSART_systeamState_t MUSART_uddtClearFlags(ST_MUART_RegistersMap_t *PS_USARTx);

EN_MUSART_systeamState_t MUSART_uddtReceiveByteSynchNonBlocking(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t *copy_u8ByteToReceive);

EN_MUSART_systeamState_t MUSART_uddtReceiveStringAsynchBlocking(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t *copy_u8ByteToReceive);

EN_MUSART_systeamState_t MUSART_uddtReceiveStringSynchNonBlocking(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t *copy_u8ByteToReceive);

EN_MUSART_systeamState_t MUSART_RxIntSetStatus(ST_MUART_RegistersMap_t *PS_USARTx, uint8_t copy_u8Status);

EN_MUSART_systeamState_t MUSART1_uddtSetCallBack(void (*ptr)(void));

EN_MUSART_systeamState_t MUSART2_uddtSetCallBack(void (*ptr)(void));

EN_MUSART_systeamState_t MUSART6_uddtSetCallBack(void (*ptr)(void));


#endif /* MCAL_USART_USART_INTERFACE_H_ */
```

D:/Programing/Embedded System Diploma/ITI/grad doc/Adaptive_Cruise_Control/Inc/MCAL/usart/usart_private.h File Reference

Data Structures

struct ST_MUART_RegistersMap_t**Macros**

- #define **MUART1_PERIPHERAL_BASE_ADDR**   (0x40011000)
- #define **MUART2_PERIPHERAL_BASE_ADDR**   (0x40004400)
- #define **MUART6_PERIPHERAL_BASE_ADDR**   (0x40011400)
- #define **MUART1_PERIPHERAL**   ((**ST_MUART_RegistersMap_t** *)**MUART1_PERIPHERAL_BASE_ADDR**)
- #define **MUART2_PERIPHERAL**   ((**ST_MUART_RegistersMap_t** *)**MUART2_PERIPHERAL_BASE_ADDR**)
- #define **MUART6_PERIPHERAL**   ((**ST_MUART_RegistersMap_t** *)**MUART6_PERIPHERAL_BASE_ADDR**)
- #define **UART_DIV_SAMPLING16**(_PCLK_, _BAUD_)   ((**uint32_t**)((((**float64_t**)(_PCLK_))*25U)/(4U*((**float64_t**)(_BAUD_)))))
- #define **UART_DIVMANT_SAMPLING16**(_PCLK_, _BAUD_)   (**UART_DIV_SAMPLING16**((_PCLK_), (_BAUD_))/100U)
- #define **UART_DIVFRAQ_SAMPLING16**(_PCLK_, _BAUD_)   ((((**UART_DIV_SAMPLING16**((_PCLK_), (_BAUD_)) - (**UART_DIVMANT_SAMPLING16**((_PCLK_), (_BAUD_)) * 100U)) * 16U) + 50U) / 100U)
- #define **UART_BRR_SAMPLING16**(_PCLK_,   _BAUD_)
- #define **UART_DIV_SAMPLING8**(_PCLK_, _BAUD_)   ((**uint32_t**)((((**float64_t**)(_PCLK_))*25U)/(2U*((**float64_t**)(_BAUD_)))))
- #define **UART_DIVMANT_SAMPLING8**(_PCLK_, _BAUD_)   (**UART_DIV_SAMPLING8**((_PCLK_), (_BAUD_))/100U)
- #define **UART_DIVFRAQ_SAMPLING8**(_PCLK_, _BAUD_)   ((((**UART_DIV_SAMPLING8**((_PCLK_), (_BAUD_)) - (**UART_DIVMANT_SAMPLING8**((_PCLK_), (_BAUD_)) * 100U)) * 8U) + 50U) / 100U)
- #define **UART_BRR_SAMPLING8**(_PCLK_,   _BAUD_)
- #define **MUSART_SR_PE_BIT**   0
- #define **MUSART_SR_FE_BIT**   1
- #define **MUSART_SR_NE_BIT**   2
- #define **MUSART_SR_ORE_BIT**   3
- #define **MUSART_SR_IDLE_BIT**   4
- #define **MUSART_SR_RXNE_BIT**   5
- #define **MUSART_SR_TC_BIT**   6
- #define **MUSART_SR_TXE_BIT**   7
- #define **MUSART_SR_LBD_BIT**   8
- #define **MUSART_SR_CTS_BIT**   9
- #define **MUSART_CR1_SBK_BIT**   0
- #define **MUSART_CR1_RWU_BIT**   1
- #define **MUSART_CR1_RE_BIT**   2
- #define **MUSART_CR1_TE_BIT**   3
- #define **MUSART_CR1_IDLEIE_BIT**   4
- #define **MUSART_CR1_RXNEIE_BIT**   5
- #define **MUSART_CR1_TCIE_BIT**   6
- #define **MUSART_CR1_TXEIE_BIT**   7
- #define **MUSART_CR1_PEIE_BIT**   8
- #define **MUSART_CR1_PS_BIT**   9
- #define **MUSART_CR1_PCE_BIT**   10
- #define **MUSART_CR1_WAKE_BIT**   11

- #define **MUSART_CR1_M_BIT** 12
- #define **MUSART_CR1_UE_BIT** 13
- #define **MUSART_CR1_OVER8_BIT** 15
- #define **MUSART_CR2_ADD0_BIT** 0
- #define **MUSART_CR2_ADD1_BIT** 1
- #define **MUSART_CR2_ADD2_BIT** 2
- #define **MUSART_CR2_ADD3_BIT** 3
- #define **MUSART_CR2_LBDL_BIT** 5
- #define **MUSART_CR2_LBDIE_BIT** 6
- #define **MUSART_CR2_LBCL_BIT** 8
- #define **MUSART_CR2_CPHA_BIT** 9
- #define **MUSART_CR2_CPOL_BIT** 10
- #define **MUSART_CR2_CLKEN_BIT** 11
- #define **MUSART_CR2_STOP_BIT** 12
- #define **MUSART_CR2_STOP0_BIT** 12
- #define **MUSART_CR2_STOP1_BIT** 13
- #define **MUSART_CR2_LINEN_BIT** 14
- #define **MUSART_CR3_CTSIE_BIT** 10
- #define **MUSART_CR3_CTSE_BIT** 9
- #define **MUSART_CR3_RTSE_BIT** 8
- #define **MUSART_CR3_DMAT_BIT** 7
- #define **MUSART_CR3_DMAR_BIT** 6
- #define **MUSART_CR3_SCEN_BIT** 5
- #define **MUSART_CR3_NACK_BIT** 4
- #define **MUSART_CR3_HDSEL_BIT** 3
- #define **MUSART_CR3_IRLP_BIT** 2
- #define **MUSART_CR3_IREN_BIT** 1
- #define **MUSART_CR3_EIE_BIT** 0

Macro Definition Documentation

#define MUART1_PERIPHERAL ((ST_MUART_RegistersMap_t *)MUART1_PERIPHERAL_BASE_ADDR)

#define MUART1_PERIPHERAL_BASE_ADDR (0x40011000)

#define MUART2_PERIPHERAL ((ST_MUART_RegistersMap_t *)MUART2_PERIPHERAL_BASE_ADDR)

#define MUART2_PERIPHERAL_BASE_ADDR (0x40004400)

#define MUART6_PERIPHERAL ((ST_MUART_RegistersMap_t *)MUART6_PERIPHERAL_BASE_ADDR)

#define MUART6_PERIPHERAL_BASE_ADDR (0x40011400)

#define MUSART_CR1_IDLEIE_BIT 4

#define MUSART_CR1_M_BIT 12

#define MUSART_CR1_OVER8_BIT 15

#define MUSART_CR1_PCE_BIT 10

#define MUSART_CR1_PEIE_BIT 8

#define MUSART_CR1_PS_BIT 9

#define MUSART_CR1_RE_BIT 2

#define MUSART_CR1_RWU_BIT 1

#define MUSART_CR1_RXNEIE_BIT 5

#define MUSART_CR1_SBK_BIT 0

#define MUSART_CR1_TCIE_BIT 6

#define MUSART_CR1_TE_BIT 3

#define MUSART_CR1_TXEIE_BIT 7

#define MUSART_CR1_UE_BIT 13

#define MUSART_CR1_WAKE_BIT 11

#define MUSART_CR2_ADD0_BIT 0

#define MUSART_CR2_ADD1_BIT 1

#define MUSART_CR2_ADD2_BIT 2

```
#define MUSART_CR2_ADD3_BIT   3

#define MUSART_CR2_CLKEN_BIT   11

#define MUSART_CR2_CPHA_BIT   9

#define MUSART_CR2_CPOL_BIT   10

#define MUSART_CR2_LBCL_BIT   8

#define MUSART_CR2_LBDIE_BIT   6

#define MUSART_CR2_LBDL_BIT   5

#define MUSART_CR2_LINEN_BIT   14

#define MUSART_CR2_STOP0_BIT   12

#define MUSART_CR2_STOP1_BIT   13

#define MUSART_CR2_STOP_BIT   12

#define MUSART_CR3_CTSE_BIT   9

#define MUSART_CR3_CTSIE_BIT   10

#define MUSART_CR3_DMAR_BIT   6

#define MUSART_CR3_DMAT_BIT   7

#define MUSART_CR3_EIE_BIT   0

#define MUSART_CR3_HDSEL_BIT   3

#define MUSART_CR3_IREN_BIT   1

#define MUSART_CR3_IRLP_BIT   2

#define MUSART_CR3_NACK_BIT   4

#define MUSART_CR3_RTSE_BIT   8

#define MUSART_CR3_SCEN_BIT   5

#define MUSART_SR_CTS_BIT   9

#define MUSART_SR_FE_BIT   1

#define MUSART_SR_IDLE_BIT   4

#define MUSART_SR_LBD_BIT   8
```

#define MUSART_SR_NE_BIT   2

#define MUSART_SR_ORE_BIT   3

#define MUSART_SR_PE_BIT   0

#define MUSART_SR_RXNE_BIT   5

#define MUSART_SR_TC_BIT   6

#define MUSART_SR_TXE_BIT   7

#define UART_BRR_SAMPLING16(_PCLK_,   _BAUD_)

```
Value:
((UART_DIVMANT_SAMPLING16((_PCLK_), (_BAUD_)) << 4U) + \

(UART_DIVFRAQ_SAMPLING16((_PCLK_), (_BAUD_)) & 0xF0U) + \

(UART_DIVFRAQ_SAMPLING16((_PCLK_), (_BAUD_)) & 0x0FU))
```

#define UART_BRR_SAMPLING8(_PCLK_,   _BAUD_)

```
Value:
((UART_DIVMANT_SAMPLING8((_PCLK_), (_BAUD_)) << 4U) + \

((UART_DIVFRAQ_SAMPLING8((_PCLK_), (_BAUD_)) & 0xF8U) << 1U) + \

(UART_DIVFRAQ_SAMPLING8((_PCLK_), (_BAUD_)) & 0x07U))
```

#define UART_DIV_SAMPLING16(_PCLK_, _BAUD_)   ((uint32_t)((((float64_t)(_PCLK_))*25U)/(4U*((float64_t)(_BAUD_)))))

#define UART_DIV_SAMPLING8(_PCLK_, _BAUD_)   ((uint32_t)((((float64_t)(_PCLK_))*25U)/(2U*((float64_t)(_BAUD_)))))

#define UART_DIVFRAQ_SAMPLING16(_PCLK_, _BAUD_)   ((((UART_DIV_SAMPLING16((_PCLK_), (_BAUD_)) - (UART_DIVMANT_SAMPLING16((_PCLK_), (_BAUD_)) * 100U)) * 16U) + 50U) / 100U)

#define UART_DIVFRAQ_SAMPLING8(_PCLK_, _BAUD_)   ((((UART_DIV_SAMPLING8((_PCLK_), (_BAUD_)) - (UART_DIVMANT_SAMPLING8((_PCLK_), (_BAUD_)) * 100U)) * 8U) + 50U) / 100U)

#define UART_DIVMANT_SAMPLING16(_PCLK_, _BAUD_)   (UART_DIV_SAMPLING16((_PCLK_), (_BAUD_))/100U)

#define UART_DIVMANT_SAMPLING8(_PCLK_, _BAUD_)   (UART_DIV_SAMPLING8((_PCLK_), (_BAUD_))/100U)

usart_private.h

Go to the documentation of this file.

```c
/*************************************************************************/
// Author        : Sherif Ashraf Khadr
// Project       : STM32F401xC
// File          : uart_private.h
// Date          : Sep 19, 2023
// GitHub        : https://github.com/sherifkhadr
/*************************************************************************/
#ifndef MCAL_USART_USART_PRIVATE_H_
#define MCAL_USART_USART_PRIVATE_H_

#define  MUART1_PERIPHERAL_BASE_ADDR            (0x40011000)
#define  MUART2_PERIPHERAL_BASE_ADDR            (0x40004400)
#define  MUART6_PERIPHERAL_BASE_ADDR            (0x40011400)



typedef struct
{

    vuint32_t MUSART_SR;
    vuint32_t MUSART_DR;
    vuint32_t MUSART_BRR;
    vuint32_t MUSART_CR1;
    vuint32_t MUSART_CR2;
    vuint32_t MUSART_CR3;
    vuint32_t MUSART_GTPR;

}ST_MUART_RegistersMap_t;


#define MUART1_PERIPHERAL   ((ST_MUART_RegistersMap_t *)MUART1_PERIPHERAL_BASE_ADDR)
#define MUART2_PERIPHERAL   ((ST_MUART_RegistersMap_t *)MUART2_PERIPHERAL_BASE_ADDR)
#define MUART6_PERIPHERAL   ((ST_MUART_RegistersMap_t *)MUART6_PERIPHERAL_BASE_ADDR)



#define UART_DIV_SAMPLING16(_PCLK_, _BAUD_) ((uint32_t)((((float64_t)(_PCLK_))*25U)/(4U*((float64_t)(_BAUD_)))))
#define UART_DIVMANT_SAMPLING16(_PCLK_, _BAUD_) (UART_DIV_SAMPLING16((_PCLK_), (_BAUD_))/100U)
#define UART_DIVFRAQ_SAMPLING16(_PCLK_, _BAUD_) ((((UART_DIV_SAMPLING16((_PCLK_), (_BAUD_)) - (UART_DIVMANT_SAMPLING16((_PCLK_), (_BAUD_)) * 100U)) * 16U) + 50U) / 100U)
/* UART_BRR = mantissa + overflow + fraction
            = (UART_DIVMANT << 4) + (UART_DIVFRAQ & 0xF0) + (UART_DIVFRAQ & 0x0FU) */
#define UART_BRR_SAMPLING16(_PCLK_, _BAUD_) ((UART_DIVMANT_SAMPLING16((_PCLK_), (_BAUD_)) << 4U) + \
(UART_DIVFRAQ_SAMPLING16((_PCLK_), (_BAUD_)) & 0xF0U) + \
(UART_DIVFRAQ_SAMPLING16((_PCLK_), (_BAUD_)) & 0x0FU))

#define UART_DIV_SAMPLING8(_PCLK_, _BAUD_) ((uint32_t)((((float64_t)(_PCLK_))*25U)/(2U*((float64_t)(_BAUD_)))))
#define UART_DIVMANT_SAMPLING8(_PCLK_, _BAUD_)          (UART_DIV_SAMPLING8((_PCLK_), (_BAUD_))/100U)
#define UART_DIVFRAQ_SAMPLING8(_PCLK_, _BAUD_) ((((UART_DIV_SAMPLING8((_PCLK_), (_BAUD_)) - (UART_DIVMANT_SAMPLING8((_PCLK_), (_BAUD_)) * 100U)) * 8U) + 50U) / 100U)
/* UART_BRR = mantissa + overflow + fraction
            = (UART_DIVMANT << 4) + ((UART_DIVFRAQ & 0xF8) << 1) + (UART_DIVFRAQ & 0x07U) */
#define UART_BRR_SAMPLING8(_PCLK_, _BAUD_) ((UART_DIVMANT_SAMPLING8((_PCLK_), (_BAUD_)) << 4U) + \
((UART_DIVFRAQ_SAMPLING8((_PCLK_), (_BAUD_)) & 0xF8U) << 1U) + \
(UART_DIVFRAQ_SAMPLING8((_PCLK_), (_BAUD_)) & 0x07U))



/* Registers Bits  */
```

```

/*********************************************/
/*               SR BITS Mapping              */
/*********************************************/
/*  Parity error                        */
#define MUSART_SR_PE_BIT    0
/*  Framing error                       */
#define MUSART_SR_FE_BIT    1
/*  Noise error flag                    */
#define MUSART_SR_NE_BIT    2
/*  Overrun error                       */
#define MUSART_SR_ORE_BIT   3
/*  IDLE line detected                  */
#define MUSART_SR_IDLE_BIT  4
/*  Read data register not empty        */
#define MUSART_SR_RXNE_BIT  5
/*  Transmission complete               */
#define MUSART_SR_TC_BIT    6
/*  Transmit data register empty        */
#define MUSART_SR_TXE_BIT   7
/*  LIN break detection flag            */
#define MUSART_SR_LBD_BIT   8
/*  CTS flag                            */
#define MUSART_SR_CTS_BIT   9



/*********************************************/
/*               CR1 BITS Mapping             */
/*********************************************/
/*  Send break  bit */
#define MUSART_CR1_SBK_BIT  0
/*  Recevier Wakeup bit */
#define MUSART_CR1_RWU_BIT  1
/*  Recevier Enable bit */
#define MUSART_CR1_RE_BIT   2
/*  Transmitter Enable bit  */
#define MUSART_CR1_TE_BIT   3
/*  IDLE interrupt enable bit   */
#define MUSART_CR1_IDLEIE_BIT   4
/*  RXNEIE interrupt enable bit */
#define MUSART_CR1_RXNEIE_BIT   5
/*  Transmission complete interrupt enable bit  */
#define MUSART_CR1_TCIE_BIT     6
/*  TXE interrupt enable bit    */
#define MUSART_CR1_TXEIE_BIT    7
/*  PE interrupt enable bit */
#define MUSART_CR1_PEIE_BIT     8
/*  Parity selection bit    */
#define MUSART_CR1_PS_BIT       9
/*  Parity control enable bit   */
#define MUSART_CR1_PCE_BIT      10
/*  Wakeup method bit   */
#define MUSART_CR1_WAKE_BIT     11
/*  Word length bit */
#define MUSART_CR1_M_BIT        12
/*  USART enable bit    */
#define MUSART_CR1_UE_BIT       13
/*  USART Oversampling bit  */
#define MUSART_CR1_OVER8_BIT    15

/*********************************************/
/*               CR2 BITS Mapping             */
/*********************************************/
/*  Address of the USART node bits  */
#define MUSART_CR2_ADD0_BIT     0
#define MUSART_CR2_ADD1_BIT     1
#define MUSART_CR2_ADD2_BIT     2
#define MUSART_CR2_ADD3_BIT     3
/*  lin break detection length bit  */
#define MUSART_CR2_LBDL_BIT 5
/*  LIN break detection interrupt enable bit    */
#define MUSART_CR2_LBDIE_BIT        6
/*  Last bit clock pulse bit    */
#define MUSART_CR2_LBCL_BIT     8
/*  Clock phase bit */
#define MUSART_CR2_CPHA_BIT     9
```

```c
/*  Clock polarity bit  */
#define MUSART_CR2_CPOL_BIT     10
/*  Clock enable bit    */
#define MUSART_CR2_CLKEN_BIT    11
/*  STOP bit start */
#define MUSART_CR2_STOP_BIT     12
/*  STOP bits   */
#define MUSART_CR2_STOP0_BIT    12
#define MUSART_CR2_STOP1_BIT    13
/*  LIN mode enable bit */
#define MUSART_CR2_LINEN_BIT    14


/***********************************************/
/*               CR3 BITS Mapping              */
/***********************************************/
/*  CTS interrupt enable bit    */
#define MUSART_CR3_CTSIE_BIT    10
/*  CTS enable bit  */
#define MUSART_CR3_CTSE_BIT     9
/*  RTS enable bit  */
#define MUSART_CR3_RTSE_BIT     8
/*  DMA enable transmitter bit  */
#define MUSART_CR3_DMAT_BIT     7
/*  DMA enable receiver bit */
#define MUSART_CR3_DMAR_BIT     6
/*  Smartcard mode enable bit   */
#define MUSART_CR3_SCEN_BIT     5
/*  Smartcard NACK enable bit   */
#define MUSART_CR3_NACK_BIT     4
/*  Half-duplex selection bit   */
#define MUSART_CR3_HDSEL_BIT    3
/*  IrDA low-power bit  */
#define MUSART_CR3_IRLP_BIT     2
/*  IrDA mode enable bit    */
#define MUSART_CR3_IREN_BIT     1
/*  Error interrupt enable bit  */
#define MUSART_CR3_EIE_BIT      0

#endif /* MCAL_USART_USART_PRIVATE_H_ */
```

Index

INDEX