9 HAL CAN Generic Driver

9.1 CAN Firmware driver registers structures

9.1.1 CAN_InitTypeDef

CAN_InitTypeDef is defined in the stm32f4xx_hal_can.h

Data Fields

- ***uint32_t Prescaler***
- ***uint32_t Mode***
- ***uint32_t SyncJumpWidth***
- ***uint32_t TimeSeg1***
- ***uint32_t TimeSeg2***
- ***FunctionalState TimeTriggeredMode***
- ***FunctionalState AutoBusOff***
- ***FunctionalState AutoWakeUp***
- ***FunctionalState AutoRetransmission***
- ***FunctionalState ReceiveFifoLocked***
- ***FunctionalState TransmitFifoPriority***

Field Documentation

- ***uint32_t CAN_InitTypeDef::Prescaler***

  Specifies the length of a time quantum. This parameter must be a number between Min_Data = 1 and Max_Data = 1024.
- ***uint32_t CAN_InitTypeDef::Mode***

  Specifies the CAN operating mode. This parameter can be a value of ***CAN_operating_mode***
- ***uint32_t CAN_InitTypeDef::SyncJumpWidth***

  Specifies the maximum number of time quanta the CAN hardware is allowed to lengthen or shorten a bit to perform resynchronization. This parameter can be a value of ***CAN_synchronisation_jump_width***
- ***uint32_t CAN_InitTypeDef::TimeSeg1***

  Specifies the number of time quanta in Bit Segment 1. This parameter can be a value of ***CAN_time_quantum_in_bit_segment_1***
- ***uint32_t CAN_InitTypeDef::TimeSeg2***

  Specifies the number of time quanta in Bit Segment 2. This parameter can be a value of ***CAN_time_quantum_in_bit_segment_2***
- ***FunctionalState CAN_InitTypeDef::TimeTriggeredMode***

  Enable or disable the time triggered communication mode. This parameter can be set to ENABLE or DISABLE.
- ***FunctionalState CAN_InitTypeDef::AutoBusOff***

  Enable or disable the automatic bus-off management. This parameter can be set to ENABLE or DISABLE.
- ***FunctionalState CAN_InitTypeDef::AutoWakeUp***

  Enable or disable the automatic wake-up mode. This parameter can be set to ENABLE or DISABLE.
- ***FunctionalState CAN_InitTypeDef::AutoRetransmission***

  Enable or disable the non-automatic retransmission mode. This parameter can be set to ENABLE or DISABLE.
- ***FunctionalState CAN_InitTypeDef::ReceiveFifoLocked***

  Enable or disable the Receive FIFO Locked mode. This parameter can be set to ENABLE or DISABLE.
- ***FunctionalState CAN_InitTypeDef::TransmitFifoPriority***

  Enable or disable the transmit FIFO priority. This parameter can be set to ENABLE or DISABLE.

9.1.2 CAN_FilterTypeDef

CAN_FilterTypeDef is defined in the stm32f4xx_hal_can.h

Data Fields

- ***uint32_t FilterIdHigh***
- ***uint32_t FilterIdLow***
- ***uint32_t FilterMaskIdHigh***
- ***uint32_t FilterMaskIdLow***
- ***uint32_t FilterFIFOAssignment***
- ***uint32_t FilterBank***
- ***uint32_t FilterMode***
- ***uint32_t FilterScale***
- ***uint32_t FilterActivation***
- ***uint32_t SlaveStartFilterBank***

Field Documentation

- ***uint32_t CAN_FilterTypeDef::FilterIdHigh***

  Specifies the filter identification number (MSBs for a 32-bit configuration, first one for a 16-bit configuration). This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF.
- ***uint32_t CAN_FilterTypeDef::FilterIdLow***

  Specifies the filter identification number (LSBs for a 32-bit configuration, second one for a 16-bit configuration). This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF.
- ***uint32_t CAN_FilterTypeDef::FilterMaskIdHigh***

  Specifies the filter mask number or identification number, according to the mode (MSBs for a 32-bit configuration, first one for a 16-bit configuration). This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF.
- ***uint32_t CAN_FilterTypeDef::FilterMaskIdLow***

  Specifies the filter mask number or identification number, according to the mode (LSBs for a 32-bit configuration, second one for a 16-bit configuration). This parameter must be a number between Min_Data = 0x0000 and Max_Data = 0xFFFF.
- ***uint32_t CAN_FilterTypeDef::FilterFIFOAssignment***

  Specifies the FIFO (0 or 1U) which will be assigned to the filter. This parameter can be a value of ***CAN_filter_FIFO***
- ***uint32_t CAN_FilterTypeDef::FilterBank***

  Specifies the filter bank which will be initialized. For single CAN instance(14 dedicated filter banks), this parameter must be a number between Min_Data = 0 and Max_Data = 13. For dual CAN instances(28 filter banks shared), this parameter must be a number between Min_Data = 0 and Max_Data = 27.
- ***uint32_t CAN_FilterTypeDef::FilterMode***

  Specifies the filter mode to be initialized. This parameter can be a value of ***CAN_filter_mode***
- ***uint32_t CAN_FilterTypeDef::FilterScale***

  Specifies the filter scale. This parameter can be a value of ***CAN_filter_scale***
- ***uint32_t CAN_FilterTypeDef::FilterActivation***

  Enable or disable the filter. This parameter can be a value of ***CAN_filter_activation***
- ***uint32_t CAN_FilterTypeDef::SlaveStartFilterBank***

  Select the start filter bank for the slave CAN instance. For single CAN instances, this parameter is meaningless. For dual CAN instances, all filter banks with lower index are assigned to master CAN instance, whereas all filter banks with greater index are assigned to slave CAN instance. This parameter must be a number between Min_Data = 0 and Max_Data = 27.

9.1.3 CAN_TxHeaderTypeDef

CAN_TxHeaderTypeDef is defined in the stm32f4xx_hal_can.h

Data Fields

- ***uint32_t StdId***
- ***uint32_t ExtId***

- ***uint32_t IDE***
- ***uint32_t RTR***
- ***uint32_t DLC***
- ***FunctionalState TransmitGlobalTime***

Field Documentation

- ***uint32_t CAN_TxHeaderTypeDef::StdId***

  Specifies the standard identifier. This parameter must be a number between Min_Data = 0 and Max_Data = 0x7FF.
- ***uint32_t CAN_TxHeaderTypeDef::ExtId***

  Specifies the extended identifier. This parameter must be a number between Min_Data = 0 and Max_Data = 0x1FFFFFFF.
- ***uint32_t CAN_TxHeaderTypeDef::IDE***

  Specifies the type of identifier for the message that will be transmitted. This parameter can be a value of ***CAN_identifier_type***
- ***uint32_t CAN_TxHeaderTypeDef::RTR***

  Specifies the type of frame for the message that will be transmitted. This parameter can be a value of ***CAN_remote_transmission_request***
- ***uint32_t CAN_TxHeaderTypeDef::DLC***

  Specifies the length of the frame that will be transmitted. This parameter must be a number between Min_Data = 0 and Max_Data = 8.
- ***FunctionalState CAN_TxHeaderTypeDef::TransmitGlobalTime***

  Specifies whether the timestamp counter value captured on start of frame transmission, is sent in DATA6 and DATA7 replacing pData[6] and pData[7].

  Note:

  – : Time Triggered Communication Mode must be enabled.

  – : DLC must be programmed as 8 bytes, in order these 2 bytes are sent. This parameter can be set to ENABLE or DISABLE.

9.1.4 CAN_RxHeaderTypeDef

CAN_RxHeaderTypeDef is defined in the stm32f4xx_hal_can.h

Data Fields

- ***uint32_t StdId***
- ***uint32_t ExtId***
- ***uint32_t IDE***
- ***uint32_t RTR***
- ***uint32_t DLC***
- ***uint32_t Timestamp***
- ***uint32_t FilterMatchIndex***

Field Documentation

- ***uint32_t CAN_RxHeaderTypeDef::StdId***

  Specifies the standard identifier. This parameter must be a number between Min_Data = 0 and Max_Data = 0x7FF.
- ***uint32_t CAN_RxHeaderTypeDef::ExtId***

  Specifies the extended identifier. This parameter must be a number between Min_Data = 0 and Max_Data = 0x1FFFFFFF.
- ***uint32_t CAN_RxHeaderTypeDef::IDE***

  Specifies the type of identifier for the message that will be transmitted. This parameter can be a value of ***CAN_identifier_type***
- ***uint32_t CAN_RxHeaderTypeDef::RTR***

  Specifies the type of frame for the message that will be transmitted. This parameter can be a value of ***CAN_remote_transmission_request***

- ***uint32_t CAN_RxHeaderTypeDef::DLC***

  Specifies the length of the frame that will be transmitted. This parameter must be a number between Min_Data = 0 and Max_Data = 8.
- ***uint32_t CAN_RxHeaderTypeDef::Timestamp***

  Specifies the timestamp counter value captured on start of frame reception.

  Note:
  - : Time Triggered Communication Mode must be enabled. This parameter must be a number between Min_Data = 0 and Max_Data = 0xFFFF.
- ***uint32_t CAN_RxHeaderTypeDef::FilterMatchIndex***

  Specifies the index of matching acceptance filter element. This parameter must be a number between Min_Data = 0 and Max_Data = 0xFF.

9.1.5 __CAN_HandleTypeDef

__CAN_HandleTypeDef is defined in the stm32f4xx_hal_can.h

Data Fields

- ***CAN_TypeDef * Instance***
- ***CAN_InitTypeDef Init***
- ***__IO HAL_CAN_StateTypeDef State***
- ***__IO uint32_t ErrorCode***

Field Documentation

- ***CAN_TypeDef* __CAN_HandleTypeDef::Instance***

  Register base address
- ***CAN_InitTypeDef __CAN_HandleTypeDef::Init***

  CAN required parameters
- ***__IO HAL_CAN_StateTypeDef __CAN_HandleTypeDef::State***

  CAN communication state
- ***__IO uint32_t __CAN_HandleTypeDef::ErrorCode***

  CAN Error code. This parameter can be a value of ***CAN_Error_Code***

9.2 CAN Firmware driver API description

The following section lists the various functions of the CAN library.

9.2.1 How to use this driver

1. Initialize the CAN low level resources by implementing the HAL_CAN_MspInit():
   - Enable the CAN interface clock using __HAL_RCC_CANx_CLK_ENABLE()
   - Configure CAN pins
     - Enable the clock for the CAN GPIOs
     - Configure CAN pins as alternate function open-drain
   - In case of using interrupts (e.g. HAL_CAN_ActivateNotification())
     - Configure the CAN interrupt priority using HAL_NVIC_SetPriority()
     - Enable the CAN IRQ handler using HAL_NVIC_EnableIRQ()
     - In CAN IRQ handler, call HAL_CAN_IRQHandler()
2. Initialize the CAN peripheral using HAL_CAN_Init() function. This function resorts to HAL_CAN_MspInit() for low-level initialization.
3. Configure the reception filters using the following configuration functions:
   - HAL_CAN_ConfigFilter()
4. Start the CAN module using HAL_CAN_Start() function. At this level the node is active on the bus: it receive messages, and can send messages.

5. To manage messages transmission, the following Tx control functions can be used:
   - HAL_CAN_AddTxMessage() to request transmission of a new message.
   - HAL_CAN_AbortTxRequest() to abort transmission of a pending message.
   - HAL_CAN_GetTxMailboxesFreeLevel() to get the number of free Tx mailboxes.
   - HAL_CAN_IsTxMessagePending() to check if a message is pending in a Tx mailbox.
   - HAL_CAN_GetTxTimestamp() to get the timestamp of Tx message sent, if time triggered communication mode is enabled.
6. When a message is received into the CAN Rx FIFOs, it can be retrieved using the HAL_CAN_GetRxMessage() function. The function HAL_CAN_GetRxFifoFillLevel() allows to know how many Rx message are stored in the Rx Fifo.
7. Calling the HAL_CAN_Stop() function stops the CAN module.
8. The deinitialization is achieved with HAL_CAN_DeInit() function.

Polling mode operation

1. Reception:
   - Monitor reception of message using HAL_CAN_GetRxFifoFillLevel() until at least one message is received.
   - Then get the message using HAL_CAN_GetRxMessage().
2. Transmission:
   - Monitor the Tx mailboxes availability until at least one Tx mailbox is free, using HAL_CAN_GetTxMailboxesFreeLevel().
   - Then request transmission of a message using HAL_CAN_AddTxMessage().

Interrupt mode operation

1. Notifications are activated using HAL_CAN_ActivateNotification() function. Then, the process can be controlled through the available user callbacks: HAL_CAN_xxxCallback(), using same APIs HAL_CAN_GetRxMessage() and HAL_CAN_AddTxMessage().
2. Notifications can be deactivated using HAL_CAN_DeactivateNotification() function.
3. Special care should be taken for CAN_IT_RX_FIFO0_MSG_PENDING and CAN_IT_RX_FIFO1_MSG_PENDING notifications. These notifications trig the callbacks HAL_CAN_RxFIFO0MsgPendingCallback() and HAL_CAN_RxFIFO1MsgPendingCallback(). User has two possible options here.
   - Directly get the Rx message in the callback, using HAL_CAN_GetRxMessage().
   - Or deactivate the notification in the callback without getting the Rx message. The Rx message can then be got later using HAL_CAN_GetRxMessage(). Once the Rx message have been read, the notification can be activated again.

Sleep mode

1. The CAN peripheral can be put in sleep mode (low power), using HAL_CAN_RequestSleep(). The sleep mode will be entered as soon as the current CAN activity (transmission or reception of a CAN frame) will be completed.
2. A notification can be activated to be informed when the sleep mode will be entered.
3. It can be checked if the sleep mode is entered using HAL_CAN_IsSleepActive(). Note that the CAN state (accessible from the API HAL_CAN_GetState()) is HAL_CAN_STATE_SLEEP_PENDING as soon as the sleep mode request is submitted (the sleep mode is not yet entered), and become HAL_CAN_STATE_SLEEP_ACTIVE when the sleep mode is effective.
4. The wake-up from sleep mode can be trigged by two ways:
   - Using HAL_CAN_WakeUp(). When returning from this function, the sleep mode is exited (if return status is HAL_OK).
   - When a start of Rx CAN frame is detected by the CAN peripheral, if automatic wake up mode is enabled.

Callback registration

9.2.2 Initialization and de-initialization functions

This section provides functions allowing to:

- HAL_CAN_Init : Initialize and configure the CAN.
- HAL_CAN_DeInit : De-initialize the CAN.
- HAL_CAN_MspInit : Initialize the CAN MSP.
- HAL_CAN_MspDeInit : DeInitialize the CAN MSP.

This section contains the following APIs:

- ***HAL_CAN_Init***
- ***HAL_CAN_DeInit***
- ***HAL_CAN_MspInit***
- ***HAL_CAN_MspDeInit***

9.2.3 Configuration functions

This section provides functions allowing to:

- HAL_CAN_ConfigFilter : Configure the CAN reception filters

This section contains the following APIs:

- ***HAL_CAN_ConfigFilter***

9.2.4 Control functions

This section provides functions allowing to:

- HAL_CAN_Start : Start the CAN module
- HAL_CAN_Stop : Stop the CAN module
- HAL_CAN_RequestSleep : Request sleep mode entry.
- HAL_CAN_WakeUp : Wake up from sleep mode.
- HAL_CAN_IsSleepActive : Check is sleep mode is active.
- HAL_CAN_AddTxMessage : Add a message to the Tx mailboxes and activate the corresponding transmission request
- HAL_CAN_AbortTxRequest : Abort transmission request
- HAL_CAN_GetTxMailboxesFreeLevel : Return Tx mailboxes free level
- HAL_CAN_IsTxMessagePending : Check if a transmission request is pending on the selected Tx mailbox
- HAL_CAN_GetRxMessage : Get a CAN frame from the Rx FIFO
- HAL_CAN_GetRxFifoFillLevel : Return Rx FIFO fill level

This section contains the following APIs:

- ***HAL_CAN_Start***
- ***HAL_CAN_Stop***
- ***HAL_CAN_RequestSleep***
- ***HAL_CAN_WakeUp***
- ***HAL_CAN_IsSleepActive***
- ***HAL_CAN_AddTxMessage***
- ***HAL_CAN_AbortTxRequest***
- ***HAL_CAN_GetTxMailboxesFreeLevel***
- ***HAL_CAN_IsTxMessagePending***
- ***HAL_CAN_GetTxTimestamp***
- ***HAL_CAN_GetRxMessage***
- ***HAL_CAN_GetRxFifoFillLevel***

9.2.5 Interrupts management

This section provides functions allowing to:

- HAL_CAN_ActivateNotification : Enable interrupts

- HAL_CAN_DeactivateNotification : Disable interrupts
- HAL_CAN_IRQHandler : Handles CAN interrupt request

This section contains the following APIs:

- ***HAL_CAN_ActivateNotification***
- ***HAL_CAN_DeactivateNotification***
- ***HAL_CAN_IRQHandler***

9.2.6 Peripheral State and Error functions

This subsection provides functions allowing to :

- HAL_CAN_GetState() : Return the CAN state.
- HAL_CAN_GetError() : Return the CAN error codes if any.
- HAL_CAN_ResetError(): Reset the CAN error codes if any.

This section contains the following APIs:

- ***HAL_CAN_GetState***
- ***HAL_CAN_GetError***
- ***HAL_CAN_ResetError***

9.2.7 Detailed description of functions

HAL_CAN_Init

Function name

HAL_StatusTypeDef HAL_CAN_Init (CAN_HandleTypeDef * hcan)

Function description

Initializes the CAN peripheral according to the specified parameters in the CAN_InitStruct.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **HAL:** status

HAL_CAN_DeInit

Function name

HAL_StatusTypeDef HAL_CAN_DeInit (CAN_HandleTypeDef * hcan)

Function description

Deinitializes the CAN peripheral registers to their default reset values.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **HAL:** status

HAL_CAN_MspInit

Function name

void HAL_CAN_MspInit (CAN_HandleTypeDef * hcan)

Function description

Initializes the CAN MSP.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_MspDeInit

Function name

void HAL_CAN_MspDeInit (CAN_HandleTypeDef * hcan)

Function description

DeInitializes the CAN MSP.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_ConfigFilter

Function name

HAL_StatusTypeDef HAL_CAN_ConfigFilter (CAN_HandleTypeDef * hcan, CAN_FilterTypeDef * sFilterConfig)

Function description

Configures the CAN reception filter according to the specified parameters in the CAN_FilterInitStruct.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
- **sFilterConfig:** pointer to a CAN_FilterTypeDef structure that contains the filter configuration information.

Return values

- **None:**

HAL_CAN_Start

Function name

HAL_StatusTypeDef HAL_CAN_Start (CAN_HandleTypeDef * hcan)

Function description

Start the CAN module.

Parameters

- **hcan:** pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **HAL:** status

HAL_CAN_Stop

Function name

HAL_StatusTypeDef HAL_CAN_Stop (CAN_HandleTypeDef * hcan)

Function description

Stop the CAN module and enable access to configuration registers.

Parameters

- **hcan:** pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **HAL:** status

HAL_CAN_RequestSleep

Function name

HAL_StatusTypeDef HAL_CAN_RequestSleep (CAN_HandleTypeDef * hcan)

Function description

Request the sleep mode (low power) entry.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **HAL:** status.

HAL_CAN_WakeUp

Function name

HAL_StatusTypeDef HAL_CAN_WakeUp (CAN_HandleTypeDef * hcan)

Function description

Wake up from sleep mode.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **HAL:** status.

HAL_CAN_IsSleepActive

Function name

uint32_t HAL_CAN_IsSleepActive (CAN_HandleTypeDef * hcan)

Function description

Check is sleep mode is active.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **Status:**
  - 0 : Sleep mode is not active.
  - 1 : Sleep mode is active.

HAL_CAN_AddTxMessage

Function name

HAL_StatusTypeDef HAL_CAN_AddTxMessage (CAN_HandleTypeDef * hcan, CAN_TxHeaderTypeDef * pHeader, uint8_t aData, uint32_t * pTxMailbox)

Function description

Add a message to the first free Tx mailbox and activate the corresponding transmission request.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
- **pHeader:** pointer to a CAN_TxHeaderTypeDef structure.
- **aData:** array containing the payload of the Tx frame.
- **pTxMailbox:** pointer to a variable where the function will return the TxMailbox used to store the Tx message. This parameter can be a value of
  - CAN_Tx_Mailboxes.

Return values

- **HAL:** status

HAL_CAN_AbortTxRequest

Function name

HAL_StatusTypeDef HAL_CAN_AbortTxRequest (CAN_HandleTypeDef * hcan, uint32_t TxMailboxes)

Function description

Abort transmission requests.

Parameters

- **hcan:** pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
- **TxMailboxes:** List of the Tx Mailboxes to abort. This parameter can be any combination of
  - CAN_Tx_Mailboxes.

Return values

- **HAL:** status

HAL_CAN_GetTxMailboxesFreeLevel

Function name

uint32_t HAL_CAN_GetTxMailboxesFreeLevel (CAN_HandleTypeDef * hcan)

Function description

Return Tx Mailboxes free level: number of free Tx Mailboxes.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **Number:** of free Tx Mailboxes.

HAL_CAN_IsTxMessagePending

Function name

uint32_t HAL_CAN_IsTxMessagePending (CAN_HandleTypeDef * hcan, uint32_t TxMailboxes)

Function description

Check if a transmission request is pending on the selected Tx Mailboxes.

Parameters

- **hcan:** pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
- **TxMailboxes:** List of Tx Mailboxes to check. This parameter can be any combination of
  - CAN_Tx_Mailboxes.

Return values

- **Status:**
  - 0 : No pending transmission request on any selected Tx Mailboxes.
  - 1 : Pending transmission request on at least one of the selected Tx Mailbox.

HAL_CAN_GetTxTimestamp

Function name

uint32_t HAL_CAN_GetTxTimestamp (CAN_HandleTypeDef * hcan, uint32_t TxMailbox)

Function description

Return timestamp of Tx message sent, if time triggered communication mode is enabled.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
- **TxMailbox:** Tx Mailbox where the timestamp of message sent will be read. This parameter can be one value of
  - CAN_Tx_Mailboxes.

Return values

- **Timestamp:** of message sent from Tx Mailbox.

HAL_CAN_GetRxMessage

Function name

HAL_StatusTypeDef HAL_CAN_GetRxMessage (CAN_HandleTypeDef * hcan, uint32_t RxFifo, CAN_RxHeaderTypeDef * pHeader, uint8_t aData)

Function description

Get an CAN frame from the Rx FIFO zone into the message RAM.

Parameters

- **hcan:** pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
- **RxFifo:** Fifo number of the received message to be read. This parameter can be a value of
  - CAN_receive_FIFO_number.
- **pHeader:** pointer to a CAN_RxHeaderTypeDef structure where the header of the Rx frame will be stored.
- **aData:** array where the payload of the Rx frame will be stored.

Return values

- **HAL:** status

HAL_CAN_GetRxFifoFillLevel

Function name

uint32_t HAL_CAN_GetRxFifoFillLevel (CAN_HandleTypeDef * hcan, uint32_t RxFifo)

Function description

Return Rx FIFO fill level.

Parameters

- **hcan:** pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
- **RxFifo:** Rx FIFO. This parameter can be a value of
  - CAN_receive_FIFO_number.

Return values

- **Number:** of messages available in Rx FIFO.

HAL_CAN_ActivateNotification

Function name

HAL_StatusTypeDef HAL_CAN_ActivateNotification (CAN_HandleTypeDef * hcan, uint32_t ActiveITs)

Function description

Enable interrupts.

Parameters

- **hcan:** pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
- **ActiveITs:** indicates which interrupts will be enabled. This parameter can be any combination of
  - CAN_Interrupts.

Return values

- **HAL:** status

HAL_CAN_DeactivateNotification

Function name

HAL_StatusTypeDef HAL_CAN_DeactivateNotification (CAN_HandleTypeDef * hcan, uint32_t InactiveITs)

Function description

Disable interrupts.

Parameters

- **hcan:** pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.
- **InactiveITs:** indicates which interrupts will be disabled. This parameter can be any combination of
  - CAN_Interrupts.

Return values

- **HAL:** status

HAL_CAN_IRQHandler

Function name

void HAL_CAN_IRQHandler (CAN_HandleTypeDef * hcan)

Function description

Handles CAN interrupt request.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_TxMailbox0CompleteCallback

Function name

void HAL_CAN_TxMailbox0CompleteCallback (CAN_HandleTypeDef * hcan)

Function description

Transmission Mailbox 0 complete callback.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_TxMailbox1CompleteCallback

Function name

void HAL_CAN_TxMailbox1CompleteCallback (CAN_HandleTypeDef * hcan)

Function description

Transmission Mailbox 1 complete callback.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_TxMailbox2CompleteCallback

Function name

void HAL_CAN_TxMailbox2CompleteCallback (CAN_HandleTypeDef * hcan)

Function description

Transmission Mailbox 2 complete callback.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_TxMailbox0AbortCallback

Function name

void HAL_CAN_TxMailbox0AbortCallback (CAN_HandleTypeDef * hcan)

Function description

Transmission Mailbox 0 Cancellation callback.

Parameters

- **hcan:** pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_TxMailbox1AbortCallback

Function name

void HAL_CAN_TxMailbox1AbortCallback (CAN_HandleTypeDef * hcan)

Function description

Transmission Mailbox 1 Cancellation callback.

Parameters

- **hcan:** pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_TxMailbox2AbortCallback

Function name

void HAL_CAN_TxMailbox2AbortCallback (CAN_HandleTypeDef * hcan)

Function description

Transmission Mailbox 2 Cancellation callback.

Parameters

- **hcan:** pointer to an CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_RxFifo0MsgPendingCallback

Function name

void HAL_CAN_RxFifo0MsgPendingCallback (CAN_HandleTypeDef * hcan)

Function description

Rx FIFO 0 message pending callback.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_RxFifo0FullCallback

Function name

void HAL_CAN_RxFifo0FullCallback (CAN_HandleTypeDef * hcan)

Function description

Rx FIFO 0 full callback.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_RxFifo1MsgPendingCallback

Function name

void HAL_CAN_RxFifo1MsgPendingCallback (CAN_HandleTypeDef * hcan)

Function description

Rx FIFO 1 message pending callback.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_RxFifo1FullCallback

Function name

void HAL_CAN_RxFifo1FullCallback (CAN_HandleTypeDef * hcan)

Function description

Rx FIFO 1 full callback.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_SleepCallback

Function name

void HAL_CAN_SleepCallback (CAN_HandleTypeDef * hcan)

Function description

Sleep callback.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_WakeUpFromRxMsgCallback

Function name

void HAL_CAN_WakeUpFromRxMsgCallback (CAN_HandleTypeDef * hcan)

Function description

WakeUp from Rx message callback.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_ErrorCallback

Function name

void HAL_CAN_ErrorCallback (CAN_HandleTypeDef * hcan)

Function description

Error CAN callback.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **None:**

HAL_CAN_GetState

Function name

HAL_CAN_StateTypeDef HAL_CAN_GetState (CAN_HandleTypeDef * hcan)

Function description

Return the CAN state.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **HAL:** state

HAL_CAN_GetError

Function name

uint32_t HAL_CAN_GetError (CAN_HandleTypeDef * hcan)

Function description

Return the CAN error code.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **CAN:** Error Code

HAL_CAN_ResetError

Function name

HAL_StatusTypeDef HAL_CAN_ResetError (CAN_HandleTypeDef * hcan)

Function description

Reset the CAN error code.

Parameters

- **hcan:** pointer to a CAN_HandleTypeDef structure that contains the configuration information for the specified CAN.

Return values

- **HAL:** status

9.3 CAN Firmware driver defines

The following section lists the various define and macros of the module.

9.3.1 CAN

CAN

CAN Error Code

HAL_CAN_ERROR_NONE

No error

HAL_CAN_ERROR_EWG

Protocol Error Warning

HAL_CAN_ERROR_EPV

Error Passive

HAL_CAN_ERROR_BOF

Bus-off error

HAL_CAN_ERROR_STF

Stuff error

HAL_CAN_ERROR_FOR

Form error

HAL_CAN_ERROR_ACK

Acknowledgment error

HAL_CAN_ERROR_BR

Bit recessive error

HAL_CAN_ERROR_BD

Bit dominant error

HAL_CAN_ERROR_CRC

CRC error

HAL_CAN_ERROR_RX_FOV0

Rx FIFO0 overrun error

HAL_CAN_ERROR_RX_FOV1

Rx FIFO1 overrun error

HAL_CAN_ERROR_TX_ALST0

TxMailbox 0 transmit failure due to arbitration lost

HAL_CAN_ERROR_TX_TERR0

TxMailbox 1 transmit failure due to tranmit error

HAL_CAN_ERROR_TX_ALST1

TxMailbox 0 transmit failure due to arbitration lost

HAL_CAN_ERROR_TX_TERR1

TxMailbox 1 transmit failure due to tranmit error

HAL_CAN_ERROR_TX_ALST2

TxMailbox 0 transmit failure due to arbitration lost

HAL_CAN_ERROR_TX_TERR2

TxMailbox 1 transmit failure due to tranmit error

HAL_CAN_ERROR_TIMEOUT

Timeout error

HAL_CAN_ERROR_NOT_INITIALIZED

Peripheral not initialized

HAL_CAN_ERROR_NOT_READY

Peripheral not ready

HAL_CAN_ERROR_NOT_STARTED

Peripheral not started

HAL_CAN_ERROR_PARAM

Parameter error

HAL_CAN_ERROR_INTERNAL

Internal error

CAN Exported Macros

__HAL_CAN_RESET_HANDLE_STATE

Description:

- Reset CAN handle state.

Parameters:

- __HANDLE__: CAN handle.

Return value:

- None

__HAL_CAN_ENABLE_IT

Description:

- Enable the specified CAN interrupts.

Parameters:

- __HANDLE__: CAN handle.
- __INTERRUPT__: CAN Interrupt sources to enable. This parameter can be any combination of
  - CAN_Interrupts

Return value:

- None

__HAL_CAN_DISABLE_IT

Description:

- Disable the specified CAN interrupts.

Parameters:

- __HANDLE__: CAN handle.
- __INTERRUPT__: CAN Interrupt sources to disable. This parameter can be any combination of
  - CAN_Interrupts

Return value:

- None

__HAL_CAN_GET_IT_SOURCE

Description:

- Check if the specified CAN interrupt source is enabled or disabled.

Parameters:

- __HANDLE__: specifies the CAN Handle.
- __INTERRUPT__: specifies the CAN interrupt source to check. This parameter can be a value of
  - CAN_Interrupts

Return value:

- The: state of __IT__ (TRUE or FALSE).

__HAL_CAN_GET_FLAG

Description:

- Check whether the specified CAN flag is set or not.

Parameters:

- __HANDLE__: specifies the CAN Handle.
- __FLAG__: specifies the flag to check. This parameter can be one of
  - CAN_flags

Return value:

- The: state of __FLAG__ (TRUE or FALSE).

__HAL_CAN_CLEAR_FLAG

Description:

- Clear the specified CAN pending flag.

Parameters:

- __HANDLE__: specifies the CAN Handle.
- __FLAG__: specifies the flag to check. This parameter can be one of the following values:
  - CAN_FLAG_RQCP0: Request complete MailBox 0 Flag
  - CAN_FLAG_TXOK0: Transmission OK MailBox 0 Flag
  - CAN_FLAG_ALST0: Arbitration Lost MailBox 0 Flag
  - CAN_FLAG_TERR0: Transmission error MailBox 0 Flag
  - CAN_FLAG_RQCP1: Request complete MailBox 1 Flag
  - CAN_FLAG_TXOK1: Transmission OK MailBox 1 Flag
  - CAN_FLAG_ALST1: Arbitration Lost MailBox 1 Flag
  - CAN_FLAG_TERR1: Transmission error MailBox 1 Flag
  - CAN_FLAG_RQCP2: Request complete MailBox 2 Flag
  - CAN_FLAG_TXOK2: Transmission OK MailBox 2 Flag
  - CAN_FLAG_ALST2: Arbitration Lost MailBox 2 Flag
  - CAN_FLAG_TERR2: Transmission error MailBox 2 Flag
  - CAN_FLAG_FF0: RX FIFO 0 Full Flag
  - CAN_FLAG_FOV0: RX FIFO 0 Overrun Flag
  - CAN_FLAG_FF1: RX FIFO 1 Full Flag
  - CAN_FLAG_FOV1: RX FIFO 1 Overrun Flag
  - CAN_FLAG_WKUI: Wake up Interrupt Flag
  - CAN_FLAG_SLAKI: Sleep acknowledge Interrupt Flag

Return value:

- None

CAN Filter Activation

CAN_FILTER_DISABLE

Disable filter

CAN_FILTER_ENABLE

Enable filter

CAN Filter FIFO

CAN_FILTER_FIFO0

Filter FIFO 0 assignment for filter x

CAN_FILTER_FIFO1

Filter FIFO 1 assignment for filter x

CAN Filter Mode

CAN_FILTERMODE_IDMASK

Identifier mask mode

CAN_FILTERMODE_IDLIST

Identifier list mode

CAN Filter Scale

CAN_FILTERSCALE_16BIT

Two 16-bit filters

CAN_FILTERSCALE_32BIT

One 32-bit filter

CAN Flags

CAN_FLAG_RQCP0

Request complete MailBox 0 flag

CAN_FLAG_TXOK0

Transmission OK MailBox 0 flag

CAN_FLAG_ALST0

Arbitration Lost MailBox 0 flag

CAN_FLAG_TERR0

Transmission error MailBox 0 flag

CAN_FLAG_RQCP1

Request complete MailBox1 flag

CAN_FLAG_TXOK1

Transmission OK MailBox 1 flag

CAN_FLAG_ALST1

Arbitration Lost MailBox 1 flag

CAN_FLAG_TERR1

Transmission error MailBox 1 flag

CAN_FLAG_RQCP2

Request complete MailBox2 flag

CAN_FLAG_TXOK2

Transmission OK MailBox 2 flag

CAN_FLAG_ALST2

Arbitration Lost MailBox 2 flag

CAN_FLAG_TERR2

Transmission error MailBox 2 flag

CAN_FLAG_TME0

Transmit mailbox 0 empty flag

CAN_FLAG_TME1

Transmit mailbox 1 empty flag

CAN_FLAG_TME2

Transmit mailbox 2 empty flag

CAN_FLAG_LOW0

Lowest priority mailbox 0 flag

CAN_FLAG_LOW1

Lowest priority mailbox 1 flag

CAN_FLAG_LOW2

Lowest priority mailbox 2 flag

CAN_FLAG_FF0

RX FIFO 0 Full flag

CAN_FLAG_FOV0

RX FIFO 0 Overrun flag

CAN_FLAG_FF1

RX FIFO 1 Full flag

CAN_FLAG_FOV1

RX FIFO 1 Overrun flag

CAN_FLAG_INAK

Initialization acknowledge flag

CAN_FLAG_SLAK

Sleep acknowledge flag

CAN_FLAG_ERRI

Error flag

CAN_FLAG_WKU

Wake up interrupt flag

CAN_FLAG_SLAKI

Sleep acknowledge interrupt flag

CAN_FLAG_EWG

Error warning flag

CAN_FLAG_EPV

Error passive flag

CAN_FLAG_BOF

Bus-Off flag

CAN Identifier Type

CAN_ID_STD

Standard Id

CAN_ID_EXT

Extended Id

CAN InitStatus

CAN_INITSTATUS_FAILED

CAN initialization failed

CAN_INITSTATUS_SUCCESS

CAN initialization OK

CAN Interrupts

CAN_IT_TX_MAILBOX_EMPTY

Transmit mailbox empty interrupt

CAN_IT_RX_FIFO0_MSG_PENDING

FIFO 0 message pending interrupt

CAN_IT_RX_FIFO0_FULL

FIFO 0 full interrupt

CAN_IT_RX_FIFO0_OVERRUN

FIFO 0 overrun interrupt

CAN_IT_RX_FIFO1_MSG_PENDING

FIFO 1 message pending interrupt

CAN_IT_RX_FIFO1_FULL

FIFO 1 full interrupt

CAN_IT_RX_FIFO1_OVERRUN

FIFO 1 overrun interrupt

CAN_IT_WAKEUP

Wake-up interrupt

CAN_IT_SLEEP_ACK

Sleep acknowledge interrupt

CAN_IT_ERROR_WARNING

Error warning interrupt

CAN_IT_ERROR_PASSIVE

Error passive interrupt

CAN_IT_BUSOFF

Bus-off interrupt

CAN_IT_LAST_ERROR_CODE

Last error code interrupt

CAN_IT_ERROR

Error Interrupt

CAN Operating Mode

CAN_MODE_NORMAL

Normal mode

CAN_MODE_LOOPBACK

Loopback mode

CAN_MODE_SILENT

Silent mode

CAN_MODE_SILENT_LOOPBACK

Loopback combined with silent mode

CAN Receive FIFO Number

CAN_RX_FIFO0

CAN receive FIFO 0

CAN_RX_FIFO1

CAN receive FIFO 1

CAN Remote Transmission Request

CAN_RTR_DATA

Data frame

CAN_RTR_REMOTE

Remote frame

CAN Synchronization Jump Width

CAN_SJW_1TQ

1 time quantum

CAN_SJW_2TQ

2 time quantum

CAN_SJW_3TQ

3 time quantum

CAN_SJW_4TQ

4 time quantum

CAN Time Quantum in Bit Segment 1

CAN_BS1_1TQ

1 time quantum

CAN_BS1_2TQ

2 time quantum

CAN_BS1_3TQ

3 time quantum

CAN_BS1_4TQ

4 time quantum

CAN_BS1_5TQ

5 time quantum

CAN_BS1_6TQ

6 time quantum

CAN_BS1_7TQ

7 time quantum

CAN_BS1_8TQ

8 time quantum

CAN_BS1_9TQ

9 time quantum

CAN_BS1_10TQ

10 time quantum

CAN_BS1_11TQ

11 time quantum

CAN_BS1_12TQ

12 time quantum

CAN_BS1_13TQ

13 time quantum

CAN_BS1_14TQ

14 time quantum

CAN_BS1_15TQ

15 time quantum

CAN_BS1_16TQ

16 time quantum

CAN Time Quantum in Bit Segment 2

CAN_BS2_1TQ

1 time quantum

CAN_BS2_2TQ

2 time quantum

CAN_BS2_3TQ

3 time quantum

CAN_BS2_4TQ

4 time quantum

CAN_BS2_5TQ

5 time quantum

CAN_BS2_6TQ

6 time quantum

CAN_BS2_7TQ

7 time quantum

CAN_BS2_8TQ

8 time quantum

CAN Tx Mailboxes

CAN_TX_MAILBOX0

Tx Mailbox 0

CAN_TX_MAILBOX1

Tx Mailbox 1

CAN_TX_MAILBOX2

Tx Mailbox 2