Error description of HAL-Driver **stm32h7xx_hal_fdcan Version 1.10.0.0**

```
HAL_StatusTypeDef HAL_FDCAN_DeactivateNotification(FDCAN_HandleTypeDef *hfdcan, uint32_t InactiveITs)
{
  uint32_t ITLineSelection;
  HAL_FDCAN_StateTypeDef state = hfdcan->State;

  /* Check function parameters */
  assert_param(IS_FDCAN_IT(InactiveITs));

  if ((state == HAL_FDCAN_STATE_READY) || (state == HAL_FDCAN_STATE_BUSY))
  {
    /* Disable the selected interrupts */
    __HAL_FDCAN_DISABLE_IT(hfdcan, InactiveITs);

    if ((InactiveITs & FDCAN_IT_TX_COMPLETE) != 0U)
    {
      /* Disable Tx Buffer Transmission Interrupts */
      CLEAR_REG(hfdcan->Instance->TXBTIE);
    }

    if ((InactiveITs & FDCAN_IT_TX_ABORT_COMPLETE) != 0U)
    {
      /* Disable Tx Buffer Cancellation Finished Interrupt */
      CLEAR_REG(hfdcan->Instance->TXBCIE);
    }
```

// The following highlighted code block (➔ reset of the ILE register) should be removed here and // are not needed!

```
// Deactivating of the interrupt in the IE register is sufficient.
// At the moment this lines here in this function can cause a configuration error in the
// ILE register.
// A interrupt line flag in the ILE register can be cleared wrongly even one another active
// interrupt is connected to this interrupt line !
// A call of the function HAL_FDCAN_ActivateNotification doesn't set the wrongly cleared
// interrupt line flag before !
//
// If someone doesn't know about this error, someone will be wondering why the wanted
// interrupt will never occur !
//
// Workaround example for TX-COMPLETE interrupt:
// Calling of the macro CLEAR_BIT(hfdcan->Instance->IE, FDCAN_IT_TX_COMPLETE) instead of
// the function HAL_FDCAN_DeactivateNotification at the moment.

  ITLineSelection = hfdcan->Instance->ILS;

  if ((hfdcan->Instance->IE | ITLineSelection) == ITLineSelection)
  {
    /* Disable Interrupt line 0 */
    CLEAR_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE0);
  }

  if ((hfdcan->Instance->IE & ITLineSelection) == 0U)
  {
    /* Disable Interrupt line 1 */
```

```
    CLEAR_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE1);
   }

   /* Return function status */
   return HAL_OK;
  }
  else
  {
   /* Update error code */
   hfdcan->ErrorCode |= HAL_FDCAN_ERROR_NOT_INITIALIZED;

   return HAL_ERROR;
  }
}
```

```
HAL_StatusTypeDef HAL_FDCAN_ActivateNotification(FDCAN_HandleTypeDef *hfdcan, uint32_t ActiveITs, uint32_t
BufferIndexes)
{
 HAL_FDCAN_StateTypeDef state = hfdcan->State;

 /* Check function parameters */
 assert_param(IS_FDCAN_IT(ActiveITs));

 if ((state == HAL_FDCAN_STATE_READY) || (state == HAL_FDCAN_STATE_BUSY))
 {
  // The following highlighted code block (➔ setting of the ILE register) should be moved into the
  // function HAL_FDCAN_ConfigInterruptLines,
  // because the interrupt lines will be configured there (➔ setting of the ILS register).
 // At the moment the lines here in this function can cause a configuration error in the
```

```
// ILE register, if this function is called first,
// before function HAL_FDCAN_ConfigInterruptLines was called.
//
// If someone doesn't know the right sequence of calling of the HAL-driver functions,
// someone will be wondering why the wanted interrupt will never occur !

  /* Enable Interrupt lines */
  if ((ActiveITs & hfdcan->Instance->ILS) == 0U)
  {
    /* Enable Interrupt line 0 */
    SET_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE0);
  }
  else if ((ActiveITs & hfdcan->Instance->ILS) == ActiveITs)
  {
    /* Enable Interrupt line 1 */
    SET_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE1);
  }
  else
  {
    /* Enable Interrupt lines 0 and 1 */
    hfdcan->Instance->ILE = (FDCAN_INTERRUPT_LINE0 | FDCAN_INTERRUPT_LINE1);
  }

  if ((ActiveITs & FDCAN_IT_TX_COMPLETE) != 0U)
  {
    /* Enable Tx Buffer Transmission Interrupt to set TC flag in IR register,
       but interrupt will only occur if TC is enabled in IE register */
    SET_BIT(hfdcan->Instance->TXBTIE, BufferIndexes);
  }

  if ((ActiveITs & FDCAN_IT_TX_ABORT_COMPLETE) != 0U)
  {
```

```
      /* Enable Tx Buffer Cancellation Finished Interrupt to set TCF flag in IR register,
         but interrupt will only occur if TCF is enabled in IE register */
      SET_BIT(hfdcan->Instance->TXBCIE, BufferIndexes);
    }

    /* Enable the selected interrupts */
    __HAL_FDCAN_ENABLE_IT(hfdcan, ActiveITs);

    /* Return function status */
    return HAL_OK;
  }
  else
  {
    /* Update error code */
    hfdcan->ErrorCode |= HAL_FDCAN_ERROR_NOT_INITIALIZED;

    return HAL_ERROR;
  }
}
```

```
HAL_StatusTypeDef HAL_FDCAN_ConfigInterruptLines(FDCAN_HandleTypeDef *hfdcan, uint32_t ITList, uint32_t
InterruptLine)
{
 HAL_FDCAN_StateTypeDef state = hfdcan->State;

 /* Check function parameters */
 assert_param(IS_FDCAN_IT(ITList));
 assert_param(IS_FDCAN_IT_LINE(InterruptLine));

 if ((state == HAL_FDCAN_STATE_READY) || (state == HAL_FDCAN_STATE_BUSY))
 {
  // The following highlighted code block (➔ setting of the ILE register) should be inserted here
  // instead in the function HAL_FDCAN_ActivateNotification,
  // because the interrupt lines will be configured here (➔setting of the ILS register).
 // At the moment this lines are present in function HAL_FDCAN_ActivateNotification.
 // It can cause a configuration error in the ILE register, if this function is called after the function
 // HAL_FDCAN_ActivateNotification.
 //
 // If someone doesn't know the right sequence of calling of the HAL-driver functions, someone
 // will be wondering why the wanted interrupt will never occur !

  /* Assign list of interrupts to the selected line */
  if (InterruptLine == FDCAN_INTERRUPT_LINE0)
  {
    CLEAR_BIT(hfdcan->Instance->ILS, ITList);
   SET_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE0);
  }
  else /* InterruptLine == FDCAN_INTERRUPT_LINE1 */
  {
```

```
    SET_BIT(hfdcan->Instance->ILS, ITList);
    SET_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE1);
   }

   /* Return function status */
   return HAL_OK;
  }
  else
  {
   /* Update error code */
   hfdcan->ErrorCode |= HAL_FDCAN_ERROR_NOT_INITIALIZED;

   return HAL_ERROR;
  }
}
```

```
HAL_StatusTypeDef HAL_FDCAN_TT_DeactivateNotification(FDCAN_HandleTypeDef *hfdcan, uint32_t InactiveTTITs)
{
  uint32_t ITLineSelection;
  HAL_FDCAN_StateTypeDef state = hfdcan->State;

  /* Check function parameters */
  assert_param(IS_FDCAN_TT_INSTANCE(hfdcan->Instance));
  assert_param(IS_FDCAN_TT_IT(InactiveTTITs));

  if ((state == HAL_FDCAN_STATE_READY) || (state == HAL_FDCAN_STATE_BUSY))
  {
   /* Disable the selected TT interrupts */
   __HAL_FDCAN_TT_DISABLE_IT(hfdcan, InactiveTTITs);
```

// The following highlighted code block (➔ reset of the ILE register) should be removed here and
// are not needed!
// Deactivating of the interrupt in the TTIE register is sufficient.
// At the moment this lines here in this function can cause a configuration error in the
// ILE register.
// A interrupt line flag in the ILE register can be cleared wrongly even one another active
// interrupt is connected to this interrupt line !
// A call of the function HAL_FDCAN_TT_ActivateNotification doesn't set the wrongly cleared
// interrupt line flag before !
//
// If someone doesn't know about this error, someone will be wondering why the wanted
// interrupt will never occur !
//
// Workaround example for REGISTER TIMEMARK interrupt:
// Calling of the macro CLEAR_BIT(hfdcan->ttcan->TTIE, FDCAN_TT_IT_REG_TIME_MARK);
// instead of the function HAL_FDCAN_TT_DeactivateNotification at the moment.

```
ITLineSelection = hfdcan->ttcan->TTILS;

if ((hfdcan->ttcan->TTIE | ITLineSelection) == ITLineSelection)
{
  /* Disable Interrupt line 0 */
  CLEAR_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE0);
}
```

```
    if ((hfdcan->ttcan->TTIE & ITLineSelection) == 0U)
    {
      /* Disable Interrupt line 1 */
      CLEAR_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE1);
    }

    /* Return function status */
    return HAL_OK;
  }
  else
  {
    /* Update error code */
    hfdcan->ErrorCode |= HAL_FDCAN_ERROR_NOT_INITIALIZED;

    return HAL_ERROR;
  }
}
```

```
HAL_StatusTypeDef HAL_FDCAN_TT_ActivateNotification(FDCAN_HandleTypeDef *hfdcan, uint32_t ActiveTTITs)
{
 HAL_FDCAN_StateTypeDef state = hfdcan->State;

 /* Check function parameters */
 assert_param(IS_FDCAN_TT_INSTANCE(hfdcan->Instance));
 assert_param(IS_FDCAN_TT_IT(ActiveTTITs));

 if ((state == HAL_FDCAN_STATE_READY) || (state == HAL_FDCAN_STATE_BUSY))
 {
  // The following highlighted code block (➔ setting of the ILE register) should be moved into the
  // function HAL_FDCAN_TT_ConfigInterruptLines,
  // because the interrupt lines will be configured there (➔ setting of the TTILS register).
 // At the moment the lines here in this function can cause a configuration error in the
 // ILE register, if this function is called first,
 // before function HAL_FDCAN_TT_ConfigInterruptLines was called.
 //
 // If someone doesn't know the right sequence of calling of the HAL-driver functions, someone
 // will be wondering why the wanted interrupt will never occur !

   /* Enable Interrupt lines */
   if ((ActiveTTITs & hfdcan->ttcan->TTILS) == 0U)
   {
     /* Enable Interrupt line 0 */
     SET_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE0);
   }
   else if ((ActiveTTITs & hfdcan->ttcan->TTILS) == ActiveTTITs)
   {
     /* Enable Interrupt line 1 */
```

```
    SET_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE1);
   }
   else
   {
    /* Enable Interrupt lines 0 and 1 */
    hfdcan->Instance->ILE = (FDCAN_INTERRUPT_LINE0 | FDCAN_INTERRUPT_LINE1);
   }

   /* Enable the selected TT interrupts */
   __HAL_FDCAN_TT_ENABLE_IT(hfdcan, ActiveTTITs);

   /* Return function status */
   return HAL_OK;
  }
  else
  {
   /* Update error code */
   hfdcan->ErrorCode |= HAL_FDCAN_ERROR_NOT_INITIALIZED;

   return HAL_ERROR;
  }
}
```

```
HAL_StatusTypeDef HAL_FDCAN_TT_ConfigInterruptLines(FDCAN_HandleTypeDef *hfdcan, uint32_t TTITList, uint32_t
InterruptLine)
{
 HAL_FDCAN_StateTypeDef state = hfdcan->State;

 /* Check function parameters */
 assert_param(IS_FDCAN_TT_INSTANCE(hfdcan->Instance));
 assert_param(IS_FDCAN_TT_IT(TTITList));
 assert_param(IS_FDCAN_IT_LINE(InterruptLine));

 if ((state == HAL_FDCAN_STATE_READY) || (state == HAL_FDCAN_STATE_BUSY))
 {
  // The following highlighted code block (➔ setting of the ILE register) should be inserted here
  // instead in the function HAL_FDCAN_TT_ActivateNotification,
  // because the interrupt lines will be configured here (➔setting of the TTILS register).
  // At the moment this lines are present in function HAL_FDCAN_TT_ActivateNotification.
  // It can cause a configuration error in the ILE register, if this function is called after the function
 // HAL_FDCAN_ActivateNotification.
 //
 // If someone doesn't know the right sequence of calling of the HAL-driver functions, someone
 //will be wondering why the wanted interrupt will never occur !


  /* Assign list of interrupts to the selected line */
  if (InterruptLine == FDCAN_INTERRUPT_LINE0)
  {
   CLEAR_BIT(hfdcan->ttcan->TTILS, TTITList);
   SET_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE0);
  }
```

```
    else /* InterruptLine == FDCAN_INTERRUPT_LINE1 */
    {
      SET_BIT(hfdcan->ttcan->TTILS, TTITList);
     SET_BIT(hfdcan->Instance->ILE, FDCAN_INTERRUPT_LINE1);
    }

    /* Return function status */
    return HAL_OK;
  }
  else
  {
    /* Update error code */
    hfdcan->ErrorCode |= HAL_FDCAN_ERROR_NOT_INITIALIZED;

    return HAL_ERROR;
  }
}
```

2022-01-20, GK_CPfal2020