

API can.h

Functions

In can.h these functions are available. Their content is described further down.

- **InitializeCAN**
- **CANReset**
- **CAN2515SetRXB0Filters**
- **CAN2515SetRXB1Filters**
- **config_SPI_SPARE**
- **config_dpi204**
- **CANTxReady**
- **CANRxReady**
- **CANGetMsg**
- **CANSendMsg**
- **Evk1100PrintDisp**
- **ClearMessages**

InitializeCAN

```
/*
 * Function:          void InitializeCAN( int Channel , int BusSpeed)
 *
 * PreCondition:     SPI port configured
 *
 * Input:            Channel: SPI channel number, 1 based
 *                  BusSpeed: Bus Speed code: CAN_1000kbps, CAN_500kbps
 *                  CAN_250kbps, or CAN_125kbps
 *
 * Output:           None
 *
 * Side Effects:     None
 *
 * Overview:         Sets upp the Rx filters and resets the CAN module
 *
 * Note:             None.
 *
 * Example:          InitializeCAN(0,CAN_125kbps)
 */
```

CANReset

```
/* *****  
 * Function:      void CANReset(int Channel)  
 *  
 * PreCondition:  SPI initialization and configuration  
 *  
 * Input:        channel: channel of type int  
 *  
 * Output:       None  
 *  
 * Side Effects:  None  
 *  
 * Overview:     Sends a software reset command to Mcp2515  
 *  
 * Note:         None.  
*****
```

Example: CANReset(0)

```
*****/
```

CAN2515SetRXB0Filters

```
/* *****  
 * Function:      void CAN2515SetRXB0Filters(int Channel, UINT16 Mask0, UINT16*  
pFlt0_1 )  
 *  
 * PreCondition:  SPI port configured  
 *  
 * Input:        Channel: SPI channel number, 1 based  
 *              Mask0: define identifier 11 bits that are must match.  
 *              pFlt0_1: Pointer to array of UINT16, 2 words,  
 *              ie. Filter 0 and Filer 1. Lower 11 bits are signicant  
 *  
 *              1 = Rcvd Idenifier bit must match filter bit  
 *  
 *              0 = Don't care Rcvd identifier bit.  
 *  
 * Output:       None  
 *  
 * Side Effects:  None  
 *  
 * Overview:     Initialize RXB0 mask and 2 filters. See MCP2515 datasheet for  
more information.  
 *  
 * Note:         None.  
*****
```

Example: CAN2515SetRXB0Filters(Channel, 0, Flt);

```
*****/
```

CAN2515SetRXB1Filters

```
/*
 * Function:    void CAN2515SetRXB1Filters(int Channel, UINT16 Mask0, UINT16*
pFlt0_1 )
 *
 * PreCondition:    SPI port configured
 *
 * Input:           Channel: SPI channel number, 1 based
 *                 Mask0: define identifier 11 bits that are must match.
 *                 pFlt2_5: Pointer to array of UINT16, 4 words,
 *
 *                 ie. Filter 2 thru Filer 5. Lower 11 bits are signicant.
 *
 *                 1 = Rcvd Idenifier bit must match filter bit
 *
 *                 0 = Don't care Rcvd identifier bit.
 *
 * Output:         None
 *
 * Side Effects:   None
 *
 * Overview:      Initialize RXB1 mask and 4 filters. See MCP2515
datasheet for more information.
 *
 * Note:         None.

Example: CAN2515SetRXB1Filters(Channel, 0, &Flt[2]);
*/
```

config_SPI_SPARE

```
/*
 * Function:      void config_SPI_SPARE(void)
 *
 * PreCondition:  None
 *
 * Input:         None
 *
 * Output:       None
 *
 * Side Effects:  None
 *
 * Overview:     This function setups the SPI spare port. This to be
able to communicate with other devices, e.g. MCP2515
 *
 * Note:         None.

Example:        config_SPI_SPARE();
*/
```

config_dpi204

```
/* *****  
 * Function:          void config_dpi204(void)  
 *  
 * PreCondition:     None  
 *  
 * Input:            None  
 *  
 * Output:           None  
 *  
 * Side Effects:     None  
 *  
 * Overview:         This function setups the Evk1100 display  
settings for the SPI interface  
 *  
 * Note:             None.  
 *  
 * Example:          config_dpi204();  
***** */
```

CANTxReady

```
/* *****  
 * Function:          BOOL CANTxReady( int Channel )  
 *  
 * PreCondition:     CAN initialized  
 *  
 * Input:            Channel: SPI channel number, 1 based  
 *  
 * Output:           Return true if the CAN controller Transmit buffer is  
available  
 *  
 * Side Effects:     None  
 *  
 * Overview:         Application can check is Tx buffer is available before  
calling CANSendMSg.  
 *  
 * Note:             None.  
 *  
 * Example:          CANTxReady(0);  
***** */
```

CANRxReady

```
/* *****  
 * Function:          BOOL CANRxReady( int Channel )  
 *  
 * PreCondition:     CAN initialized  
 *  
 * Input:            Channel: SPI channel number, 1 based  
 *  
 * Output:           Return true if the CAN controller Receive buffer is  
Full  
 *  
 * Side Effects:     None  
 *  
 * Overview:         Application can check is Rx buffer is full before  
calling CANGetMSg.  
 *  
 * Note:             None.  
 *
```

```
* Example:          CANRxReady(0);
*****/
```

CANgetMsg

```
/* *****
 * Function:        BOOL CANGetMsg( int Channel, UINT16 *pIdentifier,UINT8*
Msg, UINT8 *pMsgSize )
 *
 * PreCondition:   CAN initialized
 *
 * Input:          Channel: SPI channel number, 1 based
 *                Identifier: 11bit or 29bit data for identifier
 *                Msg: Data bytes, 8 bytes max
 *                MsgSize: number of data bytes
 *
 * Output:         Return true if a message is received
 *
 * Side Effects:   None
 *
 * Overview:       Application call this function to read the CAN message
 *                received by the CAN
controller
 *
 * Note:           None.
 *
 * Example:        CANGetMsg(0, &Ident, msg, &mSize );
*****/
```

CANsendMsg

```
/* *****
 * Function:        BOOL CANSendMsg( int Channel, UINT16 Identifier,UINT8*
Msg, UINT8 MsgSize, int R )
 *
 * PreCondition:   CAN initialized
 *
 * Input:          Channel: SPI channel number, 1 based
 *                Identifier: 11bit or 29 bit data for identifier
 *                Msg: Data bytes, 8 bytes max
 *                MsgSize: number of data bytes
 *                R: If user wants to send a remote frame or not
 *
 * Output:         Return true if the message was successfully transferred
 *                to the CAN controller Tx buffer.
 *
 * Side Effects:   None
 *
 * Overview:       Application call this function to send a message to the
CAN bus
 *
 * Note:           None.
 *
 * Example:        Channel, Identifier (max 0x1fffffff (29 bits)),
Message, Number of bytes, Remote frame R or 0 (no remote frame)
                // Standard id
                CANSendMsg( 0, 0x00, msg, 8, 0 );(no remote frame)
                CANSendMsg( 0, 0x00, msg, 8, R );(remote frame)

                // Extened id
                CANSendMsg( 0, 0x8ff, msg, 8, 0 );(no remote frame)
*****/
```

```
CANSendMsg( 0, 0x8ff, msg, 8, R );(remote frame)
```

```
*****/
```

Evk1100PrintDisp

```
/* ***** */
* Function:      void Evk1100PrintDisp(UINT32* pIdentifier, UINT8* Msg,
UINT8* pMsgSize)
*
* PreCondition:  void config_dpi204(void) has to be run first
*
* Input:         pIdentifier: Pointer to Identifier of UINT32
*                Msg:      Pointer to array of UINT8
*                pMsgSize:  Pointer to the size of the messaged of UINT8
*
* Output:        None
*
* Side Effects:  None
*
* Overview:      This function prints all eight message bytes, the
identifier and the message size
*
* Note:          None.
*
* Example:       Evk1100PrintDisp(&Ident, msg, &mSize );
/* ***** */
```

ClearMessages

```
/* ***** */
* Function:      void ClearMessages(UINT8* Msg)
*
* PreCondition:  None
*
* Input:         Pointer to array of UINT8.
*
* Output:        None
*
* Side Effects:  None
*
* Overview:      This function clears the message content
*
* Note:          None.
*
* Example:       ClearMessages(msg);
/* ***** */
```