CANMate Linux API Reference

CANMAte API Details :

- **HANDLE OpenCANMate(LPVOID lpDataCallBack = NULL, LPVOID lpEventCallBack = NULL)**

  This API enumerates all USB ports and finds the USB port to which CANMate hardware is connected. It then opens the CANMate. CANMate device internally initializes the CAN controller and replies with the status. Note : Present version of CANMate Shared Object supports only one CANMate. It will open only the first CANMate even if multiple CANMate devices are connected to a PC.

  Parameters :

  1. LPVOID lpDataCallBack
     This is a pointer to a call back function for the application to receive data

  2. LPVOID lpEventCallBack
     This is a pointer to a call back function for the application to receive events and error notifications.

  Return Value :

  Returns the USB port handle if SUCCESS. Returns NULL if there is any error.

- **int CloseCANMate(HANDLE hDev)**

  This API sends the CLOSE command to CANMate. It also closes the USB port and frees all the related resources.

Parameters :

1. HANDLE hDev
   Handle of the USB port connected to CANMate device.

Return Value :

Returns CANMate_ERROR_SUCCESS if successful

- **int SetCANBaudRate(HANDLE hDev, char chBaudRate)**

  This command sets the CAN baud rate of the CANMate.

  Parameters :

  1. HANDLE hDev
     Handle of the USB port connected to CANMate device.
  2. char chBaudRate
     Baud rate to be set. This should be one of the following.

     BAUD_RATE33K
     BAUD_RATE50K
     BAUD_RATE80K
     BAUD_RATE83K
     BAUD_RATE100K
     BAUD_RATE125K
     BAUD_RATE200K
     BAUD_RATE250K
     BAUD_RATE500K
     BAUD_RATE625K
     BAUD_RATE800K
     BAUD_RATE1000K

  Return Value :

     Returns CANMate_ERROR_SUCCESS if successful.

- **int WriteCANMessage(HANDLE hDev, CANMsg* pMsg)**

  This function transmits a single CAN message. No ack is given for the transmitted message and hence the function does not wait.

  Parameters :

  1. HANDLE hDev
     Handle of the USB port connected to CANMate device.
  2. CANMsg*pMsg
     Pointer to a can message structure

  Return Value :

Returns number of bytes written

- **int StartReception (HANDLE hDev)**

  This function sets the CANMate to transfer the received messages to the PC application. CANMate firmware will not transfer the messages to PC unless this function is invoked.

  Parameters :

  1. HANDLE hDev
     Handle of the USB port connected to CANMate device.

  Return Value :

  Returns CANMate_ERROR_SUCCESS if successful.

- **int StopReception (HANDLE hDev)**

  This function instructs the CANMate firmware to stop transferring the received messages to the PC application. It is advised to call this function if the PC app is not listening for any CAN messages.

  Parameters :

  1. HANDLE hDev
     Handle of the USB port connected to CANMate device.

  Return Value :

  Returns CANMate_ERROR_SUCCESS if successful.

- **int SetNormalMode(HANDLE hDev)**

  This function sets the CANMate in “Normal” mode. This is the default mode in which CANMate powers up. This mode should be set for normal operation of CANMate.

  Parameters :

  1. HANDLE hDev
     Handle of the USB port connected to CANMate device.

  Return Value :

Returns CANMate_ERROR_SUCCESS if successful.

- **int SetLoopbackMode(HANDLE hDev)**

  This function sets the CAN controller inside CANMate in “Loopback” mode. In loopback mode transmitted messages from PC application will not be transmitted in the CAN bus, but will be returned as received messages. This mode is useful during application development as a testing and debugging aid.

  Parameters :

  1. HANDLE hDev
     Handle of the USB port connected to CANMate device.

  Return Value :

  Returns CANMate_ERROR_SUCCESS if successful.

- **int GetCurrentMode(HANDLE hDev, int* pData)**

  This function gets the current CAN controller operating mode.

  Parameters :

  1. HANDLE hDev
     Handle of the USB port connected to CANMate device.
  2. int* pData
     The current mode will be returned in this pointer. It can be one of the following values.

     ECAN_NORMAL_MODE
     ECAN_LOOPBACK_MODE

  Return Value :

  Returns CANMate_ERROR_SUCCESS if successful.

- **int GetCurrentBaudRate(HANDLE hDev, int* pData)**

  This function gets the current CAN controller baud rate.

  Parameters :

  1. HANDLE hDev

Handle of the USB port connected to CANMate device.

2. int* pData
   The current baud rate will be returned in this pointer. It can be one of the following values.

   BAUD_RATE33K
   BAUD_RATE50K
   BAUD_RATE80K
   BAUD_RATE83K
   BAUD_RATE100K
   BAUD_RATE125K
   BAUD_RATE200K
   BAUD_RATE250K
   BAUD_RATE500K
   BAUD_RATE625K
   BAUD_RATE800K
   BAUD_RATE1000K

Return Value :

Returns CANMate_ERROR_SUCCESS if successful.

- **int GetFirmwareVersion(HANDLE hDev, int* pData)**

  This function gets the current CANMate firmware version number. Version number is a 2 byte format with MSB representing the Major version and LSB the Minor version.

  Parameters :

  1. HANDLE hDev
     Handle of the USB port connected to CANMate device.
  2. int* pData
     The current firmware version number will be returned in this pointer.

  Return Value :

  Returns CANMate_ERROR_SUCCESS if successful.

- **typedef int (*EVNT_CALLBACK)(CANEvent *pnEvnt)**

  This is the prototype of event call back.

Parameters :

1. CANEvent *pnEvnt
   Pointer to a CANEvent structure.

Return Value :

Returns CANMate_ERROR_SUCCESS.

- **typedef int (*DATA_CALLBACK)(CANMsg *pMsg, int *nNumMsgs)**

  This is the prototype of data call back.

  Parameters :

  1. CANMsg *pMsg
     Pointer to a CANMsg structure. For receiving CAN Messages.
  2. int *nNumMsgs
     Number of messages will be returned in this pointer.

  Return Value :

  Returns CANMate_ERROR_SUCCESS.

CANMate Shared Object Structures

This section explains the structures used by the CANMate

1. **CAN Message Structure**

   Fields :

   1. bExtended : This field should be non zero for extended messages and 0 for standard messages
   2. chTmStmpH : Upper byte of time stamp field.
   3. chTmStmpL : Lower byte of time stamp field
   4. Time stamp is ignored for transmit messages
   5. EArbId1 : MSB containing bits 25 to 29 in the case of extended messages

and igonored for standard messages

6. EArbId0 : 3rd Byte in the case of extended messages and igonored for standard messages
7. SArbId1 : 2nd Byte in the case of extended messages and MSB containing bits 9 to 11 for standard messages
8. SArbId0 : 1st Byte for both extended and standard messages
9. DLC : Message length (8 maximum)

2. CAN Event Structure

Fields

chErr : The type of error
This is a bit field and following is the definition

```
1. #define UART_ERROR              0x01   // 0x0000 0001
2. #define CAN_TXERROR             0x02   // 0x0000 0010
3. #define CAN_BUS_OFF             0x04   // 0x0000 0100
4. #define CAN_BUF_OVERFLOW        0x08   // 0x0000 1000
5. #define CAN_TX_PASSIVE          0x10   // 0x0001 0000
6. #define CAN_RX_PASSIVE          0x20   // 0x0010 0000
7. #define CAN_ERR_WARNING         0x40   // 0x0100 0000
8. #define PC_BUF_OVERFLOW         0x80   // 0x1000 0000
```

chTxErrCnt : Transmit error counter value from the hardware
chRxErrCnt : Receive error counter value.

Note on Error Handling : Specific CAN errors CAN_BUS_OFF, CAN_TX_PASSIVE and CAN_RX_PASSIVE will put CAN controller in CANMate hardware into an irrecoverable error mode and will require closing and opening the device again from the application software.

** Refer CANMateDll.h for defenitions used in this document*