CCANet Class Definition

Generic CAN Device Driver Interface

VITAL Systems Inc.

www.VITALSYSTEM.com

Class CCANet

Generic CAN Device Driver Interface Class

CCANet is designed to provide a generic interface for CAN adapters (currently PCI and USB) developed by VSi. The CCANet is the base class. The derived classes, e.g. CPCINet and CUSBNet, provide the implementation. These classes are thread safe.

Constructor

The constructor takes one parameter as the channel number. All methods defined by the class use this channel number to comminute with the hardware. Channel numbers start from zero.

Methods

BOOL GetMsg(ULONG& Identifier, UCHAR* Msg, UCHAR& MsgSize)

Parameters

Identifier: 11 bit identifier of the received message
Msg: Message body (max 8 bytes)
MsgSize: Size of received message

Operation

This method retrieves a message from the drivers FIFO receive queue. Each channel has its own receive FIFO queue. This is non-blocking call.

Return value

GetMsg returns true is there was atleast one message in the receive queue. The contents of the message are returned in the parameters. If the queue is empty, the method returns false.

BOOL SendMsg(ULONG Identifier, UCHAR* Msg, UCHAR MsgSize)

Parameters

Identifier: 11 bit identifier of the message
Msg: Message body (max 8 bytes)
MsgSize: Size of message to transmit

Operation

SendMsg function transmits one CAN message. The function first checks if the CAN controller chip is ready. If the buffers are available, the data is transferred to the CAN chip for delivery. The caller should retry if the function returns false, because the buffers may still be occupied by the previous send message operation. For PCI adapters, there is no transmit queue inside the driver.

Return value
The method returns true if the CAN chip accepts the message to transmit. Returns false if no buffer is available.

BOOL ResetMsgQue()

Parameters
None

Operation
This method empties the FIFO receive queue in the driver associated with the channel.

Return value
Returns true if successful.

---

void BlinkCommsLED(UCHAR ticks)

Parameters
Ticks: On Duration

Operation
Turns on the yellow LED on bracket for the given duration. Each tick is approx. 10mSec duration. For regular transmit and receive operations, the driver controls the LEDs.

Return value
None

---

void BlinkErrorLED(UCHAR ticks)

Parameters
Ticks: On Duration

Operation
Turns on the Red LED on bracket for the given duration. Each tick is approx. 10mSec duration. For regular transmit and receive operations, the driver controls the LEDs.

Return value
None

BOOL GetBoardCount(UCHAR &Count)

Parameters
Count: returns the number of boards in this variable

Operation
The methods returns the number of CAN adapters installed in the PC.

Return value
Returns true if successful.

BOOL SetBusParam(UCHAR BTr0, UCHAR BTr1,
UCHAR ACr0, UCHAR ACr1,
UCHAR ACr2, UCHAR ACr3,
UCHAR AMr0, UCHAR AMr1,
UCHAR AMr2, UCHAR AMr3)

BOOL SetBusParam(UCHAR BTr0, UCHAR BTr1,
ULONG Ident1, ULONG Mask1,
ULONG Ident2, ULONG Mask2)

Parameters
BTr0, BTr1: Bus Timming registers (Refer to Philips SJA1000 CAN controller specs for bit definitions.)

ACr0..ACR3: Acceptance Code Registers
AMr0..AMr3: Acceptance Mask Registers

Ident1, Ident2 Acceptance Identifier
Mask1, Mask2 Acceptance Mask

Operation
These methods set the bus speed as well as bus filters parameters. If you are familiar with the Philips CAN chip and require full control of the CAN chip, you can use the method with ACR and AMR registers. These register values are directly loaded into the CAN chip. For detailed description of these register, please refer to Philips SJA1000 CAN controller specifications.

For simple 11-bit identifier protocol, you can use the second function that calculates the ACR and AMR values automatically. With this method, you can set up two identifiers (Ranges) to accept. The mask register specifies the don't care bits. A 1 in mask bit indicate a don't care bit in the corresponding indentifier bit.

BTR0 and BTR1 set up the bus speed. Following are the values of common bus speeds (CAN chip running at 16Mhz):

BusSpeed	**125 kbps**	**250 kbps**	**500 Kbps**	**1 Mbps**
BTR0	3	1	0	0
BTR1	0x1C	0x1C	0x1C	0x14

Examples:

SetBusParam(3, 0x1C, 0x780, 0, 0x780, 0)
The above method sets the speed to 125 kbps. The CAN chip will only accept messages with the identifier value of 0x780.

SetBusParam(1, 0x1C, 0x780, 0xF, 0x080, 0x700)
The above method sets the speed to 250 kbps. The CAN chip will accept messages with the identifier value of 0x780 thru 0x78F and 0x080 thru 0x780.

SetBusParam(0, 0x1C, 0, 0xFFF, 0, 0xFFF)
The above method sets the speed to 500 kbps. The CAN chip will accept all messages on the network.

Return value
Returns true if successful.

BOOL GetErrorCounters(UINT &NetError, UINT &TxOverRuns, UINT &RxOverRuns)

Parameters

NetError	General Bus error Counter
TxOverRun	Transmit queue overrun
RxOverRun	Receive Queue overrun

Operation
This method retrieves the error counters associated with the channel. The TxOverRun counter is not valid in case of PCI CAN adapter.

Return value
Returns true if successful.

BOOL ResetErrorCounters()

Parameters
None

Operation
Reset error counters associated with the channel.

Return value
Returns true if successful.

BOOL DisableCAN()

Parameters
None

Operation
This method sets the CAN chip associated with the channel to BUS-OFF state. The CAN chip will not receive or transmit any message. The SetBusParam method can bring the CAN chip back to operating mode.

Return value
Returns true if successful.
