VCAT SDK

EtherCAT Master

Vecow

1.0.0 Edition 20210319

Record of Revision

Version	Date	Page	Description	Remark
1.00	2021/03/19	All	Official Release	

Disclaimer

This manual is released by Vecow Co., Ltd. for reference purpose only. All product offerings and specifications are subject to change without prior notice. It does not represent commitment of Vecow Co., Ltd. Vecow shall not be liable for direct, indirect, special, incidental, or consequential damages arising out of the use of the product or documentation or any infringements upon the rights of third parties, which may result from such use.

Declaration of Conformity

FCC

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy, and if it is not installed and used in accordance with the instruction manual, it may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

CE

The products described in this manual complies with all applicable European Union (CE) directives if it has a CE marking. For computer systems to remain CE compliant, only CE-compliant parts may be used. Maintaining CE compliance also requires proper cable and cabling techniques.

Copyright and Trademarks

Table of Contents

CHAPTER 1 INSTALL THE SOFTWARE 1

1.1 How to install the software 1

1.2 Language support 1

CHAPTER 2 DLL FUNCTIONS 3

2.1 Function Format 3

2.2 Flow chart of application implementation 5

2.3 Software Overvi ew and DLL Function 6

2.4 Error Code Table 12

1

INSTALL THE SOFTWARE

1.1 How to install the software

Please install following software :

1. WinPcap_4_1_3.exe
2. VC_redist.x32.exe (for Window 32bit distribution)
   VC_redist.x64.exe (for Window 10 64bit distribution)
3. VCAT Configuration Tool (unzip files)
4. VCAT Master Dll (unzip files)

1.2 Language support

The EtherCAT master software library is a DLL used with WinXP/7 and up. You can use these DLL with any Windows integrating development environment that can call Windows DLLs.

Building applications with the software library
The EtherCAT master function reference topic contains general information about building EtherCAT applications, describes the nature of the SDK files used in building EtherCAT applications, and explains the basics of making applications using the following tools :

Applications tools
Microsoft Visual C/C++
Microsoft Visual C#

If you are not using one of the tools listed, consult your development tool reference manual for details on creating applications that call DLLs.

EtherCAT Master Windows libraries

The master SDK for Windows function library is a DLL called VeECM.dll. Since a DLL is used, the functions are not linked into the executable files of applications. Only the information about the EtherCAT functions in the master import libraries is stored in the executable files. Import libraries contain information about their DLL-exported functions. They indicate the presence and location of the DLL routines. Depending on the development tools you are using, you can make your compiler and linker aware of the DLL functions through import libraries or through function declarations.

Refer to Table 1 to determine to which files you need to link and which to include in your development to use the functions in VECM.dll.

Header Files and Import Libraries for Different Development Environments		
Language	Header File	Import Library
Microsoft Visual C/C++	VCAT.h	VCAT.lib
Microsoft Visual C#	VCAT.cs	N/A
EtherCAT Master SDK	O	O

2

DLL FUNCTIONS

2.1 Function Format

Function format
Every master function is consist of the following format :

Status = function name (parameter 1, parameter 2, … parameter n);

Each function returns a value in the Status global variable that indicates the success or failure of the function. A returned Status equal to zero that indicates the function executed successfully. A non-zero status indicates failure that the function did not execute successfully because of an error, or executed with an error.

Note : Status is a 32-bit unsigned integer.

The first parameter to almost every function is the parameter MasterID which is located the driver of network port.

Note : MasterID is always (0) currently supported only one EtherCAT master.

These topics contain detailed descriptions of each master function. The functions are arranged alphabetically by function name. Refer to master function reference for additional information.

Variable data types
Every function description has a parameter table that lists the data types for each parameter. The following sections describe the notation used in those parameter tables and throughout the manual for variable data types.

Name	Description	Range	C/C++	C#
u8	8-bit ASCII character	0 to 255	Unsigned char	byte
I16	16-bit signed integer	-32,768 to 32,767	short	short
U16	16-bit unsigned integer	0 to 65,535	unsigned short for 32-bit compilers	ushort
I32	32-bit signed integer	-2,147,483,648 to 2,147,483,647	long	int
U32	32-bit unsigned integer	0 to 4,294,967,295	Unsigned long	uint
F32	32-bit single-precision floating-point value	-3.402823E+38 to 3.402823E+38	float	float
F64	64-bit double-precision floating-point value	-1.797683134862315E+308 to 1.797683134862315E+308	double	double

Programming language considerations

Apart from the data type differences, there are a few language-dependent considerations you need to be aware of when you use the master API. Read the following sections that apply to your programming language.

Note : Be sure to include the declaration functions of master prototypes by including the appropriate VECM header file in your source code. Refer to Building Applications with the EtherCAT master software library for the header file appropriate to your compiler.

C/C++

For C or C++ programmers, parameters listed as Input/Output parameters or Output parameters are pass-by-reference parameters, which means a pointer points to the destination variable should be passed into the function. For example, the Read Port function has the following format :

RTN_ERR = GetParameter (U16_T MasterId, U16_T ParaNum, I32_T *ParaData)

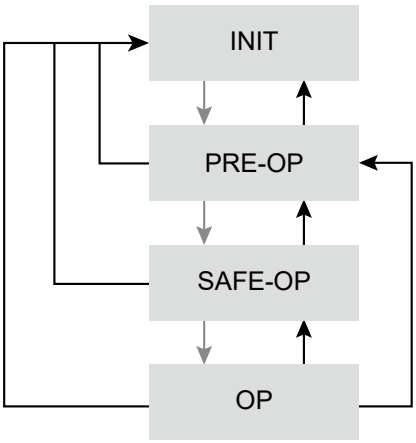
Where MasterId and port are input parameters, and data is an output parameter. Consider the following example :

U16_T MasterID, ParaNum;
I32_T *ParaData,
I32 RTN_ERR

2.2 Flow chart of application implementation

State Machine

According to the EtherCAT standard document (ETG.1000.6), the EtherCAT master station must have a state machine (EtherCAT State Machine, referred to as ECM) to handle the transition from the initialization state to the operable state between the master station and each slave station (EC-Slaves) work process. The state diagram is shown in the figure below, which contains four states :



1. INIT status
2. PREOP status
3. SAFEOP status
4. OP status

In the application of EtherCAT, before the industrial control process, the state of the state machine must be changed from the "INIT" state to the "OP" state. The implicit meaning of this action is to initialize the EC-Master network and EC-Slaves module. Group related settings, the content of which is based on the ENI (EtherCAT Network Information) file output by the utility.

In each state, the services provided by EC-Master and the operating instructions that EC-Slaves can accept are shown in the following table :

Status	Master	Slave
INIT	• Cyclic function start • No ProcessData (To slaves) • No Mailbox (To slaves)	Can't be control by master
PREOP	• Cyclic function start • Mailbox services start. (to slaves) • SDO start (to slave) • No process data Object. (to slaves)	Mailbox service start. (to master)
SAFEOP	• Cyclic function start • Mailbox services start. (to slaves) SDO start. • Rx process data object services start. • No Tx ProcessData services.	Mailbox services start. (to master) Rx Pdo services start. (process data object) Tx Pdo services not start.
OP	• Cyclic function start • Mailbox services start. (To slaves) • -SDO start. • Rx Process Data Input start. • Tx Process Data Output start.	Mailbox services start. (to master) Update Rx Pdo. Tx Pdo start.

2.3 Software Overvi ew and DLL Function

DLL list

	Function Name	Description
1	GetVersion()	Get Dll version
2	StartDriver()	Start EtherCAT master driver
3	StopDriver()	Stop EtherCAT Master driver
4	SetParameter()	Write Sdo data to slave device.
5	GetParameter()	Read Sdo data from slave device.
6	GetNetState()	Get network state.
7	StartNet()	Start network communication.
8	StartNetEx()	Start network.(add)
9	StopNet()	Stop network communication.
10	GetSlaveCount()	Get the number of EtherCAT slave.
12	GetSlaveState()	Get slave device state.
13	GetStateError()	Get master state machine error.
14	GetErrorMsg()	Get Last Error Message.
15	SetDo()	Set digital output status.
16	GetDo()	Get digital output status.
17	GetDi()	Get digital input.
18	RWSlaveProcessImage()	Read/write process data object.

The C/C++ data type used by the API is defined in v_type.h, which is described in the following table :

Data Type	C/C++	description	byte	range
BOOL	int	bool type	4	0 : False, 1 : True
U8	unsigned char	unsigned	1	0 ~ 255
U16	unsigned short	unsigned	2	0 ~ 65535
U32	unsigned int	unsigned	4	0 ~ 4294967295
U64	unsigned __int64	unsigned	8	0 ~ 18446744073709551615
I8	char	signed	1	-128 ~ 127
I16	short	signed	2	-32768 ~ 32767
I32	int	signed	4	-2147483648 ~ 2147483647
I64	__int64	Signed	8	-9223372036854775808 ~ 9223372036854775807
F32	float	32bit	4	15 digits after effective decimal
F64	double	64bit	8	15 digits after effective decimal
RTN	int	error code	4	-2147483648 ~ 2147483647

U32_T FNTYPE GetVersion()
Description : Get the current SDK version information.
Parameters : U32 * version : return the version of dll.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.
Constrain : No Constrain.

RTN FNTYPE StartDriver()
Description : Start EtherCAT master driver.
Parameters : No parameters required.
Return Value : Return error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

Void FNTYPE Stop Driver();
Description : Stop EtherCAT master driver.
Parameters : No parameters required.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE SetParameter(U16 MasterId, U16 ParaNum, I32 ParaData)
Description : Set Sdo data to slave device.
Parameters : U16 MasterId : always 0. ParaNum : Sdo Number
I32 ParaData : Sdo data.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE GetParameter(U16 MasterId, U16 ParaNum, I32 *ParaData)
Description : Get Sdo data from slave device.
Parameters : U16 MasterId : always 0. ParaNum : Sdo Number
I32 ParaData : Sdo data.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE StartNet(U16 MasterId, const char *ConfigurationFile, I32 TimeoutMs)
Description : Switch all of slave device state machine to OP mode.
Parameters : U16 MasterId : always 0.
const char * : ENI file path.(generated by VCAT's configuration tool)
I32 TimeoutMs : wait state to operation state.
If value = -1 no wait, just start communication and you can check state by call GetNetworkState().
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE StartNetEx(U16 MasterId, const char *ConfigurationFile, U32 Option, I32_T TimeoutMs)
Description : Switch all of slave device state machine to OP mode.
Parameters : U16 MasterId : always 0.
const char * : ENI file path.(generated by VCAT's configuration tool)
I32 TimeoutMs : wait state to operation state.
If value = -1 no wait, just start communication and you can check state by call GetNetworkState().
Option : 1 : Windows.
0 : RTX.(Reserved)
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE StopNet(U16 MasterId, I32 TimeoutMs)
Description : Switch all of slave device state machine to INIT mode.
Parameters : U16 MasterId : always 0.
I32 TimeoutMs : wait state to operation state. if value = -1 no wait.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE GetSlaveCount(U16 MasterId, U32 *Count)
Description : Get the current master connected slave number.//be called after StartNetwork()
Parameters : U16 MasterId : always 0.
I32 *Count : slave quantity.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE GetNetState(U16 MasterId, U16 *State);
Description : Get the current network status.
Parameters :
U16 MasterId : always 0.
State : master's state machine status.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE GetSlaveState(U16 MasterId, U16 SlaveIndex, U8 *StateArr, U16 *ArrLen)
Description : Get slave's state machine status.
Parameters :
U16 MasterId : always 0.
SlaveIndex : master's state machine status.
ArrLen : connected slave count.
U8 StateArr : Get state machine status from all connected slave devices
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE GetStateError(U16 MasterId, I32 *Code)
Description : Get the network state error code.
Parameters : U16 MasterId : always 0.
I32 *Code : Error Code.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE GetErrorMsg(U16_T MasterId, char *ErrMsg)
Description : Get EtherCAT master error code.
Parameters :
U16 MasterId : always 0.
Char * ErrMsg : get master error message char array.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

//* Application functions ****
// <<< non-synchronized DIO functions, set DO value to memory only. >>>
RTN FNTYPE SetDo(U16 MasterId, U16 SlaveAddr, U16 Offset, U16 SizeByte, const U8 *DoData)**
Description : write data to Tx process data object.
Parameters :
U16 MasterId : always 0.
SlaveAddr : slave device index (start from 0)
Offset : offset byte.
SizeByte : size of bytes.
const U8 *DoData : byte array or pointer.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE GetDo(U16_T MasterId, U16_T SlaveAddr, U16_T Offset, U16_T SizeByte, U8_T *DoData)
Description : read data from Tx process data object.
Parameters :
U16 MasterId : always 0.
SlaveAddr : slave device index (start from 0)
Offset : offset byte.
SizeByte : size of bytes.
const U8 *DoData : byte array or pointer.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE GetDi(U16_T MasterId, U16_T SlaveAddr, U16_T Offset, U16_T SizeByte, U8_T *DiData)
Description : read data from Rx process data object.
Parameters :
U16 MasterId : always 0.
SlaveAddr : slave device index (start from 0)
Offset : offset byte.
SizeByte : size of bytes.
const U8 *DiData : byte array or pointer.
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

RTN FNTYPE RWSlaveProcessImage(U16 MasterId, U16 SlaveIndex , U16 RW, U16 Offset, byte *Data, U16 Size);
Description : read/write data from/to process data object.
Parameters :
U16 MasterId : always 0.
  SlaveIndex : slave device index (start from 0)
  Offset : offset byte.(refer to slave provider)
U8 *Data = read/write data array/pointer;
  Size = size of bytes
Return Value : error code.

The call function successfully returns "ECERR_SUCCESS" (0)
Otherwise, the function call fails and returns an error code, which is defined in the VErrors.h header file.

2.4 Error Code Table

Error Code	Symbolic Name	Description
0	No Error.	No Error Occurred.
-1	ECERR_WIN32_ERROR.	
-2	ECERR_NIC_INDEX	NIC index invalid.
-3	ECERR_MASTER_ID	MasterId invalid.
-4	ECERR_IPC_TIMEOUT	IPC timeout.
-5	ECERR_CREATE_SERVICE	Only first instance can create remote service
-6	ECERR_CREATE_SOCKET	Create socket failed.
-7	ECERR_SERVER_CONNECT_FAILED	Could not establish connection
-8	ECERR_DRIVER_RESPONSE_TIMEOUT	Driver response timeout.
-9	ECERR_START_NETWORK	Start network error, check error message.
-10	ECERR_FILE_NOT_FOUND	File not found or File open failed.
-11	ECERR_DATA_BUFFER_SIZE	Data buffer size invalid (too big).
-12	ECERR_ESC_EEPROM_TIMEOUT	ESC EEPROM access timeout
-13	ECERR_ESC_EEPROM_BUSY	ESC EEPROM busy.
-14	ECERR_ESC_EEPROM_ERROR	ESC EEPROM error
-15	ECERR_SLAVE_COUNT	Slave count invalid, slave count must > 0
-16	ECERR_DATA_LEN	Data length invalid, length must > 0 or data length over limitation
-17	ECERR_LOAD_NETWORK_DESC_FILE	Open/load network file failed. Please check the path of ENI file.
-18	ECERR_MASTER_ALREADY_CREATED	Already create master, close it first.
-19	ECERR_TOO_MUCH_INIT_CMD	To much master init commands.
-20	ECERR_INIT_CMD_DATA_SIZE	INIT command size to large.
-21	ECERR_FILE_NAME_LEN	Configuration file name' length too long.
-22	ECERR_DRIVER_NOT_FOUND	1.Hardware not install, 2.Hardware driver not install, 3.Realtime driver not found/open failed
-23	ECERR_WAIT_STATE_TIMEOUT	Wait operation state timeout, please check slave's current state.
-24	ECERR_PROC_IMG_DATA_SIZE	Proceess image data command size
-25	ECERR_CYCLIC_CONFIG	Cyclic info error in configuration file or master not support
-26	ECERR_RTX_LIB_NOT_FOUND	RTX Library not exist
-28	ECERR_LOAD_RTX_LIB	Load RTX library failed
-29	ECERR_READ_ENI_ERROR	Parser ENI failed, Please check ENI file.
-30	ECERR_EC_NIC_NOT_FOUND	Scan NIC for ESC not found.
-31	ECERR_RT_DATA_BUFFER	Data length to large, data buffer overflow.

Vecow

For further support information, please visit www.vecow.com