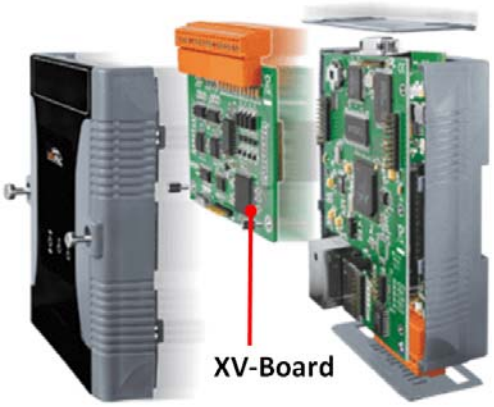
LinPAC-22xx/ 52xx Series

XV-Board API User Manual

V1.1 May 2020



Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assume no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademarks

Names are used for identification purposes only and may be registered trademarks of their respective companies.

Table of Contents

1. Introduction

This manual is intended to be used as a reference for users who need to use API functions to communicate with XV-Boards inserted into an LP-22xx/52xx series controller via the Modbus protocol.

1.1. What is the I/O Expansion Bus?

The LinPAC-22xx/52xx series provides an I/O expansion bus which can be used to implement various I/O functions, such as D/I, D/O, A/I, A/O, A/D, D/A, Timer/Counter, and other I/O functions.

Model	OS	CPU	Flash	SDRAM	Ethernet	VGA Resolution	USB	I/O Slot	Audio
LP-5231	Linux kernel 3.2.14	AM3354, 1 GHz	512 MB	512 MB	1	1024 x 768	1	I/O expansion board optional	None
LP-5231M									
LP-5231PM-3GWA									
LP-2241					2		2		

However, only one XV-Board can be inserted into each PAC at a time. For more information related to the XV-Board series, refer to:
http://www.icpdas.com/en/product/guide+Remote_I_O_Module_and_Unit+PAC_%EF%BC%86amp;_Local_I_O_Modules+XV-board

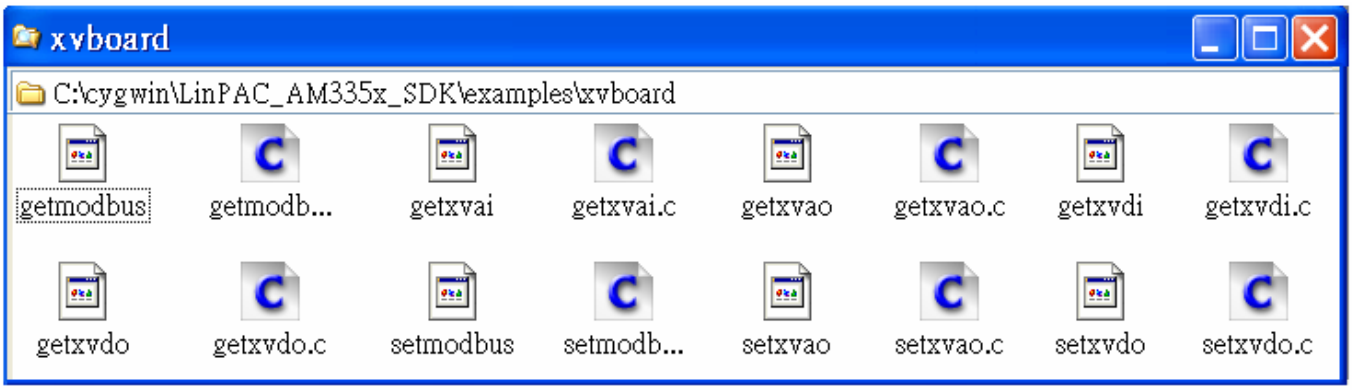
1.2. Specifications

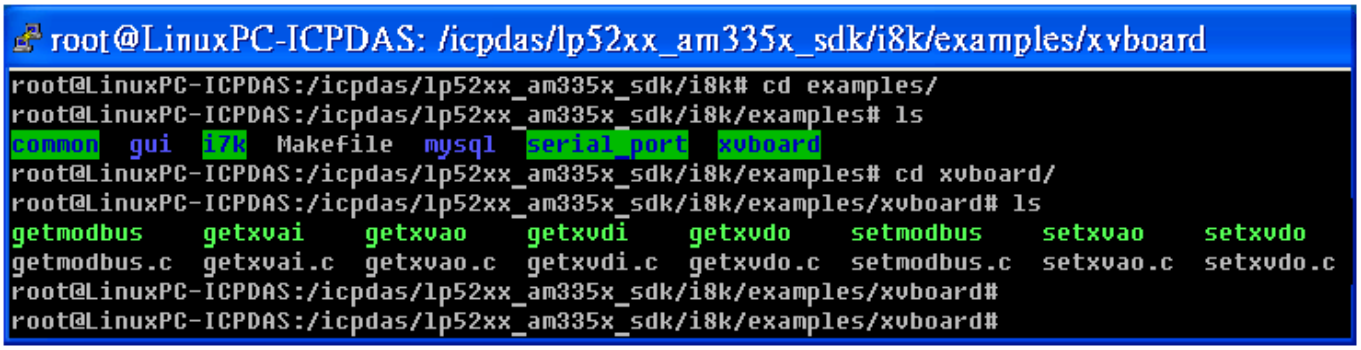
For more detailed specifications related to specific XV-Boards and the Modbus register table, refer to "**xv-board_user_manual**" file:
http://www.icpdas.com/web/product/download/pac/linux/lp-5000/document/manual/xv-board_user_manual_en.pdf

1.3. Demo programs

ICP DAS provides a range of demo programs that can be used to implement a variety of functions on the LP-22xx/52xx controller. To utilize these demo programs, transfer the files to the LP-22xx/52xx controller from the LinPAC AM335X SDK.

The demo programs can be found in either the **C:\cygwin\LinPAC_AM335x_SDK\examples\xvboard** or the root@LinuxPC-ICPDAS:**/icpdas/lp52xx_am335x_sdk/i8k/examples/xvboard/** folder.





2. XV-Board API functions

The following is an overview of the API functions used by the LP-22xx/52xx series controller that contains an XV-Board.

2.1. modbusRequest

Description:

This function is used to send a Modbus protocol request.

Syntax:

[C]

```
int   modbusRequest (char cPort, char cNetID, char cFunction, WORD wAddr,
                     WORD wCount, char szBuf, WORD wBufLen, WORD wTimeout, WORD *wT)
```

Parameters:

cPort: [**Input**] The number of the COM port for the XV-Board. 1=COM1=/dev/ttyO1.

cNetID: [**Input**] The NetID for the device.
The default slave address for the XV-Board is '1'.

cFunction: [**Input**] The Modbus RTU protocol function code.

wAddr: [**Input**] The channel address.
Note: Input the channel address mapping in **decimal** format.

wCount: [**Input**] The number of channels.

szBuf[ ]: [**Input/Output**] Used to set or read back the value from a function code.

wBufLen: [Input] The length of the szBuf[ ].

wTimeout: [**Input**] The Timeout setting. The normal value is 100 milliseconds.

*wT: [**Output**] The total duration of the send/receive interval. Unit=1 ms.

Return Values:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 3: "Error Codes" for details of other returned values.

Example:

```
int RetValue, timeout=100, count=1, netid=1;
int comport=COM1;       /* Refer to [Remark 1] */
int function=1;         /* Refer to [Remark 2] */
int addr=33;            /* Refer to [Remark 3] */
char szBuf[80];
DWORD baudrate=115200;
WORD wT;
/* Open the device file */
RetValue=Open_Com(comport, baudrate, Data8Bit, NonParity, OneStopBit);
if (RetValue > 0) {
    printf("open COM%d failed!\n",comport);
    return FAILURE;
}
RetValue=modbusRequest(comport, netid, function, addr, count, szBuf, sizeof(szBuf), timeout,
&wT);
Close_Com(comport);
if (RetValue==0) {
    if((function==1)||(function==2))
        printf("%d", szBuf[0]);
    else {
        RetValue=(szBuf[0]<<8)|szBuf[1];
        printf("%ld", RetValue);
    }
}
```

Remarks:

[1] Refer to the communication port number in the table below when sending the Modbus protocol request.

'**cPort**' parameter for the XV-Board module: 1=COM1=/dev/ttyO1.

'**cPort**' parameter for the RS-485 serial port: 2=COM2=/dev/ttyO2, 5=COM5=/dev/ttyO5.

Note: Remote I/O modules can be remotely controlled through an **RS-485** serial bus.

Refer to Chapter 4.1.2: "DIO" for details of demonstration details.

Device Name	Definition in the LP-22xx/52xx SDK	Description	Default Baud Rate
-	**/dev/ttyO1 or COM1**	Used for internal communication with the XV-Board	115200
-	Console port	RS-232 (RxD, TxD and GND); Non-isolated	115200
ttyO4	/dev/ttyO4 or COM4	RS-232 (RxD, TxD and GND); Non-isolated	9600
ttyO2	**/dev/ttyO2 or COM2**	RS-485 (Data+, Data-); Non-isolated	9600
ttyO5	**/dev/ttyO5 or COM5**	RS-485 (Data+, Data-); 2500 VDC isolated	9600

[2] Modbus RTU protocol function codes.

Function Code	Description	Input
01	Reads the status of the Digital Output (ReadCoilStatus)	1
02	Reads the status of the Digital Input (ReadInputStatus)	2
03	Reads the Analog Output channels (ReadHoldingRegister)	3
04	Reads the Analog Input channels (ReadInputRegister)	4
05	Writes to a single Digital Output channel (ForceSingleCoil)	5
06	Writes to a single Analog Output channel (PresetSingleRegister)	6
15	Writes to multiple Digital Output channels (ForceMultipleCoil)	15
16	Writes to multiple Analog Output channels (PresetMultipleRegister)	16

[3] This is the input value reference address mapping (Modbus register table) for the '**wAddr**' parameter.

More information regarding address mapping can be found at "**xv-board_user_manual**" file:

http://www.icpdas.com/en/download/show.php?num=888&model=LP-5231M

2.2. modbusRequestXV

Description:

This function is used to send a Modbus protocol request in an LP-22xx/52xx system.

Syntax:

[C]

```
int   modbusRequestXV (char cPort, char cFunction, WORD wAddr, WORD wCount,
                       char szBuf, WORD wBufLen)
```

Parameters:

cPort:	[**Input**] The number of the COM port for the XV-Board. 1=COM1=/dev/ttyO1.
cFunction:	[**Input**] The Modbus RTU protocol function code.
wAddr:	[**Input**] The channel address. **Note**: Input the channel address mapping in **decimal** format.
wCount:	[**Input**] The number of channels.
szBuf[ ]:	[**Input/Output**] Used to set or read back the value from a function code.
wBufLen:	[Input] The length of the szBuf[ ].

Return Values:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 3: "Error Codes" for details of other returned values.

Example:

```
int RetValue, count=1;
int comport=COM1;       /* Refer to [Remark 1] */
int function=1;         /* Refer to [Remark 2] */
int addr=33;            /* Refer to [Remark 3] */
char szBuf[80];
DWORD baudrate=115200;
WORD wValue;
switch(function) {
    case FC05ForceSingleCoil:
    case FC15ForceMultipleCoil:
        szBuf[0]=wValue & 0xff;
        break;
    case FC06PresetSingleRegister:
    case FC16PresetMultipleRegister:
        szBuf[0]=(wValue >> 8) & 0xff;
        szBuf[1]=wValue & 0xff;
        break;
    default:
        return FAILURE;
}

/* Open the device file */
RetValue=Open_Com(comport, baudrate, Data8Bit, NonParity, OneStopBit);
if (RetValue >0) {
    printf("open COM%d failed!\n",comport);
    return FAILURE;
}

RetValue=modbusRequestXV (comport, netid, function, addr, count, szBuf, sizeof(szBuf), timeout,
&wT);
Close_Com(comport);
```

Remarks:

[1] Refer to the communication port number in the table below to send the Modbus protocol request.

'**cPort**' parameter for the XV-Board module: 1=COM1=/dev/ttyO1.

'**cPort**' parameter for the RS-485 serial port: 2=COM2=/dev/ttyO2, 5=COM5=/dev/ttyO5.

Note: Remote I/O modules can be remotely controlled through an **RS-485** serial bus.

Refer to Chapter 4.1.2: "DIO" for details of demonstration details.

Device Name	Definition in the LP-22xx/52xx SDK	Description	Default Baud Rate
-	**/dev/ttyO1 or COM1**	Used for internal communication with the XV-Board	115200
-	Console port	RS-232 (RxD, TxD and GND); Non-isolated	115200
ttyO4	/dev/ttyO4 or COM4	RS-232 (RxD, TxD and GND); Non-isolated	9600
ttyO2	**/dev/ttyO2 or COM2**	RS-485 (Data+, Data-); Non-isolated	9600
ttyO5	**/dev/ttyO5 or COM5**	RS-485 (Data+, Data-); 2500 VDC isolated	9600

[2] Modbus RTU protocol function codes.

Function Code	Description	Input
01	Reads the status of the Digital Output (ReadCoilStatus)	1
02	Reads the status of the Digital Input (ReadInputStatus)	2
03	Reads the Analog Output channels (ReadHoldingRegister)	3
04	Reads the Analog Input channels (ReadInputRegister)	4
05	Writes to a single Digital Output channel (ForceSingleCoil)	5
06	Writes to a single Analog Output channel (PresetSingleRegister)	6
15	Writes to multiple Digital Output channels (ForceMultipleCoil)	15
16	Writes to multiple Analog Output channels (PresetMultipleRegister)	16

[3] This is the input value reference address mapping (Modbus register table) for the '**wAddr**' parameter.

More information regarding address mapping can be found at "**xv-board_user_manual**" file:

http://www.icpdas.com/en/download/show.php?num=888&model=LP-5231M

2.3. getXVBitBack

Description:

This function is used to read back the channel status value from a specified DO module in an LP-22xx/52xx system using the Modbus protocol.

Syntax:

[C]

```
int    getXVBitBack (char cPort, WORD wAddr, WORD wCount, char szBuf, WORD wBufLen)
```

Parameters:

cPort:	[**Input**] The number of the COM port for the XV-Board. 1=COM1=/dev/ttyO1.
wAddr:	[**Input**] The channel address.
	Note: Input the channel address mapping in **decimal** format.
wCount:	[**Input**] The number of channels.
szBuf[ ]:	[**Output**] Used to read back the Digital Output value.
wBufLen:	[**Input**] The length of the szBuf[ ].

Return Values:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 3: "Error Codes" for details of other returned values.

Example:

```
int RetValue, comport=COM1, count=4;
int addr=0;             /* Refer to [Remark 1] */
char szBuf[80];
DWORD baudrate=115200;
/* Open the device file */
RetValue=Open_Com(comport, baudrate, Data8Bit, NonParity, OneStopBit);
if (RetValue > 0) {
    printf("open COM%d failed!\n",comport);
    return FAILURE;
}
memset(szBuf, 0, sizeof(szBuf));
RetValue=getXVBitBack(comport, addr, count, szBuf, sizeof(szBuf));
Close_Com(comport);
if (RetValue==0) {
    if(count<=8) {
        printf("%d", szBuf[0]);
    } else if (count<=16){
        dwValue=(szBuf[1]<<8)|szBuf[0];
        printf("%u", dwValue);
    } else {
        dwValue=(szBuf[3]<<24)|(szBuf[2]<<16)|(szBuf[1]<<8)|szBuf[0];
        printf("%lu", dwValue);
    }
}
```

Remarks:

[1] This is the input value reference address mapping (Modbus register table) for the '**wAddr**' parameter.

More information regarding address mapping can be found at "**xv-board_user_manual**" file:

http://www.icpdas.com/en/download/show.php?num=888&model=LP-5231M

2.4. getXVBit

Description:

This function is used to read the channel status value from a specified DI module in an LP-22xx/52xx system using the Modbus protocol.

Syntax:

[C]

```
int    getXVBit(char cPort, WORD wAddr, WORD wCount, char szBuf, WORD wBufLen)
```

Parameters:

cPort:	[**Input**] The number of the COM port for the XV-Board. 1=COM1=/dev/ttyO1.
wAddr:	[**Input**] The channel address.
	Note: Input the channel address mapping in **decimal** format.
wCount:	[**Input**] The number of channels.
szBuf[ ]:	[**Output**] The current status of the Digital Input.
wBufLen:	[**Input**] The length of the szBuf[ ].

Return Values:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 3: "Error Codes" for details of other returned values.

Example:

```
int RetValue, comport=COM1, count=4;
int addr=32;                    /* Refer to [Remark 1] */
char szBuf[80];
DWORD baudrate=115200;
/* Open the device file */
RetValue=Open_Com(comport, baudrate, Data8Bit, NonParity, OneStopBit);
if (RetValue > 0) {
    printf("open COM%d failed!\n",comport);
    return FAILURE;
}
memset(szBuf, 0, sizeof(szBuf));
RetValue=getXVBit(comport, addr, count, szBuf, sizeof(szBuf));
Close_Com(comport);
if (RetValue==0) {
    if(count<=8) {
        printf("%d", szBuf[0]);
    } else if (count<=16){
        dwValue=(szBuf[1]<<8)|szBuf[0];
        printf("%u", dwValue);
    } else {
        dwValue=(szBuf[3]<<24)|(szBuf[2]<<16)|(szBuf[1]<<8)|szBuf[0];
        printf("%lu", dwValue);
    }
}
```

Remarks:

[1] This is the input value reference address mapping (Modbus register table) for the '**wAddr**' parameter.

More information regarding address mapping can be found at "**xv-board_user_manual**" file:

http://www.icpdas.com/en/download/show.php?num=888&model=LP-5231M

2.5. getXVRegBack

Description:

This function is used to read back the voltage float value from a specified AO module in an LP-22xx/52xx system using the Modbus protocol.

Syntax:

[C]

```
int    getXVRegBack(char cPort, WORD wAddr, WORD wCount, char szBuf, WORD wBufLen)
```

Parameters:

cPort:	[**Input**] The number of the COM port for the XV-Board. 1=COM1=/dev/ttyO1.
wAddr:	[**Input**] The channel address.
	Note: Input the channel address mapping in **decimal** format.
wCount:	[**Input**] The number of channels.
szBuf[ ]:	[**Output**] Used to read back the Analog Output value.
wBufLen:	[**Input**] The length of the szBuf[ ].

Return Values:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 3: "Error Codes" for details of other returned values.

Example:

```
int RetValue, comport=COM1, count=1;
int addr=33;          /* Refer to [Remark 1] */
char szBuf[80];
DWORD baudrate=115200;
/* Open the device file */
RetValue=Open_Com(comport, baudrate, Data8Bit, NonParity, OneStopBit);
if (RetValue > 0) {
      printf("open COM%d failed!\n",comport);
      return FAILURE;
}
memset(szBuf, 0, sizeof(szBuf));
RetValue=getXVRegBack(comport, addr, count, szBuf, sizeof(szBuf));
Close_Com(comport);
if (RetValue==0) {
      dwValue=(szBuf[0]<<8)|szBuf[1];
      printf("%u", dwValue);
}
```

Remarks:

[1] This is the input value reference address mapping (Modbus register table) for the '**wAddr**' parameter.

More information regarding address mapping can be found at "**xv-board_user_manual**" file:

http://ftp.icpdas.com.tw/pub/cd/linpac/napdos/lp-5000/lp-52xx/lp-5231/user_manual/

2.6. getXVReg

Description:

This function is used to read the current input voltage value from a specified AI module in an LP-22xx/52xx system using the Modbus protocol.

Syntax:

[C]

```
int    getXVReg(char cPort, WORD wAddr, WORD wCount, char szBuf, WORD wBufLen)
```

Parameters:

cPort:	[**Input**] The number of the COM port for the XV-Board. 1=COM1=/dev/ttyO1.
wAddr:	[**Input**] The channel address.
	Note: Input the channel address mapping in **decimal** format.
wCount:	[**Input**] The number of channels.
szBuf[ ]:	[**Output**] The current Analog Input value.
wBufLen:	[**Input**] The length of the szBuf[ ].

Return Values:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 3: "Error Codes" for details of other returned values.

Example:

```
int RetValue, comport=COM1, count=1;
int addr=3;             /* Refer to [Remark 1] */
char szBuf[80];
DWORD baudrate=115200;
/* Open the device file */
RetValue=Open_Com(comport, baudrate, Data8Bit, NonParity, OneStopBit);
if (RetValue > 0) {
    printf("open COM%d failed!\n",comport);
    return FAILURE;
}
memset(szBuf, 0, sizeof(szBuf));
RetValue=getXVReg(comport, addr, count, szBuf, sizeof(szBuf));
Close_Com(comport);
if (RetValue==0) {
    dwValue=(szBuf[0]<<8)|szBuf[1];
    printf("%u", dwValue);
}
```

Remarks:

[1] This is the input value reference address mapping (Modbus register table) for the '**wAddr**' parameter.

More information regarding address mapping can be found at "**xv-board_user_manual**" file:

http://ftp.icpdas.com.tw/pub/cd/linpac/napdos/lp-5000/lp-52xx/lp-5231/user_manual/

2.7. setXVBit

Description:

This function is used to set the status value for a specified DO module in an LP-22xx/52xx system using the Modbus protocol.

Syntax:

[C]

```
int    setXVBit(char cPort, WORD wAddr, WORD wCount, char szBuf, WORD wBufLen)
```

Parameters:

cPort:	[**Input**] The number of the COM port for the XV-Board. 1=COM1=/dev/ttyO1.
wAddr:	[**Input**] The channel address. **Note**: Input the channel address mapping in **decimal** format.
wCount:	[**Input**] The number of channels.
szBuf[ ]:	[**Input**] Use to set the Digital Output data in **decimal** format.
wBufLen:	[**Input**] The length of the szBuf[ ].

Return Values:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 3: "Error Codes" for details of other returned values.

Example:

```
int RetValue, comport=COM1, count=4;
int addr=0;             /* Refer to [Remark 1] */
char szBuf[80];
DWORD baudrate=115200, dwValue=15;
/* Open the device file */
RetValue=Open_Com(comport, baudrate, Data8Bit, NonParity, OneStopBit);
if (RetValue > 0) {
      printf("open COM%d failed!\n",comport);
      return FAILURE;
}
memset(szBuf, 0, sizeof(szBuf));
szBuf[0]=dwValue & 0xff;
szBuf[1]=(dwValue >> 8) & 0xff;
szBuf[2]=(dwValue >> 16) & 0xff;
szBuf[3]=(dwValue >> 24) & 0xff;
RetValue=setXVBit(comport, addr, count, szBuf, sizeof(szBuf));
Close_Com(comport);
```

Remarks:

[1] This is the input value reference address mapping (Modbus register table) for the '**wAddr**' parameter.

More information regarding address mapping can be found at "**xv-board_user_manual**" file:

http://ftp.icpdas.com.tw/pub/cd/linpac/napdos/lp-5000/lp-52xx/lp-5231/user_manual/

2.8. setXVReg

Description:

This function is used to set the voltage float value for a specified AO module in an LP-22xx/52xx system using the Modbus protocol.

Syntax:

[C]

```
int    setXVReg(char cPort, WORD wAddr, WORD wCount, char szBuf, WORD wBufLen)
```

Parameters:

cPort:	[**Input**] The number of the COM port for the XV-Board. 1=COM1=/dev/ttyO1.
wAddr:	[**Input**] The channel address.
	Note: Input the channel address mapping in **decimal** format.
wCount:	[**Input**] The number of channels.
szBuf[ ]:	[**Input**] Used to set the Analog Output value.
wBufLen:	[**Input**] The length of the szBuf[ ].

Return Values:

0: The function was successfully processed.

Other: The processing failed.

Refer to Chapter 3: "Error Codes" for details of other returned values.

Example:

```
int RetValue, comport=COM1, count=1;
int addr=32;            /* Refer to [Remark 1] */
char szBuf[80];
DWORD baudrate=115200, dwValue=65535;
/* Open the device file */
RetValue=Open_Com(comport, baudrate, Data8Bit, NonParity, OneStopBit);
if (RetValue > 0) {
        printf("open COM%d failed!\n",comport);
        return FAILURE;
}
memset(szBuf, 0, sizeof(szBuf));
szBuf[0]=(dwValue >> 8) & 0xff;
szBuf[1]=dwValue & 0xff;
RetValue=setXVReg(comport, addr, count, szBuf, sizeof(szBuf));
Close_Com(comport);
```

Remarks:

[1] This is the input value reference address mapping (Modbus register table) for the '**wAddr**' parameter.

More information regarding address mapping can be found at "**xv-board_user_manual**" file:

http://ftp.icpdas.com.tw/pub/cd/linpac/napdos/lp-5000/lp-52xx/lp-5231/user_manual/

3. Error Codes

The following table provides a list of error codes used by Modbus-based APIs.

Value	Return code	Description
0	NoError	OK. The function was successful.
1	FunctionError	Function use error.
2	PortError	Port open error.
3	BaudRateError	Set Baud Rate error.
4	DataError	Set data Bits error.
5	StopError	Set stop Bits error.
6	ParityError	Set parity Bits error.
7	CheckSumError	Checksum error.
8	ComPortNotOpen	The specified COM port is not open.
10	SendCmdError	Send command error.
11	ReadComStatusError	Receive command status error.
12	ResultStrCheckError	Checksum error.
13	CmdError	Send error command.
15	TimeOut	The command has received no response within the predetermined time.
19	UnderInputRange	An error has occurred because the input is below the expected range.
20	ExceedInputRange	An error has occurred because the input has exceeds the expected range.

4. Demo for XV-Boards

Making sure you are ready to install the LinPAC AM335x SDK ("lp52xx_am335x_sdk_for_windows.exe" or "lp52xx_am335x_sdk_for_linux.tar.bz2") from the ICP DAS FTP site.

User can visit to **/usr/sbin/** directory, and will find Modbus RTU commands such as - '**getmodbus**' and '**setmodbus**', which is designed for send/receive Modbus RTU communication and built-in LP-5231 and LP-22xx controller. Now, you can follow the steps described below to implement the demonstration.

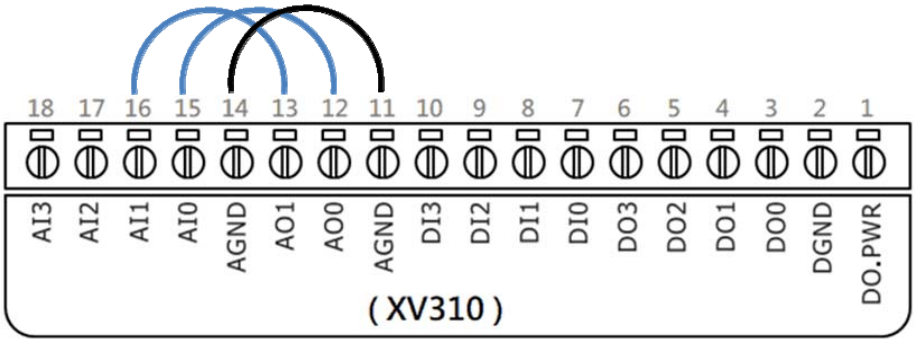
4.1. AI/O, DI/O Expansion Boards

4.1.1. AIO

➲ Location

Visit to "**\LinPAC_AM335x_SDK\examples\xvboard\"** directory in the LinPAC AM335x SDK, you will find the **setxvao.c**, **getxvao.c**, and **getxvai.c** files in details.

➲ Wiring Connections



➲ Run

Set and read back the Analog Output value for a single channel

Step1: Transfer the setxvao.exe and getxvao.exe files to the LP-22xx/52xx module.

Step2: Change the permissions for the files as shown below:

```
#  chmod  755  setxvao.exe      // Syntax of setxvao.exe: setxvao addr value
#  chmod  755  getxvao.exe      // Syntax of getxvao.exe: getxvao addr
```

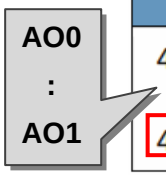
Step3: Execute the program.

```
#  ./setxvao  33  65535         // Output 10 V
#  ./getxvao  33                // Read back the value from the AO channel
```

Remarks:

The following is a more detailed description of the demonstration.

(1) In this program, the address value for the **AO1** channel is **33**.
Refer to the Modbus register table for the XV310 module for more details. The table below provides the valid range for the register addresses.



Register		Points	No. Per Point	Description	Attribute
DEC	HEX				
40032 : 40033	0020 : 0021	2	1	AO value	R/W

Refer to Chapter 1.3: "Specifications" for details of other specifications for the XV-Board together with the Modbus register table.

(2) In this example, we set the type code for the AO range to 02, which denotes that the Analog Output range is +0 V to +10 V. The table below gives an overview of the properties for the Analog Output range.

Type Code	Range	Data Format	Minimum	Maximum
02	+0 V ~ +10 V	Engineering	0	+10000
		Hexadecimal	0000h	FFFFh

(3) The default data format for the AI/AO channel is hexadecimal, so we must first convert the **hexadecimal** value to a **decimal** value. For example, the hexadecimal value **FFFF (hex)** would be expressed in decimal format as **65535 (dec)**.

Read the Analog Input value for a single channel

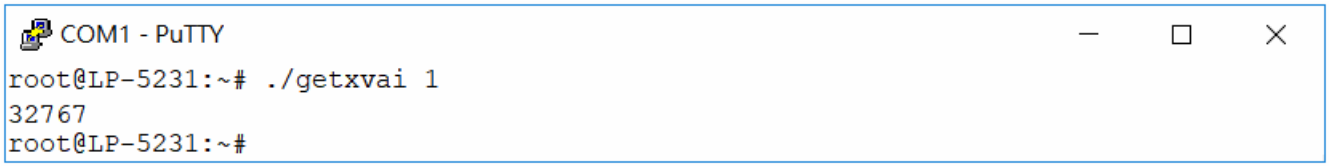
Step1: Transfer the getxvai.exe file to the LP-22xx/52xx module.

Step2: Change the permissions for the file as shown below:

```
# chmod   755   getxvai.exe          // Syntax of setxvao.exe: getxvai addr
```

Step3: Execute the program.

```
# ./getxvai   1                      // Input voltage is 10 V
```



Remarks:

The following is a description of the demonstration.

(1) In this program, the address value for the **AI1** channel is **1**. Refer to the Modbus register table for the XV310 module for more details. The table below provides the valid range for register addresses.

AI0
AI1
:
AI3

Register		Points	No. Per Point	Description	Attribute
DEC	HEX				
30000 30001 : 30003	0000 ⋮ 0003	4	1	AI value	R

Refer to Chapter 1.3: "Specifications" for details of other specifications for the XV-Board together with the Modbus register table.

(2) In this example, we set the type code of for the AI range to **08**, which denotes that the Analog Input range is +/-10 V. The table below gives an overview of the properties for the Analog Input range.

Type Code	Range	Data Format	Minimum	Maximum
08	+/- 10 V	Engineering	-10000	+10000
		Hexadecimal	8000h	7FFFh

(3) The default data format for AI/AO is hexadecimal, so we must first convert the **decimal** value to a **hexadecimal** value. For example, the decimal value **32767 (dec)** would be expressed in hexadecimal format as **7FFF (hex)**.

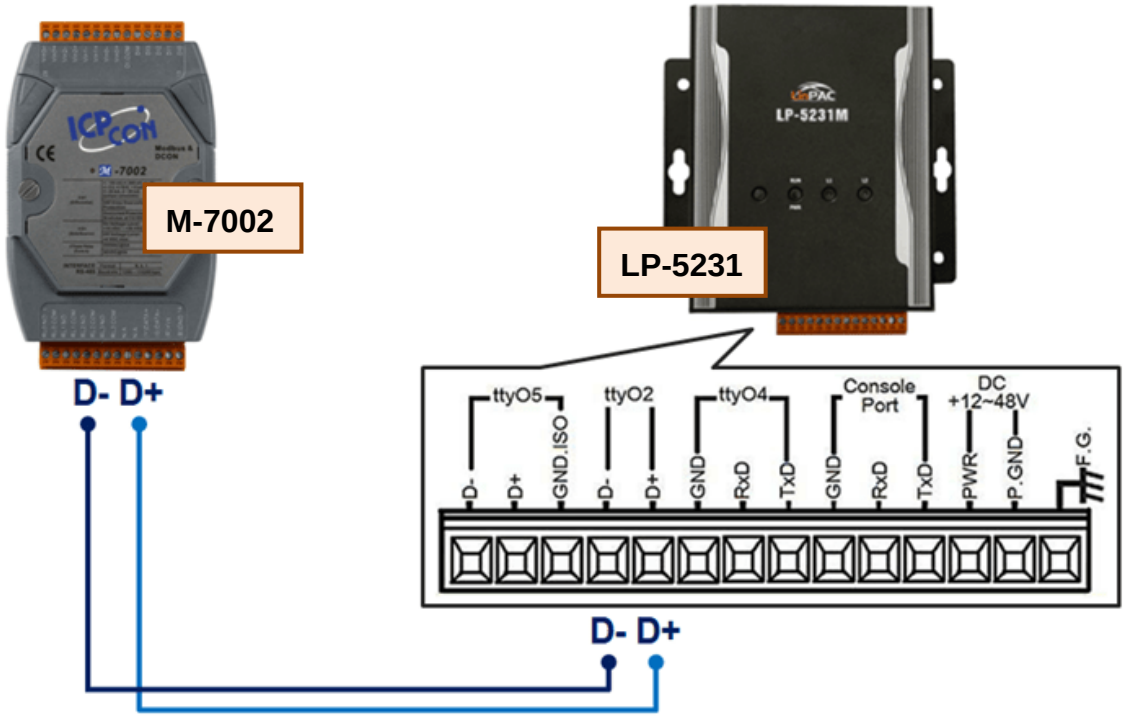
4.1.2 DIO

➲ Location

Visit to "**\LinPAC_AM335x_SDK\examples\xvboard\"** directory in the LinPAC AM335x SDK, you will find the **setxvdo.c**, **getxvdo.c**, and **getmodbus.c** files in details.

➲ Wiring Connections

The following is an example of communication between the XV310 module and the M-7000 series module. To read the M-7002 module values from an LP-52xx module, we need to connect an M-7002 module via the RS-485 serial bus, as illustrated below:

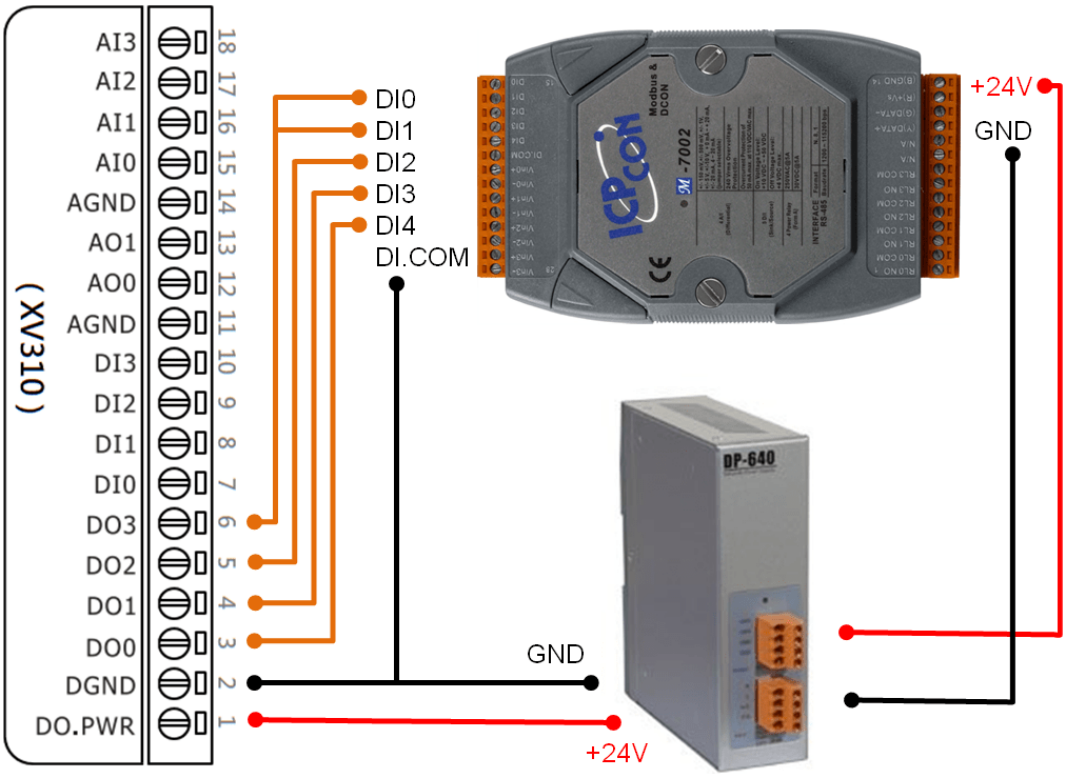


For more information related to the M-7002 module, refer to:

http://www.icpdas.com/root/product/solutions/remote_io/rs-485/i-7000_m-7000/m-7002.html

Note: The default base address for M-7000 series modules is **0**. **(Base 0)**

Connect the DO channel on the XV310 module to the DI channel on the M-7002 module, as illustrated below:



➲ Run

Set and read back the Digital Output value from multiple channels

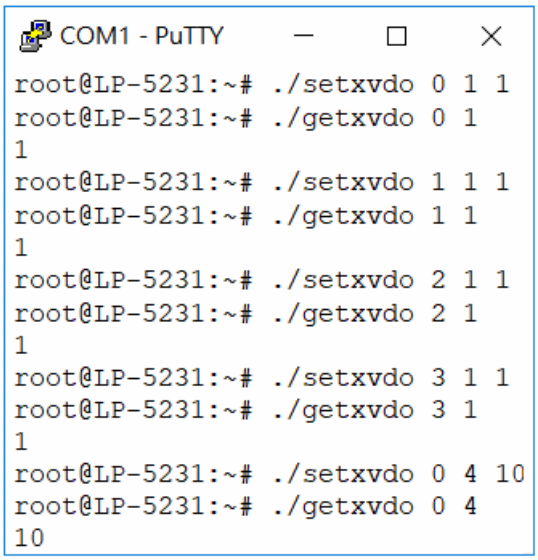
Step1: Transfer the setxvdo.exe and getxvdo.exe files to the LP-52xx module.

Step2: Change the permissions for the files as shown below:

```
# chmod   755   setxvdo.exe      // Syntax of setxvdo.exe: setxvdo addr count value
# chmod   755   getxvdo.exe      // Syntax of getxvdo.exe: getxvdo addr count
```

Step3: Execute the program.

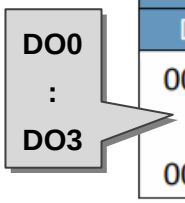
Write to the DO0 channel	# ./setxvdo 0 1 1
Write to the DO1 channel	# ./setxvdo 1 1 1
Write to the DO2 channel	# ./setxvdo 2 1 1
Write to the DO3 channel	# ./setxvdo 3 1 1
Write to all DO channels	# ./setxvdo 0 4 10
Read back the DO0 channel	# ./getxvdo 0 1
Read back the DO1 channel	# ./getxvdo 1 1
Read back the DO2 channel	# ./getxvdo 2 1
Read back the DO3 channel	# ./getxvdo 3 1
Read back all DO channels	# ./getxvdo 0 4



Remarks:

The following is a more detailed description of the demonstration.

(1) Refer to the Modbus register table for the XV310 module for more details. The table below provides the valid range for the register addresses.



Register		Points	Description	Data Format	Attribute
DEC	HEX				
00000 : 00003	0000 : 0003	4	DO value	0: Off 1: On	R/W

Refer to Chapter 1.3: "Specifications" for details of other specifications for the XV-Board, together with the Modbus register table.

(2) Set the DO value for the XV310 module. The data format is as follows:

- Single channel

DO Channel	Address	Count	Channel Status	Data Format (Value)
DO0	00000	1	ON or OFF	0: OFF 1: ON
DO1	00001			
DO2	00002			
DO3	00003			

- All channels

Start Address	Count	Channel Status				Data Format (Value)
00000	4	DO3	DO2	DO1	DO0	0~15
		ON or OFF				
Sample command # **./setxvdo 0 4 10**						
00000	4	1	0	1	0	10

In this example, set the channel states for the DO1 and DO3 channels to ON, and set the other DO channels to OFF. The default data format for multiple Digital Input or Digital Output channels is decimal, so we must first convert the **binary** value to a **decimal** value. For example, the binary value **1010 (bin)** would be expressed in decimal format as **10 (dec)**.

Read the Digital Input value from multiple channels on the M-7002 module

Step1: Transfer the getmodbus.exe file to the LP-52xx module.

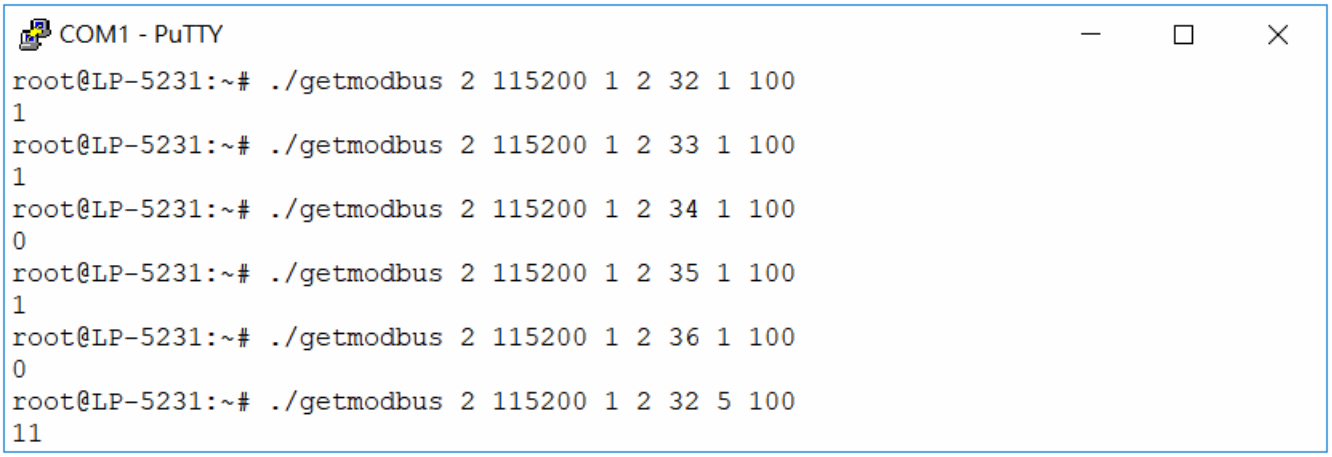
Step2: Change the permissions for the file, as shown below:

```
# chmod   755   getmodbus.exe
```

// Syntax of getmodbus: getmodbus comport baudrate netid command addr count timeout

Step3: Execute the program.

Read the DI0 channel on the M-7002 module	# ./getmodbus	2	115200	1	2	32	1	100
Read the DI1 channel on the M-7002 module	# ./getmodbus	2	115200	1	2	33	1	100
Read the DI2 channel on the M-7002 module	# ./getmodbus	2	115200	1	2	34	1	100
Read the DI3 channel on the M-7002 module	# ./getmodbus	2	115200	1	2	35	1	100
Read the DI4 channel on the M-7002 module	# ./getmodbus	2	115200	1	2	36	1	100
Read all DI channels on the M-7002 module	# ./getmodbus	2	115200	1	2	32	5	100



Remarks:

The following is a more detailed description of the demonstration.

(1) For more information related to the address mapping (Modbus register table) for the M-7002 module, refer to:

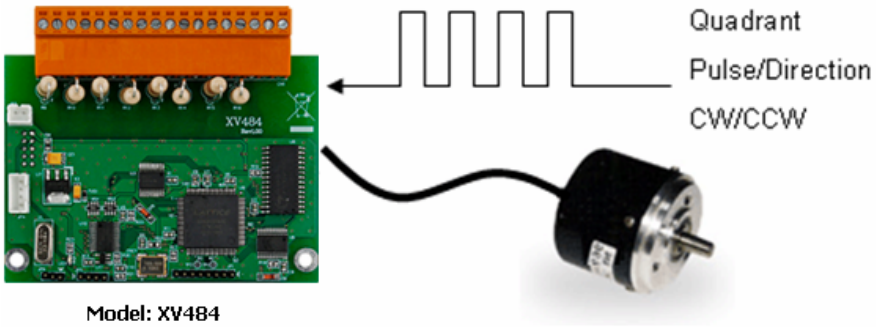
http://www.icpdas.com/products/Remote_IO/m-7000/address_mapping/m7000_address_mapping.pdf

The **base address** for the M-7000 module when communicating with the LP-5231 module is **0**. The table below provides an overview of the valid range of M-7002 module register addresses.

	Address	Description	Attribute
DI0	10032	Digital input status for channels 0 to 4	R
DI1	10033		
DI2	10034		
DI3	10035		
DI4	10036		

4.2. Counter, Frequency, Encoder Expansion Boards

4.2.1 Counter, Frequency, Encoder



Type code	Types
0x50	Counter Mode
0x51	Frequency Mode
0x54	Up/Down Counter Mode
0x55	Plus Mode
0x56	AB Phase Mode

➲ Location

Visit to "**\LinPAC_AM335x_SDK\examples\xvboard**" directory in the LinPAC AM335x SDK, you will find the **setmodbus.c** and **getmodbus.c** files in details.

➲ Run

Set and read back the Analog Output value for a single channel

Step1: Transfer the **setmodbus.c** and **getmodbus.c** files to the LP-22xx/52xx.

Step2: Change the permissions for the files as shown below:

```
# chmod   755   getmodbus.exe   setmodbus.exe
// Syntax: getmodbus   comport   baudrate   netid   command   addr   count   timeout
// Syntax: setmodbus   comport   baudrate   netid   command   addr   count   value   timeout
```

Read type code of channel

```
# getmodbus   1   115200   1   3   256   1   100
85                                          // 85 dec = 0x56 hex: Plus mode
```

Set type code of channel 0

```
# setmodbus   1   115200   1   6   256   0   86   100      // 86 dec = 0x56: AB Phase mode
# setmodbus   1   115200   1   6   256   2   86   100      // 86 dec = 0x56: AB Phase mode
```

Set Enable battery backup for counter 0 to 7

```
# setmodbus   1   115200   1   5   768   1   0   100      // 768 = Set Channel 0
```

Read counter of AB phase mode ➔ Input Register (3xxxx)

```
# getmodbus   1   115200   1   4   0   1   100
54389                                       // Read counter form channel 0 is 54389
# getmodbus   1   115200   1   4   0   1   100
57816                                       // Read counter form Channel 0 is 57816
```

Appendix

A. Revision History

This chapter provides information related to the revision history of this document.

The table below shows the revision history.

Revision	Date	Description
V1.0	Apr. 2019	**Initial issue**
V1.1	May 2020	**Add new module: XV-484**