Parker Automation

p/n YPM08130

Acroloop Communications Control

Effective: October 7, 2002

Acroloop PC Bus Communications Control

The AMCS communication control provides PC Bus communications to an Acroloop Motion Controller board. It provides functions for the binary commands as well as generic text reads and writes.

File Name

ACRXPC.OCX
Version 1,0,0,2

Operating Systems

Windows NT

Required Drivers

ACR.SYS (Windows NT only)

Remarks

A call must be made to OpenPort before any other functions are called. Each successful call to OpenPort must be matched with a call to ClosePort before the application terminates. The control uses binary data transfers for most of the functions.

Acroloop Serial Communications Control

The AMCS communication control provides serial communications to an Acroloop Motion Controller board. It provides functions for the binary commands as well as generic text reads and writes.

File Name

ACROCTL.OCX
Version 1,0,0,1

Operating Systems

Windows 95/98/NT

Required Drivers

None

Remarks

A call must be made to OpenPort before any other functions are called. Each successful call to OpenPort must be matched with a call to ClosePort before the application terminates. The control uses binary data transfers for most of the functions. It can use one of two modes for data conversion, control character prefixing or high bit stripping with control character prefixing. Each communications control corresponds to one serial port. Up to eight boards may be used on a single serial port. Additional ports may be used for additional boards.

Board Number

PC Property Only

Sets and returns the board number for the control.

Syntax

Visual Basic

(control name).BoardNumber[= Board Number]

Visual C++

(control name).BoardNumber[= Board Number];

Remarks

Valid board numbers are 0 – 7. When setting the BoardNumber at run time, it can take 150ms to set, therefore use an instance of the control for each board installed in your computer. Generates an exception if the BoardNumber is invalid.

Timeout

PC Property Only
Sets and returns the timeout value, in milliseconds, for binary reads.

Syntax

Visual Basic

(control name).Timeout[= Time Out Value]

Visual C++

(control name).Timeout[= Time Out Value];

Remarks
Minimum timeout value is 10 milliseconds. Default timeout value is 20 milliseconds.

ComPort

Serial Property Only
Sets and returns the serial port number.

Syntax
Visual Basic
(control name).ComPort[= Port Number]
Visual C++
(control name).ComPort[= Port Number];

Remarks
The serial port can be set to a number between 1 and 8. An error will be generated if the port is not available when you try to open it.

BaudRate

Serial Property Only
Sets and returns the baud rate for the serial port.

Syntax
Visual Basic
(control name).BaudRate[= Baud Rate]
Visual C++
(control name).BaudRate[== Baud Rate];

Remarks
The baud rate can be set to 2400, 4800, 9600, 19200 or 38400. An error will be generated for any other value.

Mode

Serial Property Only
Sets and returns the mode for the communication functions.

Syntax
Visual Basic

(control name).Mode[= Mode]

Visual C++

(control name).Mode[= Mode];

Remarks
The mode can be set to either 1 (control character prefixing) or 3 (control character prefixing and high bit stripping). An error will be generated if you attempt to set the mode to any other values.
Note
For a further explanation of the modes, see the Acroloop Controllers User's Guide.

Select All

Serial Method Only
Selects or “turns on” all of the cards.

Syntax

Visual Basic

```
(control name).SelectAll
```

Visual C++

```
(control name).SelectAll;
```

Remarks
This function is used during multiple board communications to select all cards.

Deselect All

Serial Method Only
Selects or “turns off” all of the cards.

Syntax

Visual Basic

(control name).DeselectAll

Visual C++

(control name).DeselectAll;

Remarks
This function is used during multiple board communications to deselect all cards.

Select

Serial Method Only
Selects or “turns on” the appropriate card.

Syntax

Visual Basic

(control name).Select *Board*

Visual C++

(control name).Select(short *Board*t);

Parameter	Basic Type	C++ Type	Description
Board	integer	short	Required. Board to select.

Remarks
This function is used during multiple board communications to select a particular card.

Deselect

Serial Method Only
Selects or “turns off” the appropriate card.

Syntax

Visual Basic

(control name).Deselect *Board*

Visual C++

(control name).Deselect(short *Board*);

Parameter	Basic Type	C++ Type	Description
Board	integer	short	Required. Board to deselect.

Remarks
This function is used during multiple board communications to deselect a particular card.

OpenPort

PC and Serial Method
Opens a communications port for I/O operations.

Syntax
Visual Basic
(control name).OpenPort
Visual C++
(control name).OpenPort();

Remarks
Opens the communications port. Generates an error if the port is unavailable.

ClosePort

PC and Serial Method
Closes the communication port.

Syntax

Visual Basic

(control name).ClosePort

Visual C++

(control name).ClosePort();

Remarks
Closes the communication port. Any time the port is successfully opened, it must be closed before the application exits.

ClearBit

PC and Serial Method
Clears the bit specified by its index.

Syntax

Visual Basic

(control name).ClearBit *Index*

Visual C++

(control name).ClearBit(short Index);

Parameter	Basic Type	C++ Type	Description
Index	integer	short	Required. Index of the bit to clear.

Remarks
This function performs a binary Clear command to clear the bit at the specified index.

SetBit

PC and Serial Method
Sets the bit specified by its index.

Syntax

Visual Basic

(control name).SetBit *Index*

Visual C++

(control name).SetBit(short *Index*);

Parameter	Basic Type	C++ Type	Description
Index	integer	short	Required. Index of the bit to set.

Remarks
This function performs a binary Set command to set the bit at the specified index.

GetLong

PC and Serial Method
Returns a long integer parameter for the specified index.

Syntax

Visual Basic

NewValue = (control name).GetLong *Index*

Visual C++

NewValue = (control name).GetLong(short *Index*);

Parameter	Basic Type	C++ Type	Description
Index	integer	short	Required. Index of the long integer to get.
NewValue	long	long	Return value.

Remarks
This function performs a binary get long to retrieve the long parameter at the specified index.

SetLong

PC and Serial Method
Sets a long integer parameter for the specified index.

Syntax

Visual Basic

(control name).SetLong *Index, NewValue*

Visual C++

(control name).SetLong(short *Index*, long *NewValue*);

Parameter	Basic Type	C++ Type	Description
Index	integer	short	Required. Index of the long integer to set.
NewValue	long	long	Return value.

Remarks
This function performs a binary set long to set the long parameter at the specified index.

GetIEEE

PC and Serial Method
Returns a floating-point parameter for the specified index.

Syntax

Visual Basic

NewValue = (control name).GetIEEE *Index*

Visual C++

NewValue = (control name).GetIEEE(short, *Index*);

Parameter	Basic Type	C++ Type	Description
Index	integer	short	Required. Index of the floating-point value to get.
NewValue	single	float	Return value.

SetIEEE

PC and Serial Method
Sets a floating-point parameter.

Syntax

Visual Basic

(control name).SetIEEE *Index, NewValue*

Visual C++

(control name).SetIEEE(short *Index,* single *NewValue*);

Parameter	Basic Type	C++ Type	Description
Index	integer	short	Required. Index of the floating-point value to set.
NewValue	single	float	Required. New value of parameter.

Remarks
This function performs a binary Set IEEE to set the floating-point parameter at the specified index.

GetLongParameter

PC and Serial Method
Retrieves the parameters specified in the mask for the specified group and index.

Syntax

Visual Basic

(control name).GetLongParameter*Group, Index, Mask, Data*

Visual C++

(control name).GetLongParameter(short *Group*, short *Index*, short *Mask*, long **Data*);

Parameter	Basic Type	C++ Type	Description
Group	integer	short	Required. Group to retrieve.
Index	integer	short	Required. Parameter index to retrieve.
Mask	integer	short	Required. Parameters to retrieve
Data	long	*long	Required. Pointer to an array to hold the data.

Remarks
This function retrieves a binary data packet using the specified group code, index and mask.

Note:

See the Acroloop Motion Controller's User's Guide for more information on this function and the appropriate values for the group, index and mask parameters.

GetFloatParameter

PC and Serial Method
Retrieves the parameters specified in the mask for the specified group and index.

Syntax

Visual Basic

(control name).GetFloatParameter *Group*, *Index, Mask, Data*

Visual C++

(control name).GetFloatParameter(short *Group*, short *Index*, short *Mask*, float **Data*);

Parameter	Basic Type	C++ Type	Description
Group	integer	short	Required. Group to retrieve.
Index	integer	short	Required. Parameter index to retrieve.
Mask	integer	short	Required. Parameters to retrieve
Data	Single	*float	Required. Pointer to an array to hold the data.

Remarks
This function retrieves a binary data packet using the specified group code, index and mask.

Note:
See the Acroloop Motion Controllers User's Guide for more information on this function and the appropriate values for the group, index and mask parameters.

GetText

PC and Serial Method
Retrieves text from the board until the operation times out.

Syntax

Visual Basic

(control name).GetText *Text*

Visual C++

(control name).GetText(BSTR **Text*);

Parameter	Basic Type	C++ Type	Description
Text	string	*BSTR	Required. Pointer to a string that receives the text.

Remarks
This function retrieves text data from the port until it times out.

SendText

PC and Serial Method
Sends a text string and returns the number of characters sent.

Syntax

Visual Basic

(control name).SendText *Text*

Visual C++

(control name).SendText(LPCTSTR *Text*);

Parameter	Basic Type	C++ Type	Description
Text	string	LPCTSTR	Required. Pointer to the string to be sent.

Remarks
This function sends the specified text to the port.

GetParameterAddress

PC and Serial Method
Returns the address of the parameter specified by its index.

Syntax

Visual Basic

(control name).GetParameterAddress *Index*

Visual C++

(control name).GetParameterAddress(short *Index*);

Parameter	Basic Type	C++ Type	Description
Index	integer	short	Required. Index of the parameter.

Remarks
This function performs a binary Parameter Address command to return the address of the specified parameter.

GetVariableAddress

PC and Serial Method
Returns the address of the specified variable type for the specified program.

Syntax

Visual Basic

(control name).GetVariableAddress *Program, VariableCode*

Visual C++

(control name).GetVariableAddress(short *Program*, short *VariableCode*);

Parameter	Basic Type	C++ Type	Description
Program	integer	short	Required. Program number.
Variable Code	integer	short	Required. Code specifying the type of variable.

Remarks
This function performs a binary Address command to return the address of the specified variable type for the specified program.

Variable Codes:

0x00	Double Variables (DV)
0x01	Double Arrays (DA)
0x02	Single Variables(SV)
0x03	Single Arrays (SA)
0x04	Long Variables (LV)
0x05	Long Arrays (LA)
0x06	String Variables ($V)
0x07	String Arrays ($A)

MaskData

PC and Serial Method
Masks the data at the specified address with the specified masks.

Syntax

Visual Basic

(control name).MaskData *Address, NAND_Mask, OR_Mask*

Visual C++

(control name).MaskData(long *Address*, long *NAND_Mask*, long *OR_Mask*);

Parameter	Basic Type	C++ Type	Description
Address	long	long	Required. Address of data to be masked.
NAND_Mask	long	long	Required. Masks the data using a binary NAND operation with the mask.
OR_Mask	long	long	Required. Masks the data using a binary OR operation with the mask.

Remarks
This function performs a binary Mask command on the data at the specified address using the specified mask parameters.

MaskParameter

PC and Serial Method
Masks the parameter at the specified index with the specified masks.

Syntax

Visual Basic

(control name).MaskParameter *Index, NAND_Mask, OR_Mask*

Visual C++

(control name).MaskParameter(long *Index,* long *NAND_Mask,* long *OR_Mask*);

Parameter	Basic Type	C++ Type	Description
Index	long	long	Required. Parameter index of data to be masked.
NAND_Mask	long	long	Required. Masks the data using a binary NAND operation with the mask.
OR_Mask	long	long	Required. Masks the data using a binary OR operation with the mask.

Remarks
This function performs a binary Parameter Mask command on the parameter at the specified index using the specified mask parameters.

PeekLong

PC and Serial Method
Reads the number of words requested at the specified address.

Syntax

Visual Basic

(control name).PeekLong *Address, Data, Count*

Visual C++

(control name).PeekLong(long *Address*, long **Data*, short*Count*t);

Parameter	Basic Type	C++ Type	Description
Address	long	long	Required. Address to read data from.
Data	long	*long	Required. Pointer to a location where returned value can be stored.
Count	integer	short	Required. Number of parameters to retrieve.

Remarks
This function performs a binary Peek Long command to read the data at the specified address.

PokeLong

PC and Serial Method
Stores the number of words provided at the specified address.

Syntax

Visual Basic

(control name).PokeLong *Address, Data, Count*

Visual C++

(control name).PokeLong(long *Address*, long **Data*, short *Count*);

Parameter	Basic Type	C++ Type	Description
Address	long	long	Required. Address to store data on the card.
Data	long	*long	Required. Pointer to a location where new data is stored.
Count	integer	short	Required. Number of parameters to store.

Remarks
This function performs a binary Poke Long command to store the data at the specified address.

PeekFloat

PC and Serial Method
Reads the number of floating point values requested at the specified address.

Syntax

Visual Basic

(control name).PeekFloat *Address, Data, Count*

Visual C++

(control name).PeekFloat(long *Address*, float **Data*, short C*ount*);

Parameter	Basic Type	C++ Type	Description
Address	long	long	Required. Address to read data from.
Data	single	*float	Required. Pointer to a location where returned data can be stored.
Count	integer	short	Required. Number of parameters to retrieve.

Remarks
This function performs a binary Peek Float command to read the data at the specified address.

PokeFloat

PC and Serial Method
Stores the number of words provided at the specified address.

Syntax

Visual Basic

(control name).PokeFloat *Address, Data, Count*

Visual C++

(control name).PokeFloat(long *Address*, float **Data*, short *Count*);

Parameter	Basic Type	C++ Type	Description
Address	long	long	Required. Address to store data on the card.
Data	single	*float	Required. Pointer to a location where new data is stored.
Count	integer	short	Required. Number of parameters to store.

Remarks
This function performs a binary Poke command to store the data at the specified address.

PeekDouble

PC and Serial Method
Reads the number of double floating point values requested at the specified address.

Syntax

Visual Basic

(control name).PeekDouble *Address, Data, Count*

Visual C++

(control name).PeekDouble(long *Address*, *double *Data*, short *Count*);

Parameter	Basic Type	C++ Type	Description
Address	long	long	Required. Address to read data from.
Data	double	*double	Required. Pointer to a location where returned value can be stored.
Count	integer	short	Required. Number of parameters to retrieve.

Remarks
This function performs a binary Peek command to read the data at the specified address.

PokeDouble

PC and Serial Method
Stores the number of double precision floating point values provided at the specified address.

Syntax

Visual Basic

(control name).PokeDouble *Address, Data, Count*

Visual C++

(control name).PokeDouble(long *Address*, double **Data*, short *Count*);

Parameter	Basic Type	C++ Type	Description
Address	long	long	Required. Address to store data on the card.
Data	double	*double	Required. Pointer to a location where new data is stored.
Count	integer	short	Required. Number of parameters to store.

Remarks
This function performs a binary Poke command to store the data at the specified address.