Parker
Automation

Acroloop
Motion Controller
CAcroLite C++ Class

Effective: October 7, 2002

INTRODUCTION

The CAcroLite class provides the C++ programmer with an encapsulated method of communicating with an Acroloop motion control card, using the ACR.SYS driver. The interface is similar to the ACRXPC.OCX ActiveX control, but does not require the overhead inherent in hosting OLE controls.
Note that this object does not communicate with the ACR8K.SYS driver, and does not offer binary move commands. If your application requires binary moves or the use of the Fast Status Buffer, the ACR8K.SYS driver and ACROWNT.DLL API interface will be required. The ACR.SYS driver and CAcroLite class are most appropriate for HMI applications where the front end is passing and displaying data, while motion control is performed directly on the Acroloop card.

The CAcroLite class does not support multiple-thread concurrent access.

CAcroLite() Constructor

#include "liteclass.h"

CAcroLite oCard(int DeviceNumber, int TextBufferSize = 4096)

CAcrolite *poCard = new CAcroLite(int DeviceNumber, int TextBufferSize = 4096)

Parameter	Type	Description
DeviceNumber	**int**	Required. Value specifying which device to initialize and use.
TextBufferSize	**int**	Optional. Value specifying internal size of the text buffer.

The DeviceNumber parameter is required. A CAcroLite object will be created to communicate with the selected Acroloop device (0-7). The object's internal text buffer may be optionally dimensioned between 1024 and 32768 bytes. The default size is 4096 bytes.

If unable to create, the constructor will throw an exception ACRLITE_E_PORTERROR. Reasons for failure include ACR driver not running, device not configured in driver, unable to allocate text buffer or text buffer size out of bounds.

Creation of a CAcroLite instance takes >100 milliseconds.

Default Constructor

CAcroLite oCard;

If the default constructor is used, the InitPort method must be called before the object can communicate with an Acroloop device.

InitPort

void **InitPort**(int DeviceNumber, int TextBufferSize)

Parameter	Type	Description
DeviceNumber	**int**	Required. Value specifying which device to initialize and use.
TextBufferSize	**int**	Optional. Value specifying internal size of the text buffer.

The DeviceNumber parameter is required. The CAcroLite object will be initialized to communicate with the selected Acroloop device (0-7). The object's internal text buffer may be optionally dimensioned between 1024 and 32768 bytes. The default size is 4096 bytes.

If unable to Initialize, InitPort() will throw an exception ACRLITE_E_PORTERROR. Reasons for failure include ACR driver not running, device not configured in driver, unable to allocate text buffer or text buffer size out of bounds.

When initializing the CAcroLite object with this method, be aware that the method takes >100 milliseconds to complete.

GetLong

Returns a long parameter for the given index using a binary data transfer.

long **GetLong**(short Index)

Parameter	Type	Description
Index	**short**	Required. Value specifying which parameter index to retrieve.

Remarks

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

SetLong

Sets a long parameter.

void **SetLong**(short Index, long NewValue)

Parameter	Type	Description
Index	**short**	Required. Specifies the parameter index to set.
NewValue	**long**	Required. New value of parameter.

Remarks

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

GetIEEE

Returns a float parameter for the given index using a binary data transfer.

float **GetIEEE**(short Index)

Parameter	Type	Description
Index	**short**	Required. Specifies which parameter index to retrieve.

Remarks

This function performs a binary get IEEE to retrieve the float parameter at the specified index.

SetIEEE

Set a floating point parameter.

void **SetIEEE**(short Index, float NewValue)

Parameter	Type	Description
Index	**short**	Required. Specifies the parameter index to set.
NewValue	**float**	Required. New value of parameter.

Remarks

This function performs a binary Set IEEE to set the float parameter at the specified index.

PeekLong

Reads the number of words requested at the specified address.

void **PeekLong**(long Address, long *Data, short Count)

Parameter	Type	Description
Address	**long**	Required. Address to read data from.
Data	**long***	Required. Pointer to array where returned data can be stored.
Count	**short**	Required. Number of parameters to retrieve.

Remarks

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

PokeLong

Stores the number of words provided at the specified address.

void **PokeLong**(long Address, long *Data, short Count)

Parameter	Type	Description
Address	**long**	Required. Address on the card where data will be sent.
Data	**long***	Required. Pointer to an array of new data to be sent to the card.
Count	**short**	Required. Number of parameters to store.

Remarks

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

PeekFloat

Reads the number of floats requested at the specified address.

void **PeekFloat**(long Address, float *Data, short Count)

Parameter	**Type**	**Description**
Address	**long**	Required. Address to read data from.
Data	**float***	Required. Pointer to array where returned data can be stored.
Count	**short**	Required. Number of parameters to retrieve.

Remarks

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

PokeFloat

Stores the number of words provided at the specified address.

void **PokeFloat**(long Address, float *Data, short Count)

Parameter	Type	Description
Address	**long**	Required. Address on the card where data will be stored.
Data	**float***	Required. Pointer to array of data.
Count	**short**	Required. Number of elements in the Data array.

Remarks

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

GetVariableAddress

Returns the address of the specified variable type for the specified program.

long **GetVariableAddress**(short Program, short VariableCode)

Parameter	Type	Description
Program	**short**	Required. Program number.
VariableCode	**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	DV	Double Variables
0x01	DA	Double Arrays
0x02	SV	Single Variables
0x03	SA	Single Arrays
0x04	LV	Long Variables
0x05	LA	Long Arrays
0x06	$V	String Variables
0x07	$A	String Arrays

GetParameterAddress

Returns the address of the parameter specified by its index.

long **GetParameterAddress**(short Index)

Parameter	Type	Description
Index	**short**	Required. Index of the parameter.

Remarks

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

SetBit

Sets the bit specified by its index.

void **SetBit**(short Index)

Parameter	Type	Description
Index	**short**	Required. Index of the bit to set.

Remarks

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

ClearBit

Clears the bit specified by its index.

void **ClearBit**(short Index)

Parameter	Type	Description
Index	**short**	Required. Index of the bit to clear.

Remarks

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

MaskData

Masks the data at the specified address with the specified masks.

void **MaskData**(long Address, long NAND_Mask, long OR_Mask)

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

Remarks

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

MaskParameter

Masks the parameter at the specified index with the specified masks.

void **MaskParameter**(short Index, long NAND_Mask, long OR_Mask)

Parameter	Type	Description
Index	**short**	Required. Index of the parameter to be masked.
NAND_Mask	**long**	Required. Masks the data using a binary NAND operation with the mask.
OR_Mask	**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.

GetFloatParameter

Retrieves the parameters specified in the mask for the specified group and index.

void **GetFloatParameter**(short Group, short Index, short Mask, float *Data)

Parameter	Type	Description
Group	**short**	Required. Specifies which group to retrieve.
Index	**short**	Required. Specifies which parameter index to retrieve.
Mask	**short**	Required. Specifies which parameters to retrieve.
Data	**float***	Required. Pointer to an array that will hold the data.

Remarks

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

Note:

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

GetLongParameter

Retrieves the parameters specified in the mask for the specified group and index.

void **GetLongParameter**(short Group, short Index, short Mask, long *Data)

Parameter	Type	Description
Group	**short**	Required. Specifies which group to retrieve.
Index	**short**	Required. Specifies which parameter index to retrieve.
Mask	**short**	Required. Specifies which parameters to retrieve.
Data	**long***	Required. Pointer to an array that will hold the data.

Remarks

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

Note:

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

GetText

Retrieves text from the board until the operation times out.

long **GetText**(UCHAR *Buf, int BufLength)

Parameter	Type	Description
Buf	**UCHAR***	Required. Pointer to an array of unsigned char that receives the text.
BufLength	**int**	Required. Length of supplied buffer.

Remarks

The function retrieves all the text from the board. Note: Stray text received in reading binary transactions is cached in the object's internal text buffer. If this buffer fills, additional stray text will be discarded. The internal buffer's contents is returned by GetText(), along with any further text available from the card, subject to the limit of the length of the buffer supplied.

SendText

Sends the text string and returns the number of character sent.

long **SendText**(LPCTSTR Text)

Parameter	Type	Description
Text	**LPCTSTR**	Required. Pointer to the string to be sent.

Remarks

This function sends the specified text to the Acroloop card.

ResetPort

Resets the communications port.

BOOL **ResetPort**()

Remarks

This function resets the internal communications port for the object. It returns FALSE on success, and TRUE if an error occurs. All buffered text will be discarded.

CAcroLite_Exceptions

Function failure is reported by throwing exceptions. CAcroLite can throw any of three exceptions:

ACRLITE_E_PORTERROR (value 0xE0000003)

Thrown by the class constructor or any member function if unable to access the ACR.SYS port for the requested device.

ACRLITE_E_WRITEERROR (value 0xE0000001)

Thrown by any member function if unable to write to the ACR.SYS port for the requested device. Note that member functions that read data from the Acroloop device must write the request down to the card, and so may throw this exception.

ACRLITE_E_READERROR (value 0xE0000002)

Thrown by any member function that reads data, if unable to read a well-formed response to the inquiry.

SetTimeOut

Sets the timeout upper bound for binary reads.

void **SetTimeOut**(WORD Value)

Parameter	Type	Description
Value	**WORD**	Required. Number of milliseconds to use for timeout.

Remarks

This function sets a timeout on binary reads. The minimum value is 10 milliseconds. The default value is 20. Internally, the function uses the GetSystemTime() call, which may be affected by changes to the system multimedia timer. If your application, or any application running on the machine, changes the default value of the multimedia timer delta, use SetTimeOut() to tune the timeout performance of the CAcroLite class instantiation accordingly.

GetTimeOut

Gets the current timeout value for binary reads.

WORD **GetTimeOut**(void)

Remarks

This function returns the current value of the timeout on binary reads. The minimum value is 10. The default value is 20.