



---

---

— *MintMT* —

---

---

# PC Programming Guide



---

# Contents

1	General Information.....	1-1
2	Introduction .....	2-1
2.1	Overview.....	2-1
2.2	Abbreviations .....	2-1
3	MintMT ActiveX Control.....	3-1
3.1	What is an ActiveX control?.....	3-1
3.2	Installing the PC developer libraries.....	3-1
3.3	MintMT ActiveX control components.....	3-2
4	Using the MintMT ActiveX Control.....	4-1
4.1	Overview.....	4-1
4.2	Communicating with controllers.....	4-1
4.3	Downloading firmware .....	4-2
4.4	MintMT programs.....	4-3
4.5	Mint Motion Library function calls .....	4-3
4.6	Error Handling .....	4-5
4.7	Limitations of PC based applications.....	4-8
4.7.1	Events .....	4-8
4.7.2	Execution Speed .....	4-8
4.8	Extended functionality of DPR controllers .....	4-8
4.8.1	Accessing Dual Port RAM .....	4-9
4.8.2	Event handling.....	4-9
5	Advanced Features .....	5-1
5.1	Event Handling .....	5-1
5.1.1	Installing ActiveX event handlers.....	5-2
5.1.2	Controlling event generation.....	5-6
5.2	Terminal window.....	5-7
5.3	Command prompt.....	5-8

---

5.4	Communicating with legacy controllers .....	5-9
5.4.1	Serial controllers .....	5-9
5.4.2	Dual Port RAM controllers.....	5-9
<b>6</b>	<b>PC Application Examples.....</b>	<b>6-1</b>
6.1	Microsoft Visual C++.....	6-1
6.2	Microsoft Visual Basic.....	6-7
6.3	Borland Delphi .....	6-10
6.4	National Instruments LabView.....	6-14
<b>7</b>	<b>MintMT ActiveX Methods .....</b>	<b>7-1</b>
7.1	Error Codes.....	7-1
7.1.1	Mint Motion Library Errors.....	7-1
7.1.2	ActiveX errors .....	7-5
7.2	Properties.....	7-10
<b>8</b>	<b>NextMove PCI DPR Map .....</b>	<b>8-1</b>
8.1	Overview .....	8-1
8.2	Dual Port RAM map.....	8-2
8.2.1	Status and control registers .....	8-3
8.2.2	Axis data .....	8-5
8.2.3	I/O data .....	8-6
8.2.4	Comms array .....	8-7
8.2.5	Immediate command mode.....	8-8
8.2.6	Pseudo serial interface.....	8-8
8.2.7	Special functions registers .....	8-9
<b>9</b>	<b>Timings.....</b>	<b>9-1</b>
9.1	MML function call timing .....	9-1
9.1.1	NextMove PCI.....	9-1
9.1.2	NextMove BX .....	9-1
9.1.3	MintDrive'' .....	9-1

Copyright Baldor (c) 2001. All rights reserved.

This manual is copyrighted and all rights are reserved. This document or attached software may not, in whole or in part, be copied or reproduced in any form without the prior written consent of Baldor. Baldor makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of fitness for any particular purpose. The information in this document is subject to change without notice. Baldor assumes no responsibility for any errors that may appear in this document.

Mint™ is a registered trademark of Baldor.

Windows 95, Windows 98, Windows ME, Windows NT and Windows 2000 are registered trademarks of the Microsoft Corporation.

#### Limited Warranty

For a period of two (2) years from the date of original purchase, Baldor will repair or replace without charge controls and accessories which our examination proves to be defective in material or workmanship. This warranty is valid if the unit has not been tampered with by unauthorized persons, misused, abused, or improperly installed and has been used in accordance with the instructions and/or ratings supplied. This warranty is in lieu of any other warranty or guarantee expressed or implied. Baldor shall not be held responsible for any expense (including installation and removal), inconvenience, or consequential damage, including injury to any person or property caused by items of our manufacture or sale. (Some countries and U.S. states do not allow exclusion or limitation of incidental or consequential damages, so the above exclusion may not apply.) In any event, Baldor's total liability, under all circumstances, shall not exceed the full purchase price of the control. Claims for purchase price refunds, repairs, or replacements must be referred to Baldor with all pertinent data as to the defect, the date purchased, the task performed by the control, and the problem encountered. No liability is assumed for expendable items such as fuses. Goods may be returned only with written notification including a Baldor Return Authorization Number and any return shipments must be prepaid.

Baldor UK Ltd  
Mint Motion Centre  
6 Bristol Distribution Park  
Hawkley Drive  
Bristol, BS32 0BF  
Telephone: +44 (0) 1454 850000  
Fax: +44 (0) 1454 850001  
Email: technical.support@baldor.co.uk  
Web site: www.baldor.co.uk

Baldor Electric Company  
Telephone: +1 501 646 4711  
Fax: +1 501 648 5792  
Email: sales@baldor.com  
Web site: www.baldor.com

Baldor ASR GmbH  
Telephone: +49 (0) 89 90508-0  
Fax: +49 (0) 89 90508-492

Baldor ASR AG  
Telephone: +41 (0) 52 647 4700  
Fax: +41 (0) 52 659 2394

Australian Baldor Pty Ltd  
Telephone: +61 2 9674 5455  
Fax: +61 2 9674 2495

Baldor Electric (F.E.) Pte Ltd  
Telephone: +65 744 2572  
Fax: +65 747 1708

Baldor Italia S.R.L.  
Telephone: +39 (0) 11 56 24 440  
Fax: +39 (0) 11 56 25 660

---

## Safety Notice

Only qualified personnel should attempt the start-up procedure or troubleshoot the equipment. The equipment may be connected to other machines that have rotating parts or parts that are controlled by the equipment. Improper use can cause serious or fatal injury. Only qualified personnel should attempt to start-up, program or troubleshoot the equipment.

### Precautions



**WARNING:** Do not touch any circuit board, power device or electrical connection before you first ensure that no high voltage is present at this equipment or other equipment to which it is connected. Electrical shock can cause serious or fatal injury. Only qualified personnel should attempt to start-up, program or troubleshoot the equipment.



**WARNING:** Be sure that you are completely familiar with the safe operation and programming of the equipment. The equipment may be connected to other machines that have rotating parts or parts that are controlled by this equipment. Improper use can cause serious or fatal injury. Only qualified personnel should attempt to program, start-up or troubleshoot the equipment.



**WARNING:** The stop input to the equipment should not be used as the single means of achieving a safety critical stop. Drive disable, motor disconnect, motor brake and other means should be used as appropriate. Only qualified personnel should attempt to program, start-up or troubleshoot the equipment.



**WARNING:** Improper operation or programming may cause violent motion of a motor shaft and driven equipment. Be certain that unexpected motor shaft movement will not cause injury to personnel or damage to equipment. Peak torque of several times the rated motor torque can occur during control failure.



**CAUTION:** The safe integration of a device into a machine system is the responsibility of the machine designer. Be sure to comply with the local safety requirements at the place where the machine is to be used. In Europe these are the Machinery Directive, the ElectroMagnetic Compatibility Directive and the Low Voltage Directive. In the United States this is the National Electrical code and local codes.



**CAUTION:** Electrical components can be damaged by static electricity. Use ESD (electro-static discharge) procedures when handling electronic components.

## 2.1 Overview

Baldor controllers are normally programmed using the MintMT language.

MintMT is a structured form of Basic, custom designed for either stepper or servo motion control applications. It was devised to allow users to quickly get started with simple motion control programs. In addition, MintMT includes a wide range of powerful commands for complex applications. The MintMT language consists of two basic components. The interpreter level that gives the MintMT language its multi-tasking functionality and the underlying Mint Motion Library (MML) which provides the hardware access. For all motion control and hardware access MintMT calls MML functions.

The PC Developer Libraries allow PC based applications to be written that communicate with MintMT controllers and access functions in the MML. This is achieved using the MintMT ActiveX Control which is a common API (Application Program Interface) for the range of MintMT based motion controllers.

Features include:

- Update firmware into FLASH / RAM
- Compile and download MintMT programs
- Immediate command mode interface to controllers which allows PC applications to call MML functions
- Read / Write to Dual Port RAM
- Support for the ASCII Mint Comms Protocol.

The MintMT ActiveX Control is suitable for use under Windows 95, 98, ME, NT and 2000.

## 2.2 Abbreviations

The following abbreviations are used in this document.

MML	Mint Motion Library
ICM	Immediate Command Mode
DPR	Dual Port RAM
IDE	Integrated Development Environment
MFC	Microsoft Foundation Classes



## 3.1 What is an ActiveX control?

ActiveX controls, formerly known as OLE controls or OCX controls, are components (or objects) that can be inserted into an application to reuse packaged functionality. For example, the ActiveX controls that are included with MintMT controllers allow PC applications to communicate with controllers to allow complete machine control from a PC.

A key advantage of ActiveX controls is that they can be used in applications written in many programming languages, including development environments such as:

- Microsoft Visual C++
- Microsoft Visual Basic
- Borland Delphi
- National Instruments LabView

Any development environment that supports ActiveX controls can use the MintMT ActiveX control making it a very versatile interface to the controller.

## 3.2 Installing the PC developer libraries

From the Baldor Motion Toolkit CD install the PC Developer Libraries from the MintMT section. This will install the MintMT ActiveX control along with various header files and examples under different development environments.

**Note:** WorkBench v5 uses the MintMT ActiveX control. If you have installed a copy of WorkBench v5 then there will already be a copy of the ActiveX control installed.

The MintMT installation folder includes a number of sub folders:

- Apps MintMT firmware for all controllers
- Bin Any executable utilities
- Docs Documentation on building PC based applications that use the MintMT ActiveX control.
- Example A selection of example applications under a variety of development environments
- Include All header files required for building PC based applications.

The ActiveX control itself will be installed into the Windows system directory.

---

## 3.3 MintMT ActiveX control components

The MintMT ActiveX control contains several components:

### Command Prompt

This component is a fully functional MintMT command prompt. It can be included in an application and provides an interface to the controller allowing MintMT keywords to be executed.

### Terminal Window

This component is an output window and can be configured to display ASCII text received from either the serial port or Dual Port RAM.

### Mint Controller

This component allows access to all functions within the ActiveX control. Both the Command Prompt and the Terminal Window interface with the Mint Controller component of the ActiveX to perform function calls.

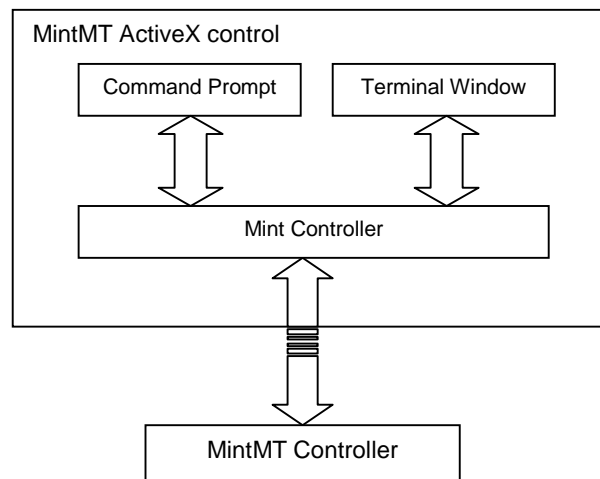


Figure 1 - MintMT ActiveX architecture

## 4.1 Overview

Using the Mint Motion Library (MML) function set, the MintMT ActiveX control provides all of the functionality that is available in the MintMT programming language. In addition to the MML functions the MintMT ActiveX control has extra functionality to allow the initialization of controllers. This includes updating firmware, downloading MintMT programs and running MintMT programs.

This allows the MintMT ActiveX control to be used for complete machine control.

## 4.2 Communicating with controllers

In order to use the MintMT ActiveX control to communicate with a MintMT controller the MintMT ActiveX control must be included in the application. Before any communication to the controller can take place a handle to the controller must be created. This tells the MintMT ActiveX control what type of controller is connected and the physical medium to use to communicate with the controller.

For example:

NextMove PCI, Card number, Node number

MintDrive", COM port, Baud rate, Node number.

Once this has been done then any MintMT ActiveX control function can be called.

Below are examples of creating handles to various controllers. Where MintController is the name of the MintMT ActiveX control in the project.

The following examples are in Microsoft Visual Basic.

### NextMove PCI

```
MintController.setNextMovePCIILink 0, 0
```

Creates a handle to a NextMove PCI controller which is card number 0 and node number 0.

### NextMove BX

```
MintController.setNextMoveBXLink 0, 1, 57600, True
```

Creates a handle to a NextMove BX controller which is node number 0 connected via com port 1 at a baud rate of 57600. The final `True` parameter specifies that the port should be opened for use.

---

## MintDrive<sup>II</sup>

```
MintController.setMintDrive2Link 0, 1, 57600, True
```

Creates a handle to a MintDrive<sup>II</sup> controller which is node number 0 connected via com port 1 at a baud rate of 57600. The final `True` parameter specifies that the port should be opened for use.

## FlexDrive<sup>II</sup>

```
MintController.setFlexDrive2Link 0, 1, 57600, True
```

Creates a handle to a FlexDrive<sup>II</sup> controller which is node number 0 connected via com port 1 at a baud rate of 57600. The final `True` parameter specifies that the port should be opened for use.

## Flex+Drive<sup>II</sup>

```
MintController.setFlexPlusDrive2Link 0, 1, 57600, True
```

Creates a handle to a Flex+Drive<sup>II</sup> controller which is node number 0 connected via com port 1 at a baud rate of 57600. The final `True` parameter specifies that the port should be opened for use.

**Note:** The MintMT ActiveX control can be used to communicate with legacy controllers (EuroSystem) and controllers running older versions of Mint (Mint v3.x and Mint v4.x) via the ASCII comms protocol. See section 5.4 for more details.

## 4.3 Downloading firmware

Firmware is the operating system that runs on the controller. In the same way that a PC runs Windows, Baldor controllers run MintMT. The firmware allows controllers to perform all the motion control, digital and analog I/O sampling, ICM decoding and run MintMT programs.

On stand alone controller MintDrive<sup>II</sup> and NextMove BX the firmware is stored in non-volatile memory and is restored at power up. Controllers that reside in the PC, NextMove PCI, do not store the firmware in non-volatile memory and need the firmware to be downloaded each time they are powered up.

The firmware for Baldor controllers is stored in a .CHX file. The version of the firmware can be examined before it is downloaded to the controller using the version locator utility on MintVer on the Baldor Motion Toolkit CD. MintVer is installed with WorkBench v5 and adds the menu option MintVersion to the Windows right click menu for firmware files.

Once a handle has been created to the controller this can be used to communicate with the controller. Firmware is downloaded using the function `doUpdateFirmware`. This function accepts a string that is the path of the firmware including the file name and extension.

For example. `MintController` is the name of the MintMT ActiveX control in the project.

---

The following examples are in Microsoft Visual Basic:

```
MintController.doUpdateFirmware "C:\MintMT\Firmware\nmPCI.CHX"
```

will update the firmware on the controller with the file nmPCI.CHX from the path C:\MintMT\Firmware.

The update firmware function will use the handle created using the `set...Link` function and pass the firmware to the controller via the appropriate communication channel.

## 4.4 MintMT programs

The MintMT ActiveX control can be used to download and run compiled MintMT programs to controllers.

During development MintMT programs are stored in a ASCII text format as a .MNT file. To run this on the controller it must be compiled into an executable format using WorkBench v5. Within WorkBench v5, rather than running the program select the Tools - Program - Compile to File menu option. This will produce a Mint executable file (.MEX).

The MintMT executable file can be downloaded to the controller using the function `doMintFileDownload`. This function accepts a string that is the path of the file including the file name and extension.

For example, where `MintController` is the name of the MintMT ActiveX control in the project.

The following examples are in Microsoft Visual Basic:

```
MintController.doMintFileDownload "C:\MintMT\Programs\myMintProgram.MEX"
```

Will download the file `myMintProgram.MEX` from the path `C:\MintMT\Programs`

If the firmware on the controller and the Mint executable file are incompatible an error message will be displayed. The version of MintMT running on the controller must be compatible with the version of the compiler used to compile the MintMT executable.

Version details of the .MEX file can be obtained before it is downloaded to the controller using the version locator utility on the Baldor Motion Toolkit CD.

Once the program has been downloaded to the controller it can be run using the function `doMintRun`. Calling this function will run the MintMT program immediately. Execution of the current program can be halted by calling the function `doMintBreak`.

## 4.5 Mint Motion Library function calls

Motion applications can be written on the host PC by calling Mint Motion Library functions via the MintMT ActiveX control. The MintMT ActiveX control uses the immediate command mode (ICM) interface to communicate with controllers. The ICM interface is a packet based communication protocol that allows PC based applications to call MML functions on the controller. The MintMT ActiveX control will create, send, receive and decode ICM packets to allow complete machine control via a PC based application.

---

Figure 2 shows a diagram of the MintMT controller software architecture.

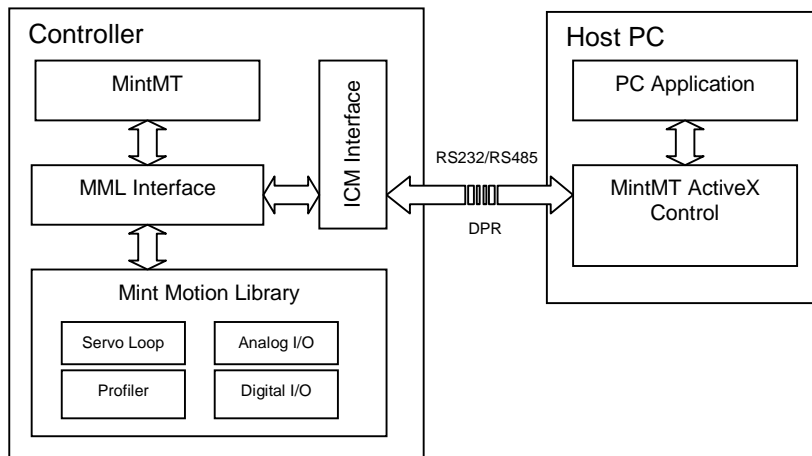


Figure 2 - MintMT software architecture

When an MML function is called by a PC application the ActiveX sends a data packet to the controller. This data packet contains data about the MML function to be called and any associated parameters. The controller then processes the MML function call and returns a data packet to the PC application containing an error code and any returned data from the MML function call.

All MML function call requests pass through the MML Interface, so only one MML function call can be in progress at any time. This means that there is no danger of data contention. MintMT controllers can process ICM function calls while running MintMT programs.

Calling functions from a PC application is particularly useful if there is a large amount of processing to do, for example the calculation of multi-axis paths. The PC application can do the processing and send commands to the controller. For example, the MintMT program could be programmed to handle the I/O and the PC application could be used to calculate positional data and load moves using the MML function call `setVectorA()`.

There is a one-to-one correlation between MintMT keywords that call MML functions, and the MintMT ActiveX control functions. For example, within a MintMT program, the `MOVER` keyword is used to create a relative positional move on an axis:

```
MOVER.0 = 10
```

Equivalent functionality can be achieved via the MintMT ActiveX control calling the function `setMoveR`.

```
MintController.setMoveR 0, 10
```

The keyword has been prefixed with `set`.

---

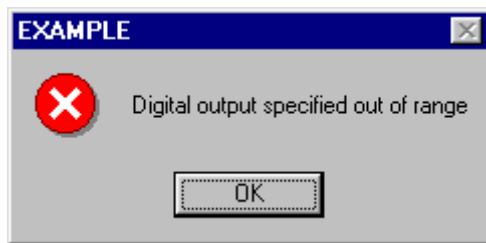
#### 4-4 Using the MintMT ActiveX Control

---

All MintMT keywords that call MML functions can be called via the MintMT ActiveX control. They are prefixed with `set` for writes, `get` for reads and `do` for commands.

## 4.6 Error Handling

Depending on the state of the controller and the parameters passed, function calls made by the MintMT ActiveX control may fail. If a function call fails for any reason a dialog box will be displayed containing a description of the error. For example:



There may be instances within an application where corrective action may be preferable to displaying a dialog box in the case of a function call failing.

If any property or method call on the MintController object fails it will generate an exception. In programming terms, an exception is an unexpected event that happens during the execution of a program.

Many modern programming languages now support some form of exception handling. Essentially, an exception is an error, and this error is 'thrown' from the place in the code where it occurred and (hopefully) 'caught' somewhere else in the code. This somewhere else could be within the same function, or the calling function or the calling function of the calling function, etc.

The exception will contain a description and also an error code. If it is not handled then a dialog box with a description of the error will be displayed. Handling the exception allows the error code returned by the function to be checked and appropriate action to be taken.

The example below is written in Visual C++ and will catch OLE Dispatch Exceptions. The error code is checked and a dialog box will display a different message based on the return code (`m_controller` is the name of the MintMT ActiveX control in the project).

---

```

// Try function call
try
{
    // Enable the drive
    m_controller.setDriveEnable ( 0, true );
}

// Catch any errors
catch ( COleDispatchException *pe )
{
    // Check the error code
    switch ( pe->m_wCode )
    {
        case erMOTION_ERROR:
            MessageBox ( "Drive in error, please re-start" );
            break;
        default:
            pe->ReportError();
            break;
    }

    pe->Delete();
}

```

The error code is part of the COleDispatchException class and can be accessed using the member variable `m_wCode`.

All error codes are defined in the format `erXXXX` in the header file `HOST_DEF.H` that can be included in Visual C++ files.

The same example in Visual Basic would be:

```

Private Sub Command1_Click()
    ' If an error occurs continue processing
    On Error GoTo Handle_Errors

    ' Enable the drive
    MintController.setDriveEnable 0, True
End Sub

Sub Handle_Errors()
    ' Check the error code
    Select Case Err.Number
        Case erMOTION_ERROR
            MsgBox "Drive in error, please re-start"
        Case Else
            MsgBox Err.Description
    End Select
End Sub

```

MintController is the name of the MintMT ActiveX control in the project.

The error code can be accessed using the `Err` structure. The element `Number` contains the error code.

---

## 4-6 Using the MintMT ActiveX Control

---

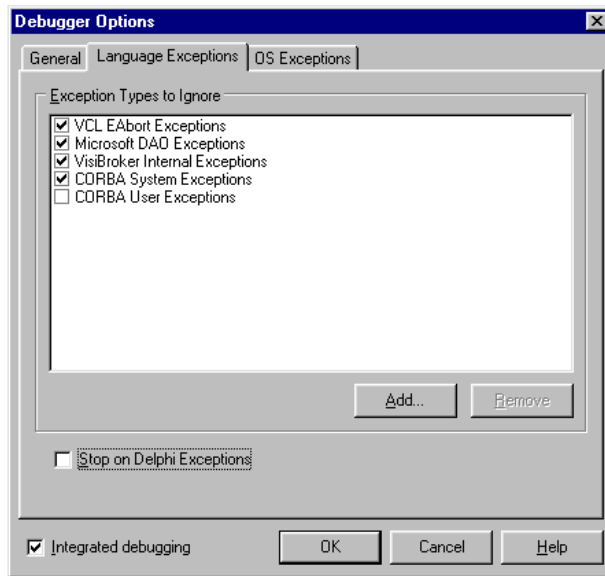
All error codes are defined in the format `erXXXX` in the header file `HOST_DEF.BAS` that can be included in Visual Basic projects.

A similar example in Borland Delphi would be:

```
// Try function call
try
  MintController.setDriveEnable ( 0, true );
// Catch any errors
except
  On E: Exception do
    MessageBox ( 0, pchar(E.Message), 'MintMT ActiveX Call Failed', 0 );
end;
```

`MintController` is the name of the MintMT ActiveX control in the project.

To prevent Delphi from halting program execution in the event of an exception the 'Stop on Delphi Exceptions' check box must be cleared. This is found in the 'Debugger Options' from the 'Tools' menu.



All error codes are defined in the format `erXXXX` in the header file `HOST_DEF.PAS` that can be included in Borland Delphi projects.

A complete list of error codes can be found in section 7.1.

**Note:** For more details on structured error handling see the manufacturer's documentation for the development environment being used.

---

## 4.7 Limitations of PC based applications

### 4.7.1 Events

There are a number of event types supported by MintMT such as ONERROR, FASTIN, TIMER, etc. These events are only supported by NextMove PCI. Serial controllers do not support MintMT events on the PC application. See section 5.1 for more detail on using events.

### 4.7.2 Execution Speed

Commands called from the PC application execute slower than if called directly on the controller. See section 9-1 for example timings.

The PC application function calls will not take priority over a MintMT program running on the controller, the processing of function calls will be shared. The MML Interface will process a MML function call request from MintMT followed by an external request. Running a PC application that communicates with the controller while also running a MintMT program will result in both applications running slower than if they were executed separately.

The MintMT program will be slowed due to the sharing of the MML Interface with the ICM function calls from the PC application and the overhead of processing the ICM packets.

On serial controllers the baud rate at which the serial communication is running affects the impact that the PC application can have. The higher the baud rate the more characters can be sent to the controller. This results in faster ICM function calls from the PC application but will cause any MintMT program to run more slowly. Conversely, the lower the baud rate the slower the character transmission rate to the controller. This results in slower ICM function calls from the PC application, but gives more CPU time to the MintMT program.

On Dual Port RAM controllers such as NextMove PCI, the ICM data is transferred to the controller over Dual Port RAM. This is shared memory and the PC can write to this without affecting the application running on the controller. When an ICM transaction occurs over Dual Port RAM the PC writes the function call details before interrupting the controller. This means the processing overhead of ICM function calls on Dual Port RAM controllers is significantly lower than on serial controllers.

## 4.8 Extended functionality of DPR controllers

Dual Port RAM controllers are resident within the PC. Rather than using a serial communication link to the PC like RS232 or RS485 they use Dual Port RAM. This is an area of shared memory on the controller that both the PC and the controller can access.

The PC can read and write to Dual Port RAM at the same time as the controller, the controller can be reading one block of data while the PC is writing another. This allows much faster data transfer than on a serial controller.

As the controller is resident within the PC certain locations of Dual Port RAM act as interrupt locations so an application on the controller can interrupt the PC or, an application on the PC can interrupt the controller. This allows support for MintMT events in a PC application.

---

## 4.8.1 Accessing Dual Port RAM

There are a number of functions to read and write data to Dual Port RAM.

Dual Port RAM is 32-bits wide and is addressable in the range 0x000 to 0xFFFF. See section 8 for details on the contents of Dual Port RAM.

Read / write a floating point value:

```
getFloat ( short nAddress, float *pfValue );  
setFloat ( short nAddress, float fValue );
```

Read / write a signed 32-bit value

```
getLong ( short nAddress, long *plValue );  
setLong ( short nAddress, long lValue );
```

Read / write a signed 16-bit value

```
getWord ( short nAddress, short *pnValue );  
setWord ( short nAddress, short nValue );
```

## 4.8.2 Event handling

There are a number of event types supported by MintMT such as ONERROR, FASTIN, TIMER, etc. The MintMT ActiveX control allows event handlers to be installed in the PC application rather than in a MintMT application. When an event occurs, the controller will interrupt the PC application passing details of the event type and any associated parameters.

This allows the PC application to install event handling routines that will be called by the controller interrupting the PC. Support for events allows complete MintMT programs to be converted to run on a PC application rather than on the controller.

**Note:** Event generation within a PC application relies on the ability of the controller to interrupt the PC application and is currently only supported by NextMove PCI. For other controllers, the event handlers must be placed in a MintMT program.



## 5.1 Event Handling

The MintMT ActiveX control supports the same selection of event handlers as MintMT, these are:

- Serial Receive event                      a character has been received
- Error event                                    an asynchronous error occurred on the NextMove card
- CAN 1 event                                  an event on CAN bus 1 (CAN Open)
- CAN 2 event                                  event on CAN bus 2 (Baldor CAN)
- Stop switch event                          a stop switch has become active
- Fast position latch event                  an axis has latched position
- Timer event                                    the timer event period has expired
- Digital input event                          a digital input has become active
- DPR event                                     the user generated a DPR event
- Move buffer low event                      the numbers of moves in a move buffer drops below a specified threshold
- Axis idle event                              an axis has become idle
- Reset event                                  the controller has reset for some reason
- Unknown event                                unknown event.

**Note:** Event generation within a PC application relies on the ability of the controller to interrupt the PC application and is currently only supported by NextMove PCI. For other controllers, the event handlers must be placed in a MintMT program.

The events are prioritized in the following order:

Priority	Event	Supported By
0: Highest	Serial Receive	All controllers
1	Error	NextMove PCI
2	CAN 1 (CANOpen)	NextMove PCI
3	CAN 2 (Baldor CAN)	NextMove PCI
4	Stop switch	NextMove PCI
5	Fast position latch	NextMove PCI
6	Timer	NextMove PCI
7	Digital input	NextMove PCI
8	Comms	NextMove PCI
9	DPR event	NextMove PCI
10	Move Buffer Low	NextMove PCI

---

Priority	Event	Supported By
11	Axis Idle	NextMove PCI
12: Lowest	Unknown	NextMove PCI

The Reset event only available on NextMove PCI. It is not generated by the controller but by the device driver and is not subject to the priority scheme.

In order for the user to be informed of an event, a handler must be installed. Event handlers may be installed in applications running on the controller (MintMT) and applications running on the PC. If both an embedded handler (MintMT or an embedded application) and a PC handler are installed then only the handler on the controller (MintMT or embedded) will be called.

When an event occurs and the appropriate handler is installed the handler is called. A higher priority event will interrupt a lower priority event. If multiple events occur within 2ms, then the above priority system will be used to decide which event to call.

### 5.1.1 Installing ActiveX event handlers

The events listed above will appear as part of the MintMT Controller object when the MintMT ActiveX control has been included in the project. Within the development environment being used, code can be added to an event handler. This code will be called when the controller generates an event of a specific type.

In order for the ActiveX control to receive events both the device driver and the controller must be told for which events handlers exist. This is done using an `installXXXXhandler` function. These functions are passed a boolean value, either `true` to install the event handler or `false` to uninstall the event handler.

Once the handler has been installed the event will get generated by the controller, this will interrupt the PC. The device driver will catch the interrupt and set an event in the ActiveX control. This event will then call the event handler code in the PC application.

#### Serial Receive

To install a serial receive event handler the function `installSerialReceiveEventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will register the event handler and allow error events to be received. Passing `false` will uninstall the event handler and prevent error events being generated or received.

When a character is transmitted from the controller to the PC a serial receive interrupt is generated. This event is used by the output window in WorkBench v5. Installing a serial receive event handler in another PC application will disable the output window in WorkBench v5.

#### Error event

To install an error event handler the function `installErrorEventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow error events to be received. Passing `false` will uninstall the event handler and prevent error events being generated or received.

---

Only asynchronous errors will generate calls to the error event handler. If a function call returns an error code this will not call the error handler. Asynchronous errors are errors that are not synchronous with program execution, like an axis exceeding its following error limit.

#### **Fast position latch event**

To install a fast position latch event handler the function `installFastInEventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow fast position latch events to be received. Passing `false` will uninstall the event handler and prevent fast position latch events being generated or received.

The source of the call to the fast position latch event handler can be established using the function `getFastLatch`.

#### **Stop switch event**

To install a stop switch event handler the function `installStopSwitchEventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow stop switch events to be received. Passing `false` will uninstall the event handler and prevent stop switch events being generated or received.

The source of the call to the stop switch event handler can be established using the function `getStopSwitch`.

#### **CAN event on bus 1**

To install a CAN event handler for CAN bus 1 the function `installCAN1EventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow CAN bus 1 events to be received. Passing `false` will uninstall the event handler and prevent CAN bus 1 events being generated or received.

The cause of the call to the CAN event handler can be established using the functions `getCANEvent` and `getCANEventInfo`.

#### **CAN event on bus 2**

To install a CAN event handler for CAN bus 2 the function `installCAN2EventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow CAN bus 2 events to be received. Passing `false` will uninstall the event handler and prevent CAN bus 2 events being generated or received.

The cause of the call to the CAN event handler can be established using the functions `getCANEvent` and `getCANEventInfo`.

---

### Timer event

To install a timer event handler the function `installTimerEventEventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow timer events to be received. Passing `false` will uninstall the event handler and prevent timer events being generated or received.

The period of the timer event can be set using the function `setTimerEvent`.

### Digital input event

To install a digital input event handler the function `installInputEventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow digital input events to be received. Passing `false` will uninstall the event handler and prevent digital input events being generated or received.

When the input event handler is called, it is passed two parameters, a bank number and a bit pattern of activated inputs. In the case of NextMove PCI where each PCI card has a bank of I/O (the main board is bank 0, the first expansion board is bank 1 and the second expansion card is bank 2) the expansion cards are read sequentially. It is advisable to set up a mask of digital inputs that will generate events using the function `setIMask`.

### Comms location changed event

To install a comms event handler the function `installCommsEventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow comms events to be received. Passing `false` will uninstall the event handler and prevent comms events being generated or received.

When the comms event handler is called, it is passed a bit pattern indicating the changed comms.

Comms events will be generated when an external source modifies the contents of comms locations 1 to 5 using the function `setComms`. Only external access to the comms array will call the embedded comms event handler. Calling the MML function `setComms` in an embedded application will not generate a call to an embedded comms event handler.

### DPR event

To install a DPR event handler the function `installDPREventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow DPR events to be received. Passing `false` will uninstall the event handler and prevent DPR events being generated or received.

DPR events are generated by an application running on the controller calling the function `doDPREvent`. This can be done in MintMT using the keyword `DPREVENT`. This function accepts a 16-bit code which is passed to the DPR event handler running on the PC application.

---

### Move buffer low event

To install a move buffer low event handler the function `installMoveBufferLowEventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow move buffer low events to be received. Passing `false` will uninstall the event handler and prevent move buffer low events being generated or received.

When the move buffer low event is generated the function called is passed a 16-bit code representing the axes which have reached the move buffer threshold. The move buffer threshold is set with the function `setMoveBufferLow`.

### Axis idle event

To install an axis idle event handler the function `installAxisIdleEventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow axis idle events to be received. Passing `false` will uninstall the event handler and prevent axis idle events being generated or received.

When the axis idle event is generated the function called is passed a 16-bit code representing the axes which have become idle.

### Unknown event

To install an unknown event handler the function `installUnknownEventHandler` must be called to install the event handler with the device driver and the controller. This function accepts a single boolean parameter. Passing `true` will install the event handler. Passing `false` will uninstall the event handler.

The unknown event handler should never get called. It will catch any unrecognized codes.

### Reset event

To install a reset event handler the function `installResetEventHandler` must be called to install the event handler with the device driver. This function accepts a single boolean parameter. Passing `true` will install the event handler and allow reset events to be received. Passing `false` will uninstall the event handler and prevent reset events being generated or received.

A reset event is generated when the controller resets unexpectedly. For example, in the case of the bus voltage dropping below a safe limit.

---

## 5.1.2 Controlling event generation

There are various functions that can be used to control event generation. The bit pattern accepted by these functions is as follows:

Bit	Event
0	<i>Reserved</i>
1	Error
2	CAN 1 (CANOpen)
3	CAN 2 (Baldor CAN)
4	Stop switch
5	Fast position latch
6	Timer
7	Digital input
8	Comms event from PC to controller (PC updated comms location)
9	Comms event from controller to PC (controller updated comms location)
10	DPR event from PC to controller
11	DPR event from controller to PC
12	<i>Reserved</i>
13	Move buffer low
14	Axis idle

The user can read which events are currently active using the function `getEventActive`.

Any currently pending events can be cleared selectively using the functions `getEventPending` and `setEventPending`. Clearing a set bit will clear the pending flag for that event. Passing a value of zero will clear all pending interrupts.

Once a handler has been installed the event generation can be disabled by using the function `setEventDisable`. Setting a bit will disable the generation of that type of event. Setting this to zero will enable all events which have a handler installed.

The function `getEventDisable` will return a bit pattern of any currently disabled interrupts. By default all digital inputs will generate events when they become active. These digital inputs can be masked so that they do not generate events using the function `setIMask`. This accepts a bank number and a bit pattern of inputs. Bit 0 corresponds to input 0; if the bit is set then the digital input will generate an event when it becomes active.

The user can interrupt an embedded application using the function `doDPREvent`. This will interrupt the controller and pass a 16-bit code to the DPR event handler on the controller.

---

## 5.2 Terminal window

The MintMT ActiveX control has a number of secondary components that work with the MintController object. One of these is the MintTerminal object. This is a window which will display any ASCII output from the controller and allow ASCII input to the controller.

The form must have a MintController object for the terminal window to function. All communications to and from a controller pass through the MintController object. In order for the terminal window to function it must be linked to a specific controller. Once this link has been established all ASCII text from the controller will be displayed in this window and all key presses performed in the window will be sent to the controller as ASCII text.

Assuming the application already has a MintController object that can be used to communicate with a controller, the ID of this controller object must be passed to the terminal window to create a link.

The ID of the MintController object can be obtained using the function `getID`.

This value can then be used to link the terminal window to a controller using the function `setMintControllerID`.

The below example is written in Microsoft Visual C++ (`m_controller` is the name of the MintController object in the project and `m_terminal` is the name of the MintTerminal object in the project).

```
// create handle to controller
m_controller.setNextMovePCILink ( 0, 0 );
// get ID of controller object
BSTR bstrID = m_controller.getID ();
// link the terminal window to the controller
m_terminal.setMintControllerID ( bstrID );
// clear up
::SysFreeString ( bstrID );
```

Notice the clean up section at the bottom that allows the re-use of memory used by the BSTR. Attempting to perform the code in one line, for example:

```
m_terminal.setMintControllerID ( m_controller.getID () );
```

will result in a memory leak, as the BSTR will go out of scope before it can be freed.

The example below is written in Microsoft Visual Basic (`MintController` is the name of the MintController object in the project and `MintTerminal` is the name of the MintTerminal object in the project).

```
Dim sID As String
' create handle to controller
MintController.setNextMovePCILink 0, 0
' get ID of controller object
sID = MintController.getID
' link the terminal window to the controller
MintTerminal.setMintControllerID sID
```

---

## 5.3 Command prompt

As well as the MintTerminal object the MintMT ActiveX control also has a MintCommandPrompt object that works with the MintController object. This allows the user to enter lines of MintMT code that will be compiled and executed in the window.

The form must have a MintController object for the MintCommandPrompt object to function. All communications to and from a controller pass through the MintController object. In order for the command prompt window to function it must be linked to a specific controller and given the compiler version and symbol table information to allow compilation.

Assuming the application already has a MintController object that can be used to communicate with a controller, the ID of this controller object must be passed to the command prompt window to create a link.

The ID of the MintController object can be obtained using the function `getID`

This value can then be used to link the MintCommandPrompt object to a MintController object using the function `setMintControllerID`.

The command prompt will also need a valid symbol table for the controller. The function `getSymbolTableForController` will ensure that a compatible version of the symbol table exists on the PC, if not it will upload one from the controller.

Following this the version of the compiler needs to be set. This is determined by the version of the MintMT virtual machine that the firmware is using. The function `getFirmwareMintVMBuild` can be used to interrogate the controller and get the version of MintMT virtual machine being used by the firmware. The command prompt can then be set up using the function `setCompiler`.

The below example is written in Microsoft Visual C++ (`m_controller` is the name of the MintController object in the project and `m_command` is the name of the MintCommandPrompt object in the project).

```
BSTR bstrID;
BSTR bstrSymbolTable;
// create handle to controller
m_controller.setNextMovePCIILink ( 0, 0 );
// get ID of controller object
bstrID = m_controller.getID ();
// link the command prompt to the controller
m_command.setMintControllerID ( bstrID );
// get symbol table for controller
m_controller.getSymbolTableForController ( VARIANT_FALSE, &bstrSymbolTable );
// link the command prompt to the symbol table
m_command.setSymbolTable ( bstrSymbolTable );
// set the compiler version for the command prompt
m_command.setCompiler ( m_controller.getFirmwareVMBuild () );
// clear up
::SysFreeString ( bstrID );
::SysFreeString ( bstrSymbolTable );
```

The example below is written in Microsoft Visual Basic (`MintController` is the name of the MintController object in the project and `MintCommandPrompt` is the name of the Command Prompt object in the project).

---

```
Dim sID As String
Dim sSymbolTable As String
` create handle to controller
MintController.setNextMovePCIILink 0, 0
` get ID of controller object
sID = MintController.getID
` link the command prompt to the controller
MintCommandPrompt.setMintControllerID ( sID );
` get symbol table for controller
MintController.getSymbolTableForController False, sSymbolTable
` link the command prompt to the symbol table
MintCommandPrompt.setSymbolTable ( sSymbolTable );
` set the compiler version for the command prompt
MintCommandPrompt.setCompiler MintController.getFirmwareVMBuild
```

## 5.4 Communicating with legacy controllers

The ICM interface between used by MintMT is not compatible with older versions of ICM used by other versions of Mint.

### 5.4.1 Serial controllers

Normally all MML functions called by the MintMT ActiveX control use the ICM protocol. This is only supported by MintMT controllers. In order to communicate to with a non-MintMT controller the ASCII comms protocol must be used.

**Note:** This protocol can only be used to read and write to the comms array on the controller.

The elements of the comms array are accessed by the functions `getComms` and `setComms`. The protocol used by these functions can be changed using the function `setCommsANSI328Mode`. This function accepts a boolean parameter.

Passing `true` will enable the ASCII comms protocol and all future use of the functions `getComms` and `setComms` will use the ASCII comms protocol.

Passing `false` will disable the ASCII comms protocol and all future use of the functions `getComms` and `setComms` will use the ICM protocol.

### 5.4.2 Dual Port RAM controllers

In order to communicate with a Dual Port RAM controller running a older version of Mint the comms array can be used.

On these controllers the comms array exists within the locations 01D6 (hex) to 029B (hex) of Dual Port RAM and is an array of floating point values. The functions `getFloat` and `setFloat` can be used to access this area. Comms values are only written to even locations.

---

Comms array element	DPR Location
1	0x1D6
2	0x1D8
3	0x1DA
4	0x1DC
.	
.	
97	0x296
98	0x298
99	0x29A

## 6.1 Microsoft Visual C++

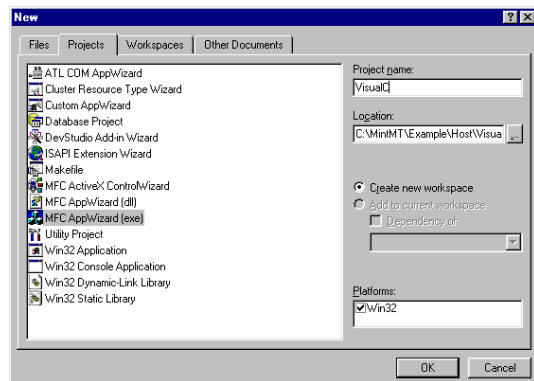
This section is a guide to creating a Microsoft Visual C++ dialog-only MFC application that uses the MintMT ActiveX control to communicate with a NextMove PCI controller.

**Note:** The MintMT ActiveX control cannot be used with Console Applications

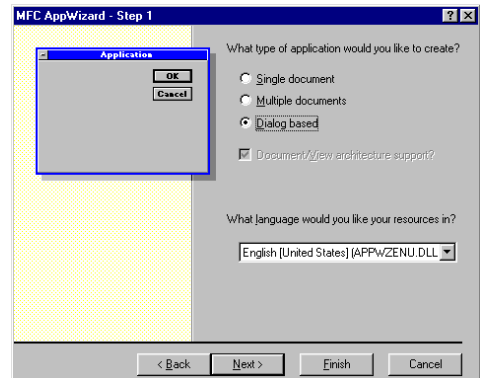
Although the example uses a NextMove PCI controller the same application can be used for a MintDrive<sup>™</sup> or NextMove BX by changing the type of controller.

This example uses Microsoft Visual C++ v6.0.

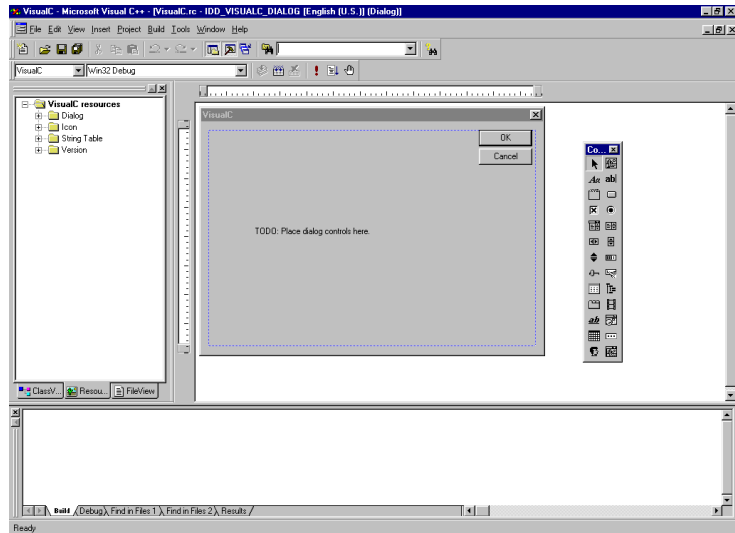
1. Open Microsoft Visual C++ and select New from the File menu. Select MFC AppWizard (exe) from the Projects tab. Enter a suitable name and click OK.



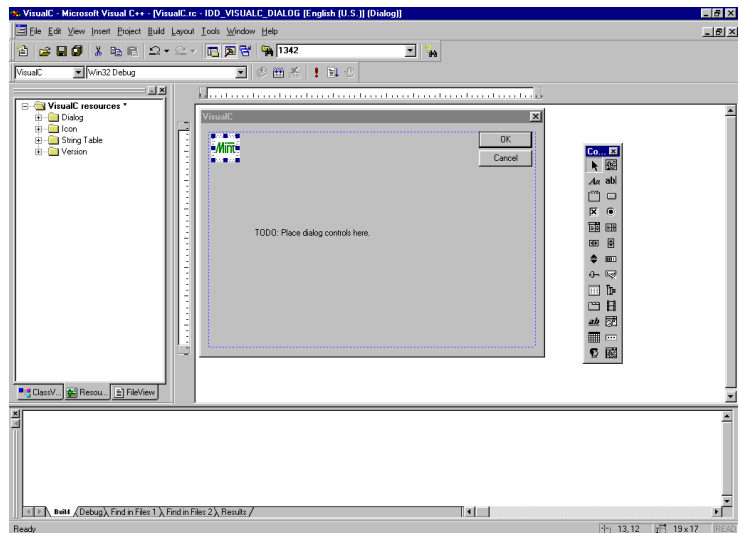
2. At step 1 of the wizard, select Dialog Based then click Finish.



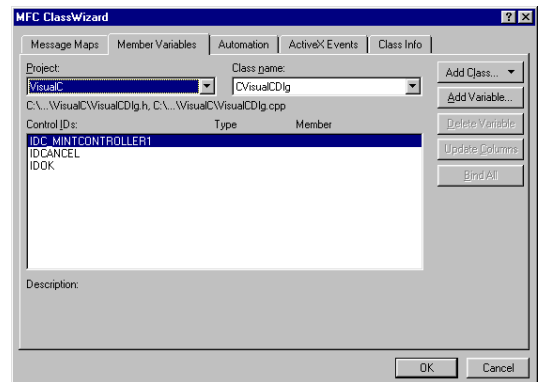
3. Click OK to accept the summary dialog. Microsoft Visual C++ will now display the resource view for the main dialog of the application.



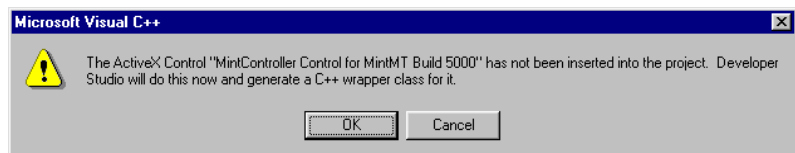
4. Right click on the dialog and choose Insert ActiveX Control.... Select the MintMT ActiveX control and click OK. In this example it is 'MintController Control for MintMT Build 5000'. The build number may vary depending on the version of the ActiveX control that is installed on the PC.  
When the ActiveX control has been inserted a Mint icon will appear on the dialog.



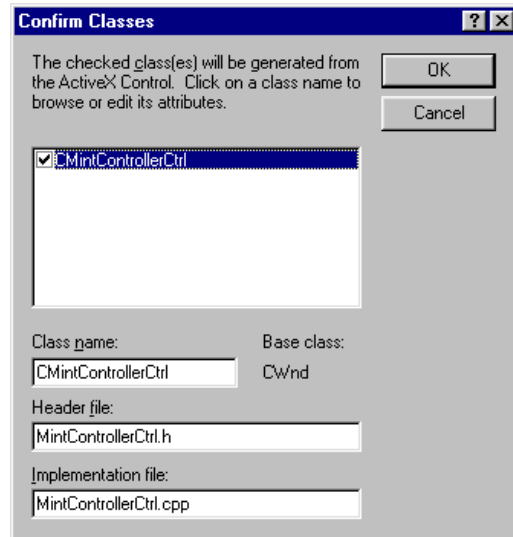
- 
5. Right click again on the dialog and choose Class Wizard.... On the Member Variables tab select the controller ID for the MintController (IDC\_MINTCONTROLLER1).



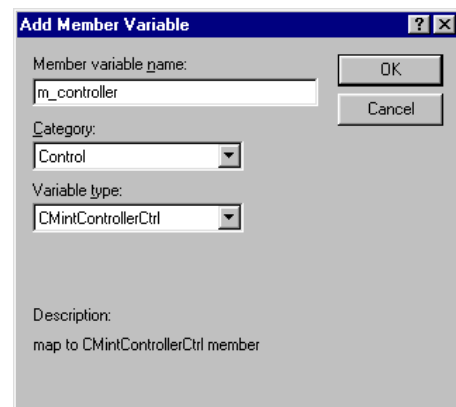
Click Add Variable. When asked whether Visual C++ should create a wrapper C++ class for the control, click OK.



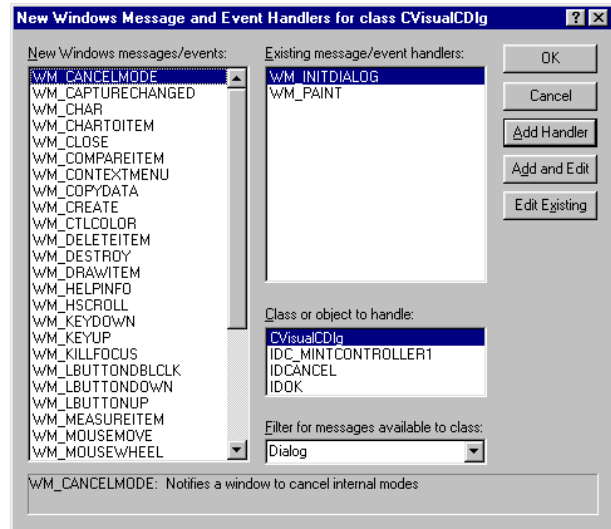
6. Click OK again to confirm the class and file names (CMintControllerCtrl, mintcontrollerctrl.h and mintcontrollerctrl.cpp respectively).



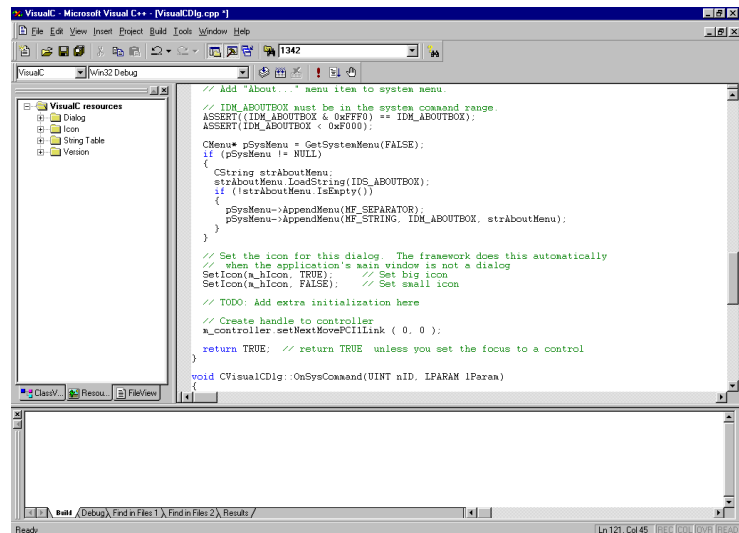
7. In the Add Member Variable dialog type in a suitable member variable name, for example 'm\_controller', then press OK.  
Click OK to dismiss the Class Wizard.



- When the application starts up it needs to create a handle to the controller. Right click on the dialog. Select Events... and select WM\_INITDIALOG, click the Edit Existing button.



This will open an editor window displaying the function that is called to initialize the dialog.



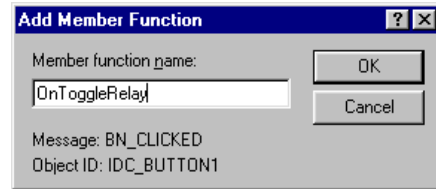
Here we need to create a handle to the controller to enable communication with the controller. At the bottom of the function before the return statement add the code:

```
m_controller.setNextMovePCILink ( 0, 0 );
```

---

This will create a handle to a NextMove PCI controller (Card 0, Node 0). All subsequent use of `m_controller` will use the NextMove PCI controller.

9. Add a button to the dialog and double click on it to add a handler function for the button press event. Type a suitable name for the function in the dialog box, for example 'OnToggleRelay'.



Visual C++ will then display an editor window with the cursor in the function. Add the code:

```
void CVisualCDlg::OnToggleRelay()
{
    int nState;

    // Read the state of the relay
    nState = m_controller.GetRelay ( 0 );
    // Invert the state of the relay
    m_controller.SetRelay ( 0, !nState );
}
```

This code will read the state of the relay and invert it. Run the application, when the button is pressed the relay will toggle.

**Note:** This example is installed by the PC developer libraries in the folder  
\\MintMT\\Example\\Host\\VisualC

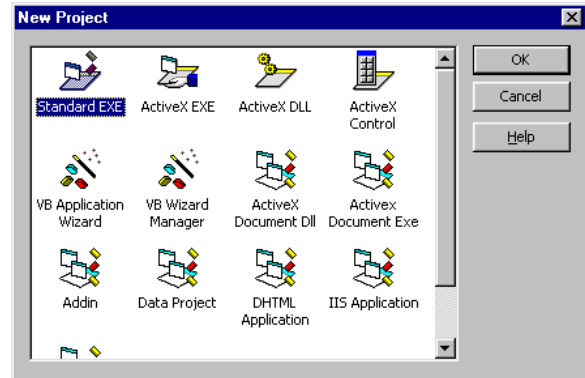
## 6.2 Microsoft Visual Basic

This section is a guide to creating a Microsoft Visual Basic application that uses the MintMT ActiveX control to communicate with a NextMove PCI controller.

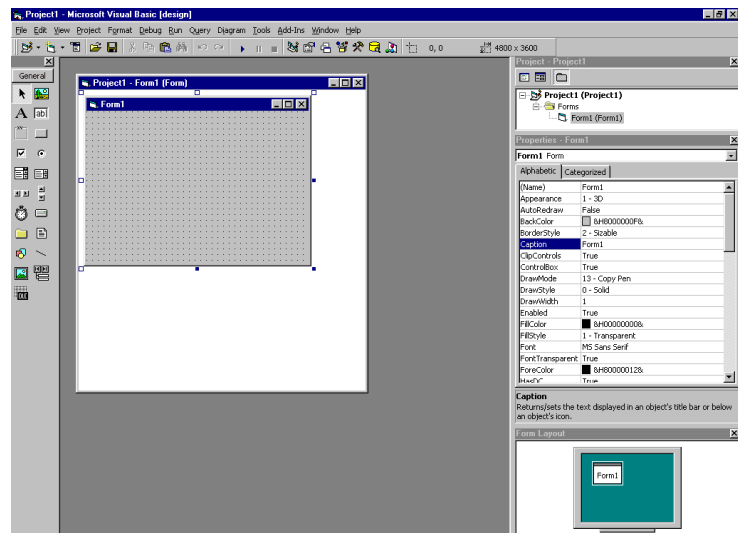
Although this example uses a NextMove PCI controller the same application can be used for a MintDrive<sup>II</sup>, NextMove BX or any other MintMT controller by changing the type of controller.

This example uses Microsoft Visual Basic v6.0.

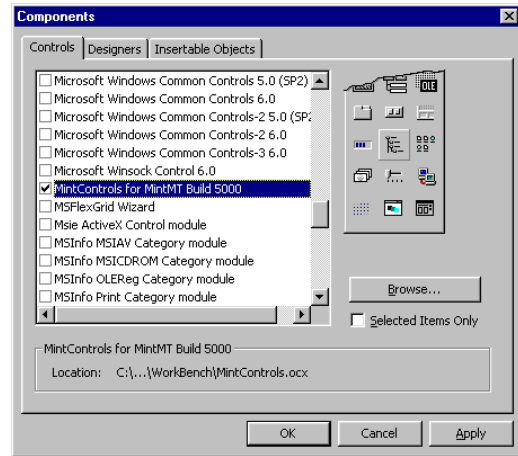
1. Open Microsoft Visual Basic and select New Project from the File menu. Select 'Standard .EXE' from the selection of project types and click OK.



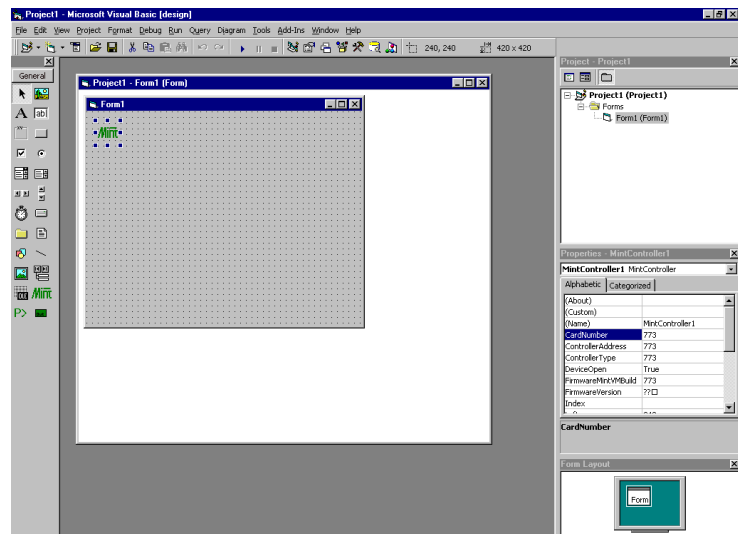
2. Microsoft Visual Basic will now display the development environment:



- From the Project menu select Components. This will display a dialog listing all ActiveX controls. Click the checkbox associated with the MintMT ActiveX control, then click OK. In this example it is 'MintController Control for MintMT Build 5000'. The build number may vary depending on the version of the ActiveX control that is installed on the PC.



- Once the MintMT ActiveX control is included in the project a Mint icon will appear in the toolbox. To use the MintMT ActiveX control in the project, place it on the form.



The properties of the ActiveX component include a field called (Name). This is the name used in the code to identify the ActiveX control, in this example it is 'MintController'.

- 
5. To use the MintMT ActiveX control to communicate with a controller a handle to the controller must be created. This can be done when the form loads. Double click on the form, to open an editor window with the cursor in the Form\_Load function. Add the code:

```
Private Sub Form_Load()  
    ' Create handle to controller  
    MintController.setNextMovePCIILink 0, 0  
End Sub
```

The above code will create a handle to a NextMove PCI controller, card number 0 and node number 0.

6. Add a command button to the form and double click on the button. This will open an editor window with the cursor in the function that is called when the command button is clicked. Add the code:

```
Private Sub Command1_Click()  
    Dim bState As Boolean  
  
    ' Read the state of the relay  
    bState = MintController.Relay(0)  
  
    ' Invert the state of the relay  
    If bState = False Then  
        bState = True  
    Else  
        bState = False  
    End If  
  
    ' Write the new relay state  
    MintController.Relay(0) = bState  
End Sub
```

This code will read the state of the relay and invert it. Run the application, when the button is pressed the relay will toggle.

**Note:** This example is installed by the PC developer libraries in the folder  
\\MintMT\Example\Host\VisualBasic

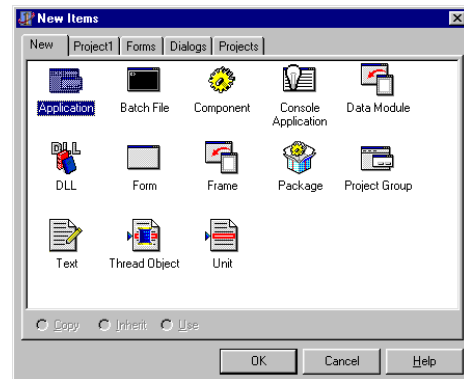
## 6.3 Borland Delphi

This section is a guide to creating a Borland Delphi application that uses the MintMT ActiveX control to communicate with a NextMove PCI controller.

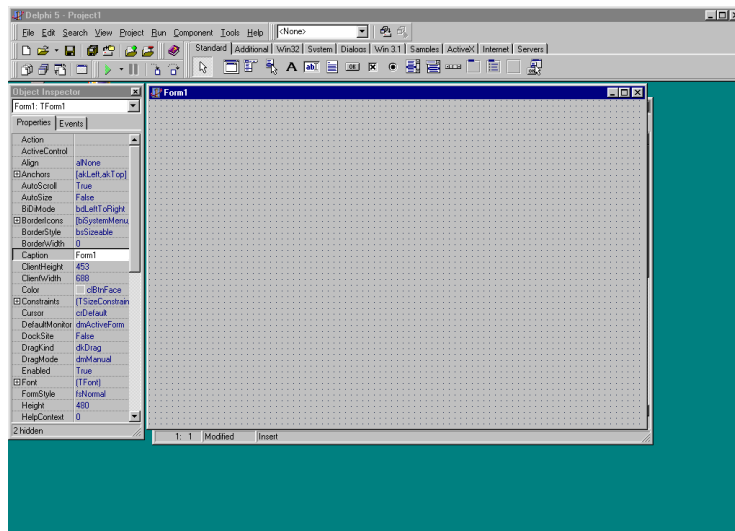
Although this example uses a NextMove PCI controller the same application can be used for a MintDrive<sup>II</sup>, NextMove BX or any other MintMT controller by changing the type of controller.

This example uses Borland Delphi v5.0.

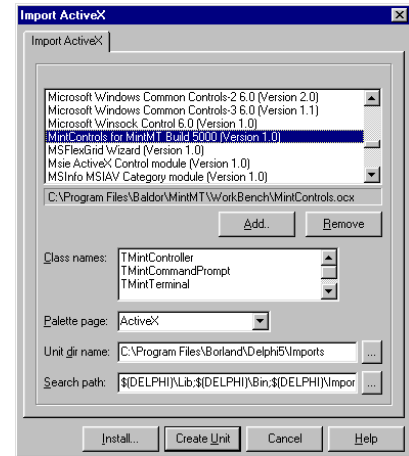
1. Open Borland Delphi and select 'New...' from the File menu. Select 'Application' from the selection of icons and click OK.



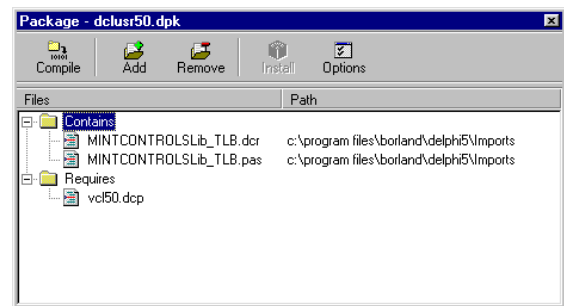
2. Borland Delphi will now display the development environment.



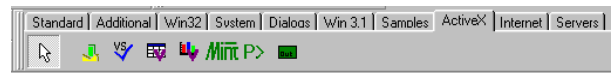
- From the Component menu select 'Import ActiveX Control...', this will display a dialog listing all ActiveX controls. Select the MintMT ActiveX control, then click 'Install' and accept the default options. In this example it is 'Mint Controls for MintMT Build 5000', the build number will vary depending on the version of the ActiveX control that is installed on the PC.



This will build a package containing the MintMT ActiveX control.

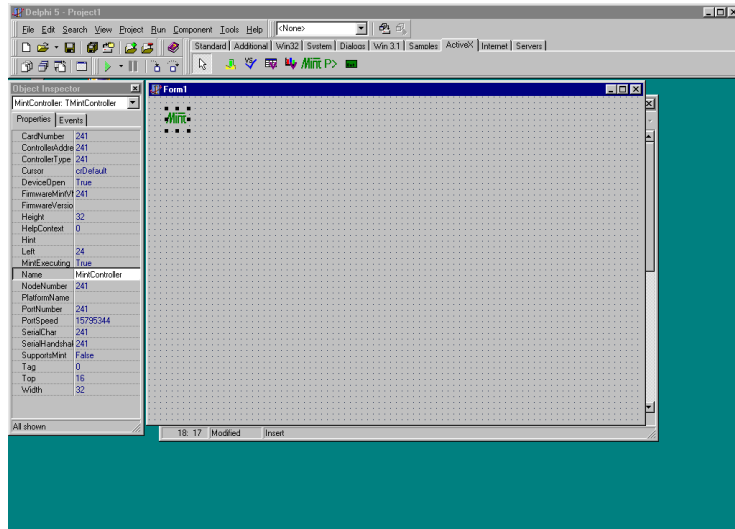


- Once the MintMT ActiveX control is included in the project the Mint icons will appear in the ActiveX tab of the component pallet.



5. To use the MintMT ActiveX control in the project, place it on the form.

The properties of the ActiveX component include a field called Name this is the name used in the code to identify the ActiveX control, in this example it is 'MintController'



6. To use the MintMT ActiveX control to communicate with a controller a handle to the controller must be created. This can be done when the form loads. Double click on the form, this will open an editor window with the cursor in the function called when the form loads. Add the code:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    // Create handle to controller
    MintController.setNextMovePCIILink (0, 0);
end;
```

The above code will create a handler to a NextMove PCI controller, card number 0 and node number 0.

Add a button to the form and double click on the button. Borland Delphi will open an editor window with the cursor in the function called when the button is clicked. Add the code:

```
procedure TForm1.Button1Click(Sender: TObject);
var
    bState: WordBool;
begin
    // Read the state of the relay
    bState := MintController.Relay [0];
    // Invert the state of the relay
    MintController.Relay [0] := not(bState);
end;
```

---

This will code will read the state of the relay and invert it. Run the application, when the button is pressed the relay will toggle.

**Note:** This example is installed by the PC developer libraries in the folder  
\\MintMT\Example\Host\Delphi

---

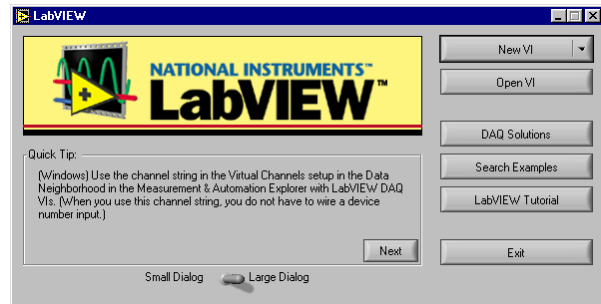
## 6.4 National Instruments LabView

This section is a guide to creating a National Instruments LabView application that uses the MintMT ActiveX control to communicate with a NextMove PCI controller.

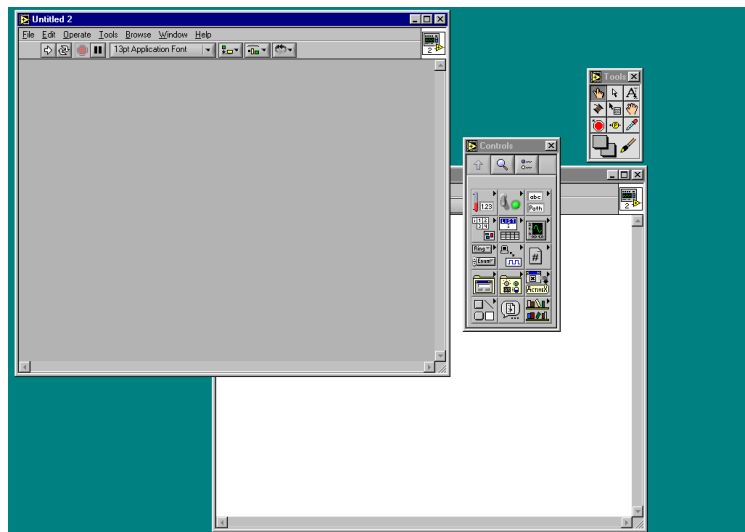
Although this example uses a NextMove PCI controller the same application can be used for a MintDrive<sup>II</sup>, NextMove BX or any other MintMT controller by changing the type of controller.

This example uses National Instruments LabView v6.0.

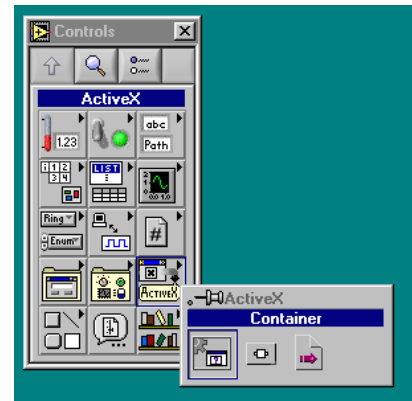
1. Open LabView and select New VI from the initial dialog.



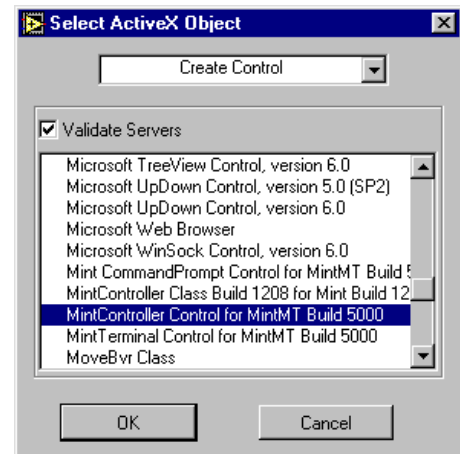
2. LabView will now display the development environment.



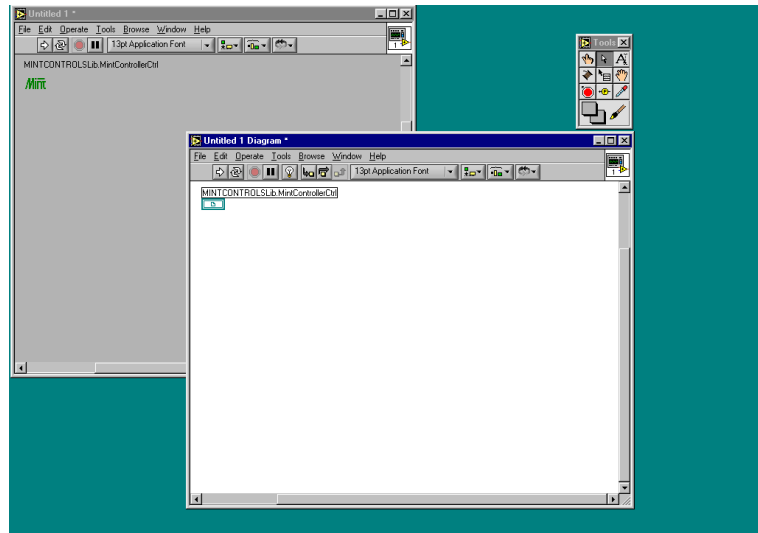
- 
- From the ActiveX icon in the Controls toolbox select a Container and place it on the front panel



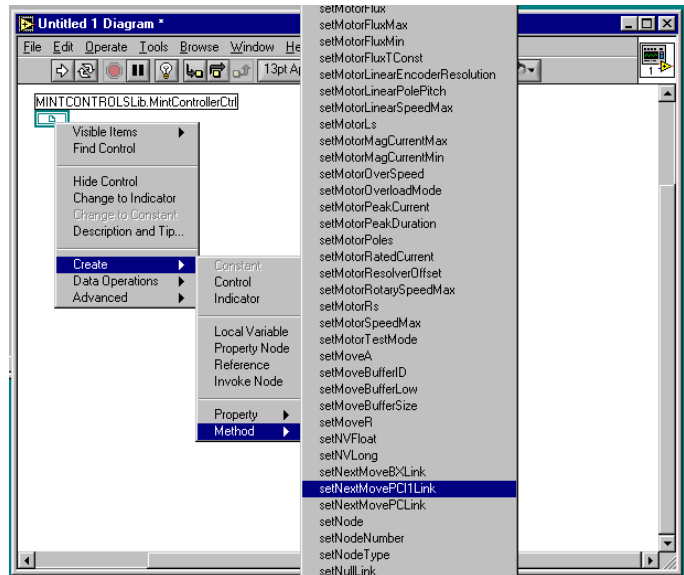
- Right click on the container in the front panel and select 'Insert ActiveX Object...'. This will display a dialog listing all ActiveX controls. Select the MintMT ActiveX control and click OK. In this example it is 'MintController Control for MintMT Build 5000', the build number will vary depending on the version of the ActiveX control that is installed on the PC.



5. Once the MintMT ActiveX control has been installed the container in the front panel will become a Mint icon.



6. Within the block diagram view right click on the MintMT object and select Create...Method...setNextMovePC1Link



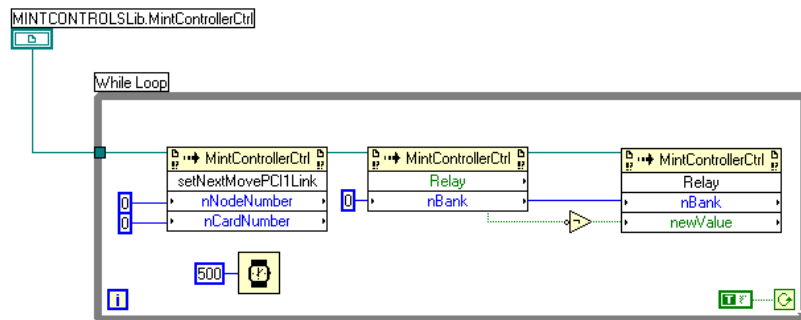
7. Set the parameters for this method to be:

- nNodeNumber        0
- nCardNumber        0

This creates a handle to a NextMove PCI card and enables further function calls to be directed to the card.



8. Other methods can now be added in the same way to build up the application. The example clicks the relay on and off at a rate of 1Hz.



**Note:** This example is installed by the PC developer libraries in the folder  
\\MintMT\Example\Host\LabView



## 7.1 Error Codes

Below is a list of all error codes at the time of printing, for a complete list see HOST\_DEF.H in the folder \MintMT\Include.

The error codes are split into two categories:

- Mint Motion Library (MML) errors
- ActiveX errors

The MML errors are error codes returned by MML function calls. These are function calls to the controller like `setSpeed()` and `getPos()`. Numbering starts at zero.

The ActiveX errors are returned by the ActiveX control. These are not returned by the controller but by the ActiveX itself. Numbering starts at 1000.

### 7.1.1 Mint Motion Library Errors

Number	Macro	Description
0	erSUCCESS	No error
1	erMML_ERROR	Synchronous MML error
2	erINVALID_AXIS	Axis specified out of range
3	erVALUE_OUT_OF_RANGE	Data specified out of range
4	erINVALID_CHANNEL	Adc / dac channel out of range
5	erNO_INPUT_SPECIFIED	Operation on home/limit etc with no i/p
6	erNO_OUTPUT_SPECIFIED	Operation on enable output with no o/p
7	erINVALID_INPUT	Digital input out of range
8	erINVALID_OUTPUT	Digital output out of range
9	erOUT_OF_MEMORY	Not enough heap for operation
10	erMOTION_IN_PROGRESS	Action denied when axis in motion
11	erAXIS_NOT_RIGHT_TYPE	Action denied when in wrong config
12	erMOTION_ERROR	General motion (async) error
13	erTABLE_ERROR	Bad Spline or cam table info
14	erCAN_ERROR	Unable to initialize the CAN bus
15	erCHANNEL_NOT_RIGHT	Channel incorrectly configured
16	erDPR_TIMEOUT	DPR timeout
17	erWRONG_PLATFORM	Not available on this controller
18	erDB_ERROR	Initialize daughter board failed

Number	Macro	Description
19	erSERIAL_ERROR	Problem with RS232 or RS485 port.
20	erWRONG_NODE_TYPE	Node referenced not expected type
21	erCAN_TIMEOUT	Failed to receive reply in time
22	erNODE_NOT_LIVE	Node is not LIVE
23	erTYPE_NOT_SUPPORTED	Type of node not supported
24	erINVALID_HARDWARE	Hardware not present
25	erCMS_DATABASE_FULL	All CMS COBIDs have been allocated
26	erINVALID_NODE_ID	CAN node number out of range
27	erREMOTE_EE_FAIL	Problem writing to EEPROM on node
28	erINVALID_REMOTE_BAUD	Node doesn't support baud rate
29	erREMOTE_SYNC_ERROR	Node reported a synchronous error
30	erREMOTE_ESTOP_ACTIVE	Node in ESTOP condition
31	erINVALID_BUS_NUMBER	CAN bus number was out of range.
32	erNO_FREE_CAN_OBJECTS	There were no free message objects left in the CAN controller.
33	erCAN_BUS_OFF	The CAN controller is bus off.
34	erCAN_TX_BUFFER_FULL	The CAN transmit buffer was full.
35	erTERMINAL_UNAVAILABLE	No terminal device
36	erTERMINAL_OUT_OF_RANGE	Port value is out of range
37	erNON_VOLATILE_MEMORY_ERROR	Problems with non-volatile memory
38	erTERMINAL_BUFFER_EMPTY	Terminal Buffer is empty
39	erTERMINAL_BUFFER_FULL	Terminal Buffer is full
40	erDRIVE_DISABLED	Drive is not enabled
41	erNO_CONNECTION	No Connection Exists
42	erCAN_PROTOCOL_ERROR	CAN Protocol error During Communication.
43	erINVALID_LOCAL_NODE	Local CAN node not correct.
44	erNOT_NETWORK_MASTER	Must be master of network
45	erCOMMS_READ_ONLY	COMMS element is read only.
46	erCOMMS_RESERVED_ELEMENT	COMMS reserved element
47	erOUTPUT_FAULT	Fault on the digital outputs.
48	erDSP_FAILED_TO_CLEAR_ERROR	DSP Failed to clear the error.
49	erOFFSET_PROFILE_ERROR	The Offset cannot be Profiled
50	erFLASH_PROGRAMMING	Error programming Flash

---

## 7-2 MintMT ActiveX Methods

Number	Macro	Description
51	erADDRESS_OUT_OF_RANGE	Error addressing Flash
52	erCRC_CHECKSUM_ERROR	Error in received Checksum
53	erFLASH_BEING_PROGRAMMED	Command invalid when Flash in use
54	erFILE_TOO_BIG	File too big for available memory
55	erCAN_INVALID_OBJECT	Invalid CAN object
56	erCAN_RESERVED_OBJECT	Reserved CAN object
57	erCAN_INVALID_CHANNEL	CAN node channel out of range
58	erCAN_VALUE_OUT_OF_RANGE	Data specified out of range
59	erMOVE_BUFFER_FULL	Move buffer is full
60	erICM_ARRAY_ERROR	ICM protocol error
61	erICM_TOO_MANY_ARRAYS	ICM protocol error
62	erICM_DATA_TIMEOUT	ICM Timeout
63	erICM_BLOCK_TOO_BIG	ICM protocol error
64	erICM_TX_SIZE_MISMATCH	ICM protocol error
65	erICM_RETURN_TIMEOUT	ICM Timeout
66	erMML_NOT_SUPPORTED	MML does not support this function
67	erINVALID_POINTER	Addresses area outside memory
68	erINVALID_MODE	Invalid error action mode
69	erDEBUG_DISABLED	Debug keywords are disabled.
70	erINVALID_MASTER_CHANNEL	Master Channel invalid for Axis.
71	erALL_AXES_MUST_BE_OFF	All Axes must be configured Off.
72	erINVALID_AXISMODE	Move not allowed in this mode.
73	erDOWNLOAD_TIMEOUT	Timeout during File Download.
74	erCAPTURE_IN_PROGRESS	Capture in progress during upload
75	erINVALID_NUM_CAP_PTS	Invalid number of capture points
76	erINVALID_BBP_TRANS_NO	BBP Transaction No. not supported
77	erINVALID_BBP_FIELD_LENGTH	BBP Transaction has wrong length
78	erBBP_DATA_OUT_OF_RANGE	BBP Transaction data invalid
79	erBBP_DATA_OUT_OF_BOUNDS	BBP Transaction data modified
80	erBBP_FAULT_PREVENTS_EXEC	BBP Transaction can't be executed
81	erBBP_MODE_PREVENTS_EXEC	BBP Transaction can't be executed
82	erBBP_BLOCK_NOT_ACCEPTED	BBP Block transfer not accepted
83	erBBP_END_OF_BLOCK_REACHED	BBP End of block reached
84	erUNKNOWN_BBP_ERROR	Unknown BBP error code
85	erDRIVE_TIMEOUT	BBP Transaction timed-out

Number	Macro	Description
86	erBBP_OVERFLOW	BBP Transaction Rx overflow
87	erINVALID_BBP_PACKET_SIZE_RXD	BBP Transaction Rxd size too big
88	erINVALID_BBP_TRANSACTION_RXD	Invalid BBP Transaction Rxd
89	erCHANNEL_IN_USE	Hardware channel required is in use
90	erDRIVE_ENABLED	Drive is enabled.
91	erINVALID_DRIVE_PARAM	Invalid Drive Parameter No.
92	erBBP_TRANSACTION_IN_PROGRESS	A BBP transaction is executing.
93	erNO_BBP_TRANSACTION_REQUESTED	No BBP transaction requested.
94	erICM_DISABLED	ICM is disabled on this channel
95	erOUTPUT_IN_USE	Output is already in use.
96	erCAPTURE_CHANNEL_MIX	Invalid capture channel mix.
97	erALL_CONTROL_AXES_IN_USE	All controllable axes in use
98	erINVALID_VAR_TYPE	Invalid variable type for RemoteObject.
99	erCAN_CONFIRMED_BUSY	A confirmed service is already in progress.
100	erFILE_PROTECTED	Mint File is Protected.
101	erREMOTE_DOWNLOAD_IN_PROGRESS	A Mint file is currently being downloaded to a remote node.
102	erBBP_REQUEST_TIMEOUT	Timeout on BBP request.
103	erPARAMETER_ACCESS_CONFLICT	Two devices updating same parameter
104	erCAN_ALREADY_CONNECTED	CANopen node is already connected to another node.
105	erREMOTE_DRIVE_DISABLED	The remote drive is disabled (CANopen).
106	erREMOTE_DRIVE_FAULT	The remote drive is in fault mode (CANopen).
107	erREMOTE_STATE_INCORRECT	Transaction aborted due to nodes state.
108	erREMOTE_DRIVE_MOVE_FAILED	The remote drive failed to accept the new move (CANopen).
109	erINCOMPATIBLE_SETTINGS	Incompatible with previous settings
110	erDSP_COMMS	Inter-processor communications error
111	erAUTOTUNE_FAILURE	Autotuning operation failed
112	erREAD_ONLY	Parameter is read only
113	erICM_DLL_ERRORS	Errors in the Rxd DLL SO telegram
114	erICM_DLL_MESSAGE_ID_MISMATCH	DLL SO telegram id doesn't match

---

#### 7-4 MintMT ActiveX Methods

Number	Macro	Description
115	erICM_TL_HOST_RETRANSMITS	Too many host retransmit requests
116	erICM_TL_TRANSMIT_REQUESTS	Too many transmit requests - timed-out
117	erICM_TL_BUSY	TL is busy - still processing command
118	erICM_TL_NO_OF_PACKETS	Invalid No of packets specified
119	erICM_TL_GROUP_MESSAGE_ID	Group message id mismatch
120	erICM_TL_GROUP_SEQUENCE	Group message packet No out of sequence
121	erEEPROM_ACCESS	Error accessing EEPROM device
122	erINITIALISATION_FAILURE	A failure occurred during initialization
123	erINVALID_ALLOCATION_TABLE	Invalid object allocation table
124	erOBJECT_NOT_FOUND	Application data object not found
125	erMINT_PROGRAM_RUNNING	A Mint program is already running
126	erINVALID_MINT_COMMAND	The Mint command is invalid
127	erINVALID_TERMINAL_PORT	Port type for terminal is not valid
128	erINVALID_TERMINAL_DEVICE	Device for terminal is not valid
129	erINVALID_TERMINAL_ADDRESS	Address type for terminal is not valid
130	erUNDEFINED_TERMINAL	Terminal is not defined
131	erSINGLE_TERMINAL_ONLY	A single terminal is required
132	erICM_HOST_BUSY	The host is not ready for the ICM reply
133	erINVALID_PLATFORM_CODE	The image files platform is invalid
134	erINVALID_IMAGE_FORMAT_CODE	The image files format is invalid

### 7.1.2 ActiveX errors

Number	Macro	Description
1001	erINITIALISING	Loader initializing
1002	erNOT_RUNNING	Loader not able to relocate
1003	erBAD_COMMAND	Unrecognized command code
1004	erBAD_ADDRESS	Invalid address received
1005	erBAD_ERASE	Flash erase failed
1006	erBAD_BURN	Flash program failed
1007	erNO_FLASH	No flash found
1008	erBAD_FLASH_HEADER	Flash not programmed properly

Number	Macro	Description
1009	erERROR_DOWNLOADING	COFF download failed
1010	erTIMEOUT	Loader did not respond in time
1011	erDPRAM_LOCATION	Dual Port Ram location out of range
1012	erNOT_ENOUGH_MEM	Insufficient memory for program
1013	erBAD_BOOT_DEVICE	Bad boot source id
1014	erCARD_NOT_FOUND	Unable to locate NextMove
1015	erINVALID_VME_TYPE	Unknown VME type
1016	erBAD_NEXTMOVE_TYPE	Unknown NextMove type
1017	erINVALID_STRING_FORMAT	Non-NULL terminated string used
1018	erNO_MINT_PROMPT	MINT not at command prompt
1019	erNO_VME_95_SUPPORT	VME not supported under Win95
1020	erCOMMAND_ABORTED	User aborted front command
1021		Error 1021 removed
1022	erCOMMAND_INTERRUPTED	Command was not loaded
1023	erRETURN_INVALID	Return code invalid. Check err
1024	erFRONT_DISABLED	PC-Front has been disabled
1025	erINVALID_HANDLE	Handle to NextMove invalid
1026		Error 1026 removed
1027	erPROTOCOL_ERROR	Invalid protocol e.g.on upload
1028	erFILE_ERROR	Error reading / writing
1029	erINVALID_FILETYPE	Invalid param : fileUp/Download
1030		Error 1030 removed
1031	erNO_NT_SUPPORT	Function not supported.
1032	erNO_RESPONSE	NextMove did not respond.
1033	erTEMP_FILE_ERROR	Error creating temp file.
1034	erCODE_ERROR	Internal error
1035		Error 1035 removed
1036		Error 1036 removed
1037		Error 1037 removed
1038		Error 1038 removed
1039	erPORT_NOT_OPEN	Serial port not opened
1040	erCORRUPTION	Corruption occurred
1041	erPORT_OUT_OF_RANGE	Specified port not available
1042	erNOTIFY	Could not enable WM_NOTIFY
1043	erCHECKSUM_ERROR	The checksum failed
1044	erNAK_RECEIVED	The controller sent NAK
1045		Error 1045 removed
1046	ROR_OPENING_PORT	Port could not be opened
1047	erINVALID_CARDNUMBER	Card number out of range
1048	erINVALID_AXIS_PARAM	Axis out of range

## 7-6 MintMT ActiveX Methods

Number	Macro	Description
1049	erINVALID_CONTROLLER_TYPE	Invalid controller enumeration
1050	erINVALID_COMMS_ADDRESS	Comms address out of range
1051		Error 1051 removed
1052		Error 1052 removed
1053	erUSER_ABORT	The user aborted the command
1054	erCONTROLLER_REPORTS_ERROR	The controller detected error
1055		Error 1055 removed
1056	erRECEIVE_BUFFER_EMPTY	The receive buffer is empty
1057	erTRANSMIT_BUFFER_FULL	The transmit buffer is full
1058	erINVALID_RETRIES	The retries parameter failed
1059	erBAD_SQUASH_FILE	Bad squash file parameter.
1060	erUNDEFINED_SERIAL_ERROR	The serial error is unknown
1061	erPSERIAL_BUFFER_CORRUPTION	The serial buffers are corrupt
1062	erFUNCTION_NOT_SUPPORTED	Not supported on this platform
1063	erCANNOT_OPEN_FILE	File bad or doesn't exist
1064	erINVALID_FORMAT	File not proper COFF format
1065	erDATA_TOO_LONG	Too much data in one chunk
1066	erINCORRECT_ARRAY_SIZE	Array size or pointer incorrect
1067	erUNKNOWN_ERROR_CODE	The error code was not known
1068	erCONTROLLER_NOT_RUNNING	The controller is not running
1069	erMML_VERSION_MISMATCH	Build number incorrect
1070	erNO_DEVICE_DRIVER_SUPPORT	Device driver not set up
1071	erBAD_COM_PORT_NUMBER	Com port not supported
1072	erBAD_BAUD_RATE	Baud rate not supported
1073		Error 1073 removed
1074		Error 1074 removed
1075		Error 1075 removed
1076		Error 1076 removed
1077		Error 1077 removed
1078	erLINE_TOO_LONG	Line too long for MINT
1079	erINVALID_PLATFORM	Invalid firmware for controller
1080	erNO_INTERRUPT_REGISTERED	No interrupt registered
1081	erINVALID_IRQ	Invalid IRQ parameter
1082	erBAD_INPUT_BUFFER	Input buffer wrong size
1083	erBAD_OUTPUT_BUFFER	Output buffer wrong size
1084	erBAD_DEVICE_DRIVER_CALL	The device driver call failed
1085	erSEMAPHORE_TIMEOUT	A semaphore was not available
1086	erINVALID_EVENT	Could not register the event
1087	erFUNCTION_NOT_AVAILABLE	Function not currently available
1088	erBOOT_TEST_FAIL	Power-up self test failed

Number	Macro	Description
1089	erBUFFER_TOO_SMALL	Not enough memory to load program
1090	erREQUIRES_DEV_DRIVER	Requires development build
1091	erICM_TX_TIMEOUT	Timeout on ICM
1092	erICM_RX_TIMEOUT	Timeout on ICM
1093	erICM_RX_SIZE_ERROR	Error in ICM protocol
1094	erICM_PROCESS_TIMEOUT	Controller too slow
1095	erDEV_DRV_UNKNOWN_IOCTL	Device driver mismatch
1096	erBBP_ACK_TIMEOUT	No response from controller
1097	erBBP_POLL_TIMEOUT	No response to poll
1098	erBBP_POLL_NO_DATA	No data ready for polling
1099	erBBP_RX_TIMEOUT	Receive data timeout
1100	erBBP_UNSUPPORTED_TRANS	Invalid/unsupported trans. no.
1101	erBBP_INVALID_DATA_LENGTH	Invalid data field length
1102	erBBP_VALUE_OUT_OF_RANGE	Data value rejected
1103	erBBP_VALUE_OUT_OF_BOUNDS	Data value modified
1104	erBBP_CONTROL_FAULT_COND	Control fault condition
1105	erBBP_STATUS_MODE_REJECT	Status / mode caused rejection
1106	erBBP_BLOCK_REJECTED	Block transfer value rejected
1107	erBBP_END_OF_BLOCK	End of block reached
1108		Error 1108 removed
1109	erAUTOTUNE_FAILED	Auto-tune function failed
1110	erNO_CAPTURED_DATA	No captured data to upload
1111	erSQ_INVALID_OUTPUT_FILE	A file could not be created
1112	erSQ_INVALID_INPUT_FILE	Bad input file
1113	erSQ_TOO_MANY_VARIABLES	Too many variables
1114	erSQ_BASIC_TABLE_NOT_FOUND	Basic.XYZ not found
1115	erSQ_MOTION_TABLE_NOT_FOUND	Motion.XYZ not found
1116	erSQ_CONSTANT_TABLE_NOT_FOUND	Constant.XYZ not found
1117	erSQ_INPUT_FILE_READ_ERROR	Error reading
1118	erSQ_OUTPUT_FILE_WRITE_ERROR	Error writing
1119	erSQ_INVALID_OUTPUT_FILE_STRING	Not NULL terminated
1120	erSQ_INVALID_INPUT_FILE_STRING	Not NULL terminated
1121	erSQ_INVALID_PATH_STRING	Not NULL terminated
1122	erSQ_TOO_MANY_BASIC_KEYWORDS	Too many basic keywords
1123	erSQ_TOO_MANY_MOTION_KEYWORDS	Too many motion keywords
1124	erSQ_TOO_MANY_CONSTANTS	Too many constants
1125	erSQ_VARIABLES_NOT_INITIALISED	Internal squash error
1126	erCANNOT_WRITE_TO_INTERRUPT	No write access to interrupts
1127	erNO_LINK_TO_CONTROLLER	Must use a setXXXLink function
1128	erFIRST_ARRAY_ELEMENT_IS_SIZE	Bad array

## 7-8 MintMT ActiveX Methods

Number	Macro	Description
1129	erPOS_ARRAY_REQUIRED	Position array must be used
1130	erARRAY_SIZE_MISMATCH	Array sizes do not match
1131	erPARAMETER_CANNOT_BE_NEGATIVE	No negative parameter.
1132	erCAN_INIT_FAILED	Initialization of CAN failed
1133	erEEPROM_CRC_FAILED	EEPROM failed CRC check
1134	erINSUFFICIENT_MEMORY	Insufficient memory to run app
1135	erCANNOT_RUN_APP	Cannot run app for unknown reason
1136	erEVENT_HANDLER_IN_USE	Event handler already installed
1137	erSERIAL_PORT_OPEN	Action not possible until closed
1138	erBAD_PARAMETER1	First parameter incorrect.
1139	erBAD_PARAMETER2	Second parameter incorrect.
1140	erBAD_PARAMETER3	Third parameter incorrect.
1141	erBAD_PARAMETER4	Fourth parameter incorrect.
1142	erBAD_PARAMETER5	Fifth parameter incorrect.
1143	erBAD_PARAMETER6	Sixth parameter incorrect.
1144	erAUTOTUNE_POLES_OR_PPR	Incorrect poles or PPR.
1145	erAUTOTUNE_INDEX_PULSE_MISSING	Index pulse missing.
1146	erAUTOTUNE_UVW_SIGNAL_LOSS	UVW Signal loss detected.
1147	erINCORRECT_UVW_PHASING	Bad UVW phasing or polarity.
1148	erLOGGING_NOT_ENABLED	Logging not enabled.
1149	erMINTCONTROLLER_NOT_FOUND	OCX problem.
1150	erMONOSPACED_FONT_REQUIRED	Must use fixed pitch font.
1151	erOBJECT_TOO_BIG	App data object is too large
1152	erMINT_CONTROLLER_ID	Must call setMintControllerID
1153	erCAPTURE_UPLOAD_RESTART	Capture restarted during upload
1154	erINVALID_ANSI328_NODE	Node number > 15.
1155	erINV_COMPILER_VERSION	Compiler ver not registered
1156	erCOMPILER_NOT_REG	Compiler not registered

---

## 7.2 Properties

This is a list of function all prototypes supported by Baldor controllers. Only a selection of these are supported by the NextMove controller. If an unsupported function is called an error code will be returned.

All functions return error codes directly of the form erXXX.

The following pages list all function prototypes supported by NextMove in alphabetical order.

Not completed

## 8.1 Overview

Each area of the address map is described below. Where an address is shown, it is the DPR location. Address offsets are added to the base address. Floating point numbers will conform to C31 format. The PC interface must convert to IEEE format before passing the data to the PC application. Likewise, IEEE floating point numbers must be converted to C31 format before writing to the DPR. All library functions do this automatically.

- The refresh rate of the data in DPR is 2ms.
- All addresses and address offsets are in hexadecimal format.

Dual Port RAM on NextMove PCI has 4K of 32-bit data, but certain areas are designated as read only. This means that if the user tries to write to these locations the data may be corrupted.

All access to Dual Port RAM on NextMove PCI is 32-bit wide.

By default the update of data is disabled. To enable the update write to the Control Register at address zero. This can be done from a PC application with the functions `setLong` or `setWord`.

Alternatively using the Command Prompt in WorkBench v5 Dual Port RAM locations can be modified using the keywords:

```
DPERLONG.address = value
```

or

```
DPERWORD.address = value
```

## 8.2 Dual Port RAM map

Address	Description	Read Only
0xFFF	Interrupt Host	✓
0xFFE	Interrupt NextMove	✓
0xFFD 0xFE0	<i>Reserved</i>	✓
0xFDF 0xBE0	User Area	
0xBDF 0x600	<i>Reserved</i>	✓
0x5FF 0x500	ICM Expansion	✓
0x4FF 0x400	Axis Data	✓
0x3FF 0x3F8	Special Function Registers	✓
0x3F7 0x29C	User Area	
0x29B 0x1D6	Comms (99 locations)	
0x1D5 0x193	Pseudo Serial Transmit Buffer	✓
0x192 0x150	Pseudo Serial Receive Buffer	✓
0x14F 0x130	ICM Interface	✓
0x12F 0x110	I/O Data	✓
0x10F 0x010	Axis Data	✓
0x00F 0x002	Status and Control Registers	✓
0x001	DPR Status Register	✓
0x000	DPR Control Register	

## 8.2.1 Status and control registers

Address	Use	Symbolic Constant	Read Only
0x000	DPR Control Register	roCONTROL	
0x001	DPR Status Register	roSTATUS	
0x002	Axis Mix	roAXIS_MIX	✓
0x003	Digital I/O Mix	roNUM_DIO	✓
0x004	Analog I/O Mix	roNUM_AIO	✓
0x005	Build ID	roBUILD	✓
0x006	2ms Timer Tick	roTIMER_TICK	✓
0x007	<i>Reserved</i>		
0x008	<i>Reserved</i>		
0x009	<i>Reserved</i>		
0x00A	<i>Reserved</i>		
0x00B	Axis Configurations (0-7 )	roAXIS_CF	✓
0x00C	Axis Configurations (8-11)	n/a	✓
0x00D	1ms Timer Tick	ro1MS_TIMER	✓
0x00F	Reserved	n/a	✓

### DPR Control Register

Bit	Meaning	Symbolic Constant
0	Lock DPR contents	btLOCK
1	Lock axis 0 DPR contents	btLOCK_AXIS_0
2	Lock axis 1 DPR contents	btLOCK_AXIS_1
3	Lock axis 2 DPR contents	btLOCK_AXIS_2
4	Lock axis 3 DPR contents	btLOCK_AXIS_3
5	Lock axis 4 DPR contents	btLOCK_AXIS_4
6	Lock axis 5 DPR contents	btLOCK_AXIS_5
7	Lock axis 6 DPR contents	btLOCK_AXIS_6
8	Lock axis 7 DPR contents	btLOCK_AXIS_7
9	Lock axis 8 DPR contents	btLOCK_AXIS_8
10	Lock axis 9 DPR contents	btLOCK_AXIS_9
11	Lock axis 10 DPR contents	btLOCK_AXIS_10
12	Lock axis 11 DPR contents	btLOCK_AXIS_11
13 - 16	Reserved	
17	Lock IO data	btLOCK_IO
18	Lock auxiliary axes	btLOCK_AUX_AXES
19-31	Reserved	

---

### DPR Status Register

Bit	Meaning	Symbolic Constant
0	DPR Contents locked if 1	btLOCKED
1	DPR contents invalid if 0	btVALID
2 - 15	Reserved	

### Axis Mix

This specifies the number and types of axes available on the NextMove variant:

- Lo-Byte - Number of stepper axes
- Hi-Byte - Number of servo axes

### Digital I/O Mix

This specifies the number of digital inputs and outputs available on the NextMove variant:

- Lo-Byte - Number of digital outputs
- Hi-Byte - Number of digital inputs

### Analog I/O Mix

This specifies the number of analog inputs and outputs available on the NextMove variant:

- Lo-Byte - Number of analogue outputs
- Hi-Byte - Number of analogue inputs

### MML Build ID

The build identifier of the Mint Motion Library running on the controller. To return the build number call `getAAABuild`.

### 2ms Timer Tick

This is a free running 16bit counter that is updated by NextMove once every 2ms and can be used to synchronize data with the DPR.

### Axis Configurations

Axis Configurations gives the current configuration of each axis in 4 bits.

Address 0x0B

Bits	31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
Axis No.	Axis 7	Axis 6	Axis 5	Axis 4	Axis 3	Axis 2	Axis 1	Axis 0

---

Address 0x0C

Bits	31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
Axis No.	-	-	-	-	Axis 11	Axis 10	Axis 9	Axis 8

#### 1ms Timer Tick:

The 1ms Timer Tick is an incrementing counter that indicates that NextMove is running. The counter increments by 1 every 1ms.

### 8.2.2 Axis data

The axis data area is divided into 12 sections, four for the main board axes and four for each of the expansion board axes. The base address for each axis is listed below:

Address	Use	Symbolic Constant
0x010	Axis 0	roAXIS_0
0x030	Axis 1	roAXIS_1
0x050	Axis 2	roAXIS_2
0x070	Axis 3	roAXIS_3
0x090	Axis 4	roAXIS_4
0x0A0	Axis 5	roAXIS_5
0x0C0	Axis 6	roAXIS_6
0x0E0	Axis 7	roAXIS_7
0x400	Axis 8	roAXIS_8
0x420	Axis 9	roAXIS_9
0x440	Axis 10	roAXIS_10
0x460	Axis 11	roAXIS_11

Each group contains the following data.

Offset	Use	Symbolic Constant	Data Size
0x00	Measured Position	roPOSITION	float
0x01	<i>Reserved</i>		
0x02	Measured Velocity	roMEASURED_SPEED	float
0x03	<i>Reserved</i>		
0x04	Speed*	roDEMAND_SPEED	float
0x05	<i>Reserved</i>		
0x06	Mode of motion	roMODE_OF_MOTION	int 32
0x07	<i>Reserved</i>		
0x08	Axis error	roMOTION_ERROR	int 32
0x09	Following Error	roFOLLOWING_ERROR	float

Offset	Use	Symbolic Constant	Data Size
0x0A	<i>Reserved</i>		
0x0B	Kprop*	roP_GAIN	float
0x0C	<i>Reserved</i>		
0x0D	Kvel*	roV_GAIN	float
0x0E	<i>Reserved</i>		
0x0F	KvelFF*	roFF_GAIN	float
0x10	<i>Reserved</i>		
0x11	Kderiv*	roD_GAIN	float
0x12	<i>Reserved</i>		
0x13	Kint*	roI_GAIN	float
0x14	<i>Reserved</i>		
0x15	KintLimit(%)*	roI_RANGE	float
0x16	<i>Reserved</i>		
0x17	Next Mode of motion	roNEXT_MODE	int 32
0x18	<i>Reserved</i>		
0x19	DAC value	roDAC_VALUE	int 16
0x1A	Free Spaces in buffer	roFREE_SPACES	int 16
0x1B	Move buffer ID	roMOVE_ID	int 16
0x1C	Demand Position	roDEMAND_POS	float
0x1D	<i>Reserved</i>		
0x1E	Demand Velocity	roDEMAND_VEL	float
0x1F	<i>Reserved</i>		

All data (except those marked \*) is written every 2ms by NextMove. Locations marked \* are only written when they change.

### 8.2.3 I/O data

Address	Use	Symbolic Constant	Data Size
0x110	Analog 0	roANALOG_0	int 16
0x111	Analog 1	roANALOG_1	int 16
0x112	Analog 2	roANALOG_2	int 16
0x113	Analog 3	roANALOG_3	int 16
0x114	Expansion Analog 4	roANALOG_4	int 16
0x115	Expansion Analog 5	roANALOG_5	int 16
0x116	Expansion Analog 6	roANALOG_6	int 16
0x117	Expansion Analog 7	roANALOG_7	int 16
0x118	Base Digital inputs	roINPUTS	int 32
0x119	<i>Reserved</i>		

## 8-6 NextMove PCI DPR Map

Address	Use	Symbolic Constant	Data Size
0x11A	Base Digital Outputs	roOUTPUTS	int 16
0x11B	Stop / Error bits	roMG_STATUS	int 16
0x11C	<i>Reserved</i>		
0x11D	Auxiliary Encoder 0	roAUXENC_0_POS	float
0x11E	<i>Reserved</i>		
0x11F	Auxiliary Encoder 0 vel	roAUXENC_0_VEL	float
0x120	<i>Reserved</i>		
0x121	Auxiliary Encoder 1	roAUXENC_1_POS	float
0x122	Auxiliary Encoder 1 vel	roAUXENC_1_VEL	float
0x123	Auxiliary Encoder 2	roAUXENC_2_POS	float
0x124	Auxiliary Encoder 2 vel	roAUXENC_2_VEL	float
0x125	Expansion 1 Digital Inputs	roEXP1_INPUTS	int 32
0x126	Expansion 1 Digital Outputs	roEXP1_OUTPUTS	int 32
0x127	Expansion 2 Digital Inputs	roEXP2_INPUTS	int 32
0x128	Expansion 2 Digital Outputs	roEXP2_OUTPUTS	int 32
0x129	<i>Reserved</i>		
0x12A	<i>Reserved</i>		
0x12B	<i>Reserved</i>		
0x12C	<i>Reserved</i>		
0x12D	<i>Reserved</i>		
0x12E	<i>Reserved</i>		
0x12F	<i>Reserved</i>		

All data is written every 2ms.

## 8.2.4 Comms array

The Comms area simulates protected Comms communications on serial based controllers. The Comms array uses an area of DPR from address 0x1D6 to 0x29A. The data is accessed as:

Address	Comms Location
0x1D6	location 1
0x1D8	location 2
0x1DA	location 3
....	
0x298	location 98
0x29A	location 99

Each location is a floating point value. Comms is accessed using the COMMS keyword in MintMT or the `getComms / setComms` functions.

## 8.2.5 Immediate command mode

This area is reserved for immediate command mode (ICM) function calls.

## 8.2.6 Pseudo serial interface

The serial interface works by implementing a 64 word circular buffer within DPR. There is one such buffer for the receive buffer and one for the transmit buffer. Head and tail pointers are also located in DPR allowing both sides of DPR to check the status of the buffers.

The serial interface occupies DPR locations 0x150 to 0x1D5 in the following configuration:

Txd Buffer	0x85
Txd Reserved	0x46
Txd Tail	0x45
Txd Head	0x44
Rxd Buffer	0x43
Rxd Reserved	0x42
Rxd Tail	0x03
Rxd Head	0x02
Rxd Tail	0x01
Rxd Head	0x00

The buffer itself has two sets of symbolic constants, depending on which side, NextMove or host, that is using them.

Offset	Symbolic Constant - Host	Symbolic Constant - NextMove
0x00	ofTXD_HEAD	ofNM_RXD_HEAD
0x01	ofTXD_TAIL	ofNM_RXD_TAIL
0x03	ofTXD_BUFFER	ofNM_RXD_BUFFER
0x43	ofRXD_HEAD	ofNM_TXD_HEAD
0x44	ofRXD_TAIL	ofNM_TXD_TAIL
0x46	ofRXD_BUFFER	ofNM_TXD_BUFFER

The offsets from the start of the serial interface are shown in hex. The start of the serial I/O buffer has a symbolic constant of `ofSERIAL_IO_BASE`.

## 8.2.7 Special functions registers

Address	Use	Symbolic Constant
0x3F8	ICM Handshaking	roICM_HANDSHAKE
0x3F9	Data associated with events	roINTERRUPT_DATA_1
0x3FA	Data associated with events	roINTERRUPT_DATA_2
0x3FB	Application Code Register	roAPPLICATION_CODE
0x3FC	Functionality Code Register	roFUNCTION_CODE
0x3FD	Scratchpad Register	roSCRATCH_PAD

The way in which DPR is used may vary from application to application. All applications should use the registers detailed in this document in the same way. This will allow host resident code to determine whether it recognizes the application and the protocol used for communication.

There is no hardware restriction upon those locations that may be read or written from either side. Both NextMove and the host have full read and write access to all locations.

### Application Code Register (3FB)

This register identifies the software running on NextMove. The host may use this to determine how to communicate with the software or better interpret the bits within the Functionality Code Register. Each application program should have a unique identifier. Of the 65536 possible codes, the first half are reserved. Codes 32768 to 65535 may be used to identify user programs. Application programs may prime this register after initialization. It is recommended that the host does not write to this location.

For an embedded application the MML will write a value of 7 to this location.

Code	Description of Program	Symbolic Constant
0	Unidentified program or no program running.	apNONE
1	Loader running.	apLOADER
2	Immediate Command Mode supported.	apFRONT
3	NextMove test program running.	apNM_TEST
4	Mint for NextMove supported.	apNM_MINT
5	Mint for NextMove supported.	apFRONT_MINT
6	Custom Version.	apRPD_MINT
7	Mint Motion Library. (Embedded)	apEMBEDDED_APP
8	MintMT	apMINT_MT
9+	Reserved	

---

### Functionality Code Register (3FC)

This register describes the capabilities of the software running on NextMove. The register may be used by a host to determine how it should communicate with the software, what data is stored in dual port RAM, etc. The register contains a series of bits each of which indicate whether a specific feature is supported. The table below describes the current list of standard application capabilities. It is expected that this list will grow over time. Application programs should set the relevant bits in this register after all other initialization. It is recommended that the host does not write to this location.

Bit	Description of Feature	Symbolic Constant
0	Loader communication protocol.	fcLOADER_COMMS
1	MML update of locations 0x000 to 0x012F.	fcAUTO_UPDATE
2	ICM communication protocol.	fcFRONT_COMMS
3	Pseudo Serial Port Buffer.	fcSERIAL_PORT
4	Mint interpretation of serial buffer communications (Comms Protocol)	fcCOMMS_ON
5	Mint running	fcMINT_RUNNING
6 - 15	Reserved	

### Scratchpad Register (3FD)

This register is a general purpose register used only by the host. It is only written to by the Loader immediately after reset when it is cleared to zero. It may be used by the host to determine that a NextMove may be installed on the bus. As NextMove will not write to this location the host can write codes and read them back in the knowledge that they should not have changed. After use by the PC host, the scratchpad should be returned to the value it originally contained.

It is recommended that NextMove application programs do not write to this register.

## 9.1 MML function call timing

These timings show the time taken to call Immediate Command Mode (ICM) functions from a PC application.

The tests were performed on a 600 MHz Pentium III PC. On both MintDrive<sup>II</sup> and NextMove PCI the timings were the same on Windows 98 and Windows NT. All communication to serial controllers was performed at a baud rate of 57600. The PC application was written in Microsoft Visual C++.

### 9.1.1 NextMove PCI

Function	Called from MintMT	Called from PC
getPos		
setJog		
setSpeed		

### 9.1.2 NextMove BX

Function	Called from MintMT	Called from PC
getPos		
setJog		
setSpeed		

### 9.1.3 MintDrive<sup>II</sup>

Function	Called from MintMT	Called from PC
getPos		
setJog		
setSpeed		

# **BALDOR<sup>®</sup>** **MOTORS AND DRIVES**

Baldor UK Ltd  
Mint Motion Centre  
6 Bristol Distribution Park  
Hawkey Drive, Bristol  
BS32 0BF, UK

<b>UK</b> TEL: +44 1454 850000 FAX: +44 1454 850001	<b>US</b> TEL: +1 501 646-4711 FAX: +1 501648-5792	<b>MX</b> TEL: +52 47 61 2030 FAX: +52 47 61 2010
<b>CH</b> TEL: +41 52 647 4700 FAX: +41 52 659 2394	<b>D</b> TEL: +49 89 90 50 80 FAX: +49 89 90 50 8491	<b>F</b> TEL: +33 145 10 7902 FAX: +33 145 09 0864
<b>I</b> TEL: +39 11 562 4440 FAX: +39 11 562 5660	<b>AU</b> TEL: +61 29674 5455 FAX: +61 29674 2495	<b>CC</b> TEL: +65 744 2572 FAX: +65 747 1708



MNxxxx 09/2001

**Printed in UK**  
**© Baldor UK Ltd**