



---

---

— *MintMT* —

---

---

# Embedded Programming Guide



---

# Contents

1	General Information.....	1-1
2	Introduction.....	2-1
2.1	Overview .....	2-1
2.2	Abbreviations.....	2-1
3	Creating an Embedded Application.....	3-1
3.1	Overview .....	3-1
3.2	Tools.....	3-1
3.3	Installing the embedded developer libraries .....	3-1
3.4	Embedded example .....	3-3
3.4.1	Building the example.....	3-3
3.4.2	Downloading the example.....	3-5
3.5	Debugging .....	3-5
3.5.1	Creating debug information.....	3-5
3.5.2	Using the C source debugger .....	3-7
4	Mint Motion Library Features.....	4-1
4.1	Mint Motion Library (MML) data types.....	4-1
4.2	Calling API Functions.....	4-1
4.3	Accessing Dual Port RAM (DPR).....	4-2
4.4	Event handling.....	4-2
4.4.1	Events within MML.....	4-3
4.4.2	Event control .....	4-5
4.4.3	Event handling example.....	4-6
5	Open Architecture .....	5-1
5.1	Data available.....	5-1
5.2	Servo loop .....	5-5
5.2.1	Installing servo loop hooks.....	5-5
5.2.2	The servo loop flow diagram .....	5-7
5.2.3	Servo loop example .....	5-9
5.3	Motion profiler.....	5-9
5.3.1	Installing a motion profiler hook .....	5-9

---

5.3.2	Writing the new demand position.....	5-10
5.3.3	Error handling.....	5-10
5.3.4	Motion profiler example.....	5-11
5.4	Calling APIs from within the handlers.....	5-11
5.5	Code execution time.....	5-11
<b>6</b>	<b>TMS320C31 Compiler.....</b>	<b>6-1</b>
6.1	Overview.....	6-1
6.2	Compiling code.....	6-1
6.3	Run-time libraries.....	6-2
6.4	COFF files.....	6-3
6.5	Environment settings.....	6-3
6.6	Memory models.....	6-4
6.7	Argument passing.....	6-4
6.8	Two pass parsing / out of memory errors.....	6-4
6.9	Entry point.....	6-4
6.10	Sections.....	6-4
6.11	Data types.....	6-5
6.12	Division.....	6-6
6.13	Variable initialization.....	6-6
<b>7</b>	<b>NextMove PCI DPR Map.....</b>	<b>7-1</b>
7.1	Overview.....	7-1
7.2	Dual Port RAM map.....	7-2
7.2.1	Status and control registers.....	7-3
7.2.2	Axis data.....	7-5
7.2.3	I/O data.....	7-6
7.2.4	Comms array.....	7-7
7.2.5	Immediate command mode interface.....	7-8
7.2.6	Pseudo serial interface.....	7-8
7.2.7	Special functions registers.....	7-9
<b>8</b>	<b>Enumerations.....</b>	<b>8-1</b>
8.1	Function Call Enumerations.....	8-1
8.2	System Error Codes.....	8-12

Copyright Baldor (c) 2005. 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.

UL and cUL are registered trademarks of Underwriters Laboratories.

## 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  
Hawley Drive  
Bristol, BS32 0BF  
Telephone: +44 (0) 1454 850000  
Fax: +44 (0) 1454 850001  
Email: [technical.support@baldor.co.uk](mailto:technical.support@baldor.co.uk)  
Web site: [www.baldor.co.uk](http://www.baldor.co.uk)

Baldor Electric Company  
Telephone: +1 501 646 4711  
Fax: +1 501 648 5792  
Email: [sales@baldor.com](mailto:sales@baldor.com)  
Web site: [www.baldor.com](http://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

NextMove 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 Mint language consists of two basic components. The Mint Virtual Machine (MVM) which gives the MintMT language it's 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.

There may be applications in which Mint does not provide sufficiently high execution speed. For these applications, it is possible to access the MML directly by writing an embedded application.

The MML is a library of functions that provides access to all the features of NextMove usually accessible through MintMT. Embedded applications are written in the 'C' programming language. This code can then be compiled and linked to the MML using the Texas C3x cross compiler to produce an application that replaces Mint on the controller.

For example, the MintMT code:

```
SPEED.0=10  
MOVER.0=100
```

becomes

```
SetSpeed( 0, 10.0F );  
SetMoveR( 0, 100.0F );
```

in embedded 'C'.

**Note:** Embedded applications are not multi-tasking.

This manual makes no reference to the general subject of 'C' programming and assumes a basic level of competence.

## 2.2 Abbreviations

The following abbreviations are used in this document.

DSP Digital Signal Processor

C3x Texas Instruments TMS320C3x processors

COFF Common Object File Format

DPR Dual Port RAM

MML Mint Motion Library



## 3.1 Overview

Programming NextMove using 'C' requires a considerably higher level of expertise and commitment of resources than using Mint. The tools required to program in this way must be sourced from third parties.

Source code files are generated on the host using any development environment. Object code and executables are generated using C3x software development tools. These tools are not supplied by Baldor UK Ltd. By linking with the supplied libraries, an executable application is created which is downloaded to NextMove replacing the normal MintMT firmware.

## 3.2 Tools

All NextMove controllers use the Texas TMS320C3x processor. To create an embedded application, you will need the Texas Instruments TMS320C3x compiler (code TMDS3243855-02). The libraries supplied by Baldor are built using v5.11 of the compiler, however all versions from v5.10 onward are compatible.

See section 6 for more details on the compiler.

The debugger and emulator package is useful for debugging but not essential (code TMDR3260130).

The TI web site has details on all the products at <http://dspvillage.ti.com/>

Distributors for the tools include Avnet, Arrow, SEI and Macro.

This document and all example applications assume that the compiler has been installed into the folder:

```
C:\C31v5_11
```

with

```
C:\C31v5_11\BIN           Executable files
C:\C31v5_11\INCLDUE      Header files
C:\C31v5_11\LIB          Library files
```

**Note:** The examples will need to be modified if this path is not correct. The compiler should be in the PATH.

## 3.3 Installing the embedded developer libraries

Run the 'NextMove Embedded Libraries' installation program. This will install the Mint Motion Libraries along with the required header files, example build files and example embedded application. The default installation folder is C:\MINTMT

The MintMT installation folder includes a number of sub folders:

- 
- Bin Any executable files required for building embedded applications.
  - Example An example embedded application with build files for all NextMove variants.
  - Include All header files required for building embedded applications
  - Libraries All library files required for building embedded applications.

---

## 3.4 Embedded example

The example embedded application, *example.c*, can be built and run on any NextMove variant. It shows the use of a user installed profile loop and servo loop together with a move buffer low event handler and a comms event handler.

### 3.4.1 Building the example

To build the application, open a command window in the example directory. The BUILD.BAT file is passed a parameter which specifies the NextMove product for which the example is being built:

```
NextMove PCI: BUILD PCI
NextMove BX: BUILD BX
NextMove BX II: BUILD BXII
NextMove ESB: BUILD ESB
NextMove ES: BUILD ES
NextMove ST: BUILD ST
```

For example, to build the example for NextMove PCI, at the C:\MINTMT\EXAMPLE> prompt type:

```
BUILD PCI
```

The BUILD.BAT file uses the compiler shell CL30 in order to compile the files specified in the COMPILE.CMD command file. The linker LNK30 is then called with the LINKxxx.CMD command file to produce an EMBEDDED.CHX application file. There is a separate link command file for each product which contains the linker options and details of the memory map for the specific controllers.

The output of the Texas Instruments compiler/linker is a .OUT file in COFF 0 format<sup>1</sup>. This file is then compressed to create a .CHX file that can be downloaded to the controller. This compressed format substantially decreases the download time for serial controllers.

The NextMove ES, ESB and ST examples use the CANopen variant of the embedded libraries. To use the BaldorCAN variant of the libraries, the LINKxxx.cmd files must be edited. The required lines are commented out in the file.

Typical compiler output is shown below:

```
C:\Proj\MintMT\Development Firmware\Nextmove\Embedded\Example>build PCI
[example.c]
TMS320C3x/4x ANSI C Compiler           Version 5.11
Copyright (c) 1987-1999 Texas Instruments Incorporated
"example.c" ==> main
"example.c" ==> setupAxes
"example.c" ==> myGenerator
"example.c" ==> myPreServoLoop
"example.c" ==> myServoLoop
```

---

<sup>1</sup> When using the TMS320C3x C Source debugger the .OUT application file must be used.

---

```
"example.c" ==> myPostServoLoop
"example.c" ==> myCommsEventHandler
"example.c" ==> myMoveBufferLowEventHandler
TMS320C3x/4x ANSI C Optimizer          Version 5.11
Copyright (c) 1987-1999 Texas Instruments Incorporated
"example.c" ==> main
"example.c" ==> setupAxes
"example.c" ==> myGenerator
"example.c" ==> myPreServoLoop
"example.c" ==> myServoLoop
"example.c" ==> myPostServoLoop
"example.c" ==> myCommsEventHandler
"example.c" ==> myMoveBufferLowEventHandler
TMS320C3x/4x ANSI C Code Generator     Version 5.11
Copyright (c) 1987-1999 Texas Instruments Incorporated
"example.c" ==> main
"example.c" ==> setupAxes
"example.c" ==> myGenerator
"example.c" ==> myPreServoLoop
"example.c" ==> myServoLoop
"example.c" ==> myPostServoLoop
"example.c" ==> myCommsEventHandler
"example.c" ==> myMoveBufferLowEventHandler
TMS320C3x/4x COFF Assembler           Version 5.11
Copyright (c) 1987-1999 Texas Instruments Incorporated
PASS 1
PASS 2

No Errors, No Warnings
<Linking>
TMS320C3x/4x COFF Linker               Version 5.11
Copyright (c) 1987-1999 Texas Instruments Incorporated

BUILD complete.
TMS320C3x/4x Hex Converter             Version 5.11
Copyright (c) 1987-1999 Texas Instruments Incorporated

BUILD: HEX conversion completed OK.

CompHex v1.02
Compressing output\embedded.out
Initial size = 721366
Size of compressed binary image = 335834 (46.6%)
Embedding banner: MintMT for NextMove PCI Build 5331 CANopen BaldorCAN
Verifying... Okay

BUILD: File compression completed OK.
```

---

## 3.4.2 Downloading the example

The firmware file (.CHX) can be downloaded to the controller from Mint *WorkBench* or from a host application.

In *WorkBench*, Select 'Download Firmware...' from the tools menu. Then select the 'Advanced' tab and 'Download Firmware file...'. Browse to the 'Embedded.chx' file and click 'Ok'. This will overwrite any existing firmware on the controller.

When writing your own application, it is important to use the compile and linker settings from the relevant example in order to produce an application that runs correctly. See the 'TMS320C3x Optimising C Compiler User's Guide' for details of the switches and options used.

You may easily build up your own application by specifying the files to use at the bottom of `COMPILE.CMD` and at the top of `LINKxxx.CMD`.

## 3.5 Debugging

Debugging an embedded application can prove difficult without an in-circuit emulator (ICE). However there are several ways to identify problems; these are described in section 3.5.1.

### 3.5.1 Creating debug information

#### DoPrint

The simplest form of debugging is to insert statements into the application that let you know where in the code the problem occurs, or the state of a variable. The first method is to make use of the Mint terminal with the `DoPrint()` function. For example:

```
__int8 cString[255];

DoPrint ( TERM1, "\n\rAbout to set Speed" );
SetSpeed ( 0, 1000.0 );
sprintf ( cString, "\n\rValue of speed is %f", GetSpeed ( 0 ) );
DoPrint ( TERM1, cString );
```

Using *WorkBench*, select the Edit & Debug tool from the toolbox, then any text printed from an embedded application will appear in the Output window. In the previous example the following text would be displayed:

```
About to set Speed
Value of speed is 1000
```

Using this method of debugging adds a lot of processing overhead to the application. If the section of code that you are trying to debug is time critical it may not be practical to add the delay of printing to the pseudo serial buffer. Also, as the text is buffered the host will read it some time after the embedded application has written the characters into the buffer. This can give a false impression of what line is being executed.

See the 'Terminal input/output' section in the MintMT Help File for details on terminal ports and printing.

---

## Dual Port RAM

When using NextMove PCI, an alternative to printing text is to write values to Dual Port RAM (DPR). The code overhead with this method is much smaller and so can be used in time critical sections of code. The value in DPR will be updated in real-time so the point of execution can be determined more accurately.

The user area of DPR can be used to write debug information (see section 7 for a DPR map). For example:

```
SetDPRLong ( 0xBE0, 1 );
SetSpeed ( 0, 1000.0 );
read = GetSpeed ( 0 );
SetDPRFloat ( 0xBE1, read );
```

The value in a specific location in DPR can be read using the Command Prompt in WorkBench. The keywords `DPRLONG` and `DPRFLOAT` can be used to read and write to DPR locations. The syntax for these is:

```
DPRtype.address = value
value = DPRtype.address
```

The DPR WatchWindow can also be used to view DPR locations.

## LED Display

Most NextMove products have an LED display. This can be used as an alternative to printing text. The code overhead with this method is much smaller so it can be used in time critical sections of code. Although it is difficult to display real numbers on the LED display, it can be used to indicate the point of execution. In order to write to the LED display it must be first put in user mode using the function `SetLED()`. The different LED segments can be illuminated by writing to the display using the function `SetLEDDisplay()`. The LED display will be updated on the next 2ms tick.

For example:

```
SetLED (-2);
SetLEDDisplay (0x0001);
SetSpeed (0, 1000.0);
SetLEDDisplay (0x0002);
```

## HSS

For a more graphical representation the HSS output pins can be toggled during execution. The state of these pins can be monitored using an oscilloscope to determine what section of code is being executed. The code overhead of writing to the HSS port is very small so this method can be used in real-time environments.

The HSS pins can also be used to time execution speeds of an embedded application. See section 5.5 for more details on the HSS port.

---


## 3.5.2 Using the C source debugger

It is possible to debug an application on NextMove using a TMS320C3x C Source debugger in conjunction with an in-circuit emulator pod.

An 11-pin footprint on the rear of the card marked 'ICE' provides access to the processor for boundary scan emulation. To access this a two row 12-pin 0.1" pitch surface mount pin header with pin 8 missing must be fitted. The connections are those specified by Texas Instruments. Please contact your local Baldor representative to obtain a NextMove with this header fitted.

The only recommended emulator pod is the Texas Instruments XDS510. Both the XDS510 debugger software and the Code Composer Suite have been used for application debugging.

In order to single-step through an application, the system watchdog will need to be disabled in order to avoid the watchdog barking and putting the card into reset.

 **WARNING:** Disabling the watchdog will remove fail safe protection for the firmware. If the emulator is stopped with a move in progress, the DAC outputs will not be cleared down which will result in axes moving in an uncontrolled fashion. It is NOT recommended that debugging is performed on the application machine but preferably in a 'laboratory' environment where no physical or personal harm can be inflicted.

Contact your local Baldor representative for information on how to disable the watchdog functionality.



## 4.1 Mint Motion Library (MML) data types

Although all data types on NextMove are 32-bit (with the exception of the extended precision floating point number, see section 6.11), it is still useful to use a data type that is meaningful to the data being held.

The Mint Motion Libraries define a number of data types for this purpose:

```
__int8      - signed 8-bit
__int16     - signed 16-bit
__int32     - signed 32-bit

__uint8     - unsigned 8-bit
__uint16    - unsigned 16-bit
__uint32    - unsigned 32-bit

BOOL       - boolean value
```

The MML functions use data types appropriate to the size of the data being passed as arguments.

For example, the prototype for `setSpeed()` is:

```
__int16 SetSpeed ( __int16 nAxis, float fSpeed );
```

The return error code is defined as a signed 16-bit value, the axis number as a signed 16-bit value and the speed to set as a 32-bit floating point value.

## 4.2 Calling API Functions

The MintMT API allows motion functions to be called from embedded C. The prototypes for the functions can be found in the file `MMLAPI.H`. Pre-defined constants can be found in the file `MIL.H`.

If a function call fails, a system error is generated. For example, trying to set a parameter on an invalid axis will cause an 'erINVALID\_AXIS' error which has a value of 2. The `GetSystemError` function can be used to return the error code from the last function called.

```
__int16 nError;
SetSpeed ( -1, 10.0 );          /* Axis -1 is invalid */
nError = GetSystemError ();    /* nError will equal 2 */
SetSpeed ( 0, 10.0 );         /* Axis 0 is valid */
nError = GetSystemError ();    /* nError will equal 0 */
```

Error codes can be found in chapter 7.

A system error will also cause an entry in the error log that can be read using WorkBench. Go to the Application pane and click on 'Error Log'. System errors will be displayed as type 'System' with a fault description of the form:

*MML call (426) failed : 'Axis specified out of range'*

The function call number is an enumeration of all API functions. These can be found in chapter 7.

---

## 4.3 Accessing Dual Port RAM (DPR)

On NextMove PCI, there are a number of functions to read and write data to DPR from an embedded application. DPR is 32-bits wide and is addressable in the range 0x000 to 0xFFF. See section 7 for details on the contents of DPR.

Read / write a float

```
float   GetDPRFloat ( __int16 nAddress );
void    SetDPRFloat ( __int16 nAddress, float fValue );
```

Read / write a signed 32-bit value

```
__int32 GetDPRLong ( __int16 nAddress );
void    SetDPRLong ( __int16 nAddress, __int32 lValue );
```

## 4.4 Event handling

Events can be generated in response to the following conditions:

- Asynchronous Errors – an asynchronous error occurred on the NextMove card.
- Fast position latch – an axis has latched position
- Stop switch – a stop switch has become active
- Bus event -an event on CAN bus 1 or 2
- Timer – the timer event period has expired
- Digital input active – a digital input has become active
- DPR event – the user generated a DPR event.
- Move Buffer Low - the numbers of moves in a move buffer drops below a specified threshold.
- Axis Idle - an axis has become idle.

**Note :** For more details on events see the MintMT help file.

The events are prioritized in the following order:

Priority	Event
0: Highest	Error
1	Bus
2	Stop switch
3	Fast position latch
4	Timer
5	Digital input
6	Comms

---

Priority	Event
7	DPR event
8	Move Buffer Low
9	Axis Idle

In order for the user to be informed of an event, a handler must be installed. Event handlers may be installed in embedded and host applications. If both an embedded handler (Mint or MML) and a host handler are installed then only the embedded handler will be called.

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

#### 4.4.1 Events within MML

For events to be generated each event requires a handler to be installed. The handlers are installed with the following functions:

##### Error handler

The install function for error events accepts a pointer to a function, if this is a NULL pointer the handler is uninstalled.

```
typedef void TErrorHandler ( void );  
void InstallErrorHandler ( TErrorHandler *pHandler );
```

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 handler

The install function for fast position latch events accepts a pointer to a function, if this a NULL pointer the handler is uninstalled.

```
typedef void TFastInEventHandler ( void );  
void InstallFastInEventHandler ( TFastInEventHandler *pHandler );
```

The source of the call to the fast position latch handler can be established using the function `GetFastLatch()`.

---

### Stop switch handler

The install function for stop switch events accepts a pointer to a function, if this is a NULL pointer the handler is uninstalled.

```
typedef void TStopSwitchEventHandler ( void );
void InstallStopSwitchEventHandler ( TStopSwitchEventHandler *pHandler );
```

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

### BUS event handler

The install function for BUS events accepts a pointer to a function, if this is a NULL pointer the handler is uninstalled. The `lBusBitmap` parameter is a bit pattern representing the bus that caused the event.

```
typedef void TBusEventHandler ( __int32 lBusBitmap );
void InstallBusEventHandler ( TBusEventHandler *pHandler );
```

### Timer event handler

The install function for timer events accepts a pointer to a function, if this is a NULL pointer the handler is uninstalled. The parameter passed to the function will always be zero.

```
typedef void TTimerEventHandler ( __int16 nTimerEventNumber );
void InstallTimerEventHandler ( TTimerEventHandler *pHandler );
```

### Digital input handler

The install function for digital input events accepts a pointer to a function, if this is a NULL pointer the handler is uninstalled. When the input event handler is called, it is passed a bit pattern of activating 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 bank 2) the expansion cards are read sequentially. The events for digital inputs on each card have the same priority.

```
typedef void TInputEventHandler ( __int16 nBank,
                                  __int32 lActivatedInputs );
void InstallInputEventHandler ( TInputEventHandler *pHandler );
```

### Comms location changed event

The install function for comms events accepts a pointer to a function, if this is a NULL pointer the handler is uninstalled. When the comms event handler is called, it is passed a bit pattern indicating the changed comms locations. Comms events will only be generated for comms location 1 to 5 inclusive.

```
typedef void TCommsEventHandler ( __int32 lCommsEventPending );
void InstallCommsEventHandler ( TCommsEventHandler *pHandler );
```

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 from the Host

The install function for DPR events accepts a pointer to a function, if this is a NULL pointer the handler is uninstalled. When the DPR event is generated the function called is passed a 16-bit code, as passed by the host.

```
typedef void TDPREventHandler ( __int16 nCode );  
void InstallDPREventHandler ( TDPREventHandler *pHandler );
```

### Move-Buffer-Low

The install function for Move-Buffer-Low events accepts a pointer to a function; if this is a NULL pointer the handler is uninstalled. 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.

```
typedef void TMoveBufferLowEventHandler ( __int16 nAxisBitPattern );  
void InstallMoveBufferLowEventHandler ( TMoveBufferLowEventHandler *pHandler );
```

### Axis Idle

The install function for axis idle events accepts a pointer to a function; if this is a NULL pointer the handler is uninstalled. When the axis idle event is generated the function called is passed a 16-bit code representing the axes which have become idle.

```
typedef void TAxisIdleEventHandler ( __int16 nAxisBitPattern );  
void InstallAxisIdleEventHandler ( TAxisIdleEventHandler *pHandler );
```

## 4.4.2 Event control

There are various functions which can be used to control event generation. These are detailed below. The bit pattern accepted by these functions is described in the following table:

Bit	Event
0	<i>Reserved</i>
1	Error
2	Bus
3	<i>Reserved</i>
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)

---

Bit	Event
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 function `GetEventPending` and `SetEventPending`. This accepts the same bit pattern. 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`. This function accepts the same bit pattern. 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 function accepts a bit pattern which represents all digital inputs; if the bit is set then the digital input will generate an event when the input becomes active.

The function `GetIMask` will return a bit pattern representing those digital inputs which will generate an event when they become active.

The user can interrupt a host application using the function: This will interrupt the host and pass a code to the host's DPR event handler. Although the parameter passed is a 16-bit variable, due to the way the code is passed across Dual Port RAM only an 8 bit code can be passed.

### 4.4.3 Event handling example

The `TimerEvent` example can be found in the folder `\MintMT\Example\Embedded\TimerEvent`.

The example will install a Timer Event handler and set the period between calls to 1000ms. When called the Timer Event handler will display a text string.

The idea of the Mint open architecture is to allow users to add their own functionality to Mint. This allows users to:

- modify or replace the servo loop for any/all axes.
- replace the motion profiler for any/all axes.

For an embedded application, the user links with the MML libraries as normal to generate a user application file. Hooks are provided which allows the user to install custom servo loops and profile loops.

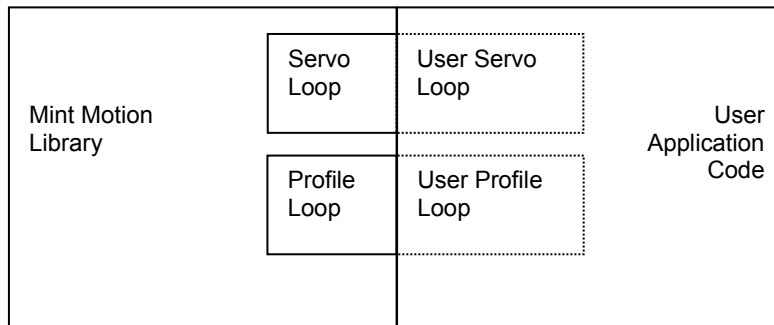


Figure 1 - Mint open architecture

## 5.1 Data available

In order to use the open architecture, the user must have access to a number of internal and external (user) parameters. Internal NextMove data can be accessed through a number of structures:

```
_TServoLoopData
_TAxisData
_TAuxAxisData
_TIOData
```

These structures provide copies of a number of selected common internal parameters that might be useful within both a servo loop and a trajectory generator. Any other parameters may be accessed through the standard MML function calls.

`TServoLoopData` contains data that is pertinent to the servo loop, axis position, servo loop gain terms etc.

```
typedef struct {
    float *actualPosition;
    float *actualSpeed;
    float *demandPosition;
    float *demandSpeed;
```

---

```

float *demandAcceleration;
float *followingError;
float *kProp;
float *kInt;
float *kDeriv;
float *kVel;
float *kVelFff;
float *kAccel;
float *DACValue;
BOOL *useDAC;
} _TServoLoopData;

```

<code>actualPosition</code>	This is the actual position of the axis, normally derived from the encoder position.
<code>actualSpeed</code>	This is the difference in <code>actualPosition</code> in consecutive servo loops.
<code>demandPosition</code>	This is the demand position for the axis, normally derived from the profiled position.
<code>demandSpeed</code>	This is the demand speed of the axis. Normally this is only updated when new positions are loaded as, due to linear interpolation of position, the velocity between profiled positions is constant.
<code>demandAcceleration</code>	This is the demand acceleration of the axis, normally only updated when new positions are loaded.
<code>followingError</code>	This is the difference between <code>demandPosition</code> and <code>actualPosition</code> .
<code>kProp</code>	This is the proportional gain term as set by the MML function <code>setKProp()</code> .
<code>kInt</code>	This is the integral gain term as set by the MML function <code>setKInt()</code> .
<code>kDeriv</code>	This is the derivative gain term as set by the MML function <code>setKDeriv()</code> .
<code>kVel</code>	This is the velocity feedback gain term as set by the MML function <code>setKVel()</code> .
<code>kVelFff</code>	This is the velocity feedforward gain term as set by the MML function <code>setKVelFF()</code> .
<code>kAccel</code>	This is the acceleration feedforward gain term as set by the MML function <code>setKAccel()</code> .
<code>DACValue</code>	This is the value to output to the DAC.
<code>useDAC</code>	If this is set to TRUE then the DAC on the NextMove card will be used, otherwise the DAC output will not be applied to the hardware.

These pointers are setup during initialization so they can be used without any pre-initialization. Where applicable all the data is in counts. All numerical data is in 32-bit floating point format. The position range of the NextMove controllers is -8388608 to +8388607.

These two values are defined as macros in MIL.H:

```

#define MAX_FLOAT          8388607.0F /* max. MML_FLOAT (24 bit mantissa) */
#define MIN_FLOAT         -8388608.0F /* min. MML_FLOAT (24 bit mantissa) */

```

---

All positional data must be wrapped to keep it within these limits. Failure to do this will lead to unexpected behavior.

TAxisData contains data for all controlled axes on the controller.

```
typedef struct {
    float    previousDemandPosition;
    __int32  deltaPosition;
    __int32  deltaEncoder;
    __int32  measuredPosition;
    __int32  encoderPosition;
    float    scaleFactor;
    float    invScaleFactor;
    float    encScaleFactor;
    float    invEncScaleFactor;
    float    speed;
    float    acceleration;
    float    deceleration;
    float    errorDeceleration;
    __int32  flags;
} _TAxisData;
```

previousDemandPosition	The demand position generated by the previous profile loop in user units.
deltaPosition	The number of encoder counts or stepper pulses received since the last profile loop; this is in effectively the axis velocity.
deltaEncoder	The number of encoder counts received since the last profile loop. This is the velocity of the encoder.
measuredPosition	The current measured position in encoder counts or stepper pulses.
encoderPosition	The current encoder position in encoder counts.
scaleFactor	The axis scale factor. This is the number of encoder counts / stepper pulses in a user unit, as set by the MML function <code>setScale()</code> .
invScaleFactor	The inverse of the scale factor (1/scale factor)
encScaleFactor	The encoder scale factor. The number of encoder counts in a user unit, as set by the MML function <code>setEncoderScale()</code> .
invEncScaleFactor	The inverse of the encoder scale factor (1/encoder scale factor).
speed	The axis speed value in user units per profile loop, as set by the MML function <code>setSpeed()</code> .
acceleration	The axis acceleration value in user units per profile loop per profile loop, as set by the MML function <code>setAccel()</code> .
deceleration	The axis deceleration value in user units per profile loop per profile loop, as set by the MML function <code>setDecel()</code> .
errorDeceleration	The axis deceleration value in user units per profile loop per profile loop used when the axis is in error. This is set by the MML function <code>setErrorDecel()</code> .

---

flags

See section 5.3.3

TAuxAxisData contains data for all auxiliary axes on the controller.

```
typedef struct {
    __int32 deltaEncoder;
    __int32 encoderPosition;
    float encScaleFactor;
    float invEncScaleFactor;
} _TAuxAxisData;
```

`deltaEncoder` The number of encoder counts received since the last profile loop, this is the velocity of the encoder.

`encoderPosition` The current encoder position in encoder counts.

`encScaleFactor` The encoder scale factor, to convert the number of encoder counts to a user unit. This is set by the MML function `setEncoderScale()`.

`invEncScaleFactor` The inverse of the encoder scale factor (1/encoder scale factor)

TIOData contains data for all inputs on the controller.

```
typedef struct {
    __int32 inputBank0;
    __int32 inputBank0Raw;
    __int32 inputBank1;
    __int32 inputBank0Raw;
    __int32 inputBank2;
    __int32 inputBank0Raw;
    float analog[12];
} _TIOData;
```

`inputBank0` Bit pattern of inputs on the main board, this is processed data. It reflects any INPUTACTIVELEVEL's which have been specified. In the case of edge triggered inputs the activating edges are latched until cleared by calling the MML function `getIN()`.

`inputBank0Raw` Bit pattern of input levels on the main board. The inputs are sampled every millisecond. This is raw data and reflects the state of the inputs as read prior to the processing of the profiler.

`inputBank1` Bit pattern of inputs on expansion board 0, this is processed data. It reflects any INPUTACTIVELEVEL's which have been specified. In the case of edge triggered inputs the activating edges are latched until cleared by calling the MML function `getIN()`.

`inputBank1Raw` Bit pattern of input levels on expansion board 0. The inputs are sampled every millisecond. This is raw data and reflects the state of the inputs as read prior to the processing of the profiler.

---

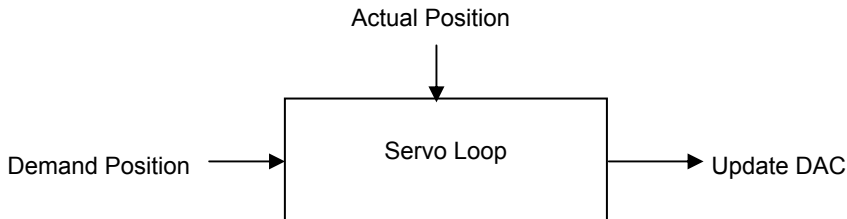
inputBank2	Bit pattern of inputs on the expansion board 1, this is processed data. It reflects any INPUTACTIVELEVEL's which have been specified. In the case of edge triggered inputs the activating edges are latched until cleared by calling the MML function getIN().
inputBank2Raw	Bit pattern of input levels on expansion board 1. The inputs are sampled every millisecond. This is raw data and reflects the state of the inputs as read prior to the processing of the profiler.
analog	Analog values for all ADC channels on the main board and expansion boards.

These are accessible to the user with the `servoLoopData[]`, `axisData[]`, `auxAxisData[]` and `IOData` variables.

## 5.2 Servo loop

The purpose of the servo loop is to execute the control law on each axis.

It reads the actual position of the axis and calculates the actual speed as well as demand position, speed and acceleration then generates a DAC output using the control law.



By default the closure time for the servo loop is every 1000µs. This can be changed using the MML function `SetLoopTime`.

### 5.2.1 Installing servo loop hooks

There are three user installable servo loop options:

#### Pre-Servo Loop

This is called directly after the actual and demand positions have been calculated, but prior to the point that the following error is calculated. This is designed to allow users to overload either the actual position (read from the encoder) or the demand position (generated by the profiler). If either of these two values are modified the user must modify the associated speed and acceleration parameters to allow the control law to function correctly.

To install a pre servo loop handler use the function `InstallPreServoLoop()`:

```

typedef void TPreServoLoop ( __uint8 ucAxis );
void InstallPreServoLoop ( __uint8 ucAxis, TPreServoLoop *pHandler );
  
```

---

The installed handler will be called automatically by the MML libraries. To uninstall a handler, pass a NULL pointer as the handler.

### **Main Servo Loop**

This is called in place of the PIDVFA control law that is used to generate the DAC demand value. If this hook is installed then no control law will be processed on the controller. This is designed to allow users to write their own control law.

To install a main servo loop handler use the function `installMainServoLoop()`:

```
typedef void TMainServoLoop ( __uint8 ucAxis );  
void InstallMainServoLoop ( __uint8 ucAxis, TMainServoLoop *pHandler );
```

The installed handler will be called automatically by the MML libraries. To uninstall a handler, pass a NULL pointer as the handler.

### **Post Servo Loop**

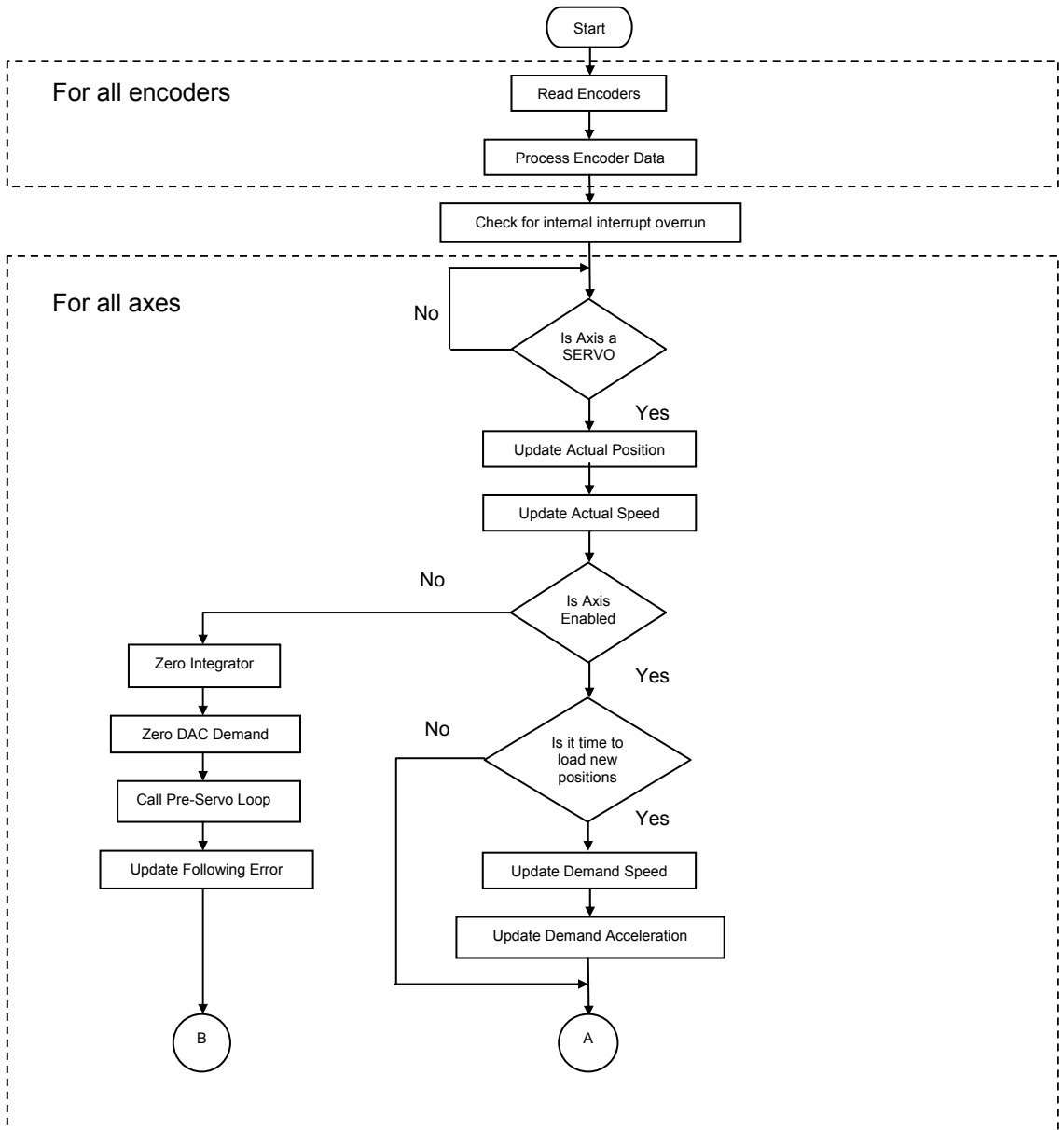
This is called directly before the DAC value is written to the hardware. This is designed to allow a DAC filter to be implemented or to allow the DAC value to be written to a different DAC.

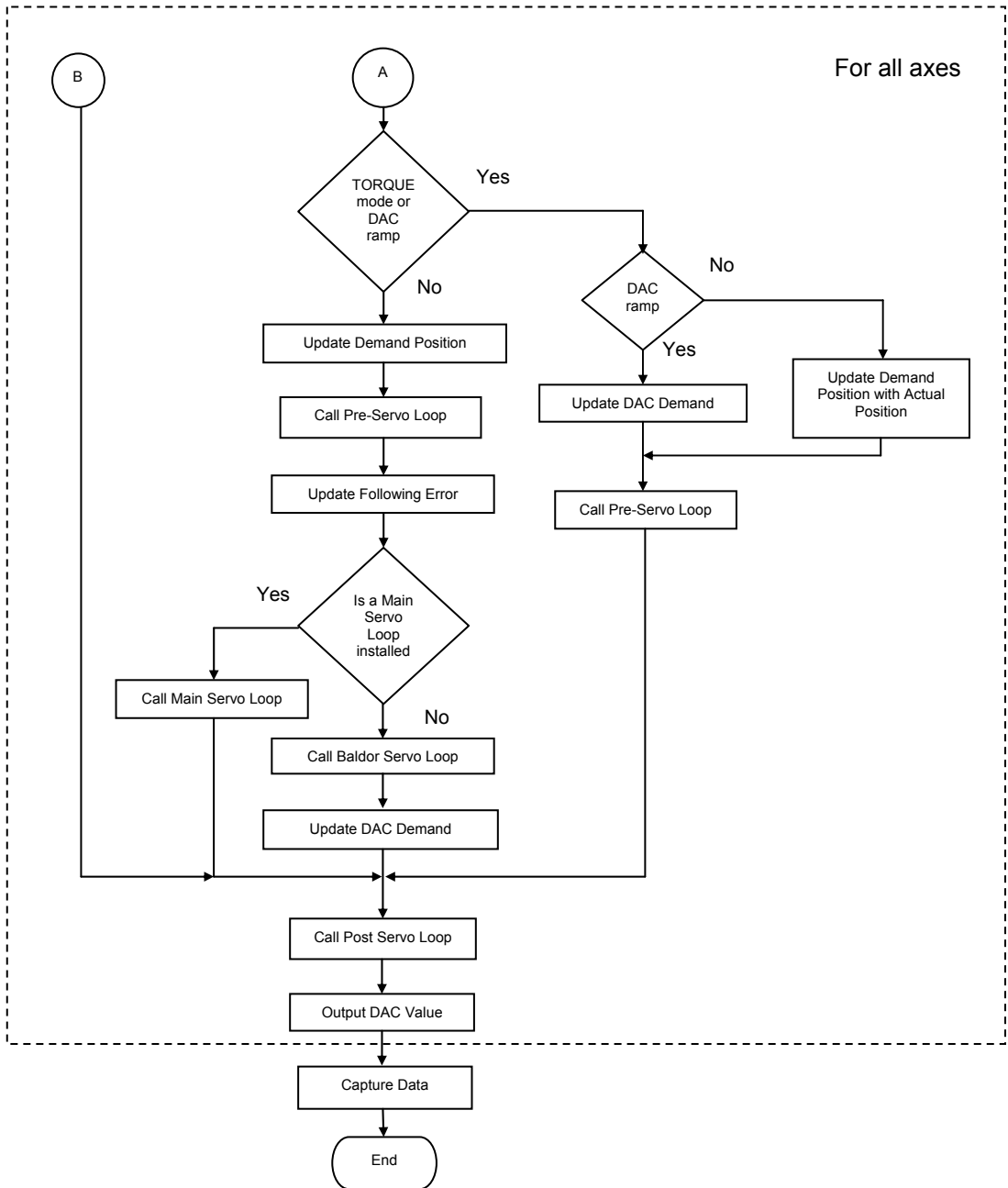
To install a post servo loop handler use the function `installPostServoLoop()`:

```
typedef void TPostServoLoop ( __uint8 ucAxis );  
void InstallPostServoLoop ( __uint8 ucAxis, TPostServoLoop *pHandler );
```

The installed handler will be called automatically by the MML libraries. To uninstall a handler, pass a NULL pointer as the handler.

## 5.2.2 The servo loop flow diagram





---

### 5.2.3 Servo loop example

The ServoLoop example can be found in the folder \MintMT\Example\Embedded\ServoLoop.

This example installs pre, main and post servo loops for axis zero.

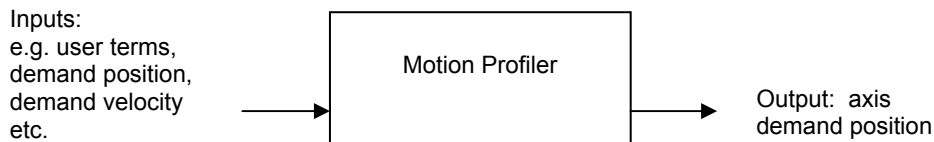
The pre-servo loop keeps a copy of the actual position of the axis but wraps it at 16384 giving a position range of 0 to 16383. This value is then written to DPR at location 0xBE0. The code is wrapped in a toggle of timer pin 1 to allow timing.

The main servo loop is a simple PID controller which generates a DAC demand value based on the actual and demand position of the axis. The code is wrapped in a toggle of timer pin 2 to allow timing.

The post servo loop applies a smoothing function to the DAC output. The code is wrapped in a toggle of timer pin 3 to allow timing.

## 5.3 Motion profiler

By default the motion profiler runs every 2ms and performs a number of functions. For each axis in turn, it checks if a move type is loaded and triggered for motion and then calls the specific trajectory generation routine for that move type. It then processes the new demand position produced for the axis so that it can be picked up by the next servo loop. The output of a trajectory generator is a floating point demand position scaled to user units.



The inputs to the trajectory generator may be any user parameters as well as a number of internal parameters made available to the users trajectory generator.

### 5.3.1 Installing a motion profiler hook

A motion profiler is simply a function or set of functions that is specified to be the motion profiler for an axis. It is installed using the `InstallProfileLoop()` function.

```
typedef void TProfileLoop ( __uint8 ucAxis )  
void InstallProfileLoop ( __uint8 ucAxis, TProfileLoop *pHandler )
```

Once the user motion profiler is installed, it can be considered as a new move type called 'user'. It does not replace the existing motion generation routines such as homing or linear moves but is an addition to them. A separate profiler may be installed for each axis or multiple axes may use the same one. A profiler is not limited to creating motion only on the axis for which it is installed. It may create motion for any number of axes.

In order to use the new motion profiler, the user must load a move. The function `SetUserMotion` will perform some range checking and check the state of the axis. In order to load a move on an axis certain conditions must be satisfied:

- 
- The axis must not be in error.
  - The axis must be configured as either a servo, stepper or a virtual axis.
  - The drive must be enabled.
  - The axis must be idle (i.e. there cannot be any other moves loaded on the axis).

If any of these conditions are not met the function will return an error code.

The state parameter allows the user move type to be turned on and off. A state of 1 will load the move. A state of zero effectively unloads the move and stops the user motion profiler from being called.

`GetAxisMode()` will return to idle.

Once the move has been loaded, reading the current mode of motion will be `mdUSER` and the axis will no longer be idle. The move must be started by calling the `doGo()` function. Every 2ms, the user motion profiler will then be called for that axis until the move is stopped by the user or by an error condition.

### 5.3.2 Writing the new demand position

In order to write the new demand position for any axis, the `SetDemandPosition()` function must be called. The function will perform some range checking and check the state of the axis. In order to update the demand position of an axis certain conditions must hold.

- The axis must not be in error.
- The axis must be configured as either a servo or a stepper.
- The drive must be enabled.
- The mode of the axis must be `mdUSER` (i.e. there must be a user motion enabled for this axis).

If any of these conditions are not met the function will return an error code. A user motion profiler may calculate a demand position for multiple axes as long as those axes are in user mode.

### 5.3.3 Error handling

Most error and event handling is performed automatically. All 'crash stop' type situations are performed internally.

All ramp down style events must be performed in the user's motion profiler. This applies to any error condition that has an error mode set to `emERROR_DECEL`. It also applies to the `DoStop()` and `SetSuspend()` functions and the stop input. When the ramp down condition is indicated, the user should bring motion to a controlled stop. The user motion profiler does not have to 'ramp' the motion to a halt. Any profile is acceptable, but if the user does not support this functionality then the `DoStop()` and `SetSuspend()` functions will not operate. The axis will also fail to respond to the error conditions (which could result in mechanical damage) if the error mode is `emERROR_DECEL`.

The flags parameter of the `TAxisData` structure is used to pass information and instructions to the users motion profiler. The flags field is a bit pattern as follows:

---

Value	Meaning
0	No action required.
1	Deceleration required. This is caused by the doStop() function, the Suspend() function and by the stop switch being activated.
3	Error deceleration required. This is caused by an error condition or by the stop switch being activated.

Failure to check bit 0 of the flags field on all axes performing a user move will mean that the axis will not stop as normal when requested. Failure to check bit 1 of the flags field on all axes performing a user move could result in an axis not stopping in an error condition.

**Note:** Error modes should NOT be set to error deceleration unless code to support this has been written in the motion profiler.

### 5.3.4 Motion profiler example

The Profiler example can be found in the folder \MintMT\Example\Embedded\Profiler.

This example installs a custom profiler for axes 0 to 3. The profiler generates a trapezoidal velocity profile to move an axis to a specified target position. This example will respond correctly to requests to decelerate in the event of an error.

## 5.4 Calling APIs from within the handlers

Since the open architecture handlers must run quickly, it is not possible to call the majority of API calls from within the handlers. Attempting to do so will cause the internal ticks to overrun and the controller to crash. The following APIs may safely be called from within a handler:

```
SetDemandPosition  
GetDPRFloat  
SetDPRFloat  
GetDPRLong  
SetDPRLong
```

## 5.5 Code execution time

NextMove code is interrupt driven, meaning both the servo loop and the profiler are triggered at regular intervals. Because of this it is essential that both the servo loop and the profiler complete within the allocated time slice.

The servo loop has a dedicated interrupt, which by default is called every 1ms.

The profiler shares its interrupt with the housekeeping routine. These run alternately every 2ms.

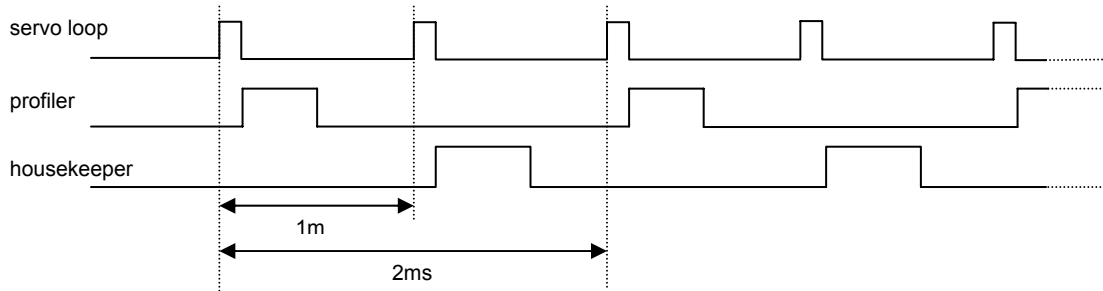


Figure 2 - Interrupt Timing Diagram

The servo loop interrupt is a higher priority than the profiler/housekeeper interrupt. Higher priority interrupts will interrupt the lower priority interrupts if their time slices overlap, so it is essential that these functions complete in the specified time slice.

The execution time of these routines can be monitored via the HSS header.

On NextMove PCI a 10-pin footprint on the rear of the card marked 'HSS' provides access to the HSS port. To access this a two row 10-pin 0.1" pitch surface mount pin header must be fitted. Please contact your local Baldor representative to obtain a NextMove with this fitted.

On NextMove BX the HSS port connector comes fitted as standard, it is located is next to the bootloader EEPROM and labeled 'JP13'.

The servo loop, profiler and housekeeping routines are always output via timer pins. The remaining 3 timer pins are available to the user via the macros:

```
TIMER_PIN1_ON
TIMER_PIN1_OFF

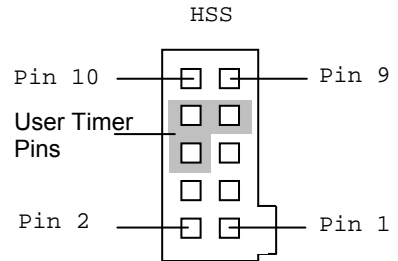
TIMER_PIN2_ON
TIMER_PIN2_OFF

TIMER_PIN3_ON
TIMER_PIN3_OFF
```

Sections of code can be wrapped in these macros. When a timer is turned on, it causes a pin of the HSS port to be set high. When the timer is turned off, the pin goes low. An oscilloscope may be used to time the duration of the pulse.

Details of the HSS pin out are described in the following table.

Pin	Function
1	Ground
2	Ground
3	Servo Loop
4	Profiler
5	Housekeeper
6	Timer Pin 1
7	Timer Pin 2
8	Timer Pin 3
9	Ground
10	Ground



The NextMove will keep track of the timings as much as possible. In the event of a timing violation bit 8 of the miscellaneous error bit pattern will be set and the user installed error handler will be called. The condition which will cause a timing violation if a house keeper tick occurs while a profiler is in progress.

Using the entire time slice up will result in the main application code running slowly, so the servo loop and profiler must be as fast as possible.

Although the NextMove will generate an error in the event of a timing violation the only real way to determine which sections of code are taking the most time is to monitor the timing pins on the HSS header using an oscilloscope.



## 6.1 Overview

For full details on the Texas Instruments tools, refer to the documentation supplied with them. The following sections provide some useful information when using these tools.

## 6.2 Compiling code

The Texas Instruments (TI) compiler is an ANSI 'C' cross compiler with associated tools (assembler, linker, hex converter, etc.) that run on a PC. The tools are command line driven utilities.

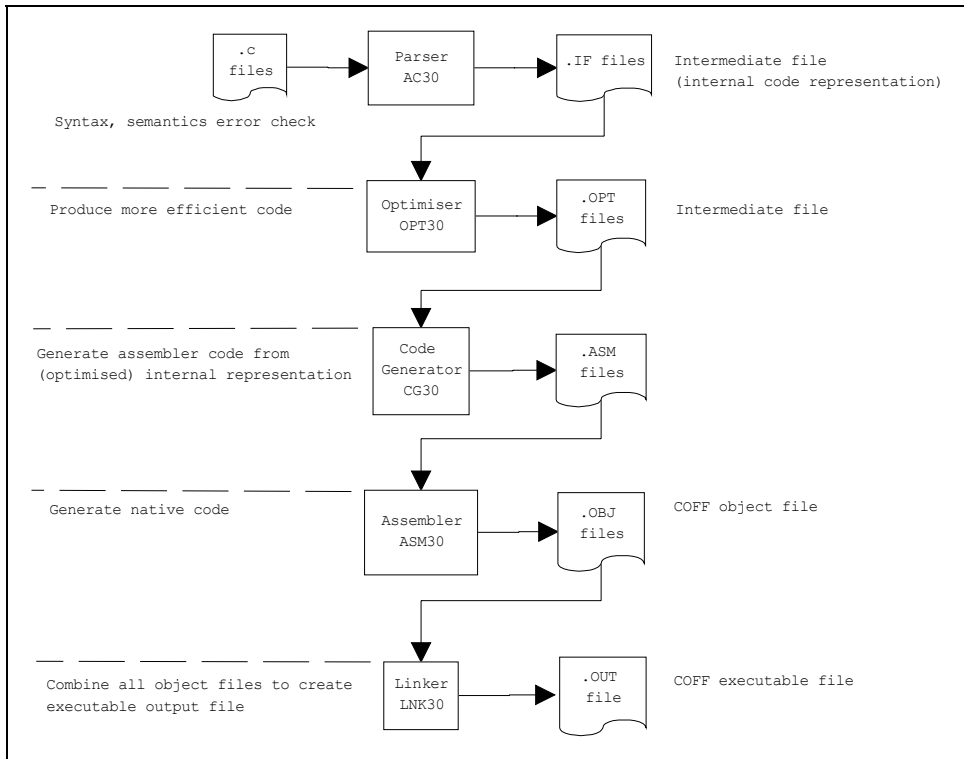
The Texas Instruments 'Code Composer' suite may be also be used. It is important that the compiler and linker settings are transferred from the example build files to the Integrated Development Environment (IDE) to ensure correct production of an application.

The C31 compiler tools include:

- AC30        compiler (parser)
- ASM30     assembler (makes COFF object)
- CG30       code generator
- OPT30     optimizer
- LNK30     COFF linker (makes COFF executable)
- CL30       compiler shell, combines above 5 tools
- HEX30     hex converter (convert COFF executable to intel Hex format for NextMove BX)
- CLIST      interlist utility
- AR30       archiver

Although the 'C' compiler conforms to the ANSI 'C' standard there are a few special considerations that must be taken into account due to the processor architecture and design of the NextMove controller.

The quickest way to compile code is to use the compiler shell CL30.EXE. This is a utility that allows you to compile, assemble and optionally link code in a single step.



**Figure 3 - Compilation Stages**

The shell runs one or multiple files through the following individual tools:

- compiler
  - parser (AC30)
  - optimiser (OPT30)
  - code generator (CG30)
- assembler (ASM30)
- linker (LNK30)
- hex conversion (HEX30)

## 6.3 Run-time libraries

Texas Instruments supply a run-time library in the form of a .LIB file that contains object code for common 'C' functions like strcpy(), malloc(), etc. They also supply a set of header files that contain the function prototypes, macro definitions and some global variables. Source for the run-time library is also provided as an archive file.

---

The standard run time library RTS.SRC includes math functions written in 'C'. Texas Instruments provide alternative source code for these functions written in assembler (MATHASM.SRC). These are claimed to be faster than the 'C' code versions, however they do change the functionality of some of the math functions.

The NextMove code is built on the standard RTS30.LIB run-time library.

It is recommended that the RTS30.LIB library is used.

## 6.4 COFF files

COFF is an abbreviation for Common Object Format File. This is a structured file that may contain symbolic information required by source level debuggers, relocation information for smart loaders, etc. as well as the code itself. Version 5.0 of the compiler introduces two new versions of the COFF format. WorkBench only supports COFF 0 format.

## 6.5 Environment settings

### **C\_DIR and A\_DIR Environment Variable**

These environment variables are used by the TI compiler. They describes search paths that the compiler should use to locate macro libraries and include files.

```
SET C_DIR=<dirs>          to set cpp and linker search paths
SET A_DIR=<dirs>          to set asm and linker search paths
```

You can specify multiple paths separated by ';', for example:

```
SET C_DIR=C:\C31v5_11\BIN;C:\C31v5_11\LIB;C:\C31v5_11\INCLUDE
```

This will be setup by the build files to be:

```
SET A_DIR=C:\C31v5_11\BIN
SET
C_DIR=C:\C31v5_11\BIN;C:\C31v5_11\LIB;C:\C31v5_11\INCLUDE;..\..\..\INCLUDE
```

### **C\_MODE Environment Variable**

There are two versions of each of the Texas Instruments tools, a protected mode version and a real mode version. The compiler shell CL30 will normally invoke the real mode versions of the tools. You may find that during some stages of the compilation process the process is aborted with a 'not enough memory' error. The protected mode versions of the tools allow more memory to be used. The C\_MODE environment variable is used by the shell to determine which version of the tools it should invoke. The examples use the protected mode versions of the tools:

```
SET C_MODE=PROTECTED
```

---

## 6.6 Memory models

The Texas Instruments compiler can generate code for two memory models - big and small. The big memory model is not used and should therefore not be used by any code linking to the libraries. This is because the big model produces larger / slower code and is not suitable for real-time control.

The small model places no restriction on the size of code, stack or heap. All external variables, static variables and compiler generated constants in the entire program (after linking) must fit into 64k x 32. The .bss section cannot cross any 64k page boundaries.

Should you require to generate large data structures that push your program data requirements beyond the 64k limit, dynamically allocate memory from the heap. This method allows access to the entire address range with no real speed or size penalty save for the allocation / de-allocation overhead.

## 6.7 Argument passing

The Texas Instruments compiler allows you to specify whether function arguments are passed on the stack or within C31 registers. The library code is compiled for stack argument passing. User code that is to be linked with the libraries should be compiled in the same way. Code will not run if you link modules that have been compiled with different options.

## 6.8 Two pass parsing / out of memory errors

Most of the development tools allocate memory from the heap to build tables during processing. The tools may fail with 'Out Of Memory' errors if there is insufficient memory on your machine.

The parser normally runs a one pass process. When it does this it uses PC memory to store macro definitions and symbol definitions. When run in two passes these functions can be separated. The first pass performs pre-processing so that memory is only required for the macro definitions. In the second pass there are no macro definitions so the maximum amount of memory is available for the symbol table. If you experience 'out of memory' errors when compiling a solution may be to parse in two passes. See the 'TMS320 Floating Point DSP Optimizing C Compiler User Guide'.

## 6.9 Entry point

The file MML\_BOOT.LIB contains the program entry point `c_int00`. When a program begins running, MML\_BOOT is executed first.

## 6.10 Sections

The compiler produces six relocatable blocks of code and data. These blocks are called sections:

- `.TEXT`      executable code

- 
- .CINIT      tables for initializing variables
  - .CONST     string constants and switch tables
  - .BSS        global and static variables
  - .STACK     system stack
  - .SYSMEM    heap

The compiler treats memory as a linear block that is partitioned into smaller blocks of code and data. Each block of code or data that a 'C' program generates will be placed in its own contiguous space in memory. It is the linker that defines the memory map and allocates code and data into target memory.

You specify where these sections (and any sections that you define yourself) should be located in the NextMove memory map within a linker command file. The example linker command file LINK.CMD illustrates how this is done.

## 6.11 Data types

Most of the C31 data types are represented in 32-bits, with the exception of the extended precision floating point number which is 40-bits.

Data type	Size	sizeof
char	32 bits	1
short	32 bits	1
int	32 bits	1
long	32 bits	1
float	32 bits	1
double	32 bits	1
long double	40 bits	2

The range of values that a type may take is the range that can be represented in 32-bits. char, int, long and float are all the same size. Note that the sizeof operator will return 1 for all types except the long double.

The extended precision floating point format is stored in two memory words and will require two instruction cycles to load/store. Note that only addition and subtraction benefit from the extended precision; all multiplication uses a truncated 32-bit version. There is also an instruction overhead when converting from 40-bit floating point to 32-bit floating point due to the use of the RND instruction.

As all data types are 32-bit, declaring a variable as an 8-bit value will not limit its range. For example, incrementing a variable declared as a 8 bit value will not wrap at 0xFF back to 0x00 but will continue to increment. To ensure that none of the upper bits are set, masking with an appropriate mask such as 0xFF will limit the range.

---

## 6.12 Division

There are no divide instructions on the C31 so all division is done within library functions. These functions may take some time to execute. If you need to perform division in time critical portions of code you should store the inverse and multiply if possible.

## 6.13 Variable initialization

Notice that variables are not zero initialized as standard so you must do this in your code. You should not assume the value of any variable or pointer unless you have initialized it yourself.

## 7.1 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 DPR data is disabled. To enable the DPR update write to the DPR Control Register at address zero. This can be done from an embedded application with the functions:

```
void SetDPRLong ( __int16 nAddress, __int32 lValue );
```

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

```
DPRLONG.address = value
```

## 7.2 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	

## 7.2.1 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	✓
0x00E	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:

- Low-Byte - Number of stepper axes
- High-Byte - Number of servo axes

### Digital I/O Mix

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

- Low-Byte - Number of digital outputs
- High-Byte - Number of digital inputs

### Analog I/O Mix

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

- Low-Byte - Number of analogue outputs
- High-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 16-bit counter that is updated by NextMove once every 2ms and can be used to synchronize data with the DPR.

### Axis Configurations

This 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 a 32-bit incrementing counter that indicates that NextMove is running. The counter increments by 1 every 1ms.

## 7.2.2 Axis data

The axis data area is divided into 12 sections, four for the main board axes and four for 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
0x0A	<i>Reserved</i>		

Offset	Use	Symbolic Constant	Data Size
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>		

\*These locations are only written when they change. All other data is written every 2ms.

### 7.2.3 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>		
0x11A	Base Digital Outputs	roOUTPUTS	int 16
0x11B	Stop / Error bits	roMG_STATUS	int 16

Address	Use	Symbolic Constant	Data Size
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.

## 7.2.4 Comms array

The Comms array simulates protected Comms communications on serial based controllers. It uses an area of DPR from address 0x1D6 to 0x29A.

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

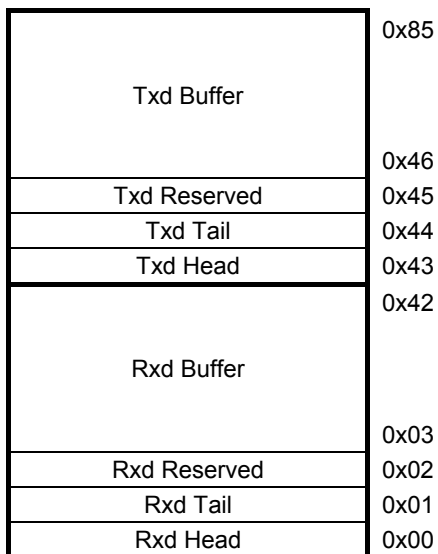
Each location is a float value. Comms is accessed using the COMMS keyword in MintMT or the `GetComms()` / `SetComms()` functions.

## 7.2.5 Immediate command mode interface

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

## 7.2.6 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 allow both sides of DPR to check the status of the buffers. The serial interface occupies DPR locations 0x150 to 0x1D5 in the following configuration:



The buffer itself has two sets of symbolic constants, depending on which side, NextMove or host, 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 start of the serial I/O buffer has a symbolic constant of *ofSERIAL\_IO\_BASE*.

## 7.2.7 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	apNM_MINT
5	Mint for NextMove (with ICM interface)	apFRONT_MINT
6	Reserved	
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 DPR, 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.

## 8.1 Function Call Enumerations

If an API call fails, an entry will be added to the error. The function enumeration of the call that failed will be displayed. The following list shows the function enumerations.

AAABuild	0
Abort	1
AbortMode	2
Accel	3
AccelTime	4
ActiveRS485Node	6
ADC	7
ADCError	11
ADCErrorMode	12
ADCMax	14
ADCMin	15
ADCMode	16
ADCMonitor	17
AsyncError	25
AsyncErrorPresent	26
AuxDAC	32
AuxEncoder	33
AuxEncoderMode	34
AuxEncoderScale	35
AuxEncoderVel	37
AuxEncoderWrap	38
AuxEncoderZLatch	42
AxisChannel	43
AxisError	44
AxisMode	45
AxisStatus	46
AxisWarning	47
AxisWarningDisable	48
Backlash	49
BacklashInterval	50
BacklashMode	51
Boost	55
Cam	59
CamAmplitude	60
CamEnd	61
CamIndex	62
CamPhase	63

---

CamPhaseStatus	64
CamSegment	65
CamStart	66
CamTable	67
BusBaud	68
BusReset	69
BusState	70
Cancel	71
CancelAll	72
BusEvent	73
BusEventInfo	74
Capture	75
CaptureAxis	76
CaptureEvent	79
CaptureEventAxis	80
CaptureEventDelay	81
CaptureInfo	83
CaptureInterval	84
CaptureMode	85
CaptureNumPoints	86
CapturePeriod	87
ChannelType	88
CircleA	89
CircleR	90
Cls	91
CommsRemote	92
CommsMode	93
CompareLatch	94
CompareMode	95
ComparePos	96
Config	97
Connect	98
ConnectInfo	99
ConnectStatus	100
ContourMode	101
DAC	109
DACLimitMax	110
DACMode	111
DACMonitorAxis	112
DACMonitorGain	113
DACMonitorMode	114
DACOffset	117
DACRamp	118
Decel	122

---

DecelTime	123
Default	125
DefaultAll	126
DPREvent	129
DPRFloat	130
DPRLong	131
DriveEnable	140
DriveEnableOutput	142
Encoder	156
EncoderMode	157
EncoderPreScale	158
EncoderScale	159
EncoderVel	160
EncoderWrap	161
EncoderZActiveLevel	162
EncoderZLatch	163
ErrorDecel	164
ErrorInput	165
ErrorInputMode	166
ErrorMask	167
EventActive	181
EventDisable	182
EventPending	183
FastAuxEnable	187
FastAuxEncoder	188
FastAuxLatch	189
FastAuxLatchMode	190
FastAuxSelect	191
FastEnable	192
FastEncoder	193
FastLatch	194
FastLatchMode	195
FastPos	196
FastSelect	198
Feedrate	201
FeedrateMode	202
FeedrateOverride	203
Fly	205
FolError	206
FolErrorFatal	207
FolErrorMode	208
FolErrorWarning	209
Follow	210
FollowMode	212

---

Freq	213
Gearing	214
GearingMode	215
GlobalErrorOutput	217
Go	218
Group	219
GroupComms	220
GroupInfo	221
GroupMaster	222
GroupMasterStatus	223
GroupStatus	224
Home	227
HomeBackoff	228
HomeInput	229
HomePos	231
HomeSpeed	232
HomeSwitch	233
HTA	237
HTAChannel	238
HTADamping	239
HTADeadBand	240
HTAFilter	241
HTAKInt	242
HTAKProp	243
Idle	244
IdlePos	245
IdleVel	246
IMask	247
In	248
IncA	249
IncR	250
InitError	251
InputActiveLevel	252
InputMode	253
InputNegTrigger	254
InputPosTrigger	255
InState	256
InStateX	257
InX	260
Jog	261
KAccel	270
KDeriv	271
KInt	277
KIntLimit	278

---

KIntMode	279
KProp	281
KVel	284
KVelFF	285
LED	290
LEDDisplay	291
LifeTime	292
Limit	293
LimitForward	294
LimitForwardInput	295
LimitMode	296
LimitReverse	297
LimitReverseInput	298
LocateCursor	301
LoopTime	302
MasterChannel	303
MasterDistance	304
MasterSource	305
MaxSpeed	306
MiscError	308
MiscErrorDisable	309
MoveA	337
MoveBufferFree	339
MoveBufferID	340
MoveBufferIDLast	341
MoveBufferLow	342
MoveBufferSize	343
MoveBufferStatus	344
MoveR	351
MoveStatus	352
Node	353
NodeLive	354
NodeScan	355
NodeType	356
NumberOf	357
NVFloat	358
NVLong	359
Offset	360
OffsetDistance	361
OffsetMode	362
OffsetStatus	363
Out	364
OutputActiveLevel	365
OutX	366

---

Platform	369
Pos	373
PosDemand	374
PosRemaining	375
PosTarget	376
PrecisionIncrement	378
PrecisionMode	379
PrecisionOffset	380
PrecisionTable	381
ProfileTime	384
PWMOnTime	386
PWMPeriod	387
ReadKey	388
Relay	389
RemoteADC	390
RemoteADCDelta	391
RemoteBaud	392
RemoteDAC	393
RemoteDebounce	394
RemoteEStop	396
RemoteIn	397
RemoteInputActiveLevel	398
RemoteInX	399
RemoteMode	400
RemoteNode	401
RemoteObject	402
RemoteOut	405
RemoteOutputActiveLevel	406
RemoteOutputError	407
RemoteOutX	408
RemoteReset	409
RemoteStatus	411
Reset	412
ResetAll	413
ScaleFactor	417
SerialBaud	419
SoftLimitForward	422
SoftLimitMode	423
SoftLimitReverse	424
Speed	426
Spline	441
SplineEnd	442
SplineIndex	443
SplineStart	444

---

SplineSuspendTime	445
SplineTable	446
SplineTime	447
SRamp	448
StepperIO	449
Stop	450
StopInput	451
StopInputMode	452
StopSwitch	453
Suspend	454
SystemSeconds	457
TerminalAddress	459
TerminalDevice	460
TerminalMode	461
TerminalPort	462
Time	463
TimerEvent	464
Torque	466
UserMotion	477
Vel	478
VelDemand	479
VelError	480
VelFatal	481
VelFatalMode	482
ICMChannel	488
FactoryDefaults	489
CaptureBufferSize	500
MaxAccel	501
MaxDecel	502
TerminalFlush	504
VectorA	505
VectorR	506
SplineSegment	517
Wait	523
CaptureModeParameter	540
OkToLoadMove	556
ContourAngle	562
HomePhase	581
HomeStatus	582
EventPort	583
CommsMultipleRemote	629
PulseCounter	630
TriggerMode	631
InputDebounce	632

---

AuxEncoderPreScale	637
HSSTiming	654
AccelJerk	675
AccelJerkTime	676
DecelJerk	677
DecelJerkTime	678
ProfileMode	679
Comms	683
CommsMultiple	684
AxisVelEncoder	694
Print	716
StepperDelay	724
HelixA	726
HelixR	727
PosRollover	728
MultipleInterpolatedMoves	729
MoveDwell	735
PulseOutX	736
MovePulseOutX	737
MoveOutX	738
MoveOut	739
RemoteInhibitTime	740
ErrorLogString	741
Blend	742
BlendDistance	743
BlendMode	744
CaptureDuration	745
PhaseSearchInput	746
PhaseSearchCurrent	747
AccelSensorVelErrorFatal	748
AccelSensorFilterFreq	749
AccelSensorOffset	750
AccelSensorScale	751
AccelSensorVelError	752
AccelSensorMode	753
AccelSensorType	754
CompareOutput	755
FastSource	756
CompareEnable	757
EraseLine	758
BusCommandTelegram	759
BusReceiveTelegram	760
BusProcessDataInTelegram	761
BusProcessDataOutTelegram	762

---

BusTelegramDiagnostics	763
BusTelegramDiagnosticStrings	764
ContourParameter	765
SpeedFilterDepth	766
RemoteEmergencyMessage	767
RemoteError	768
TorqueFilterType	769
TorqueFilterFreq	770
TorqueFilterBand	771
TorqueFilterDepth	772
ModuleName	773
IdleTime	774
SpeedDemandFilterType	775
SpeedDemandFilterFreq	776
RemoteInBank	777
RemoteOutBank	778
MPGCount	779
MotorDirectionOutput	780
CurrentSensorMode	781
CompareSource	782
BusCommandMask	783
MotorBrake	784
MotorBrakeDelay	785
MotorBrakeOutput	786
CommsMapMode	787
CommsMapParameter	788
CommsMapInfo	789
CommsMapList	790
FastLatchDistance	791
RemotePDOIn	792
RemotePDOOut	793
KnifeMode	794
KnifeMasterAxis	795
KnifeStatus	796
MotorBrakeMode	797
TriggerValue	798
CamBoxData	799
CamBox	800
IdleSettlingTime	801
IdleMode	802
Knife	803
CapturePoint	804
CaptureChannelUpload	805
DiagnosticIndexedString	806

---

RedirectAPICall	807
ExecuteAPICall	808
ProductSerialNumber	809
ProductCatalogNumber	810
IDCTParameter	811
DriveTestMode	812
PosScaleFactor	813
VelScaleFactor	814
AccelScaleFactor	815
ControlRate	816
ControlRefSource	817
TuningTestSchedule	818
VelRef	819
APIDefinition	820
APIArgumentDefinition	821
MotorType	822
LoadFriction	823
LoadOffset	824
PhaseSearchOutput	825
PhaseSearchSwitch	826
PosScaleUnits	827
VelScaleUnits	828
AccelScaleUnits	829
FlashFileHandle	830
FlashFileData	831
FlashFileReleaseHandle	832
FastAuxLatchDistance	833
ParamTableValue	834
ParamTableFamilyName	835
ObjectDictionaryEntry	836
ForceAbort	837
AppDataFreeAddress	838
AppDataObjectInfo	839
TerminalValid	840
PDODataElements	841
PDODataIndicesIn	842
PDODataIndicesOut	843
AxisPosEncoder	844
ExtKnifeEventHandler	845
EncoderSpeed	846
PosRolloverTarget	847
FeedrateParameter	848
PosTargetLast	849
PosWrap	850

---

JogAway	851
JogAwayOffset	852
MoveCorrection	853
PosRolloverTargetLast	854
PosRolloverDemand	855
SystemState	856
DiagnosticIndexedParameter	857
MonitorParameter	858
ContourRatio	859
FileHandle	860
FileClose	861
FileData	862
FileDelete	863
FileSystemDetails	864
FileSystemEntry	865
FileSize	866
ProductPowerCycles	867
RemoteObjectString	868
EncoderOutZCount	869
KITime	870
AxisDAC	871
AxisPDOOutput	872
SerialParity	873
OperatingMode	874
PrechargeRatio	875
RelayOut	876
RelayOutX	877
BusProcessDataInDatatype	878
BusProcessDataOutDatatype	879
MACAddress	880
CommsInteger	881
BusAppObjectForPDO	882
BusCommsObjectForPDO	883
BusPDOConfig	884
BusPDOContent	886
InternalFunction1	888
InternalFunction2	889
BusNode	890
EncoderType	891
EncoderResolution	892
EncoderCycleSize	893
EncoderStatus	894
AbsEncoderTurns	895
DriveTestParameter	896

---

CaptureTriggerChannel	897
CaptureTriggerAbsolute	898
CaptureTriggerValue	899
CaptureTriggerType	900
CaptureTrigger	901
CapturePreTriggerDuration	902
CaptureProgress	903
AnalogInputSetup	904
AnalogOutputSetup	905
DigitalInputBankSetup	906
DigitalOutputBankSetup	907
RelayOutputBankSetup	908
EncoderSetup	909
MappedDevice	910
RemoteNumberOf	911
CaptureTriggerMode	912
CaptureTriggerSource	913
ParamTableMode	914
APIArgumentValues	915
Stepper	916
BusProcessDCF	917
AuxEncoderRollover	918
BusCycleTime	919

## 8.2 System Error Codes

erSUCCESS	0	/* no error	*/
erMML_ERROR	1	/* synchronous MML error	*/
erINVALID_AXIS	2	/* axis specified out of range	*/
erVALUE_OUT_OF_RANGE	3	/* data specified out of range	*/
erINVALID_CHANNEL	4	/* adc / dac channel out of range	*/
erNO_INPUT_SPECIFIED	5	/* op on home/limit etc with no i/p	*/
erNO_OUTPUT_SPECIFIED	6	/* op on enable output with no o/p	*/
erINVALID_INPUT	7	/* digital input out of range	*/
erINVALID_OUTPUT	8	/* digital output out of range	*/
erOUT_OF_MEMORY	9	/* not enough heap for operation	*/
erMOTION_IN_PROGRESS	10	/* action denied when axis in motion	*/
erAXIS_NOT_RIGHT_TYPE	11	/* action denied when in wrong config	*/
erMOTION_ERROR	12	/* general motion (async) error	*/
erTABLE_ERROR	13	/* Bad Spline or cam table info	*/
erCAN_ERROR	14	/* Unable to initialise the CAN bus	*/
erCHANNEL_NOT_RIGHT	15	/* Channel incorrectly configured	*/
erDPR_TIMEOUT	16	/* DPR timeout	*/

---

```

erWRONG_PLATFORM          17 /* Not available on this controller */
erDB_ERROR                18 /* Initialise daughter board failed */
erSERIAL_ERROR            19 /* Problem with RS232 or RS485 port. */
erWRONG_NODE_TYPE        20 /* node referenced not expected type */
erCAN_TIMEOUT             21 /* Failed to receive reply in time */
erNODE_NOT_LIVE           22 /* node is not LIVE */
erTYPE_NOT_SUPPORTED      23 /* type of node not supported */
erINVALID_HARDWARE        24 /* Hardware not present */
erCMS_DATABASE_FULL       25 /* All CMS COBIDs have been allocated */
erINVALID_NODE_ID         26 /* CAN node number out of range */
erREMOTE_EE_FAIL          27 /* Problem writing to EEPROM on node */
erINVALID_REMOTE_BAUD     28 /* Node doesn't support baud rate */
erREMOTE_SYNC_ERROR       29 /* Node reported a synchronous error */
erREMOTE_ESTOP_ACTIVE     30 /* Node in ESTOP condition */
erINVALID_BUS_NUMBER      31 /* CAN bus number was out of range. */
erNO_FREE_CAN_OBJECTS     32 /* There were no free message objects left
/* in the CAN controller. */
erCAN_BUS_OFF             33 /* The CAN controller is bus off. */
erCAN_TX_BUFFER_FULL      34 /* The CAN transmit buffer was full. */
erTERMINAL_UNAVAILABLE    35 /* No terminal device */
erTERMINAL_OUT_OF_RANGE   36 /* Port value is out of range */
erNON_VOLATILE_MEMORY_ERROR 37 /* Problems with non-volatile memory */
erTERMINAL_BUFFER_EMPTY   38 /* Terminal Buffer is empty */
erTERMINAL_BUFFER_FULL    39 /* Terminal Buffer is full */
erDRIVE_DISABLED          40 /* Drive is not enabled */
erNO_CONNECTION           41 /* No Connection Exists */
erCAN_PROTOCOL_ERROR      42 /* CAN Protocol Error During Communication.*/
erINVALID_LOCAL_NODE      43 /* Local CAN node not correct. */
erNOT_NETWORK_MASTER      44 /* Must be master of network */
erCOMMS_READ_ONLY         45 /* COMMS element is read only. */
erCOMMS_RESERVED_ELEMENT  46 /* COMMS reserved element */
erOUTPUT_FAULT            47 /* Fault on the digital outputs. */
erDSP_FAILED_TO_CLEAR_ERROR 48 /* DSP Failed to clear the error. */
erOFFSET_PROFILE_ERROR    49 /* The Offset cannot be Profiled */
erFLASH_PROGRAMMING       50 /* Error programming Flash */
erADDRESS_OUT_OF_RANGE    51 /* Error addressing Flash */
erCRC_CHECKSUM_ERROR      52 /* Error in received Checksum */
erFLASH_BEING_PROGRAMMED  53 /* Command invalid when Flash in use */
erFILE_TOO_BIG            54 /* File too big for available memory */
erCAN_INVALID_OBJECT      55 /* Invalid CAN object */
erCAN_RESERVED_OBJECT     56 /* Reserved CAN object */
erCAN_INVALID_CHANNEL     57 /* CAN node channel out of range */
erCAN_VALUE_OUT_OF_RANGE  58 /* Data specified out of range */
erMOVE_BUFFER_FULL        59 /* Move buffer is full */

```

---

```

erICM_ARRAY_ERROR          60 /* ICM protocol error          */
erICM_TOO_MANY_ARRAYS     61 /* ICM protocol error          */
erICM_DATA_TIMEOUT        62 /* ICM Timeout                  */
erICM_BLOCK_TOO_BIG       63 /* ICM protocol error          */
erICM_TX_SIZE_MISMATCH    64 /* ICM protocol error          */
erICM_RETURN_TIMEOUT      65 /* ICM Timeout                  */
erMML_NOT_SUPPORTED       66 /* MML does not support this function */
erINVALID_POINTER        67 /* addresses area outside memory */
erINVALID_MODE           68 /* invalid error action mode     */
erDEBUG_DISABLED        69 /* Debug keywords are disabled.  */
erINVALID_MASTER_CHANNEL 70 /* Master Channel invalid for Axis. */
erALL_AXES_MUST_BE_OFF   71 /* All Axes must be configured Off. */
erINVALID_AXISMODE       72 /* Move not allowed in this mode.  */
erDOWNLOAD_TIMEOUT       73 /* Timeout during File Download.  */
erCAPTURE_IN_PROGRESS    74 /* Capture in progress during upload */
erINVALID_NUM_CAP_PTS    75 /* Invalid number of capture points */
erINVALID_BBP_TRANS_NO   76 /* BBP Transaction No. not supported */
erINVALID_BBP_FIELD_LENGTH 77 /* BBP Transaction has wrong length */
erBBP_DATA_OUT_OF_RANGE  78 /* BBP Transaction data invalid   */
erBBP_DATA_OUT_OF_BOUNDS 79 /* BBP Transaction data modified  */
erBBP_FAULT_PREVENTS_EXEC 80 /* BBP Transaction can't be executed */
erBBP_MODE_PREVENTS_EXEC 81 /* BBP Transaction can't be executed */
erBBP_BLOCK_NOT_ACCEPTED 82 /* BBP Block transfer not accepted */
erBBP_END_OF_BLOCK_REACHED 83 /* BBP End of block reached       */
erUNKNOWN_BBP_ERROR      84 /* Unknown BBP error code         */
erDRIVE_TIMEOUT          85 /* BBP Transaction timed-out      */
erBBP_OVERFLOW           86 /* BBP Transaction Rx overflow    */
erINVALID_BBP_PACKET_SIZE_RXD 87 /* BBP Transaction Rxd size too big */
erINVALID_BBP_TRANSACTION_RXD 88 /* Invalid BBP Transaction Rxd    */
erCHANNEL_IN_USE         89 /* Hardware channel required is in use */
erDRIVE_ENABLED          90 /* Drive is enabled.              */
erINVALID_DRIVE_PARAM    91 /* Invalid Drive Parameter No.    */
erBBP_TRANSACTION_IN_PROGRESS 92 /* A BBP transaction is executing.  */
erNO_BBP_TRANSACTION_REQUESTED 93 /* No BBP transaction requested.  */
erICM_DISABLED           94 /* ICM is disabled on this channel */
erOUTPUT_IN_USE          95 /* Output is already in use.      */
erCAPTURE_CHANNEL_MIX    96 /* Invalid capture channel mix.   */
erALL_CONTROL_AXES_IN_USE 97 /* All controllable axes in use   */
erINVALID_VAR_TYPE       98 /* Invalid variable type for RemoteObject. */
erCAN_CONFIRMED_BUSY     99 /* A confirmed service is already in
/* progress.                  */
erFILE_PROTECTED         100 /* Mint File is Protected.        */
erREMOTE_DOWNLOAD_IN_PROGRESS 101 /* A Mint file is currently being
/* downloaded to a remote node.  */

```

---

---

```

erBBP_REQUEST_TIMEOUT          102 /* Timeout on BBP request. */
erPARAMETER_ACCESS_CONFLICT    103 /* Two devices updating same parameter */
erCAN_ALREADY_CONNECTED        104 /* CANOpen node is already connected to
/* another node. */
erREMOTE_DRIVE_DISABLED        105 /* The remote drive is disabled (CANOpen). */
erREMOTE_DRIVE_FAULT           106 /* The remote drive is in fault mode
/* (CANOpen). */
erREMOTE_STATE_INCORRECT       107 /* Transaction aborted due to nodes state. */
erREMOTE_DRIVE_MOVE_FAILED     108 /* The remote drive failed to accept the
/* new move (CANOpen). */
erINCOMPATIBLE_SETTINGS        109 /* Incompatible with previous settings */
erDSP_COMMS                     110 /* Inter-processor communications error */
erAUTOTUNE_FAILURE             111 /* Autotuning operation failed */
erREAD_ONLY                     112 /* Parameter is read only */
erICM_DLL_ERRORS               113 /* Errors in the Rxd DLL SO telegram */
erICM_DLL_MESSAGE_ID_MISMATCH  114 /* DLL SO telegram id doesn't match */
erICM_TL_HOST_RETRANSMITS      115 /* Too many host retransmit requests */
erICM_TL_TRANSMIT_REQUESTS     116 /* Too many transmit requests - timed-out */
erICM_TL_BUSY                  117 /* TL is busy - still processing command */
erICM_TL_NO_OF_PACKETS         118 /* Invalid No of packets specified */
erICM_TL_GROUP_MESSAGE_ID      119 /* Group message id mismatch */
erICM_TL_GROUP_SEQUENCE        120 /* Group message packet No out of sequence */
erEEPROM_ACCESS                121 /* Error accessing EEPROM device */
erINITIALISATION_FAILURE       122 /* A failure occurred during initialisation*/
erINVALID_ALLOCATION_TABLE      123 /* Invalid object allocation table */
erOBJECT_NOT_FOUND             124 /* Application data object not found */
erMINT_PROGRAM_RUNNING         125 /* A Mint program is already running */
erINVALID_MINT_COMMAND         126 /* The Mint command is invalid */
erINVALID_TERMINAL_PORT        127 /* Port type for terminal is not valid */
erINVALID_TERMINAL_DEVICE      128 /* Device for terminal is not valid */
erINVALID_TERMINAL_ADDRESS     129 /* Address type for terminal is not valid */
erUNDEFINED_TERMINAL           130 /* Terminal is not defined */
erSINGLE_TERMINAL_ONLY          131 /* A single terminal is required */
erICM_HOST_BUSY                132 /* The host is not ready for the ICM reply */
erINVALID_PLATFORM_CODE        133 /* The image files platform is invalid */
erINVALID_IMAGE_FORMAT_CODE    134 /* The image files format is invalid */
erAXIS_NOT_COMMISSIONED        135 /* The axis/drive is not commissioned. */
erFEEDBACK_NOT_RESPONDING      136 /* Feedback device comms is not responding */
erFEEDBACK_MESSAGE_CORRUPT     137 /* Feedback device comms message is
/* corrupt */
erFEEDBACK_COMMS_BUSY          138 /* Feedback device comms is busy */
erFEEDBACK_ERROR               139 /* Feedback device has error */
erCAN_INVALID_SUBINDEX         140 /* Invalid subIndex for CAN object */
erCAN_INVALID_OBJECT_ACCESS    141 /* Invalid access to a CAN object */

```

---

---

```

erMOVE_BUFFER_NOT_EMPTY      142 /* Move buffer is not empty.          */
erINCOMPATIBLE_CONTROL_MODE  143 /* Incompatible control mode.         */
erVARIABLE_NOT_FOUND         144 /* Static variable not found.         */
erINVALID_STATIC_HANDLE     145 /* Invalid handle for static variable. */
erINVALID_STATIC_CHUNK      146 /* Invalid chunk specified for static. */
erSTATIC_DATA_OVERRUN       147 /* Static filled, but data remains.    */
erSTATIC_DATA_UNDERRUN      148 /* Data consumed, but static not filled.*/
erINCORRECT_REF_SOURCE      149 /* Reference source is not Host when trying*/
                             /* to set a speed ref using Mint keyword */
erFEEDBACK_DATUM_ERROR      150 /* Feedback device is unable to datumise */
erPHASE_SEARCH_RUNNING       151 /* Phase search is in progress         */
erFEEDBACK_MOTOR_DATA_ERROR  152 /* Motor data stored on encoder is invalid */
erFIELDMARSHAL_EEPROM_ACCESS 153 /* Error accessing FieldMarshal EEPROM */
erFIELDBUS_INIT_ERROR       154 /* Error initialising the Fieldbus card */
erPARAMETER_TABLE_VERSION   155 /* Parameter Table version not supported */
erPARAMETER_TABLE_PLATFORM   156 /* Parameter Table doesn't match platform */
erPARAMETER_TABLE_DOWNLOAD   157 /* Can't download Parameter Table.     */
erCANNOT_CONTOUR_AND_BLEND   158 /* Can't contour and blend at same time. */
erPARAMETER_TABLE_INDEX     159 /* Parameter index is out of sequence   */
erTRANSFER_IN_PROGRESS      160 /* A parameter table transfer is already */
                             /* in progress                          */
erTRANSFER_NOT_READY        161 /* Drive has not been prepared for     */
                             /* parameter table transfer             */
erKNIFE_HANDLER_NOT_INSTALLED 162 /* Need a KNIFE handler for knife control */
erKNIFE_AXES_NOT_CONFIGURED  163 /* The axes for knife control not set   */
erCAN_MESSAGE_COBID_DISABLED  164 /* CAN message is disabled in the COB_ID */
erCAN_MESSAGE_NMT_DISABLED    165 /* CAN message is disabled by the NMT state*/
erKNIFE_ROTATION_OUT_OF_RANGE 166 /* Knife rotation out of range         */
erTOO_MANY_TASKS            167 /* Too many tasks in Mint program       */
erNO_DATA_STORED            168 /* No data stored to recover            */
erDATA_STORED               169 /* Data already stored                  */
erEVENT_HANDLER_ALREADY_INSTALLED 170 /* The event handler is already installed. */
erTEST_TIMEOUT              171 /* Test has timed out                  */
erTEST_OVERTRAVEL          172 /* Test has caused overtravel          */
erTEST_COMMUTATION_INCORRECT 173 /* Commutation settings incorrect      */
erCOULD_NOT_OPEN_FLASH_DRIVER 174 /* Could not open the specified file in */
                             /* the E1's flash filing system        */
erINVALID_FLASH_FILE_HANDLE  175 /* Invalid handle to open E1 flash file */
erCAN_SDO_ABORT             176 /* An SDO transaction has been aborted  */
erEPL_SDO_ABORT             177 /* An SDO transaction has been aborted  */
erEPL_UNKNOWN_ERROR         178 /* Encountered unrecognized EPL error code */
erFILE_HANDLE_INVALID       179 /* File handle is not valid            */
erFILE_NOT_FOUND           180 /* File does not exist                 */
erFILE_IN_USE               181 /* File is in use                      */

```

---

---

erTOO_MANY_FILES	182	/* Attempting to store too many files	*/
erFILE_TYPE_INCORRECT	183	/* File has the wrong type	*/
erTOO_MANY_FILES_OPEN	184	/* Too many files open at once	*/
erFILE_WRITE_INVALID	185	/* Problems writing to file in flash	*/
erFILE_READ_INVALID	186	/* Problems reading from file in flash	*/
erFILE_SECTOR_INVALID	187	/* Problems with file sector allocation	*/
erRESOURCE_UNAVAILABLE	188	/* Unable to assign software resource	*/
erINVALID_OBJECT_INDEX	189	/* Invalid Object Dictionary index	*/
erINVALID_OBJECT_SUBINDEX	190	/* Invalid Object Dictionary subindex	*/
erREDIRECT_STATE_ERROR	191	/* Encountered a redirecting API seq error	*/
erREDIRECT_TIMEOUT	192	/* A redirect has timed out	*/
erREDIRECT_INVALID_DATA	193	/* Invalid data rx'ed during redirection	*/
erEPL_TIMEOUT	194	/* Failed to receive reply in time	*/
erAXIS_NOT_IN_REMOTE_MODE	195	/* Axis will not accept remote commands	*/
erSYSTEM_ENABLED	196	/* System must be disabled	*/
erINVALID_POS_ENCODER	197	/* No position encoder has been assigned	*/
erINVALID_VEL_ENCODER	198	/* No velocity encoder has been assigned	*/
erINVALID_DAC	199	/* No DAC has been assigned	*/
erINVALID_PDOUTPUT	200	/* No pulse/direction output has been assigned	*/
erREDIRECT_IN_PROGRESS	201	/* A redirect call is already in progress	*/
erINVALID_OBJECT_ACCESS	202	/* Invalid Object Dictionary access	*/
erMOTION_TYPE_NOT_SUPPORTED	203	/* Move is not supported by remote profiler	*/
erUNABLE_TO_PEND_EVENT	204	/* Unable to pend event	*/







Baldor UK Ltd  
Mint Motion Centre  
6 Bristol Distribution Park  
Hawkley 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



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