LXE CE API Programming Guide

E-SW-WINAPIPG-E

Notices

LXE's CE API Programming Guide *contains software developed by LXE for use on LXE certified equipment. Any reference, whether direct or implied, to any LXE wireless or RF equipment requires the reader to refer to the specific equipment's User Manuals for cautions, warnings and federal notices (e.g. FCC, EMC, UL, CE, etc.).*

Revision Notice

LXE CE API Programming Guide

Update from Revision D to Revision E

Rev.	Section	Description
B	Chapter 2 – API Calls	Changed LXEInfoRAMID to not supported for all computers. Updated LXEPcmciaHasCard to reflect second PCMCIA slot on VX6 and VX7. Added extended support for these computers. Updated LXEPcmciaDisableSlot and LXEPcmciaEnableSlot to reflect second PCMCIA slot on VX6 and VX7. Updated LXEPcmciaWriteProtect to include extended values and extended support for MX3X computers. Updated LXEBatteryIsCharging and LXEBatteryIsDischarging to clarify API's are for backup battery. Updated LXEScannerSetKey for new values: SCANKEY_FLDEXIT and SCANKEY_RFID. Revised introduction to Scanner API section with updated buffer clearing information. Added MX5 PPC, MX5X and MX6 scanner initialization note. Added new API's: LXEHasIntRFID and LXEIsTurboOn.
C	Chapter 1 – Introduction	Added MX3-RFID to "Identify your Windows CE Equipment". Added "Getting Help" section.
C	Chapter 2 – API Calls	Added MX3-RFID to product tables for all API calls. Updated LXEScannerSetWindow to reflect that the message is only sent to the top window. Updated LXEScannerAttachPort, LXEScannerSetSerial and LXEScannerSetPower with note for MX3-RFID. Added new APIs: LXEScannerCtrlCodeOff, LXEScannerCtrlCodeOn.
C	Cover pages	Updated LXE logo and date.
C	Chapter 3 – RFID Specific API Calls	Added new chapter.
D	Chapter 1 – Introduction	Added MX5-IS to "Identify your Windows CE Equipment".
D	Chapter 2 – API Calls	Added MX5-IS to product tables for all API calls.
D	Chapter 3 – RFID Specific API Calls	Revised LXERFIDSetPostamble and LXERFIDSetSeparator. Added support for LXERFIDMaskDefine and LXERFIDMaskRead.
E	Chapter 1 – Introduction	Added MX7 to "Identify your Windows CE Equipment".
E	Chapter 2 – API Calls	Added new APIs: LXEScannerStripLead, LXEScannerStripTrail, LXEScannerPrefix, LXEScannerSuffix. Revised LXEPcmciaDisableSlot and LXEPcmciaEnableSlot. Added MX7 to product tables for all API calls.
E	Appendix A – Character Code Tables	Added new appendix.

Table of Contents

Chapter 1 Introduction

Application Programming Interface (API)

The LXE® CE Application Programmer's Interface (API) is designed to enable application programmers to access the functionality of Microsoft® Windows CE® equipped LXE computer hardware without requiring them to understand the details of the hardware design. This programming guide describes, in a general way, the functions that comprise the API. This guide is for the application programmer working with LXE Microsoft® Windows CE® based computers / devices only.

Note: The LXE® DOS API Programming Guide (E-SW-DOSAPIPG) is directed toward the programmer working with LXE devices with Datalight® ROM-DOS® or Microsoft® MS-DOS® Operating Systems.

API Defined

Also known as Application Programmer's Interface, an API is a specification of the methods an application programmer can use to access services provided by a software module.

APIs are implemented by writing function calls in the program, which provide the linkage to the required subroutine for execution. Thus, an API implies that some program module is available in the computer to perform the operation or that it must be linked into the existing program to perform the tasks.

Identify your Windows CE Equipment

This document details the LXE specific API calls for LXE's family of computers running various versions of the Microsoft® Windows® CE operating system. The supported computers and operating systems are displayed in the chart below.

Note: Although some products appear physically similar (i.e. the MX3-CE and the MX3X), there are significant differences in the API specifications for each computer type. Please use the chart below for help in identifying your equipment. For more information, please refer to the reference guide for the appropriate LXE computer.

Although the computers may appear physically similar, this manual DOES NOT cover devices with operating systems other than Microsoft Windows CE. Examples of equipment not covered include the MX3 with DOS and VX4 with DOS or Microsoft Windows 98, 2000 or XP.

MX3 CE 3.0

- Microsoft Windows CE 3.0
- Intel® StrongARM™ processor
- Color or monochrome display, 640 x 240 pixels

Note: The MX3-CE computer is obsolete.

MX3X CE .NET

- Microsoft Windows CE .NET 4.2
- Intel® XScale® processor
- Color or monochrome display, 640 x 240 pixels

MX3-RFID

- Microsoft Windows CE .NET 4.2
- Intel® XScale® processor
- Color display, 640 x 240 pixels
- Internal RFID module

MX5 PPC

- Microsoft Pocket PC® 2000/2002
- Intel® StrongARM™ processor
- Color display, 320 x 240 pixels
- Identified by “Pocket PC” label below keypad

Note: The MX5 PPC computer is obsolete.

MX5X CE .NET

- Microsoft Windows CE .NET 4.2
- Intel® XScale® processor
- Color display, 320 x 240 pixels
- Identified by “CE .NET” label below keypad

MX5-IS

- Microsoft Windows CE .NET 4.2
- Intel® XScale® processor
- Color display, 320 x 240 pixels
- For use in hazardous locations
- Identified by “I-SAFE” label and blue keypad overlay

MX6 PPC

- Microsoft Pocket PC 2002/2003
- Intel® XScale® processor
- Color display, 320 x 240 pixels

Note: The MX6 PPC computer is obsolete.

MX6 WinMobile

- Microsoft Windows Mobile 2003 for Pocket PC
- Intel® XScale® processor
- Color display, 320 x 240 pixels

MX7

- Microsoft Windows CE .5.0
- Intel® XScale® processor 400MHz
- Color display, 320 x 240 pixels

VX4 CE .NET

- Microsoft Windows CE .NET 4.2
- Intel® Pentium® or Celeron® processor
- Full screen color display, 800 x 600 pixels

VX6

- Microsoft Windows CE .NET 4.2
- Intel® XScale® processor
- Half screen color display, 800 x 320 pixels

VX7

- Microsoft Windows CE .NET 4.2
- Intel® XScale® processor
- Full screen color display, 800 x 600 pixels

Document Contents

This document details only the LXE-specific API calls. It also shows which calls from the standard LXE API are and are not supported on each of the LXE computers.

It is intended as an appendix to the standard Microsoft Windows Pocket PC or Windows CE API documentation.

The APIs documented are included in the file LXEAPI.DLL, which is extracted from API.CAB on the CF card.

These LXE specific API calls and registry settings are details in Chapter 2, “API Calls” and cover the following:

- Battery
- PCMCIA
- Power Management
- Scanner
- Version Control
- Display
- Audio
- Keyboard
- Miscellaneous

A Note About Obsolete Equipment

Some of the equipment discussed in this manual is obsolete. Obsolete/archived manuals are not available on the LXE Manuals CD. They are available for download from the ServicePass website only.

A Note About the MX3-RFID

The MX3-RFID supports many of the APIs detailed in Chapter 2, “API Calls”. In addition, the MX3-RFID supports several RFID specific APIs contained in Chapter 3, “RFID Specific API Calls”.

A Note About the MX5-IS

The MX5-IS supports the same APIs as the MX5X CE .NET.

Caution 	The user is strongly cautioned to review the MX5 Intrinsically Safe User’s Guide for warning and cautions on using the MX5-IS in a hazardous environment.

Getting Help

LXE user guides are now available on CD and they can also be viewed/downloaded from the LXE ServicePass website. Contact your LXE representative to obtain the LXE Manuals CD or access to the LXE ServicePass website. You can also check the LXE ServicePass website for the latest manual releases.

Note: Obsolete/archived manuals are not available on the LXE Manuals CD. They are available for download from the ServicePass website only.

You can get help from LXE by calling the telephone numbers listed on the LXE Manuals CD, in the file titled "Contacting LXE". This information is also available on the LXE website.

Explanations of terms and acronyms used in this guide are located in the file titled "LXE Technical Glossary" on the LXE Manuals CD and on the LXE website.

Related Manuals

Equipment Manuals

• MX3X Reference Guide	MX3X CE .NET and MX3-RFID
• MX5 CE .NET Reference Guide	MX5X CE .NET and MX5-IS
• MX6 Reference Guide	MX6 WinMobile
• MX7 Reference Guide	MX7 CE 5.0
• VX4 CE .NET Reference Guide	VX4 CE .NET
• VX6 Reference Guide	VX6 CE .NET
• VX7 Reference Guide	VX7 CE .NET

Obsolete Equipment Manuals

• MX3-CE Reference Guide	MX3 CE 3.0
• MX5 Reference Guide	MX5 PPC
• MX6 PPC Reference Guide	MX6 PPC

Reference Manuals

- Contacting LXE
- LXE Technical Glossary

Chapter 2 API Calls

General Notes

API support is listed for each LXE computer with a Microsoft Windows CE operating system indicated by the following designations:

- **Yes** – The LXE computer supports this API. If the API definition includes extended values, the extended values ARE NOT supported for this computer.
- **Yes (extended)** – The LXE computer supports this API. If the API definition includes extended values, all values, including the extended values, are supported for this computer.
- **No** – The LXE computer does not support this API. On these computers, the API returns a not supported function result to the calling application.
- **N/A** – The LXE computer does not support this API. The API was created after the release of the computer and the API is not defined for this model of LXE computer. Calling this API on a computer marked as N/A may give unpredictable results including the possibility the LXE computer may lock up and require rebooting.

GetLastError

GetLastError() is an optional Windows CE API call built into many of the Windows CE images loaded on the LXE computer. When an API has indicated an error has occurred, this API can be used to get a detailed error message.

*Note: The VX4 CE .NET **DOES NOT** support the* **GetLastError()** *function.*

If an error does occur, **GetLastError()** should be used immediately after the error to get a meaningful error code. This is necessary because successful APIs may use **SetLastError(0)**, wiping out any previously set error message.

For complete details on using **GetLastError()**, please refer to Microsoft Windows CE documentation. This information is also contained on Microsoft's web site, Microsoft.com.

Battery API

LXEBatteryIsCharging

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID CE.NET	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEBatteryIsCharging(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function returns:

- TRUE if the backup battery is charging, or
- FALSE if the backup battery is not charging.

LXEBatteryIsDischarging

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEBatteryIsDischarging(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function returns:

- TRUE if the backup battery is discharging, or
- FALSE if the backup battery is not discharging.

LXEBatteryChargeBackup

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEBatteryChargeBackup(void);
```

This function generates a system event which causes the battery driver to start charging the backup battery. In normal operation, this API would not be used as the power management driver monitors the battery voltage and initiates a charge as necessary.

This function returns

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEBatteryDischargeBackup

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEBatteryDischargeBackup(void);
```

This function generates a system event which causes the battery driver to start discharging the backup battery. This is used to condition the NiCad battery. When the battery discharges to a sufficiently low value, the hardware detects this and automatically begins the recharge cycle.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEBatteryIsACPower

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEBatteryIsACPower(int *status);
```

This function sets the status variable to:

- 1 if the computer has external power supplied, or
- 0 if the computer is powered only by battery.

This reflects the power state as of the last status loop of the battery driver which occurs every 500 ms.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEBatteryReadVoltage

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEBatteryReadVoltage(int *vmain, int *vback);
```

This function returns the current voltage of both the main battery (**vmain**) and the backup battery (**vback**), in millivolts. This is the value read by the last poll of the battery driver, which occurs every 10 seconds.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEBatteryReadPercent

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEBatteryReadPercent(int *pmain, int *pback);
```

Note: For the MX5, this API is supported for the Main Battery only. The **pback** *return value is meaningless.*

This function returns the current percentage of full charge of both the main battery (**pmain**) and the backup battery (**pback**), in percent notation. This is the value read by the last poll of the battery driver, which occurs every 10 seconds. In addition, the battery driver assumes a linear charge lifetime, which is only approximately accurate. This value reflects the distance between fully charged (100%) and discharged to the critically low voltage trip point (0%).

Note: The 0% end of the percentage range DOES NOT correspond to zero battery power. Instead, it corresponds to the critically low point below which the computer does not have enough power to operate.

This function returns

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEBatteryReadmAhr

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	No
MX3-RFID	No
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	Yes
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEBatteryReadmAhr(int *pmain);
```

This function returns the current charge in mA/hours of the main battery (**`pmain`**). This is the value read by the last poll of the battery driver which occurs every 10 seconds.

This function returns

- 0 on error,
- or 1 on success.

GetLastError() may be used to get a detailed error message.

PCMCIA API

For the PCMCIA APIs, the parameter **`slot`** can have the following values (where X indicates the slot is present on the device):

Constant	Value	MX3-CE	MX3X MX3-RFID	VX6 VX7	MX5	MX5X	MX7
#define SLOT_PCMCIA	0	X	X	X	X	X	
#define SLOT_PCMCIA2	1			X			
#define SLOT_COMPACTFLASH	1	X	X				
#define SLOT_INTATA	2		X	X	X	X	
#define SLOT_SMSC (on-board Ethernet controller)	3		X	X		X	
#define SLOT_SDMMC	4			X			X
#define SLOT_USB_RADIO	5						X

LXEPcmciaHasCard

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes (extended)
MX3-RFID	Yes (extended)
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes (extended)
VX7 CE.NET	Yes (extended)

```
int LXEPcmciaHasCard(int slot, int *status);
```

Note: Please refer to the table at the beginning of this section for a list of valid options for the **`slot`** *parameter.*

This function returns the current status of the specified slot:

- If a card is inserted in the slot, a 1 is returned in status.
- Otherwise, 0 is returned.

This function returns

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPcmciaDisableSlot

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes (extended)
MX3-RFID	Yes (extended)
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	Yes (extended)
VX7 CE.NET	Yes (extended)

```
int LXEPcmciaDisableSlot(int slot);
```

Note: Please refer to the table at the beginning of this section for a list of valid options for the **`slot`** *parameter.*

This function disables the specified slot. It does this by setting an internal flag which causes any card insertions on the slot to be ignored. It then removes power from the slot. Normally, the card would be detected by the PCMCIA driver and power restored to the slot. However, the internal flag prevents this happening, and the card is disabled. The slot can still be powered and accessed by applications or diagnostic software, but it is inhibited from detection by the PCMCIA driver.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPcmciaEnableSlot

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes (extended)
MX3-RFID	Yes (extended)
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	Yes (extended)
VX7 CE.NET	Yes (extended)

```
int LXEPcmciaEnableSlot(int slot);
```

Note: Please refer to the table at the beginning of this section for a list of valid options for the **`slot`** *parameter.*

This function enables the specified slot, generally after being deactivated by the disable function, listed previously. It does this by clearing an internal flag which causes any card insertions on the slot to be ignored. The card is then detected by the PCMCIA driver and power is restored to the slot, restoring the card to normal operation.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPcmciaWriteProtect

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes (extended)
MX3-RFID	Yes (extended)
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPcmciaWriteProtect(int slot, int val);
```

This function causes the card in the specified slot to be write protected. This sets an internal flag which inhibits the PCMCIA driver from allowing writes to the card. It is undefined what happens when a non-SRAM card is write protected; it is safe to assume the card will not function properly.

Note: Please refer to the table at the beginning of this section for a list of valid options for the **`slot`** *parameter.*

The argument **`val`** is:

- 1 to write protect the card, and
- 0 to allow writes to the card.

This function returns

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Power Management API

This section assumes the user is familiar with the power management features and power management states (or power modes) of the particular LXE computer. For more details on power management and primary events, please refer to the reference guide for the LXE computer.

LXEPowerMgrEnable

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrEnable(void);
```

This function generates a system event which causes the power management driver to return to normal operation. The most common usage is following an **LXEPowerMgrAlwaysOn()** command, detailed later in this document.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrToNormal

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrToNormal(void);
```

This function generates a system event which causes the power management driver to return to normal operation. The most common usage is following an **LXEPowerMgrAlwaysOn()** command, detailed later in this document. This is the same call as **LXEPowerMgrEnable()**, listed previously.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrAlwaysOn

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrAlwaysOn(void);
```

This function generates a system event which causes the power management driver to disable operation, leaving all power always on. This disables all display and backlight controls, as well as the OS suspend timer.

For CE .NET equipped computers, this function spins off a thread which generates user events once per second to keep the CE .NET power manager in the On state.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrDisplayNormal

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrDisplayNormal(void);
```

This function generates a system event which causes the power management driver to start handling display on/off timing and events normally. The most common usage is following an **LXEPowerMgrDisplayOn()** command, detailed later in this document.

This function returns

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrDisplayOn

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrDisplayOn(void);
```

This function generates a system event which causes the power management driver to disable monitoring display on/off timing and events, leaving display power always on. This has no effect on other power management subsystems.

For CE .NET equipped computers, this function spins off a thread which generates system events once per second to keep the CE .NET power manager in the User Idle state.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrBacklightNormal

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrBacklightNormal(void);
```

This function generates a system event which causes the power management driver to start handling backlight on/off timing and events normally. The most common usage is following an **LXEPowerMgrBacklightOn()** command, detailed later in this document.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrBacklightOn

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrBacklightOn(void);
```

This function generates a system event which causes the power management driver to disable monitoring backlight on/off timing and events, leaving backlight power always on. This has no effect on other power management subsystems.

For CE .NET equipped computers, this function spins off a thread which generates user events once per second to keep the CE .NET power manager in the On state.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrLoopTime

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	No
MX3-RFID	No
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrLoopTime(int val);
```

This function changes the value of the power manager loop interval (given in milliseconds) and saves it to the registry. The function then generates the event necessary to cause the driver to reload its registry information, so that the new value takes effect immediately.

The argument **`val`** is the loop time in milliseconds. By default it is 5000.

Note: CE .NET equipped computers use the OS-native power management so this function is not implemented for those computers.

This function returns

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrBacklightTimeout

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrBacklightTimeout(int valbatt, int valac);
```

This function changes the values of the backlight power management timeouts (given in seconds) and saves them to the registry. The function then generates the event necessary to cause the driver to reload its registry information, so that the new value takes effect immediately.

The argument **`valbatt`** is the timeout when running under battery power, and the argument **`valac`** is the timeout when running under external AC power.

For CE .NET equipped computers, this function sets timeouts for PM state User Idle, which is entered when there are no user events in the given timeout period.

This function returns

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrDisplayTimeout

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrDisplayTimeout(int valbatt, int valac);
```

This function changes the values of the display power management timeouts (given in seconds) and saves them to the registry. The function then generates the event necessary to cause the driver to reload its registry information, so that the new value takes effect immediately.

The argument **`valbatt`** is the timeout when running under battery power, and the argument **`valac`** is the timeout when running under external AC power.

For CE .NET equipped computers, this function sets timeouts for PM state System Idle, which is entered when there are no system events in the given timeout period.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrPrimaryEvents

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrPrimaryEvents(unsigned long val);
```

This function changes the bitmask of primary events that affect waking from suspend, and saves it to the registry. The function then generates the event necessary to cause the driver to reload its registry information, so that the new value takes effect immediately.

Valid values within the bitmask are defined in **LXEAPI.H**, as follows (blank entries in table are undefined):

Symbol	CE 3.0		CE .NET		CE 5.0	
	Value	Default	Value	Default	Value	Default
PRIMARYEVT_POWER	0x00000001	X	0x00000001	X	0x00000001	X
PRIMARYEVT_KEY	0x00000002		0x00010000	X	0x00010000	X
PRIMARYEVT_TOUCH	0x00000004	X	0x00000020	X	0x00000020	X
PRIMARYEVT_COM1	0x00000100	X	0x00000800	X	0x00000800	X
PRIMARYEVT_COM2	0x01000000					
PRIMARYEVT_COM3	0x00001000	X	0x00001000	X	0x00001000	X
PRIMARYEVT_SCAN	0x02000000		0x00100000		0x00100000	
PRIMARYEVT_USB	0x10000000		0x00400000	X	0x00400000	X
PRIMARYEVT_DOCKED	0x40000000		0x00020000	X	0x00020000	X
PRIMARYEVT_TRIGGER			0x00800000		0x00800000	
PRIMARYEVT_PCMCIA	0x00000018		0x0C000000		0x0C000000	
PRIMARYEVT_PCMCIA0			0x04000000		0x04000000	
PRIMARYEVT_PCMCIA1			0x08000000		0x08000000	
PRIMARYEVT_PCMSTS			0x03000000		0x03000000	
PRIMARYEVT_PCMSTS0			0x01000000		0x01000000	
PRIMARYEVT_PCMSTS1			0x02000000		0x02000000	

Because of hardware limitations, **`PRIMARYEVT_COM2`** and **`PRIMARYEVT_SCAN`** can never be wakeup events.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrBacklightPrimaryEvents

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	No
MX3-RFID	No
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrBacklightPrimaryEvents(unsigned long val);
```

This function changes the bitmask of primary events that affect backlight power on/off, and saves it to the registry. The function then generates the event necessary to cause the driver to reload its registry information, so that the new value takes effect immediately.

The values within the bitmask are given previously, under **LXEPowerMgrPrimaryEvents()**. Default events are:

- PRIMARYEVT_KEY,
- PRIMARYEVT_TOUCH,
- PRIMARYEVT_DOCKED,
- PRIMARYEVT_COM1,
- PRIMARYEVT_COM2,
- PRIMARYEVT_COM3,
- PRIMARYEVT_SCAN, and
- PRIMARYEVT_USB.

For CE .NET equipped computers, these events are not selectable, so this function is not implemented.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEPowerMgrDisplayPrimaryEvents

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	No
MX3-RFID	No
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEPowerMgrDisplayPrimaryEvents(unsigned long val);
```

This function changes the bitmask of primary events that affect display blanking, and saves it to the registry. The function then generates the event necessary to cause the driver to reload its registry information, so that the new value takes effect immediately.

The values within the bitmask are given previously, under **LXEPowerMgrPrimaryEvents()**. Default events are:

- PRIMARYEVT_POWER,
- PRIMARYEVT_KEY,
- PRIMARYEVT_TOUCH,
- PRIMARYEVT_DOCKED,
- PRIMARYEVT_COM1,
- PRIMARYEVT_COM2,
- PRIMARYEVT_COM3,
- PRIMARYEVT_SCAN, and
- PRIMARYEVT_USB.

For CE .NET equipped computers, these events are not selectable, so this function is not implemented.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEForcePowerKeyPrimaryEvent

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEForcePowerKeyPrimaryEvent(void);
```

This function generates a system event which simulates a power keypress to the power management driver. This is primarily of use in diagnostic software, or to force a wakeup of the power management system.

Note: This event has no effect on the computer going into suspend, only the display / backlight controls.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEForceKeyPrimaryEvent

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEForceKeyPrimaryEvent(void);
```

This function generates a system event which simulates a keypress to the power management driver. This is primarily of use in diagnostic software, or to force a wakeup of the power management system.

Note: This event has no effect on the computer going into suspend, only the display / backlight controls.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEForceTouchPrimaryEvent

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEForceTouchPrimaryEvent(void);
```

This function generates a system event which simulates a screen touch to the power management driver. This is primarily of use in diagnostic software, or to force a wakeup of the power management system.

Note: This event has no effect on the computer going into suspend, only the display / backlight controls.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEForceUserPrimaryEvent

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEForceUserPrimaryEvent(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function generates a system event which simulates a user event to the power management driver. This is primarily of use in diagnostic software, or to force a wakeup of the power management system.

Note: This event has no effect on the computer going into suspend, only the transition to power management state User Idle.

For compatibility, this function is also mapped to the following API calls:

```
int LXEForcePowerKeyPrimaryEvent(void);
int LXEForceKeyPrimaryEvent(void);
int LXEForceTouchPrimaryEvent(void);
int LXEForceScanPrimaryEvent(int onoff);
```

The parameter **`onoff`** in some of the calls is unused, but is present for compatibility with CE 3.0 usage. The original usage was:

- 1 activates the event, and
- 0 deactivates it.

This value is not processed in the API.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEForceSystemPrimaryEvent

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEForceSystemPrimaryEvent(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function generates a system event which simulates a system event to the power management driver. This is primarily of use in diagnostic software, or to force a wakeup of the power management system.

Note: This event has no effect on the computer going into suspend, only the transition to power management state System Idle.

For compatibility, this function is also mapped to the following API calls:

```
int LXEForcePCMCIAPrimaryEvent(void);
int LXEForceCOM1PrimaryEvent(int onoff);
int LXEForceCOM2PrimaryEvent(int onoff);
int LXEForceCOM3PrimaryEvent(int onoff);
int LXEForceUSBPrimaryEvent(int onoff);
int LXEForceDockedPrimaryEvent(int onoff);
```

The parameter **`onoff`** in some of the calls is unused, but is present for compatibility with CE 3.0 usage. The original usage was:

- 1 activates the event, and
- 0 deactivates it.

This value is not processed in the API.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Scanner API

The scanner has two different usage methods, depending upon whether or not the scanner wedge is active.

If the scanner wedge is not active, the scanner data can be read through direct file I/O to the COM port configured for use with the scanner.

If the scanner wedge is active, the serial port (or ports) to which it is configured is locked so other applications cannot open it. The scanner buffers all data sent through it, which may be retrieved with standard file open / read / close calls to the scanner wedge, device WDG0:. This data may also be sent as keystroke messages to the frontmost window, allowing use of the scanner with applications which otherwise do not support it. Even if sent as keystrokes, the buffer is cleared when some API (for example, LXEScannerGetData) calls are made,

The buffer can hold up to 2048 characters of data to support 2D scanners.

Please refer to the applicable reference guide for the LXE computer for details on available COM ports and scanners.

For the MX5 PPC, MX5X and MX6: Default settings for the scanner interface are not initialized by the operating system. Therefore, after a cold boot the scanner settings must be initialized using the scanner control panel before calling any of the scanner API's. The scanner settings are retained through a warm boot.

LXEHasIntScanner

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEHasIntScanner(void)
```

This function polls the computer to see if it has an internal scanner. Tethered scanner connections generate a 0 result.

This function returns:

- 1 if the computer has an internal scanner,
- else 0.

LXEHasIntRFID

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEHasIntRFID(void)
```

This function polls the computer to see if it has an internal RFID reader. Currently, only the MX3X offers an RFID reader (and is identified as an MX3-RFID when so equipped). However, the MX7, VX6 and VX7 support this API call for application compatibility.

This function returns:

- 1 if the computer has an internal RFID reader,
- else 0.

LXEScannerEnable

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEScannerEnable(void);
```

Note: After making this call, wait 1.5 seconds before attempting a scan. This allows the COM ports to be configured.

This function generates a system event which causes the scanner driver to return to normal operation, usually after a disable command. When resuming operation, it rereads all registry settings, so any unsaved settings (setup via API calls) revert to their permanent settings.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerDisable

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEScannerDisable(void);
```

This function generates a system event which causes the scanner driver to disable operation. This closes all serial ports being monitored by the scanner driver, and any unread data in the scanner driver buffer is lost. This API can be called to free the serial port for use by another application (such as ActiveSync).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerStart

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEScannerStart(void);
```

This function generates a system event which causes the scanner driver to start a scan operation on the internal scanner.

Note: This API call has no effect on external tethered scanners.

This turns the laser on to execute a read, and continues until:

- a successful read occurs,
- the laser-on timeout is reached, or
- the **LXEScannerStop()** API call (detailed later in this document) is executed.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerGetStatus

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEScannerGetStatus(void)
```

This function returns the status of the scanner:

- SCAN_SCANNING (1) if scan is in progress.
- SCAN_COMPLETE (0) if scan has finished.
- SCAN_NOSCAN (2) if scan was unsuccessful.
- SCAN_ERROR (3) if an error occurred during scanning, or if the scanner is not configured correctly.
- SCAN_BUSY (4) if the scanner is still busy from a previous scan.

LXEScannerGetData

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEScannerGetData(char *buf, int *buflen)
```

Call with a pointer to the output buffer in **buf**, and a pointer to a length value in **buflen**. On call, set **buflen** to the length of the actual buffer. On return, **buflen** has the length of the scanned string. The string is automatically terminated with a zero byte.

The call itself returns:

- 0 on error (unable to open or read failure), or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerStop

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEScannerStop(void);
```

This function generates a system event which causes the scanner driver to terminate a scan operation on the internal scanner.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerKeysOff

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEScannerKeysOff(void);
```

This function generates a system event which causes the scanner driver to stop processing scanned data as keystroke messages.

*Note: This setting is **NOT** saved in the registry. To make the change permanent, it must be changed in the Scanner control panel on the device.*

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerKeysOn

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEScannerKeysOn(void);
```

This function generates a system event which causes the scanner driver to start processing scanned data as keystroke messages sent to the frontmost window.

*Note: This setting is **NOT** saved in the registry. To make the change permanent, it must be changed in the Scanner control panel on the device.*

This function returns

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerKeyStatus

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
Bool LXEScannerKeyStatus(void);
```

This function returns:

- True if the scanner wedge is processing data as keystroke messages, and
- False if the data is being buffered.

LXEScannerPowerOff

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEScannerPowerOff(void);
```

This function generates a system event which causes the scanner driver to power down all active scanner devices.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerPowerOn

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEScannerPowerOn(void);
```

Note: After making this call, wait one second before attempting a scan. This is necessary to allow scanner power to stabilize before scanning.

This function generates a system event which causes the scanner driver to apply power to all active scanner devices.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerForcePower

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
// values for scanner port
#define SCANPORT_COM1       1
#define SCANPORT_COM3       3
#define SCANPORT_INTERNAL   4

int LXEScannerForcePower(int port, int value);
```

This function forces the serial port to supply power to a scanner, regardless of the registry setting, or whether this is an active serial port or not. This allows an application to activate a scanner independently from the scanner driver. **`value`** is set to 1 to supply output power, or set to 0 to turn power off:

- Using this call on ports COM1 and COM3 applies or removes 5 volt power to pin 9.
- Using this call on the internal scanner powers it off and on.

Using this call on any other port returns an error.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerAttachPort

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes (see note)
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
// values for wedge attachment port
#define WEDGEPORT1          1
#define WEDGEPORT2          2
#define WEDGEPORT3          3     /*MX7 only*/
// values for scanner port
#define SCANPORT_DISABLED   0
#define SCANPORT_COM1       1
#define SCANPORT_COM2       2
#define SCANPORT_COM3       3
#define SCANPORT_INTERNAL   4

int LXEScannerAttachPort(int port, int val);
```

This function changes the wedge port (**`port`**) attached to the scanner driver as specified (**`val`**) and saves it to the registry. The function then generates the event necessary to cause the driver to reload its registry information so that the new value takes effect immediately.

There are two possible ports to attach to the scanner driver, indicated by **`WEDGEPORT1`** and **`WEDGEPORT2`**. These can be attached to any of the indicated ports. By default the ports are disabled, unless there is an internal scanner. In that case **`WEDGEPORT1`** is attached to the internal scanner.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Note: On an MX3-RFID terminal, a call with WEDGEPORT2 or SCANPORT_COM1 performs no action. An error message may be returned.

LXEScannerSetKey

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
// values for scan keys
#define SCANKEYLEFT         1
#define SCANKEYRIGHT        2
#define SCANKEYTRIG         3     /* handle trigger */
// values for key action
#define SCANKEY_DISABLED    0
#define SCANKEY_SCAN        1
#define SCANKEY_ENTER       2
#define SCANKEY_TAB         3
#define SCANKEY_VIRTKEY     4
#define SCANKEY_FLD_EXIT    5     (only in MX3X, MX3RFID)
#define SCANKEY_RFID        6     (only in MX3X, MX3RFID)
#define SCANKEY_ESC         7     (only in MX5 PPC, MX5X, MX5-
IS)

int LXEScannerSetKey(int port, int val);
```

This function changes the action of the scan key specified (**`port`**) to the action specified (**`val`**) and saves it to the registry. The function then generates the event necessary to cause the driver to reload its registry information, so that the new value takes effect immediately.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerSetSerial

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes (see note)
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
#define SCANPORT_DISABLED   0
#define SCANPORT_COM1       1
#define SCANPORT_COM2       2
#define SCANPORT_COM3       3
#define SCANPORT_INTERNAL   4

int LXEScannerSetSerial(int port, int baud, int data,
int parity, int stop);
```

This function changes the configuration of the scanner serial port on the indicated port and saves it to the registry. The function then generates the event necessary to cause the driver to reload its registry information, so that the new value takes effect immediately.

Note: This setting has no effect on the serial ports unless they are handled by the scanner driver. Regular serial ports are configured using the standard Win32 API calls.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Note: On an MX3-RFID terminal, a call with SCANPORT_COM1 performs no action. An error message may be returned.

LXEScannerSetPower

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes (see note)
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
#define SCANPORT_DISABLED   0
#define SCANPORT_COM1       1
#define SCANPORT_COM2       2
#define SCANPORT_COM3       3
#define SCANPORT_INTERNAL   4

int LXEScannerSetPower(int port, int val);
```

This function changes the value of the scanner serial port power pin on the indicated serial port and saves it to the registry. The valid options for **`val`** are:

- 0 Power pin is set to ring indicator, or
- 1 Power pin is set to 5 volts.

This function is only valid for COM1 and COM3 serial ports, and returns an error for all others. The function then generates the event necessary to cause the driver to reload its registry information, so that the new value takes effect immediately.

This function returns

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Note: On an MX3-RFID terminal, a call with SCANPORT_COM1 performs no action. An error message may be returned.

LXEScannerSetWindow

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
#define WM_LXE_SCANDONE     0x600
#define WM_LXE_SCANERR      0x601
#define WM_LXE_SCANFULL     0x602

int LXEScannerSetWindow(HWND window);
```

This function attaches an application window to the scanner driver. When a scan occurs, the scanner driver then sends a Windows message to the application window, indicating that a scan has occurred.

The message **`WM_LXE_SCANDONE`** or **`WM_LXE_SCANFULL`** indicates the scan is complete, and the data may be read using standard Win32 file/device API calls, opening the scanner device. The message **`WM_LXE_SCANERR`** indicates a configuration error which prevents the scanner from processing data correctly. There is no message generated on a failed scan. Other messages may be added to this list in the future as new capabilities emerge.

This function may be called up to 8 times. All 8 window handles are sent the message when the events occur. The window handle is tested for validity before sending the message; invalid handles are removed from the list. Thus, the application is not required to clean up the connection.

This function returns

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Note: **`WM_LXE_SCANFULL`** *is an updated replacement for* **`WM_LXE_SCANDONE`**. **`WM_LXE_SCANDONE`** *is retained in this API for backwards compatibility.*

Note: This function is not required under the CE .NET operating system, since Windows messages can be broadcast across the system. When broadcast, the message is only received by the root (topmost) window.

LXEScannerCtrlCodeOff

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	No
MX3-RFID	No
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEScannerCtrlCodeOff(void);
```

This function disables processing of ASCII values less than 0x20 (control codes) when scanner data is processed as keystroke messages. The control code data is discarded

*Note: This setting is **NOT** saved in the registry. To permanently disable processing of control codes, use the Scanner control panel.*

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerCtrlCodeOn

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	No
MX3-RFID	No
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEScannerCtrlCodeOff(void);
```

This function enables processing of ASCII values less than 0x20 (control codes) when scanner data is processed as keystroke messages. The control code data is translated from the ASCII value to its equivalent control key sequence and is sent as keystroke messages just like the printable data.

*Note: This setting is **NOT** saved in the registry. To permanently enable processing of control codes, use the Scanner control panel.*

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerStripLead

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	No
MX3-RFID	No
MX5 PPC	No
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	Yes
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEScannerStripLead(BOOL bEnable, int Count);
```

This function enables or disables stripping of characters from the beginning of a barcode. **`Count`** specifies the number of character to strip. The maximum number of characters that can be stripped is 99. If the number of characters to be stripped is greater than the number of characters in the barcode, a good beep is sounded but all barcode data is discarded.

Note: If this feature is used with the AddPrefix or AddSuffix features, the leading characters are stripped before the prefix or suffix is added.

*Note: This setting is **NOT** saved in the registry. To permanently enable character stripping, use the Scanner control panel.*

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerStripTrail

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	No
MX3-RFID	No
MX5 PPC	No
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	Yes
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEScannerStripTrail (BOOL bEnable, int Count);
```

This function enables or disables stripping of characters from the end of a barcode. **`Count`** specifies the number of character to strip. The maximum number of characters that can be stripped is 99. If the number of characters to be stripped is greater than the number of characters in the barcode, a good beep is sounded but all barcode data is discarded.

Note: If this feature is used with the AddPrefix or AddSuffix features, the ending characters are stripped before the prefix or suffix is added.

*Note: This setting is **NOT** saved in the registry. To permanently enable character stripping, use the Scanner control panel.*

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerPrefix

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	No
MX3-RFID	No
MX5 PPC	No
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	Yes
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEScannerPrefix(int bEnable, int Mode, char* Prefix, int
PrefixSize, char *PrefixText);
```

This function enables or disables whether a prefix is added to the beginning of a barcode. **`Mode`** specifies whether the prefix is intended for Key (1) or Block (0) mode. **`Prefix`** is a pointer to a data array of ASCII values or VK_codes (depending on the Mode) that is prepended to the barcode data. Up to 19 characters can be specified for the prefix. In Key Mode, the data array must contain two bytes for each VK_code. The first byte is the VK_code and the second byte is the shift state of the VK_code (0 for "not shifted" and 1 for "shifted").

Note: This setting is saved in the registry.

The characters can be text or control characters, like tab or carriage return. The characters can be entered by typing from the keypad, entering their hex equivalent, or entering in '^' delimited (hat encoded, 8-bit code table) notation. (See Appendix B, "Character Code Tables", for a listing of hex equivalent and hat encoded characters.)

There are two modes in which the prefix is processed, key message or block mode.

- In key message mode (**`Mode = 1`**), all keys on the keypad can be entered into the configuration. In this mode, the prefix, barcode, and suffix (if specified, see LXEScannerSuffix) are sent as keystrokes to the application with the focus.
- In block mode (**`Mode = 0`**) ASCII characters (0x0 – 0x7F), plus Backspace, Tab, Delete, Return, and Escape (open issue) can be specified. In this mode, the prefix/suffix data is added to the beginning and end of the buffered barcode data that can then be read by an application from the WDG: device.

Control codes specified in the prefix are translated according to the "Translate Control Codes" setting. This may be set via:

- the Scanner Control Panel (please refer to the appropriate computer reference guide)
- an API (please refer to LXEScannerCtrlCodeOff / LXEScannerCtrlCodeOn, earlier in this section).

Note: If this feature is used with the StripLead or StripTrail features, the characters are stripped before the prefix is added.

Note: This setting is saved in the registry.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEScannerSuffix

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	No
MX3-RFID	No
MX5 PPC	No
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	Yes
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEScannerSuffix(int bEnable, int Mode, char* Suffix, int
SuffixSize, char *SuffixText);
```

This function enables or disables whether a suffix is added to the end of a barcode. **`Mode`** specifies whether the prefix is intended for Key (1) or Block (0) mode. **`Suffix`** is a pointer to a data array of ASCII values or VK_codes (depending on the Mode) that is appended to the barcode data. Up to 19 characters can be specified for the suffix. In Key Mode, the data array must contain two bytes for each VK_code. The first byte is the VK_code and the second byte is the shift state of the VK_code (0 for "not shifted" and 1 for "shifted").

Note: This setting is saved in the registry.

The characters can be text or control characters, like tab or carriage return. The characters can be entered by typing from the keypad, entering their hex equivalent, or entering in '^' delimited (hat encoded, 8-bit code table) notation. (See Appendix B, "Character Code Tables", for a listing of hex equivalent and hat encoded characters.)

There are two modes in which the suffix is processed, key message or block mode.

- In key message mode (**`Mode = 1`**), all keys on the keypad can be entered into the configuration. In this mode, the prefix (if specified, see LXEScannerPrefix), barcode, and suffix are sent as keystrokes to the application with the focus.
- In block mode (**`Mode = 0`**) ASCII characters (0x0 – 0x7F), plus Backspace, Tab, Delete, Return, and Escape (open issue) can be specified. In this mode, the prefix/suffix data is added to the beginning and end of the buffered barcode data that can then be read by an application from the WDG: device.

Control codes specified in the suffix are translated according to the "Translate Control Codes" setting. This may be set via:

- the Scanner Control Panel (please refer to the appropriate computer reference guide)
- an API (please refer to LXEScannerCtrlCodeOff / LXEScannerCtrlCodeOn, earlier in this section).

Note: If this feature is used with the StripLead or StripTrail features, the characters are stripped before the suffix is added.

Note: This setting is saved in the registry.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Version Control API

LXEVersionOS

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEVersionOS(char *str);
```

This function returns the version number and patch information of the Windows CE OS and formats the number into a string with the following format:

```
Windows CE n.n build nnn patched through mm/dd/yy
```

The space for this string may be allocated using the global value from **LXEAPI.h**, as follows:

```
char VersionString[VERSION_STRSIZE]
```

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEVersionOAL

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEVersionOAL(char *str);
```

This function returns the version number of the OAL code and OS image and formats the number into a string with the following format:

```
MX3-CE vn.n.n ARM SA1100 04/01/2002 12:35
```

The date and time specified are the actual date and time that the OAL was compiled. The space for this string may be allocated using the global value from **LXEAPI.h**, as follows:

```
char VersionString[VERSION_STRSIZE]
```

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEVersionBoot

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEVersionBoot(char *str);
```

This function returns the version number of the bootloader code and formats the number into a string with the following format:

```
MX3-CE vn.n.n ARM SA1100 04/01/2002 12:35
```

The date and time specified are the actual date and time that the bootloader was compiled. The space for this string may be allocated using the global value from **LXEAPI.h**, as follows:

```
char VersionString[VERSION_STRSIZE]
```

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEVersionFPGA

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEVersionFPGA(char *str);
```

This function returns the version number of the FPGA code and formats the number into a string with the flowing format:

```
FPGA rev nn
```

The space for this string may be allocated using the global value from **LXEAPI.h**, as follows:

```
char VersionString[VERSION_STRSIZE]
```

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEVersionAPI

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEVersionAPI(char *str);
```

This function returns the version number of the API library and formats the number into a string with the following format:

```
MX3-CE xxxxxxxxxx ARM SA1100 04/01/2002 12:35
```

The date and time specified are the actual date and time that the API was compiled. The space for this string may be allocated using the global value from **LXEAPI.h**, as follows:

```
char VersionString[VERSION_STRSIZE]
```

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEInfoCopyright

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEInfoCopyright(WCHAR *str, int len);
```

This function returns the image build copyright as a Unicode string.

The space for this string may be allocated using the global value from **LXEAPI.h**, as follows:

```
WCHAR CopyrightString[COPYRIGHT_STRSIZE]
```

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEInfoGetCodecInfo

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEInfoGetCodecInfo(char *str)
```

This function returns information about the hardware codec, formatted into a string as follows:

```
1004 = UCB1200
```

Where:

- 1004 is the hardware code returned, and
- UCB1200 is the text interpretation of this code.

The space for this string may be allocated using the global values from **LXEAPI.h**, as follows:

```
char InfoString[VERSION_STRSIZE];
```

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEInfoGetCPUInfo

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEInfoGetCPUInfo(char *str)
```

This function returns information about the hardware codec, formatted into a string as follows:

```
2577 [8] = SA1110 B4 206 MHz (jumper=206)
```

Where:

- the first number is the type code of the CPU (2577=SA1110),
- the second number in brackets is the revision of the CPU (8 = rev B4 of StrongARM CPU),
- the speed is read from the CPU registers,
- and the jumper number represents the position of the onboard speed jumper.

The space for this string may be allocated using the global values from **LXEAPI.h**, as follows:

```
char InfoString[VERSION_STRSIZE];
```

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEInfoROMID

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEInfoROMID(char *str);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function returns information about the Flash memory in a string as follows:

```
<mfg> <device ID> <boot (top or bottom)>
```

This function returns 0 on error, or 1 on success. **GetLastError()** may be used to get a detailed error message.

LXEInfoRAMID

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	No
MX3-RFID	No
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This API is not supported as none of the LXE computers above currently use SIMM DRAMs.

LXEInfoGetROMInfo

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
typedef struct {
    unsigned long BlockSize;
    unsigned long BlockCount;
    unsigned long BusWidth;
    unsigned long BaseAddress;
    unsigned long TotalSize;
    unsigned long BootloaderBase;
    unsigned long BootloaderSize;
    unsigned long OsBase;
    unsigned long OsSize;
    unsigned long RegistryBase;
    unsigned long RegistrySize;
    unsigned long FilesysBase;
    unsigned long FileSysSize;
} ROMINFO;

int LXEInfoGetROMInfo(ROMINFO *rtn);
```

This function returns information about the ROM (flash memory) in the system in the data block specified. Because this information is established at image compile time, these values do not change.

`BlockSize` is the minimum erasable block size in bytes. This is a function of the flash chip hardware. This is the value of the chip erase block times the number of chips in the array.

`BlockCount` is the total number of **`BlockSize`** blocks in all of the flash devices.

`BusWidth` returns the flash bus width, in bits. This is normally 32.

`BaseAddress` is the memory address in mapped kernel space where the flash memory starts.

`TotalSize` is the total number of bytes of flash ROM in the computer. This is **`BlockSize`** times **`BlockCount`**.

`BootloaderBase` and **`BootloaderSize`** return the base address and allocated size for the bootloader partition of flash.

OsBase and **OsSize** return the base address and allocated size for the operating system image partition of flash. These values are empty for XScale boot flash systems.

RegistryBase and **RegistrySize** return the base address and allocated size for the Persistent Memory registry configuration backup partition. These values are empty for XScale boot flash systems.

FilesysBase and **FileSysSize** return the base address and allocated size for the Persistent Memory flash file system partition. These values are empty for XScale boot flash systems.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEInfoGetRAMInfo

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
typedef struct {
    unsigned long BaseAddress;
    unsigned long TotalSize;
    unsigned long StorageSize;
    unsigned long StorageFree;
    unsigned long ProgramSize;
    unsigned long ProgramFree;
    unsigned long PagefileSize;
    unsigned long PagefileFree;
    unsigned long VirtualSize;
    unsigned long VirtualFree;
} RAMINFO;

int LXEInfoGetRAMInfo(RAMINFO *rtn);
```

This function returns information about the RAM in the system in the data block specified. Because this information is read using standard Win32 APIs, it represents the state of the system at the moment the API call is made.

Note: This represents the total of all RAM in the system, when there is more than one RAM device or type of RAM.

`BaseAddress` is the memory address in mapped kernel space where the RAM starts. This does not change.

`TotalSize` is the total detected RAM in the computer. This is the sum of **`StorageSize`** and **`ProgramSize`**.

`StorageSize` and **`StorageFree`** return the total size and bytes free values for storage memory. This is set as a proportion of physical RAM by the System control panel.

`ProgramSize` and **`ProgramFree`** return the total size and bytes free values for program memory. This is set as a proportion of physical RAM by the System control panel.

`PagefileSize` and **`PagefileFree`** return the total size and bytes free values for memory. In a computer equipped with Windows CE, these should be 0.

`VirtualSize` and **`VirtualFree`** return the total size and bytes free values for virtual memory.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEInfoGetUUID

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEInfoGetUUID(char *str);
```

This function reads the UUID (Universal Unit Identifier) out of global memory and formats the number into a string of a generic format as follows:

```
XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

The space for this string may be allocated using the global value from **LXEAPI.h**, as follows:

```
Char UUIDString[GUID_STRSIZE]
```

This function returns 0 on error, or 1 on success. **GetLastError()** may be used to get a detailed error message.

Display API

LXEHasColorLCD

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	Yes (see note)
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
#define DISPLAY_MONO              0
#define DISPLAY_COLOR_TRANSFLECT  1
#define DISPLAY_COLOR_ACTIVE      2     /* TFT display */
#define DISPLAY_COLOR_TRANSMISS   3
#define DISPLAY_COLOR_ACTIVE      4 (See Note)
#define DISPLAY_INVALID           -1

int LXEHasColorLCD(void);
```

Note: For computers equipped with an active color (TFT) display, this API returns:
2 for all computers except VX4 CE .NET
4 for VX4 CE .NET.

This function allows an application to test for a color display. This usually is not necessary in most applications, but may provide enhancements on a color terminal. The possible return codes are listed above. Because monochrome display is zero, the test can be simple to avoid the (probably unnecessary) need to parse for color display type:

```
if (!LXEHasColorLCD())
    // process as monochrome
else
    // process as color
```

LXEShowTaskbar

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	No
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
void LXEShowTaskbar(int flag)
```

This API provides a convenient function to show or hide the taskbar, similar to PocketPC.

Flag is **SW_SHOW** or **SW_HIDE**, like standard Win32 API calls.

SW_HIDE must occur before main application window is created (right after **RegisterClass()**, or similar)

SW_SHOW should happen right before **PostQuitMessage()**.

Note: **RegisterClass()** *and* **PostQuitMessage()** *are standard Windows CE API calls. For more information on these API calls, please refer to Microsoft documentation, such as that available on Microsoft.com.*

LXEGetContrast

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEGetContrast(int *val)
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This API provides a convenient function to wrap the **ExtEscape()** call to the display driver. The current contrast setting is returned in **`val`**, which is a value from 0 (lowest) to 31 (highest).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Note: **ExtEscape()** *is a standard Windows CE API call. For more information on this API call, please refer to Microsoft documentation, such as that available on Microsoft.com.*

LXESetContrast

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	No
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXESetContrast(int val)
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This API provides a convenient function to wrap the **ExtEscape()** call to the display driver. Contrast is set to the value in **`val`**, which is a value from 0 (lowest) to 31 (highest).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Note: **ExtEscape()** *is a standard Windows CE API call. For more information on this API call, please refer to Microsoft documentation, such as that available on Microsoft.com.*

LXEGetBrightness

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXEGetBrightness(int *val)
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This API provides a convenient function to wrap proprietary extensions to the **ExtEscape()** call to the display driver. Current backlight brightness setting is returned in **`val`**, which is a percentage from 0 (off) to 100 (full on).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Note: **ExtEscape()** *is a standard Windows CE API call. For more information on this API call, please refer to Microsoft documentation, such as that available on Microsoft.com.*

LXESetBrightness

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	No
VX7 CE.NET	No

```
int LXESetBrightness(int val)
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This API provides a convenient function to wrap proprietary extensions to the **ExtEscape()** call to the display driver. Backlight brightness is set to the value in **`val`**, which is a percentage from 0 (off) to 100 (full on).

Note: Values less than 20% have little if any visible effect.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Note: **ExtEscape()** *is a standard Windows CE API call. For more information on this API call, please refer to Microsoft documentation, such as that available on Microsoft.com.*

Audio APIs

LXEAudioGetGain

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioGetGain(int *val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function returns the current record gain from the audio driver in the variable **`val`**, which can range from 0 (0 db gain) to 225 (22.5 db gain) in 1.5 db increments (limited by the codec resolution).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSetGain

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSetGain(int val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function sets the record gain from the audio driver to the value passed in the variable **`val`**, which can range from 0 (0 db gain) to 225 (22.5 db gain) in 1.5 db increments (limited by the codec resolution).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioLoadGain

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioLoadGain(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function loads a record gain value from the registry into internal variables in the audio driver.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSaveGain

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSaveGain(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function saves the current record gain from the internal variables in the audio driver to the registry.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioGetBoost

Product	API supported
MX3 CE 3.0	N/A
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	N/A
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioGetBoost(int *val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function returns the current setting of the +20 dB record gain boost from the audio driver in the variable **`val`**. The value is:

- 1 if the +20 dB boost is enabled,
- 0 if disabled.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSetBoost

Product	API supported
MX3 CE 3.0	N/A
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	N/A
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSetBoost(int val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function sets the +20 dB record gain boost to the value passed in the variable **`val`**. The values are:

- 1 to enable boost or
- 0 to disable boost.

This function returns

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioLoadBoost

Product	API supported
MX3 CE 3.0	N/A
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	N/A
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioLoadBoost(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function loads the boost setting from the registry into internal variables in the audio driver.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSaveBoost

Product	API supported
MX3 CE 3.0	N/A
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	N/A
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSaveBoost(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function saves the current boost setting from the internal variables in the audio driver to the registry.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioGetVolume

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioGetVolume(int *val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function returns the current Windows volume from the audio driver in the variable **`val`**, which can range from 0 (lowest volume) to 15 (highest volume).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSetVolume

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSetVolume(int val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function sets the Windows volume from the audio driver to the value passed in the variable **`val`**, which can range from 0 (lowest volume) to 15 (highest volume).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioLoadVolume

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioLoadVolume(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function loads the Windows volume value from the registry into internal variables in the audio driver.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSaveVolume

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSaveVolume(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function saves the Windows volume from the internal variables in the audio driver to the registry.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioGetMasterVolume

Product	API supported
MX3 CE 3.0	N/A
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	N/A
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioGetMasterVolume(int *val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function returns the current master output volume from the audio driver in the variable **`val`**, which can range from 0 (0 dB attenuation) to 31 (-46.5 dB attenuation).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSetMasterVolume

Product	API supported
MX3 CE 3.0	N/A
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	N/A
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSetMasterVolume(int val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function sets the master output volume from the audio driver to the value passed in the variable **`val`**, which can range from 0 (0 dB attenuation) to 31 (-46.5 dB attenuation).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioLoadMasterVolume

Product	API supported
MX3 CE 3.0	N/A
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	N/A
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioLoadMasterVolume(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function loads the master output volume value from the registry into internal variables in the audio driver.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSaveMasterVolume

Product	API supported
MX3 CE 3.0	N/A
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	N/A
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSaveMasterVolume(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function saves the current master output volume from the internal variables in the audio driver to the registry.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioGetRecordIn

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioGetRecordIn(int *val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function returns the current record input selected from the audio driver in the variable **`val`**. The following values are valid:

```
#define RECORDIN_NONE 0
#define RECORDIN_MIC1 1
```

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSetRecordIn

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSetRecordIn(int val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function sets the record input select from the audio driver to the value passed in the variable **`val`**. The following values are valid:

```
#define RECORDIN_NONE 0
#define RECORDIN_MIC1 1
```

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioLoadRecordIn

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioLoadRecordIn(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function loads a record input select value from the registry into internal variables in the audio driver.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSaveRecordIn

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSaveRecordIn(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function saves the current record input select value from the internal variables in the audio driver to the registry.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioGetSidetone

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioGetSidetone(int *val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function returns the current record sidetone from the audio driver in the variable **`val`**, which can range from 120 (+12.0 db) to –345 (-34.5 db), in 1.5 db increments (limited by the codec resolution).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSetSidetone

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSetSidetone(int val);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function sets the record sidetone from the audio driver to the value passed in the variable **`val`**, which can range from 120 (+12.0 db) to –345 (-34.5 db), in 1.5 db increments (limited by the codec resolution).

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioLoadSidetone

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioLoadSidetone(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function loads a record sidetone value from the registry into internal variables in the audio driver.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

LXEAudioSaveSidetone

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEAudioSaveSidetone(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function saves the current record sidetone from the internal variables in the audio driver to the registry.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Keyboard API

LXEKeyboardSetLayout

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEKeyboardSetLayout(int id);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function sets the active keyboard layout to the value in the variable **`id`**. This layout must exist in the registry, and have been created with the LXE KeyComp utility before using this function.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Note: API is a wrapper for the standard CE .NET API call ActivateKeyboardLayout().

LXEKeyboardGetLayout

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEKeyboardGetLayout(int *id);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function returns the ID of the current keyboard layout in the variable **`id`**.

This function returns:

- 0 on error, or
- 1 on success.

GetLastError() may be used to get a detailed error message.

Note: API is a wrapper for the standard CE .NET API call GetKeyboardLayout().

Miscellaneous API

LXEBoot

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	Yes
MX6 WinMobile	Yes
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEBoot(void);
```

This function performs a warm boot, preserving the contents of RAM and all configuration settings. However, any network sessions are lost, and any data in running applications not specifically flushed to storage may be lost.

Note: It is up to the application making this call to validate the API's usage. For example, the application may be designed to provide a warning, ask for a password, etc. Calling this function arbitrarily can result in loss of data.

This function returns:

- 0 on error, or
- 1 on success.

LXEBootClear

Product	API supported
MX3 CE 3.0	Yes
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	Yes
MX5X CE.NET	Yes
MX5-IS CE.NET	Yes
MX6 PPC	No
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEBootClear(void);
```

This function performs a cold boot, erasing the contents of RAM and returning all registry configuration settings to factory defaults. This routine does not return to the calling application as the calling application is terminated on the cold boot.

Note: Currently, in implementations with Intel Persistent Storage Manager, the last saved registry values are reloaded on startup, not the factory defaults.

Note: It is up to the application making this call to validate the API's usage. For example, the application may be designed to provide a warning, ask for a password, etc. Calling this function arbitrarily can result in loss of data.

This function returns:

- 0 on error, or
- 1 on success.

LXEIsTurboOn

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXEIsTurboOn(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function can be used to determine if the CPU is currently set to turbo (double speed) mode.

Note: Power drain increases substantially in turbo mode.

This function returns:

- 1 if the CPU is currently set to double speed
- 0 if the CPU is not set to double speed.

LXETurboOn

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXETurboOn(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function sets the CPU to double speed for processor intensive applications. It is implemented only on the XScale CPU.

Note: Power drain increases substantially in turbo mode.

This function returns:

- 1 for success, or
- 0 if not implemented.

LXETurboOff

Product	API supported
MX3 CE 3.0	No
MX3X CE.NET	Yes
MX3-RFID	Yes
MX5 PPC	N/A
MX5X CE.NET	No
MX5-IS CE.NET	No
MX6 PPC	N/A
MX6 WinMobile	No
MX7 CE 5.0	Yes
VX4 CE.NET	No
VX6 CE.NET	Yes
VX7 CE.NET	Yes

```
int LXETurboOff(void);
```

Note: Calling this API on a computer marked as N/A may give unpredictable results including the possibility the computer may lock up and require rebooting.

This function sets the CPU to half its maximum speed to save power when CPU intensive applications are not running. It is implemented only on the XScale CPU.

This function returns:

- 1 for success, or
- 0 if not implemented.

Chapter 3 RFID Specific API Calls

Introduction

The API calls in this chapter are for the LXE MX3-RFID computer with a Microsoft Windows CE .NET operating system and an internal RFID reader.

API support is indicated by the following designations:

- **Yes** – The LXE computer supports this API.
- **No** – The LXE computer does not support this API. On these computers, the API returns a not supported function result to the calling application.

For status codes returned by the API calls, please see "Status Field Codes" later in this chapter.

Please refer to the RFID information in the MX3X Reference Guide for more information on RFID functions and parameters.

RFID API

LXERFIDSystemNoChg

Product	API supported
MX3-RFID	Yes

```
int LXERFIDSystemNoChg( BYTE *status, INT *ldrSwVer,
INT *appSwVer, BYTE *mode );
```

This API causes no change (NO_CHG function) to the RFID computer. It is used to retrieve the firmware version.

Input

N/A

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

INT **ldrSwVer** – The current loader software version mark
Range: 0x0000 - 0xFFFF

INT **appSwVer** – The current application software version mark
Range: 0x0000 - 0xFFFF

BYTE **mode** – The current mode setting
Range: Function currently returns 0

LXERFIDTag0Kill

Product	API supported
MX3-RFID	No

This API corresponds to the TAG_0_KILL function. This API is not currently supported.

LXERFIDTag0Set

Product	API supported
MX3-RFID	Yes

```
int LXERFIDTag0Set( BYTE *status, BYTE rfLevel,
BYTE modDepth );
```

Function

This API performs the TAG_0_SET function. For Class 0 tags, this function sets the read parameters:

- RF power level and
- Modulation depth

Input

BYTE **rfLevel –** RF power level to be used for reading Class 0 tags

Range: 0x00 – 0x10 where:
0x00 = NO_CHG – The current setting is to remain unchanged
0x01 – 0x10 = +15dbm to +30dbm in 16 steps of +1db
Default: 0x10 (+30dbm)

BYTE **modDepth –** Modulation depth to be used for reading Class 0 tags

Range: 0x00 – 0x20 where:
0x00 = NO_CHG – The current setting is to remain unchanged
0x01 – 0x20 = 20% to 95% in steps of ~2.42%
Default: 0x20 (95%)

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

LXERFIDTag0Read

Product	API supported
MX3-RFID	Yes

```
int LXERFIDTag0Read( BYTE *status, BYTE singulationMode, BYTE
filterBitCount, BYTE *filterBits, INT *tagCount, BYTE **tagID,
INT *tagIDLen );
```

This API performs the TAG_0_READ function. Class 0 tag IDs are read using the parameters set by TAG_0_SET

Input

BYTE **singulationMode –** Identifiers for singulation algorithm

Range: 0x00 – 0x02 where:
0x00 = ID0
0x01 = ID1
0x02 = ID2
0x99 = NO_CHG – The current setting is to remain unchanged

Default: 0x02 (ID2)

Note: ID2 is the only option that is implemented so far in the reader.

BYTE **filterBitCount –** Number of filter bits in the filterBits array

Range: 0x00 – 0x60 = treat as ID2 data
0x80 – 0xe0 = treat as negative value of filter bit count and treat FILTER_BITS as ID1 data.

Default: 0x00

BYTE **filterBits**[16] – Filter for the Reader to use when filtering the tags.

Default: ‘’

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

INT **tagCount** – Number of tags read

BYTE ***tagID** – A string containing a ***Preamble***, a sequence of 1 or more tag IDs separated by a ***Separator***, and ending with a ***Postamble***.

INT **tagIDLen** – Length of **tagID.**

Note: API will allocate space for **tagID**. *The caller must free the memory by calling:* ***free(*****tagID)**.

LXERFIDTag1Kill

Product	API supported
MX3-RFID	No

This API corresponds to the TAG_1_KILL function. . This API is not currently supported.

LXERFIDTag1Set

Product	API supported
MX3-RFID	Yes

```
int LXERFIDTag1Set( BYTE *status, BYTE rfLevel,
BYTE modDepth );
```

This API performs the TAG_1_SET function. For Class 1 tags, this function sets the read parameters:

- RF power level and
- Modulation depth

Description

Set read parameters (RF power level and Modulation depth)

Input

BYTE **rfLevel** – RF power level to be used for reading Class 1 tags
Range: 0x00 – 0x10 where:
0x00 = NO_CHG – The current setting is to remain unchanged
0x01 – 0x10 = +15dbm - +30dbm in 16 steps of +1db
Default: 0x10 (+30dbm)

BYTE **modDepth –** Modulation depth to be used for reading Class 1 tags
Range: 0x00 – 0x20 where:
0x00 = NO_CHG – The current setting is to remain unchanged
0x01 – 0x20 = 20% - 95% in steps of ~2.42%
Default: 0x20 (95%)

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

LXERFIDTag1Read

Product	API supported
MX3-RFID	Yes

```
int LXERFIDTag1Read( BYTE *status, BYTE filterBitCount, BYTE
*filterBits, INT *tagCount, BYTE **tagID, INT *tagIDLen );
```

This API performs the TAG_1_READ function. Class 1 tag IDs are read using the parameters set by TAG_1_SET

Input

BYTE **filterBitCount** – Number of filter bits in the filterBits array
Range: 0x00 – 0x60
Default: 0x00

BYTE **filterBits**[16] **–** Filter for the Reader to use when filtering the tags.
Default: ‘’

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

INT **tagCount** – Number of tags read

BYTE ***tagID –*** A string containing a ***Preamble***, a sequence of 1 or more tag IDs separated by a ***Separator***, and ending with a ***Postamble***.

INT **tagIDLen** – Length of **tagID.**

Note: API will allocate space for **tagID**. *The caller must free the memory by calling:* ***free(*****tagID*****)***.

LXERFIDTag1ProgramID

Product	API supported
MX3-RFID	Yes

```
int LXERFIDTag1ProgramID( BYTE *status, BYTE tagIDBitCount,
BYTE password, BYTE *tagID );
```

This API performs the TAG_1_PROGRAM_ID function. All Class 1 tags receiving this command will program the specified tag ID in memory.

Input

BYTE **tagIDBitCount** – Number of ID bits to follow (up to 0x60 – 96 bits for 12 byte tag)
Values: 0x40 for 64-bit tag
0x60 for 96-bit tag
All other values will produce an error from the Reader.

BYTE **password –** Tag password. LXERFIDTag1ProgramID command programs the password along with the Tag ID. Once the password is programmed in, Lock and Kill commands sent to the tag are required to provide this password.

BYTE **tagID**[12] – Tag ID to program (up to 12 bytes)

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

LXERFIDTag1VerifyID

Product	API supported
MX3-RFID	Yes

```
typedef struct {
    BYTE tagLen;
    BYTE epcIDCode[12];
    BYTE password;
    BYTE crc[2];
}TAG_INFO;
typedef struct {
    BYTE status;
    INT tagCount;
    TAG_INFO tagInfo[100];
}RFID_VERIFY;

RFID_VERIFY *pRFIDVerifyPtr;

int LXERFIDTag1VerifyID( RFID_VERIFY **pRFIDVerifyPtr  );
```

This API performs the TAG_1_VERIFY_ID function. All tags receiving this command reply with:

- their CRC, followed by their entire ID Code, followed by their Password.

A tag that has successfully executed the LOCK_ID command ignores the VERIFY_ID command.

Input

N/A

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

Content of RFID_VERIFY structures returned:

Summary info:

BYTE **status** – Error Code

INT **tagCount** – number of tags verified

For each tag:

BYTE **tagLen –** Length of tagID (in bytes) stored in **epcIDCode**.

BYTE **epcIDCode** [12] – Contains the EPC tag ID code (up to 96 bits), MSB first

BYTE **password** - The 8-bit password received from the EPC tag. MSB first. Lock and Kill commands sent to the tag are required to provide this password.

BYTE **crc**[2] – Contains the 16-bit CRC response from the tag. MSB first.

Note: API will allocate space for **pRFIDVerifyPtr**. *The caller must free the memory by calling:* ***free(*****pRFIDVerifyPtr*****)**.

LXERFIDTag1LockID

Product	API supported
MX3-RFID	Yes

```
int LXERFIDTag1LockID( BYTE *status, BYTE password );
```

This API performs the TAG_1_LOCK_ID function. This command prevents any further modification of the tag ID, CRC, and Password.

Input

BYTE **password –** 8-bit password that is programmed into the tag.

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

LXERFIDTag1EraseID

Product	API supported
MX3-RFID	Yes

```
int LXERFIDTag1EraseID( BYTE *status );
```

This API performs the TAG_1_ERASE_ID function. This command sets all bits of the tag ID, CRC, and Password to '0'. A tag that has successfully executed the LOCK_ID command ignores the ERASE_ID command

Input

N/A

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE s**tatus** – Error Code

LXERFIDTag1Write

Product	API supported
MX3-RFID	Yes

```
int LXERFIDTag1Write( BYTE *status, BYTE tagIDBitCount, BYTE
password, BYTE *tagID );
```

This API performs the TAG_1_WRITE function, which combines the PROGRAM_ID and LOCK_ID operations.

Input

BYTE **tagIDBitCount** – Number of ID bits to follow

Values: 0x40 for 64-bit tag
0x60 for 96-bit tag
All other values will produce an error from the Reader.

BYTE **password –** password

BYTE **tagID**[12] – Tag ID to program

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

LXERFIDMaskDefine

Product	API supported
MX3-RFID	Yes

```
#define FIELD_NAME_LENGTH               40
#define MASK_VALUE_LENGTH               24

typedef struct {
    TCHAR fieldName[FIELD_NAME_LENGTH+1];
    INT maskOffset;
    INT filterCheck;
    TCHAR maskValue[MASK_VALUE_LENGTH+1];
}MASK_DEFINE;

typedef struct {
    BYTE status;
    MASK_DEFINE maskDefine[6];
}MASK_INFO;

MASK_INFO *MASKINFOPTR;

int LXERFIDMaskDefine( BYTE *status, MASK_INFO *pMaskInfoPtr );
```

Function to define up to six masks used to filter the Read operations.

Input

Contents of MASK_INFO structure:

Summary info:

BYTE **status –** Error Code, used internally. Passed in value ignored

For each mask:

TCHAR **fieldName[FIELD_NAME_LENGTH+1]** – Name of the mask, to be used in MaskRead.

INT **maskOffset** – Specifies the first character for the Mask matching, zero-indexed.

INT **filterCheck** – Specifies whether the mask is selected.

TCHAR **maskValue[MASK_VALUE_LENGTH+1]** – Mask to be used for filtering Tag IDs.

Output

The function returns TRUE if Status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE s**tatus** – Error Code

LXERFIDMaskRead

Product	API supported
MX3-RFID	Yes

```
#define FIELD_NAME_LENGTH               40

typedef struct {
    BYTE status;
    TCHAR fieldName1[FIELD_NAME_LENGTH+1];
    TCHAR fieldName2[FIELD_NAME_LENGTH+1];
    TCHAR fieldName3[FIELD_NAME_LENGTH+1];
    TCHAR fieldName4[FIELD_NAME_LENGTH+1];
    TCHAR fieldName5[FIELD_NAME_LENGTH+1];
    TCHAR fieldName6[FIELD_NAME_LENGTH+1];
}MASK_READ;

MASK_READ *MASKREADPTR;

int LXERFIDMaskRead( BYTE *status, BYTE tagType, MASK_READ
*pMaskReadPtr, INT *tagCount, BYTE **tagID, INT *tagIDLen );
```

Function to define up to six masks used to filter the Read operations.

Input

BYTE tagType – Tag type to read (0 = Class 0, 1 = Class 1)

Contents of MASK_READ structure:

TCHAR **fieldName1[FIELD_NAME_LENGTH+1]** – Name of the first mask to use.

TCHAR **fieldName2[FIELD_NAME_LENGTH+1]** – Name of the second mask to use.

TCHAR **fieldName3[FIELD_NAME_LENGTH+1]** – Name of the third mask to use.

TCHAR **fieldName4[FIELD_NAME_LENGTH+1]** – Name of the fourth mask to use.

TCHAR **fieldName5[FIELD_NAME_LENGTH+1]** – Name of the fifth mask to use.

TCHAR **fieldName6[FIELD_NAME_LENGTH+1]** – Name of the sixth mask to use.

Note: FieldNames selected in MaskRead() call should match those defined in the MaskDefine() call. Tags will be filtered based on the logical OR of the Masks selected.

Output

The function returns TRUE if Status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

INT **tagCount –** Number of tags read

BYTE ***tagID** – A string containing a ***Preamble***, a sequence of 1 or more tag IDs separated by a ***Separator***, and ending with a ***Postamble***.

INT **tagIDLen –** Length of **tagID.**

Note: API will allocate space for **tagID**. *The caller must free the memory by calling:* ***free(*****tagID*****)***.

LXERFIDSetPreamble

Product	API supported
MX3-RFID	Yes

```
int LXERFIDSetPreamble( BYTE *status, TCHAR *preamble );
```

This API performs the SET_PREAMBLE function. The preamble is set for the output of all read commands (TAG_0_READ, TAG_1_READ, and MASK_READ)

Input

TCHAR **preamble**[11] **–** Up to 5 characters that can be specified by combination of 7-bit ASCII characters and hat-encoded characters. A hat-encoded character is treated as one character.

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

LXERFIDSetPostamble

Product	API supported
MX3-RFID	Yes

```
int LXERFIDSetPostamble( BYTE *status, TCHAR *postamble );
```

This API performs the SET_POSTAMBLE function. The postamble is set for the output of all read commands (TAG_0_READ, TAG_1_READ, and MASK_READ)

Input

TCHAR **postamble** [11] **–** Up to 5 characters that can be specified by a combination of 7-bit ASCII characters and hat-encoded characters. A hat-encode character is treated as one character.

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

LXERFIDSetSeparator

Product	API supported
MX3-RFID	Yes

```
int LXERFIDSetSeparator( BYTE *status, TCHAR *separator );
```

This API performs the SET_SEPARATOR function. The tag separator is set for the output of all read commands (TAG_0_READ, TAG_1_READ, and MASK_READ).

Input

TCHAR **separator** [5] **–** Up to 2 characters that can be specified by combination of 7-bit ASCII characters and hat-encoded characters. A hat-encoded character is treated as one character.

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

LXERFIDReaderPowerTimeout

Product	API supported
MX3-RFID	Yes

```
int LXERFIDReaderPowerTimeout( BYTE *status, BYTE readerTimeout
);
```

This API performs the READER_POWER_TIMEOUT function. The timeout is set for the Reader Power Management to kick in. When the driver is inactive for the specified time, it puts the reader into "Standby" mode to conserve power.

Input

BYTE **readerTimeout –** Enumerated timeout value.

Range: 0x00 – 0x17 where:
- 0x00 = Disable feature. Reader never goes into disabled state
- 0x01 = 3 sec
- 0x02 = 4 sec
- 0x03 = 5 sec
- 0x04 = 10 sec
- 0x05 = 15 sec
- 0x06 = 20 sec
- 0x07 = 30 sec
- 0x08 = 45 sec
- 0x09 = 1 min
- 0x0a = 2 min
- 0x0b = 3 min
- 0x0c = 4 min
- 0x0d = 5 min
- 0x0e = 6 min
- 0x0f = 7 min
- 0x10 = 8 min
- 0x11 = 9 min
- 0x12 = 10 min
- 0x13 = 11 min
- 0x14 = 12 min
- 0x15 = 13 min
- 0x16 = 14 min
- 0x17 = 15 min

Default: 0x01

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

LXERFIDNotifyReadSuccess

Product	API supported
MX3-RFID	Yes

```
int LXERFIDNotifyReadSuccess( BYTE *status, BYTE beepOn );
```

Function

This API performs the NOTIFY_READER_SUCCESS function. It turns ON or OFF a beep on a read operation that results in one or more tags read.

Input

BYTE **beepOn** – Flag can be set to either 0 (no beep) or 1 (beep).
Range: 0x00 – 0x01 where:
0x00 = Do not beep
0x01 = Beep

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** – Error Code

LXERFIDNotifyReadOn

Product	API supported
MX3-RFID	Yes

```
int LXERFIDNotifyReadOn( BYTE *status, BYTE buzzOn );
```

This API performs the NOTIFY_READER_ON function. It turns ON or OFF a buzz on a read operation that does not produce a beep.

Input

BYTE **buzzOn** – Flag can be set to either 0 (no buzz) or 1 (buzz).
Range: 0x00 – 0x01 where:
0x00 = Do not buzz
0x01 = Buzz

Output

The function returns TRUE if status is 0x00 (ERR_NONE), FALSE otherwise. Check status and GetLastError() for error information.

BYTE **status** - Error Code

LXERFIDGetData

Product	API supported
MX3-RFID	Yes

```
#define   WM_LXE_RFIDFULL                    0x604
int LXERFIDGetData( BYTE **tagData, INT *tagDataLen );
```

This API performs the GET_DATA function. The message WM_LXE_RFIDFULL indicates the RFID read is complete, and the data may be read by calling LXERFIDGetData function.

Input

N/A

Output

BYTE ***tagData** – Data from the last read operation.

INT **tagDataLen** – Number of bytes in **tagData**

*Note: API will allocate space for **tagData**. The caller must free the memory by calling: **free(tagData)**.*

Status Field Codes

ERROR	Code	Description
ERR_NONE	0x00	No errors or faults encountered
ERR_UNDEFINED	0xFF	Unknown error/fault encountered
ERR_INVALID_CMD	0xFE	Command not recognized
ERR_INVALID_PARAM	0xFD	Command parameter(s) out of range
ERR_INSUFFICIENT_DATA	0xFC	Insufficient data supplied with command
ERR_ANT_FAULT	0xFB	An antenna fault (open/short) was detected
ERR_PLL_LOCK_FAIL	0xFA	PLL failed to lock
ERR_DC_FAIL	0xF9	DC input voltage out of range (+/-5%)
ERR_RF_FAIL	0xF8	Low RF power (not used in response to SYSTEM command)
ERR_OVR_TEMP	0xF7	TX isolator over-temp
ERR_DNL_BAD_CRC	0xF6	Invalid data CRC during DOWNLOAD command
ERR_DNL_INVALID_ADDR	0xF5	Invalid address during DOWNLOAD command
ERR_DNL_INVALID_SUBCMD	0xF4	DOWNLOAD sub-command not supported by application
ERR_DNL_ERASE_TIMEOUT	0xF3	Unable to erase flash for DOWNLOAD command due to a timeout condition for flash erase
ERR_DNL_WRITE_TIMEOUT	0xF2	Unable to write to flash for DOWNLOAD command due to a timeout condition for flash write
ERR_DNL_WRITE_FAIL	0xF1	Unable to write to flash for DOWNLOAD command due to flash write failure
ERR_FP_READ_FAIL	0xF0	Invalid Forward Power level reading
ERR_VOLT_MON_FAIL	0xEF	Invalid Voltage Monitor circuit reading
ERR_READ_ANT_SENSE_FAIL	0xEE	Invalid Antenna Sense reading
ERR_READ_PA_TEMP_FAIL	0xED	Invalid Power Amp Temperature reading
ERR_INVALID_SUB_CMD	0xEC	Invalid Subcommand received from Host
ERR_PROG_ID_FAIL	0xEB	Failed to program tag ID
ERR_TAG_READ_FAIL	0xEA	Failed to read a tag
ERR_ERASE_ID_FAIL	0xE9	Failed to erase the tag
ERR_LOCK_ID_FAIL	0xE8	Failed to lock the tag
ERR_KILL_ID_FAIL	0xE7	Failed to kill the tag

Output to MX3-RFID Keyboard Buffer

The RFID Driver provides a runtime utility for directing the output of read operations to the device keyboard buffer. Only read operations are supported through this utility. When an "RFID Read" button is pressed, a READ command will be called. Based on the "Tag Types to Read" setting (set through the LXE RFID configuration Utility), the READ command performs one of the following:

- TAG_0_READ,
- TAG_1_READ,
- or both TAG_0_READ and TAG_1_READ.

Appendix A Character Code Tables

Hexadecimal and Hat Encoded Characters

Desired ASCII	Hexadecimal Value	Hat Encoded		Desired ASCII	Hexadecimal Value	Hat Encoded
NUL	00	^@		ESC	1B	^[
SOH	01	^A		FS	1C	^\\
STX	02	^B		GS	1D	^]
ETX	03	^C		RS	1E	^^
EOT	04	^D		US	1F	^_ (underscore))
ENQ	05	^E		(space)	20	
ACK	06	^F		!	21	
BEL	07	^G		“	22	
BS	08	^H		#	23	
HT	09	^I		$	24	
LF	0A	^J		%	25	
VT	0B	^K		&	26	
FF	0C	^L		‘	27	
CR	0D	^M		(	28	
SO	0E	^N		)	29	
SI	0F	^O		*	2A	
DLE	10	^P		+	2B	
DC1(XON)	11	^Q		‘	2C	
DC2	12	^R		–	2D	
DC3 (XOFF)	13	^S		.	2E	
DC4	14	^T		/	2F	
NAK	15	^U		0	30	
SYN	16	^V		1	31	
ETB	17	^W		2	32	
CAN	18	^X		3	33	
EM	19	^Y		4	34	
SUB	1A	^Z		5	35	

Desired ASCII	Hexadecimal Value	Hat Encoded		Desired ASCII	Hexadecimal Value	Hat Encoded
6	36			V	56	
7	37			W	57	
8	38			X	58	
9	39			Y	59	
:	3A			Z	5A	
;	3B			[	5B	
<	3C			\	5C	
=	3D			]	5D	
>	3E			^	5E	
?	3F			_	5F	
@	40			`	60	
A	41			a	61	
B	42			b	62	
C	43			c	63	
D	44			d	64	
E	45			e	65	
F	46			f	66	
G	47			g	67	
H	48			h	68	
I	49			i	69	
J	4A			j	6A	
K	4B			k	6B	
L	4C			l	6C	
M	4D			m	6D	
N	4E			n	6E	
O	4F			o	6F	
P	50			p	70	
Q	51			q	71	
R	52			r	72	
S	53			s	73	
T	54			t	74	
U	55			u	75	

Desired ASCII	Hexadecimal Value	Hat Encoded		Desired ASCII	Hexadecimal Value	Hat Encoded
v	76			SPA	96	~^V
W	77			EPA	97	~^W
X	78				98	~^X
Y	79				99	~^Y
Z	7A				9A	~^Z
{	7B			CSI	9B	~^[
|	7C			ST	9C	~^\\
}	7D			OSC	9D	~^]
~	7E			PM	9E	~^^
	7F			APC	9F	~^_ (underscrore)
	80	~^@		(no-break-space)	A0	~ (tilde and space)
	81	~^A		¡	A1	~!
	82	~^B		¢	A2	~″
	83	~^C		£	A3	~#
IND	84	~^D		¤	A4	~$
NEL	85	~^E		¥	A5	~%
SSA	86	~^F		¦	A6	~&
ESA	87	~^G		§	A7	~′
HTS	88	~^H		¨	A8	~(
HTJ	89	~^I		©	A9	~)
VTS	8A	~^J		ª	AA	~*
PLD	8B	~^K		«	AB	~+
PLU	8C	~^L		¬	AC	~,
RI	8D	~^M		(soft hypen)	AD	~- (dash)
SS2	8E	~^N		®	AE	~.
SS3	8F	~^O		¯	AF	~/
DCU	90	~^P		°	B0	~0
PU1	91	~^Q		±	B1	~1
PU2	92	~^R		²	B2	~2
STS	93	~^S		³	B3	~3
CCH	94	~^T		´	B4	~4
MW	95	~^U		µ	B5	~5

Desired ASCII	Hexadecimal Value	Hat Encoded		Desired ASCII	Hexadecimal Value	Hat Encoded
¶	B6	~6		Ö	D6	~V
·	B7	~7		×	D7	~W
¸	B8	~8		Ø	D8	~X
¹	B9	~9		Ù	D9	~Y
º	BA	~:		Ú	DA	~Z
»	BB	~;		Û	DB	~[
¼	BC	~<		Ü	DC	~\\
½	BD	~=		Ý	DD	~]
¾	BE	~>		Þ	DE	~\^
¿	BF	~?		ß	DF	~_
À	C0	~@		à	E0	~`
Á	C1	~A		á	E1	~a
Â	C2	~B		â	E2	~b
Ã	C3	~C		ã	E3	~c
Ä	C4	~D		ä	E4	~d
Å	C5	~E		å	E5	~e
Æ	C6	~F		æ	E6	~f
Ç	C7	~G		ç	E7	~g
È	C8	~H		è	E8	~h
É	C8	~I		é	E9	~i
Ê	CA	~J		ê	EA	~j
Ë	CB	~K		ë	EB	~k
Ì	CC	~L		ì	EC	~l
Í	CD	~M		í	ED	~m
Î	CE	~N		î	EE	~n
Ï	CF	~O		ï	EF	~o
Ð	D0	~P		ð	F0	~p
Ñ	D1	~Q		ñ	F1	~q
Ò	D2	~R		ò	F2	~r
Ó	D3	~S		ó	F3	~s
Ô	D4	~T		ô	F4	~t
Õ	D5	~U		õ	F5	~u

Desired ASCII	Hexadecimal Value	Hat Encoded		Desired ASCII	Hexadecimal Value	Hat Encoded
ö	F6	~v		û	FB	~{
÷	F7	~w		ü	FC	~|
ø	F8	~x		ý	FD	~}
ù	F9	~y		þ	FE	~~
ú	FA	~z		ÿ	FF	~^?