

System Library Manual

(Version 1.08)

CASIO Computer Co., Ltd.

Copyright ©2011. All rights reserved.

February 2011

Table of the Contents

	Editorial Record	5
Chapter 1.	Overview	6
Chapter 2.	Operation Environment	7
Chapter 3.	Functions List	8
3.1	SysGetModelName	15
3.2	SysGetDeviceIDCode	17
3.3	SysGetUserIDCode	18
3.4	SysGetSystemName	19
3.5	SysSetBootup	21
3.6	SysGetBootup	23
3.7	SysSetOffMaskTime	24
3.8	SysGetOffMaskTime	25
3.9	SysSetStorageOffMaskTime	26
3.10	SysGetStorageOffMaskTime	27
3.11	SysSetPowerOnKeyTime	28
3.12	SysGetPowerOnKeyTime	29
3.13	SysSetPowerOffKeyTime	30
3.14	SysGetPowerOffKeyTime	31
3.15	SysPowerOff	32
3.16	SysDisablePowerOff	33
3.17	SysEnablePowerOff	34
3.18	SysGetPowerOff	35
3.19	SysDisableAPO	36
3.20	SysEnableAPO	37
3.21	SysGetAPO	38
3.22	SysSoftReset	39
3.23	SysCheckIOBOX	40
3.24	SysCheckCharger	41
3.25	SysSetBackupLife	42
3.26	SysGetBackupLife	43
3.27	SysSetWakeOn	44
3.28	SysGetWakeOn	45
3.29	SysSetVirtualOffMode	46
3.30	SysSetVirtualOffModeEx	48
3.31	SysGetVirtualOffMode	50
3.32	SysSetSystemManagedVirtualOffMode	51
3.33	SysGetSystemManagedVirtualOffMode	52
3.34	SysDisplayOff	53
3.35	SysDisplayOn	54
3.36	SysGetDisplayPowerState	56
3.37	SysTouchPanelOn	57
3.38	SysTouchPanelOff	58
3.39	SysGetTouchPanelState	59
3.40	SysSetEmulateMouseState	60
3.41	SysGetEmulateMouseState	61
3.42	SysAudioOff	62
3.43	SysAudioOn	63

3.44	SysGetAudioPowerState	64
3.45	SysKeyBackLightOn	65
3.46	SysKeyBackLightOff	66
3.47	SysGetKeyBackLightState	67
3.48	SysMultiTouchOn	68
3.49	SysMultiTouchOff	69
3.50	SysGetMultiTouchState	70
3.51	SysSetLED	71
3.52	SysGetLED	74
3.53	SysPrepareLED	75
3.54	SysUpdateLED	77
3.55	SysSetLEDState	79
3.56	SysGetLEDState	81
3.57	SysDisableCardDetect	82
3.58	SysEnableCardDetect	83
3.59	SysGetCardDetect	84
3.60	SysDisableWLAN	85
3.61	SysEnableWLAN	86
3.62	SysGetWLAN	87
3.63	SysRenewPartition	88
3.64	SysPlayBuzzer	89
3.65	SysStopBuzzer	91
3.66	SysSetBuzzerVolume	92
3.67	SysGetBuzzerVolume	93
3.68	SysSetBuzzerMute	94
3.69	SysGetBuzzerMute	95
3.70	SysSetCPUMode	96
3.71	SysGetCPUMode	97
3.72	SysSetDefaultCPUMode	98
3.73	SysSet180Rotate	99
3.74	SysGet180Rotate	100
3.75	SysSetBLBattery	101
3.76	SysGetBLBattery	102
3.77	SysSetBLExpPower	103
3.78	SysGetBLExpPower	104
3.79	SysGetBLMaximum	105
3.80	SysSetBLOffTime	106
3.81	SysGetBLOffTime	107
3.82	SysPlayVibrator	108
3.83	SysStopVibrator	110
3.84	SysSetVibratorMute	111
3.85	SysGetVibratorMute	112
3.86	SysSetFnKeyLock	113
3.87	SysGetFnKeyLock	114
3.88	SysSetOneKeyLock	115
3.89	SysGetOneKeyLock	116
3.90	SysSetAllKeyLock	117
3.91	SysGetAllKeyLock	118
3.92	SysSetInputMode	119
3.93	SysGetInputMode	120

3.94	SysSetNormalUserDefineKey	121
3.95	SysGetNormalUserDefineKey	135
3.96	SysSetUserDefineKey	136
3.97	SysGetUserDefineKey	138
3.98	SysSetFnUserDefineKey	140
3.99	SysGetFnUserDefineKey	142
3.100	SysSetOtherUserDefineKey	143
3.101	SysGetOtherUserDefineKey	145
3.102	SysSetKeyRepeat	146
3.103	SysGetKeyRepeat	148
3.104	SysSetFakeRepeat	149
3.105	SysGetFakeRepeat	150
3.106	SysSetAllKeyRepeat	151
3.107	SysGetAllKeyRepeat	152
3.108	SysDeleteUserDefineKey	153
3.109	SysDeleteUserDefineKeyEx	154
3.110	SysGetDefaultKey	155
3.111	SysSetUserDefineKeyState	156
3.112	SysGetUserDefineKeyState	157
3.113	SysSetResetUserDefineKeyState	158
3.114	SysGetResetUserDefineKeyState	159
3.115	SysSetEnableKeyMode	160
3.116	SysGetEnableKeyMode	161
3.117	SysSetEnableTriggerKey	162
3.118	SysGetEnableTriggerKey	163
3.119	SysSetFnKeyOperation	164
3.120	SysGetFnKeyOperation	165
3.121	SysWaitForEvent	166
3.122	SysTerminateWaitEvent	168

No part of this document may be produced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of CASIO Computer Co., Ltd. in Tokyo Japan. Information in this document is subject to change without advance notice. CASIO Computer Co., Ltd. makes no representations or warranties with respect to the contents or use of this manual and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose.

Editorial Record

Manual Version no.	Date edited	Page	Content
1.00	October 2008		Original
1.01	January 2009	40	The description about the parameters is corrected.
		41	The description about the parameters is corrected.
1.02	March 2009	18	In Chapter 3.4, the required memory size is changed from "40 characters" to "80 characters".
		21	In Chapter 3.5, the content in Table 3.11 is corrected.
		53 to 54	In Chapter 3.35, note for virtual OFF mode is added.
		69	In Chapter 3.52, the return values are corrected.
		84	In Chapter 3.64, the content in Table 3.17 is corrected.
		115	In Chapter 3.92, the description about the parameters is corrected.
		116	In Chapter 3.93, the description about the parameters is corrected.
		123	In Chapter 3.94, Table 3.24 is corrected.
		7	In Chapter 2, "Microsoft® Windows Mobile 6.1" is added to "OS".
		14	In Chapter 3.1, "DT-X30 Windows Mobile" is added to the model names.
1.03	August 2009	-	IT-800 is added.
1.04	January 2010	121	In Chapter 3.91, the content in Table 3.21 is corrected.
1.05	May 2010	74	In Chapter 3.52, the description about the parameters is corrected.
		79	In Chapter 3.55, the description about the parameters is corrected.
1.06	May 2010	121	In Chapter 3.94, "DT-X30-R50" is added.
1.07	January 2011	-	IT-300 and DT-X8 are added.
		68	In Chapter 3.48, SysMultiTouchOn function is added.
		69	In Chapter 3.49, SysMultiTouchOff function is added.
		70	In Chapter 3.50, SysGetMultiTouchState function is added.
1.08	February 2011	121	In Chapter 3.94, Table 3.21 is modified.

1. Overview

The **System Library** provides specific expanded functions for running the system such as key control, power control, and etc. integrated in each model of the CASIO handheld terminals.

The **System Class Library** is a wrapper library layer. The library can be directly manipulated by .NET Compact Framework application.

The use of **System Library** enables the enhancement of source code compatibility for mobile applications regardless of handheld terminal model.

With regard to the function for either the "unsupported error" or "parameter error", at time of process execution, the user is notified that the function is unsupported or that the process itself should be disabled. In this way it is possible to describe a source code that is aware of the "functions" rather than a source code that is aware of the handheld terminal "models" (and the devices integrated in those handheld terminal models), which enables the development of a business application style that attaches "importance to functions".

Note:

The aim of this library is to enhance the source code compatibility for mobile business application, it is not intended to guarantee the compatibility of functions of integrated devices. In actuality it decides whether the "unsupported error" or "parameter error" is required to notify the user that functions are not supported or that the actual process should be disabled.

Also, for the business application to actually run in multiple handheld terminal models, it must be rebuilt to agree with the CPU types (ARMV4T, ARMV4 ...).

2. Operation Environment

Applicable Handheld Terminals

- IT-600 series
- DT-X11 series
- DT-X7 series
- DT-X30 series
- IT-3100 series
- IT-800 series
- IT-300 series
- DT-X8 series

OS

- Microsoft® WindowsCE 5.0
- Microsoft® WindowsCE 6.0
- Microsoft® Windows Mobile 6.1
- Microsoft® Windows Mobile 6.5
- Microsoft® Windows Mobile 6.5.3

Development Environment

- Microsoft® eMbedded C++ Version4.0 + Service Pack 4
- Microsoft® Visual Studio 2005 + Service Pack 1
- Microsoft® Visual Studio 2008 + Service Pack 1

Supplied Files

- SystemLib.h
- SystemLib.lib
- SystemLib.dll
- SystemLibNet.dll (Class Library)

Steps to Start Up

For Visual C++:

1. Include the header file, **SystemLib.h**, in the source program and define it as the library that uses the **SystemLib.lib** file.
2. **SystemLib.dll** is already integrated by default in the terminal.

For Visual Basic and Visual C#:

1. Add **SystemLibNet.dll** to the Project Reference.
2. **SystemLib.dll** is already integrated by default in the terminal.
3. Copy **SystemLibNet.dll** into the same folder where the execution module is stored.

3. Functions List

Table 3.1 Security functions

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysGetModelName	Retrieves terminal information.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetDeviceIDCode	Retrieves Device ID.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetUserIDCode	Retrieves User ID.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetSystemName	Retrieves folder and port names different by model.	Y	Y	Y	Y	Y	Y	Y	Y

Table 3.2 Power Supply functions

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysSetBootup	Sets up "Enable" or "Disable" for turning on power.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetBootup	Retrieves "Enabled" or "Disable" status for turning on power.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetOffMaskTime	Sets up a period of time for disabling turning off power.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetOffMaskTime	Retrieves time period set for disabling turning off power.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetStorageOffMaskTime	Sets up a period of time for disabling turning off power after use of storage.	Y	Y	Y	Y	-	Y	Y	Y
SysGetStorageOffMaskTime	Retrieves time period set for disabling turning off power after use of storage.	Y	Y	Y	Y	-	Y	Y	Y
SysSetPowerOnKeyTime	Sets up period of time for pressing the Power key continuously until the power is turned on.	-	-	-	-	-	-	-	-
SysGetPowerOnKeyTime	Retrieves time period set for pressing the Power key continuously until the power is turned on.	-	-	-	-	-	-	-	-
SysSetPowerOffKeyTime	Sets up a period of time for pressing the Power key until the power is turned off.	-	-	-	-	-	-	-	-
SysGetPowerOffKeyTime	Retrieves time period set for pressing the Power key until the power is turned off.	-	-	-	-	-	-	-	-
SysPowerOff	Turns off the Power on the terminal.	Y	Y	Y	Y	Y	Y	Y	Y
SysDisablePowerOff	Sets up "Disable" for turning off the power on the terminal.	Y	Y	Y	Y	Y	Y	Y	Y
SysEnablePowerOff	Sets up "Enable" or "Disable" for turning off the power on the terminal.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetPowerOff	Retrieves "Enable" or "Disable" status for turning off the power on the terminal.	Y	Y	Y	Y	Y	Y	Y	Y

Continue.

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysDisableAPO	Sets up "Disable" for turning off the power automatically.	Y	Y	Y	Y	Y	Y	Y	Y
SysEnableAPO	Sets up "Enable" for turning off the power automatically.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetAPO	Retrieves "Enable" or "Disable" status for turning off the power automatically.	Y	Y	Y	Y	Y	Y	Y	Y
SysSoftReset	Resets the terminal.	Y	Y	Y	Y	Y	Y	Y	Y
SysCheckIOBOX	Retrieves status of connection with cradle.	Y	Y	Y	Y	Y	Y	Y	Y
SysCheckCharger	Retrieves status of connection with Battery charger.	-	-	-	-	-	-	-	-
SysSetBackupLife	Sets up period of time for data backup into RAM.	-	-	-	-	-	-	-	-
SysGetBackupLife	Retrieves time period set for data backup into RAM.	-	-	-	-	-	-	-	-
SysSetWakeOn	Sets up "Enable" or "Disable" for WakeOn function of module.	-	-	-	-	Y	-	-	-
SysGetWakeOn	Retrieves "Enable" or "Disable" status for WakeOn function of module.	-	-	-	-	Y	-	-	-
SysSetVirtualOffMode	Sets up "Enable" or "Disable" for virtual turning off the power.	-	Y	Y	Y	-	Y	Y	Y
SysSetVirtualOffModeEx	Sets up "Enable" or "Disable" for virtual turning off for touch panel, screen, key operation, prohibition on APO, prohibition on turning off the power, and setting CPU speed.	-	Y	Y	Y	-	Y	Y	Y
SysGetVirtualOffMode	Retrieves "Enable" or "Disable" status for virtual turning off the power.	-	Y	Y	Y	-	Y	Y	Y
SysSetSystemManagedVirtualOff Mode	Sets up "Enable" or "Disable" for the system management virtual OFF mode.	-	-	Y	C	-	-	-	Y
SysGetSystemManagedVirtualOff Mode	Retrieves the status of "Enable" or "Disable" for the system management virtual OFF mode.	-	-	Y	C	-	-	-	Y
SysDisplayOff	Turns off the power virtually on the terminal with the display turned off.	-	Y	Y	Y	-	Y	Y	Y
SysDisplayOn	Turns off the power virtually on the terminal with the display turned on.	-	Y	Y	Y	-	Y	Y	Y
SysGetDisplayPowerState	Retrieves "Enable" or "Disable" status for turning off the power virtually on the terminal.	-	Y	Y	Y	-	Y	Y	Y

Continue.

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysTouchPanelOn	Turns on the touch panel virtually with the touch panel turned on.	-	Y	Y	Y	-	Y	Y	Y
SysTouchPanelOff	Turns off the touch panel virtually with the touch panel turned off.	-	Y	Y	Y	-	Y	Y	Y
SysGetTouchPanelState	Retrieves "Enable" or "Disable" status for turning off the touch panel virtually.	-	Y	Y	Y	-	Y	Y	Y
SysSetEmulateMouseState	Sets up "Enable" or "Disable" for the Mouse Emulator.	-	-	Y	-	-	-	-	-
SysGetEmulateMouseState	Retrieves the status of "Enable" or "Disable" for the Mouse Emulator.	-	-	Y	-	-	-	-	-
SysAudioOff	Turns off the audio virtually with the audio turned off.	-	Y	Y	Y	-	Y	Y	Y
SysAudioOn	Turns on the audio virtually with the audio turned on.	-	Y	Y	Y	-	Y	Y	Y
SysGetAudioPowerState	Retrieves "Enable" or "Disable" status for turning off the audio virtually.	-	Y	Y	Y	-	Y	Y	Y
SysKeyBackLightOn	Turns on the key backlight.	-	-	-	Y	-	Y	Y	Y
SysKeyBackLightOff	Turns off the key backlight.	-	-	-	Y	-	Y	Y	Y
SysGetKeyBackLightState	Retrieves the status of "Enable" or "Disable" for the key backlight.	-	-	-	Y	-	Y	Y	Y
SysMultiTouchOn		-	-	-	-	-	-	Y	-
SysMultiTouchOff		-	-	-	-	-	-	Y	-
SysGetMultiTouchState		-	-	-	-	-	-	Y	-

Table 3.3 LED functions

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysSetLED	Sets up "Enable" or "Disable" for turning on the LED.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetLED	Retrieves "Enable" or "Disable" status for turning on the LED.	Y	Y	Y	Y	Y	Y	Y	Y
SysPrepareLED	Prepares necessary process for turning on the LED.	Y	-	-	-	Y	-	-	-
SysUpdateLED	Turns on the LED.	Y	-	-	-	Y	-	-	-
SysSetLEDState	Sets up "Enable" or "Disable" for turning on the System LED.	-	Y	Y	Y	-	Y	Y	Y
SysGetLEDState	Retrieves "Enable" or "Disable" status for turning on the System LED.	-	Y	Y	Y	-	Y	Y	Y

Table 3.4 Card functions

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysDisableCardDetect	Inserts and removes virtual card with the power to the virtual card turned off.	Y	Y	-	Y	Y	Y	Y	Y
SysEnableCardDetect	Inserts and removes virtual card with the power to the virtual card turned on.	Y	Y	-	Y	Y	Y	Y	Y
SysGetCardDetect	Retrieves status of the card detection terminals for virtual card.	Y	Y	-	Y	Y	Y	Y	Y
SysDisableWLAN	Turns off the power to integrated WLAN module.	-	Y	Y	Y	-	Y	Y	Y
SysEnableWLAN	Turns on the power to integrated WLAN module	-	Y	Y	Y	-	Y	Y	Y
SysGetWLAN	Retrieves status of the power to integrated WLAN module.	-	Y	Y	Y	-	Y	Y	Y
SysRenewPartition	Carries out "Demount" to "Mount" on partition of storage so that the system recognizes the storage again.	-	Y	Y	Y	-	Y	Y	Y

Table 3.5 Buzzer functions

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysPlayBuzzer	Sounds the buzzer.	Y	Y	Y	Y	-	Y	Y	Y
SysStopBuzzer	Turns off the buzzer's sound.	Y	Y	Y	Y	-	Y	Y	Y
SysSetBuzzerVolume	Sets up sound volume of the buzzer.	Y	Y	Y	Y	-	Y	Y	Y
SysGetBuzzerVolume	Retrieves sound volume of the buzzer.	Y	Y	Y	Y	-	Y	Y	Y
SysSetBuzzerMute	Sets up sound volumes for all the parameters and individual mutes.	Y	Y	Y	Y	-	Y	Y	Y
SysGetBuzzerMute	Retrieves statuses of all the sound volumes and individual mutes.	Y	Y	Y	Y	-	Y	Y	Y

Table 3.6 CPU functions

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysSetCPUMode	Sets up the CPU frequency control.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetCPUMode	Retrieves status of the CPU frequency control.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetDefaultCPUMode	Returns the CPU speed setting to the factory-default.	Y	Y	Y	Y	Y	Y	Y	Y

Table 3.7 Display functions

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysSet180Rotate	Sets up flip angle for the screen.	Y	Y	Y	Y	Y	Y	Y	Y
SysGet180Rotate	Retrieves status of flip angle for the screen.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetBLBattery	Retrieves brightness of the screen when the power is supplied by battery pack.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetBLBattery	Sets up brightness of the screen when the power is supplied by battery pack.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetBLExpower	Sets up brightness of the backlight when the power is supplied by external power.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetBLExpower	Retrieves brightness of the backlight when the power is supplied by external power.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetBLMaximum	Retrieves the maximum value of brightness for the backlight.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetBLOffTime	Sets up a period of time for pressing the power key continuously until the power is turned off.	-	-	-	-	-	-	-	-
SysGetBLOffTime	Retrieves time period set for pressing the power key until the power is turned off.	-	-	-	-	-	-	-	-

Table 3.8 Vibrator functions

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysPlayVibrator	Turns on the vibrator.	-	Y	Y	-	-	Y	Y	Y
SysStopVibrator	Turns off the vibrator.	-	Y	Y	-	-	Y	Y	Y
SysSetVibratorMute	Sets up "Enable" or "Disable" for all the parameters for the vibrator and individual mutes.	-	Y	Y	-	-	Y	Y	Y
SysGetVibratorMute	Retrieves statuses of all the parameters for the vibrator and individual mutes.	-	Y	Y	-	-	Y	Y	Y

Table 3.9 Keys functions

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysSetFnKeyLock	Sets up "Enable" or "Disable" for the Fn key to activate.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetFnKeyLock	Retrieves "Enable" or "Disable" status for the Fn key to activate.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetOneKeyLock	Sets up "Enable" or "Disable" for lock with individual keys.	-	-	-	-	-	-	-	-
SysGetOneKeyLock	Retrieves status of "Enable" or "Disable" for lock with individual keys.	-	-	-	-	-	-	-	-
SysSetAllKeyLock	Sets up "Enable" or "Disable" for lock with specified key.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetAllKeyLock	Retrieves "Enable" or "Disable" status for lock with specified key.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetInputMode	Sets up the operation by the input changeover key.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetInputMode	Retrieves the operation by the input changeover key.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetNormalUserDefineKey	Sets up key codes in normal mode.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetNormalUserDefineKey	Retrieves key codes in normal mode.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetUserDefineKey	Sets up user defined keys which are freely issued in either numeric, hiragana, katakana, alphabets in uppercase, or alphabets in lowercase mode.	-	-	Y	Y	-	Y	Y	Y
SysGetUserDefineKey	Retrieves user defined keys.	-	-	Y	Y	-	Y	Y	Y
SysSetFnUserDefineKey	Sets up key codes in Fn mode.	-	-	-	-	-	-	-	-
SysGetFnUserDefineKey	Retrieves key codes in Fn mode.	-	-	-	-	-	-	-	-
SysSetOtherUserDefineKey	Sets up key codes in alphanumeric mode	-	-	-	-	-	-	-	-
SysGetOtherUserDefineKey	Retrieves key codes in alphanumeric mode	-	-	-	-	-	-	-	-
SysSetKeyRepeat	Sets up key codes of the keys that perform key repeat.	-	-	-	-	-	-	-	-
SysGetKeyRepeat	Retrieves key codes of the keys that perform key repeat.	-	-	-	-	-	-	-	-

Continue.

SysSetFakeRepeat	Sets up key codes of the keys that perform fake key repeat.	-	-	-	-	-	-	-	-
SysGetFakeRepeat	Retrieves key codes of the keys that perform fake key repeat.	-	-	-	-	-	-	-	-
SysSetAllKeyRepeat	Sets up key repeat on all the keys.	-	-	-	-	-	-	-	-
SysGetAllKeyRepeat	Retrieves all keys that perform key repeat.	-	-	-	-	-	-	-	-
SysDeleteUserDefineKey	Deletes user defined keys.	Y	Y	Y	Y	Y	Y	Y	Y
SysDeleteUserDefineKeyEx	Deletes user defined keys.	-	-	Y	Y	-	Y	Y	Y
SysGetDefaultKey	Retrieves default keys.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetUserDefineKeyState	Sets up "Enable" or "Disable" for user defined keys.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetUserDefineKeyState	Retrieves "Enable" or "Disable" status for the user defined keys.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetResetUserDefineKeyState	Sets up "Enable" or "Disable" for user defined keys at time of reset on the terminal	Y	Y	Y	Y	Y	Y	Y	Y
SysGetResetUserDefineKeyState	Retrieves "Enable" or "Disable" status for user defined keys at time of reset on the terminal.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetEnableKeyMode	Sets up "Enable" or "Disable" for the key mode transition at time of changing key input mode.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetEnableKeyMode	Retrieves "Enable" or "Disable" status for the key mode transition at time of changing key input mode.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetEnableTriggerKey	Sets up "Enable" or "Disable" for Trigger key.	Y	Y	Y	Y	Y	Y	Y	Y
SysGetEnableTriggerKey	Retrieves "Enable" or "Disable" status for the Trigger key.	Y	Y	Y	Y	Y	Y	Y	Y
SysSetFnKeyOperation	Sets up "Enable" or "Disable" for the Fn key operation.	-	Y	Y	Y	-	Y	Y	Y
SysGetFnKeyOperation	Retrieves "Enable" or "Disable" status for the Fn key operation.	-	Y	Y	Y	-	Y	Y	Y

Table 3.10 Event control functions

Function	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
SysWaitForEvent	Waits for an event occurred in driver, etc.	Y	Y	Y	Y	Y	Y	Y	Y
SysTerminateWaitEvent	Forces to terminate the wait for event to occur.	Y	Y	Y	Y	Y	Y	Y	Y

Y: Supported.

C: Supported on DT-X30 Windows CE series. Not supported on DT-X30 Windows Mobile series.

-: Error is returned when the function is called.

3.1 SysGetModelName

This function retrieves information about the handheld terminal such as model no. of the terminal, the OS version and the platform version.

Calling Sequences

```
[C++]
DWORD SysGetModelName(
    DWORD *dwModel,
    DWORD *dwVersion,
    DWORD *dwPlatform
)
```

```
[Visual Basic]
Public Shared Function SysGetModelName( _
    ByRef dwModel As Int32, _
    ByRef dwVersion As Int32, _
    ByRef dwPlatform As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetModelName(
    ref Int32 dwModel,
    ref Int32 dwVersion,
    ref Int32 dwPlatform
);
```

Parameters

dwModel

This parameter is for capturing the value of model no. The values and model numbers are as follows.

IT-3100	: 4
IT-600	: 7
DT-X11	: 8
DT-X7	: 9
DT-X30 Winodws CE	: 10
DT-X30 Windows Mobile	: 11
IT-800 Windows Mobile	: 13
IT-300	: 15
DT-X8	: 16

dwVersion

This parameter is for retrieving the OS build ID.

dwPlatform

This parameter is for retrieving WindowsCE version.

Return Values

TRUE	: Normal end
FUNCTION_UNSupport	: Unsupported error

3.2 SysGetDeviceIDCode

This function retrieves device ID written in the EEPROM.

Calling Sequences

```
[C++]
DWORD SysGetDeviceIDCode(
    TCHAR *pdwDevID
)
```

```
[Visual Basic]
Public Shared Function SysGetDeviceIDCode( _
    ByVal pdwDevID As Char() _
) As Int32
```

```
[C#]
public static Int32 SysGetDeviceIDCode(
    Char[] pdwDevID
);
```

Parameters

pdwDevID

This parameter is for retrieving UUID that expresses the device ID. See notes below.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

Notes:

- The UUID returns a character string that consists of characters up to 32. For this reason, be sure to allocate an area in the memory for more than 33 characters.
- Before calling this function, be sure to clear the area with zero.

3.3 SysGetUserIDCode

This function retrieves the user ID written in the EEPROM.

Calling Sequences

```
[C++]
DWORD SysGetUserIDCode(
    DWORD *pwUserID
)
```

```
[Visual Basic]
Public Shared Function SysGetUserIDCode( _
    ByRef pwUserID As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetUserIDCode(
    ref Int32 pwUserID
);
```

Parameters

pwUserID

This parameter is for retrieving the user ID.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.4 SysGetSystemName

This function retrieves names and/or paths for the folders and ports listed below. Each name listed below is different according to model number of CASIO handheld terminals.

- Flash folder
- CF1 Card folder
- CF2 Card folder
- SD Card folder
- Bluetooth Client Serial port
- Bluetooth Server Serial port
- Bluetooth Dial-up port
- Bluetooth LAN port
- IrDA Raw COM port
- USB Serial COM port
- Serial COM port #1
- Serial COM port #2
- Card Serial COM port #1
- Card Serial COM port #2
- Storage path for the Laser configuration file
- Storage path for the Imager configuration file

Calling Sequences

```
[C++]
DWORD SysGetSystemName(
    DWORD      dwValueID,
    TCHAR      *lpValueName
)
```

```
[Visual Basic]
Public Shared Function SysGetSystemName( _
    ByVal dwValueID As Int32 _
    ByVal lpValueName As Char() _
) As Int32
```

```
[C#]
public static Int32 SysGetSystemName(
    Int32 dwValueID,
    Char[] lpValueName
);
```

Parameters

dwValueID

This parameter is for setting IDs listed below for names of the ports and/or paths that you wish to retrieve.

STORAGE_Name_Flash	: Flash folder
STORAGE_Name_CF1	: CF1 Card folder
STORAGE_Name_CF2	: CF2 Card folder
STORAGE_Name_SD	: SD Card folder
BT_ClientSerial	: Bluetooth Client Serial port
BT_ServerSerial	: Bluetooth Server Serial port
BT_DaialUP	: Bluetooth Dial-up port
BT_LAN	: Bluetooth LAN port
IrDA_Raw	: IrDA RawCOM port
USB_Serial	: USB Serial COM port
Serial_Port1	: Serial COM port #1
Serial_Port2	: Serial COM port #2
Card_Serial1	: Card Serial COM port #1
Card_Serial2	: Card Serial COM port #2
OBRIni_FilePath	: Storage path for Laser configuration file
IMGIni_FilePath	: Storage path for Imager configuration file

lpValueName

This parameter is to store the folder names, port names, and storage paths specified with each ID. For the storage, the memory must be secured for 80 characters. If any undefined folder name, port name or storage path is specified, the function returns "Normal end", but the specified name will not be stored.

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UNSUPPORTED	: Unsupported error

3.5 SysSetBootup

This function sets up "Enable" or "Disable" for turning on the power by a wakeup factor. While the terminal is kept off, the power on the terminal can be turned on by a wakeup factor. It enables control on turning on the power.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetBootup** function.

Calling Sequences

```
[C++]
DWORD SysSetBootup(
    DWORD BootMode
)
```

```
[Visual Basic]
Public Shared Function SysSetBootup( _
    ByVal BootMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetBootup(
    Int32 BootMode
);
```

Parameters

BootMode

This parameter specifies a sum of the values in OR operation selected in the below table to control the power ON.

Table 3.11 Setting values

Setting Value	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
BOOT_MULTI	Turns on the power with the Multi key.	-	-	-	-	-	-	-	-
BOOT_LEFT	Turns on the power with the Left key.	-	-	-	-	-	-	-	-
BOOT_RIGHT	Turns on the power with the Right key.	-	-	-	-	-	-	-	-
BOOT_LEFTTRIGGER	Turns on the power with the Left trigger key.	Y	Y	Y	Y	Y	Y	Y	Y
BOOT_RIGHTTRIGGER	Turns on the power with the Right trigger key.	Y	Y	Y	Y	Y	Y	Y	Y
BOOT_IOBOX	Turns on the power by putting the terminal on cradle.	Y	Y	Y	Y	Y	Y	Y	Y

Continue.

BOOT_NONE	Disables turning on the power with a wakeup factor.	Y	-	-	-	Y	Y	Y	Y
BOOT_PGBUTTON	Turns on the power with the Program key.	-	-	-	-	-	-	-	-
BOOT_APL1	Turns on the power with application key 1.	-	-	-	-	-	-	-	-
BOOT_APL2	Turns on the power with application key 2.	-	-	-	-	-	-	-	-
BOOT_APL3	Turns on the power with application key 3.	-	-	-	-	-	-	-	-
BOOT_APL4	Turns on the power with application key 4.	-	-	-	-	-	-	-	-
BOOT_APL5	Turns on the power with application key 5.	-	-	-	-	-	-	-	-
BOOT_APL6	Turns on the power with application key 6.	-	-	-	-	-	-	-	-
BOOT_USB	Turns on the power when USB connection is established.	-	-	-	-	-	-	-	-
BOOT_CFCARD	Turns on the power when a CF card is inserted.	-	-	-	-	-	-	-	-
BOOT_GUNTRIGGER	Turns on the power with the Trigger grip key.	-	Y	-	-	-	-	-	-
BOOT_CENTERTRIGGER	Turns on the power with the Center trigger key.	-	Y	Y	Y	-	Y	-	-

Return Values

TRUE : Normal end
 FALSE : Internal error
 SYS_PARAMERR : Parameter error
 FUNCTION_UN SUPPORT : Unsupported error

3.6 SysGetBootup

This function retrieves "Enable" or "Disable" status for turning on the power by a wakeup factor.

Calling Sequences

```
[C++]
DWORD SysGetBootup(
    DWORD *pBootMode
)
```

```
[Visual Basic]
Public Shared Function SysGetBootup( _
    ByRef pBootMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBootup(
    ref Int32 pBootMode
);
```

Parameters

pBootMode

This parameter is for retrieving "Enable" or "Disable" status of turning on the power. See **SysSetBootup** function for value to retrieve.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.7 SysSetOffMaskTime

This function sets up a period of time between when the power is turned on and until when the power is turned off.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetOffMaskTime** function.

Calling Sequences

```
[C++]
DWORD SysSetOffMaskTime(
    DWORD dwTime
)
```

```
[Visual Basic]
Public Shared Function SysSetOffMaskTime( _
    ByVal dwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetOffMaskTime(
    Int32 dwTime
);
```

Parameters

dwTime

This parameter is for specifying a period of time in second that the power cannot be turned off with the Power key after it has been turned on. The defaults values for each model are as follows. See note below.

IT-600, DT-X7, DT-X30, IT-800, IT-300, and DT-X8	: 10
Other models	: 5

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

Do not set up a value in *dwTime* parameter smaller than the default indicated above for the respective models.

3.8 SysGetOffMaskTime

This function retrieves the time period that disables turning off the power after it has been turned on.

Calling Sequences

```
[C++]  
DWORD SysGetOffMaskTime(  
    DWORD *pdwTime  
)
```

```
[Visual Basic]  
Public Shared Function SysGetOffMaskTime( _  
    ByRef pdwTime As Int32 _  
) As Int32
```

```
[C#]  
public static Int32 SysGetOffMaskTime(  
    ref Int32 pdwTime  
);
```

Parameters

pdwTime

This parameter is for retrieving a period of time that the power cannot be turned off with the Power key after it has been turned on.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.9 SysSetStorageOffMaskTime

This function sets up a period of time that disables turning off the power after use of either one of the memory storages.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetStorageOffMaskTime** function.

Calling Sequences

```
[C++]
DWORD SysSetStorageOffMaskTime(
    DWORD dwStorage,
    DWORD dwStorageType
)
```

```
[Visual Basic]
Public Shared Function SysSetStorageOffMaskTime( _
    ByVal dwStorage As Int32 _
    ByVal dwStorageType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetStorageOffMaskTime(
    Int32 dwStorage
    Int32 dwStorageType
);
```

Parameters

dwStorage

This parameter is for setting up a time period in the unit of millisecond.

dwStorageType

This parameter is for setting storage type selecting one of the values listed below.

STORAGE_MASK_CF	: CF card
STORAGE_MASK_SD	: SD card
STORAGE_MASK_FD	: Flash disc
STORAGE_MASK_PCMCIA	: PCMCIA card

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.10 SysGetStorageOffMaskTime

This function retrieves the period of time that disables turning off the power after use of either one of the memory storages. See the *dwStorageType* parameter of **SysSetStorageOffMaskTime** function for memory storage kinds.

Calling Sequences

```
[C++]
DWORD SysGetOFFMaskTime(
    DWORD *pdwStorage,
    DWORD *pdwStorageType
)
```

```
[Visual Basic]
Public Shared Function SysGetStorageOffMaskTime( _
    ByRef pdwStorage As Int32 _
    ByRef pdwStorageType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetStorageOffMaskTime(
    ref Int32 pdwStorage
    ref Int32 pdwStorageType
);
```

Parameters

pdwStorage

This is for retrieving the time period. See **SysSetStorageOffMaskTime** function for value to retrieve.

pdwStorageType

This parameter is for retrieving storage type. See **SysSetStorageOffMaskTime** function for value to retrieve.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.11 SysSetPowerOnKeyTime

This function sets up a period of time for continuously pressing the Power key until when the power is turned on.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetPowerOnKeyTime** function.

Calling Sequences

```
[C++]
DWORD SysSetPowerOnKeyTime(
    DWORD dwTime
)
```

```
[Visual Basic]
Public Shared Function SysSetPowerOnKeyTime( _
    ByVal dwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetPowerOnKeyTime(
    Int32 dwTime
);
```

Parameters

dwTime

This parameter is for specifying a period of time either 16 milliseconds or a value in the range of 250 to 1750 milliseconds increment of 250 that allows continuously pressing the Power key until when the power is turned on.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.12 SysGetPowerOnKeyTime

This function retrieves the period of time for continuously pressing the Power key until when the power is turned on.

Calling Sequences

```
[C++]
DWORD SysGetPowerOnKeyTime(
    DWORD *pdwTime
)
```

```
[Visual Basic]
Public Shared Function SysGetPowerOnKeyTime( _
    ByRef pdwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetPowerOnKeyTime(
    ref Int32 pdwTime
);
```

Parameters

pdwTime

This parameter is for retrieving the time period. See **SysSetPowerOnKeyTime** function for value to retrieve.

Return Values

TRUE	: Normal end
FUNCTION_UN SUPPORT	: Unsupported error

3.13 SysSetPowerOffKeyTime

This function sets up a period of time for continuously pressing the Power key until when the power is turned off.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetPowerOffKeyTime** function.

Calling Sequences

```
[C++]
DWORD SysSetPowerOffKeyTime(
    DWORD dwTime
)
```

```
[Visual Basic]
Public Shared Function SysSetPowerOffKeyTime( _
    ByVal dwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetPowerOffKeyTime(
    Int32 dwTime
);
```

Parameters

dwTime

This parameter is for specifying a period of time in the range of 250 to 1750 milliseconds increment of 250 that allows continuously pressing the Power key until when the power is turned off.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.14 SysGetPowerOffKeyTime

This function retrieves the period of time for continuously pressing the Power key until when the power is turned off.

Calling Sequences

```
[C++]
DWORD SysGetPowerOffKeyTime(
    DWORD *pdwTime
)
```

```
[Visual Basic]
Public Shared Function SysGetPowerOffKeyTime( _
    ByRef pdwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetPowerOffKeyTime(
    ref Int32 pdwTime
);
```

Parameters

pdwTime

This parameter is for retrieving the period of time set for continuously pressing the Power key. See **SysSetPowerOffKeyTime** function for values to retrieve.

Return Values

TRUE	: Normal end
FUNCTION_UN SUPPORT	: Unsupported error

3.15 SysPowerOff

This function turns off the power on the terminal.

In the Device Emulator, the function does not perform any.

Calling Sequences

```
[C++]  
DWORD SysPowerOff( )
```

```
[Visual Basic]  
Public Shared Function SysPowerOff() As Int32
```

```
[C#]  
public static Int32 SysPowerOff()
```

Parameters

None

Return Values

None	: When the function is called by a handheld terminal that supports the function. See Table 3.2.
FUNCTION_UNSUPPORTED	: Unsupported error

3.16 SysDisablePowerOff

This function disables turning off the power with the Power key. In case where turning off the power has been set disabled, PBT_APMSUSPEND or WM_POWERBROADCAST will be issued when the Power key is pressed. In your application, design the application so that this message is retrieved to precede necessary routine works before turning soft-off on the terminal. See also notes below.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetPowerOff** function.

Calling Sequences

```
[C++]  
DWORD SysDisablePowerOff( )
```

```
[Visual Basic]  
Public Shared Function SysDisablePowerOff() As Int32
```

```
[C#]  
public static Int32 SysDisablePowerOff()
```

Parameters

None

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

Notes:

- As this setting is cancelled whenever the power is turned on, set it again as required.
- Opening the battery cover intentionally or accidentally or emergency power OFF forces the power to be immediately turned off even if this setting has been set effect.

3.17 SysEnablePowerOff

This function permits turning off the power with the Power key.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetPowerOff** function.

Calling Sequences

```
[C++]  
DWORD SysEnablePowerOff( )
```

```
[Visual Basic]  
Public Shared Function SysEnablePowerOff() As Int32
```

```
[C#]  
public static Int32 SysEnablePowerOff()
```

Parameters

None

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.18 SysGetPowerOff

This function retrieves the status of "Enable" or "Disable" for permitting turning off the power with the Power key.

Calling Sequences

```
[C++]  
DWORD SysGetPowerOff( )
```

```
[Visual Basic]  
Public Shared Function SysGetPowerOff() As Int32
```

```
[C#]  
public static Int32 SysGetPowerOff()
```

Parameters

None

Return Values

TRUE	: Power OFF prohibited
FALSE	: Power OFF permitted (Default)
FUNCTION_UNSUPPORTED	: Unsupported error

3.19 SysDisableAPO

This function disables the Auto Power OFF function. If the function is carried out, turning off the power does not automatically take place irrespective of the setting at the Control Panel.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetAPO** function.

Calling Sequences

```
[C++]  
DWORD SysDisableAPO( )
```

```
[Visual Basic]  
Public Shared Function SysDisableAPO() As Int32
```

```
[C#]  
public static Int32 SysDisableAPO()
```

Parameter

None

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.20 SysEnableAPO

This function enables the Auto Power OFF possible. If the function is carried out, turning off the power automatically takes place according to the setting at the Control Panel.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetAPO** function.

Calling Sequences

```
[C++]  
DWORD SysEnableAPO( )
```

```
[Visual Basic]  
Public Shared Function SysEnableAPO() As Int32
```

```
[C#]  
public static Int32 SysEnableAPO()
```

Parameter

None

Return Values

TRUE	: Normal end
FUNCTION_UN SUPPORT	: Unsupported error

Note:

The Auto Power OFF function will not be set effect while other processes or threads are carrying out **SysDisableAPO** function even if this function has been carried out. To make the Auto Power OFF function effect, this function must be carried out for the same number of times that **SysDisableAPO** function has been carried out.

3.21 SysGetAPO

This function retrieves "Enable" or "Disable" status for the Auto Power OFF (APO) function.

Calling Sequences

```
[C++]  
DWORD SysGetAPO( )
```

```
[Visual Basic]  
Public Shared Function SysGetAPO() As Int32
```

```
[C#]  
public static Int32 SysGetAPO()
```

Parameter

None

Return Values

TRUE	: APO disabled
FALSE	: APO enabled (Default)
FUNCTION_UN SUPPORT	: Unsupported error

3.22 SysSoftReset

This function resets the terminal.

In the Device Emulator, the function does not perform.

Calling Sequences

```
[C++]  
DWORD SysSoftReset( )
```

```
[Visual Basic]  
Public Shared Function SysSoftReset() As Int32
```

```
[C#]  
public static Int32 SysSoftReset()
```

Parameter

None

Return Values

None	: When the function is supported by the terminal.
FUNCTION_UN SUPPORT	: Unsupported error

Note:

When this function is carried out, the terminal will be reset immediately, and files and data being accessed may be lost. Be sure to close all files if any is opened before carrying out this function.

3.23 SysCheckIOBOX

This function retrieves the status of contact established between the terminal and cradle. The condition between the two devices can be monitored by setting timeout time. See also note below.

Calling Sequences

```
[C++]  
DWORD SysCheckIOBOX(  
    DWORD time_out  
)
```

```
[Visual Basic]  
Public Shared Function SysCheckIOBOX( _  
    ByVal time_out As Int32 _  
) As Int32
```

```
[C#]  
public static Int32 SysCheckIOBOX(  
    Int32 time_out  
);
```

Parameters

time_out

This parameter is for specifying a period of time for monitoring the contact status selecting one of the values listed below. Any value below 200 will be automatically rounded off to 200.

200 to 3600000	: Contact status monitoring time (in millisecond)
INFINITE	: Infinite monitoring time (no timeout)

Return Values

0	: Contact detected
-1	: Timeout occurred
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

If the function is carried out continuously, be sure to set up an interval of one second or longer before the subsequent execution of the function takes place.

3.24 SysCheckCharger

This function retrieves the contact status between the terminal and battery charger. The condition between the two devices can be monitored by setting timeout time. See also note below.

Calling Sequences

```
[C++]
DWORD SysCheckCharger(
    DWORD time_out
)
```

```
[Visual Basic]
Public Shared Function SysCheckCharger( _
    ByVal time_out As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysCheckCharger(
    Int32 time_out
);
```

Parameter

time_out

This parameter is for specifying a contact status monitoring time selecting one of the values listed below. Any value below 200 will be automatically rounded off to 200.

200 to 3600000	: Contact status monitoring time (in millisecond)
INFINITE	: Infinite monitoring time (no timeout)

Return Values

0	: Contact detected
-1	: Timeout occurred
FUNCTION_UN SUPPORT	: Unsupported error

Note:

If the function is carried out continuously, be sure to set up an interval of one second or longer before the subsequent execution of the function takes place.

3.25 SysSetBackupLife

This function sets up a period of time to back up data in the RAM. Data in the RAM will be backed up for approximately 1.5 days if **NORMAL_LIFE** is specified in the *dwSetBackupLife* parameter or 3.0 days if **LONG_LIFE** is specified.

Calling Sequences

```
[C++]
DWORD SysSetBackupLife(
    DWORD dwSetBackupLife
)
```

```
[Visual Basic]
Public Shared Function SysSetBackupLife( _
    ByVal dwSetBackuplife As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetBackupLife(
    Int32 dwSetBackuplife
);
```

Parameters

dwSetBackupLife

This parameter is for specifying data backup period in the RAM selecting either of the values listed below.

LONG_LIFE	: Approx 3.0 days
NORMAL_LIFE	: Approx 1.5 days

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.26 SysGetBackupLife

This function retrieves RAM backup period. Counting the period starts when VDET1 occurs.

Calling Sequences

```
[C++]  
DWORD SysGetBackupLife( )
```

```
[Visual Basic]  
Public Shared Function SysGetBackupLife() As Int32
```

```
[C#]  
public static Int32 SysGetBackupLife()
```

Parameter

None

Return Values

LONG_LIFE	: Approx. 3.0 days
NORMAL_LIFE	: Approx. 1.5 days
FUNCTION_UN SUPPORT	: Unsupported error

3.27 SysSetWakeOn

This function sets up "Enable" or "Disable" status for the WakeOn function of module. It can control the WakeOn function for module integrated in the IT-3100 series.

Calling Sequences

```
[C++]
DWORD SysSetWakeOn(
    DWORD dwWakeOnModules
)
```

```
[Visual Basic]
Public Shared Function SysSetWakeOn( _
    ByVal dwWakeOnModules As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetWakeOn(
    Int32 dwWakeOnModules
);
```

Parameters

dwWakeOnModules

This parameter is for specifying "Enable" or "Disable" status for the WakeOn function selecting either of the values listed below.

WAKEON_SERIAL14PIN	: Enables WakeOn function with Serial 14-pin interface.
WAKEON_NON	: Disables WakeOn function.

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UNSUPPORTED	: Unsupported error

3.28 SysGetWakeOn

This function retrieves "Enable" or "Disable" status for the WakeOn function of module. It can control the WakeOn function for module integrated in the IT-3100 series.

Calling Sequences

```
[C++]
DWORD SysGetWakeOn(
    DWORD *pdwWakeOnModules
)
```

```
[Visual Basic]
Public Shared Function SysGetWakeOn( _
    ByRef pdwWakeOnModules As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetWakeOn(
    ref Int32 pdwWakeOnModules
);
```

Parameters

pdwWakeOnModules

This parameter is for retrieving "Enable" or "Disable" status for the WakeOn function.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.29 SysSetVirtualOffMode

This function sets up "Enable" or "Disable" for the Virtual OFF function. The Virtual OFF function makes the terminal become in such a condition that the power on the terminal has been turned off, but the power to the integrated WLAN module (operable only on models with the WLAN module integrated) is kept on for an immediate access right after the power on the terminal is turned on. See note on the next page to set the terminal in the virtual OFF mode.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetVirtualOffMode** function.

Calling Sequences

```
[C++]
DWORD SysSetVirtualOffMode(
    DWORD dwVirtualOFFMode
)
```

```
[Visual Basic]
Public Shared Function SysSetVirtualOffMode( _
    ByVal dwVirtualOFFMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetVirtualOffMode(
    Int32 dwVirtualOFFMode
);
```

Parameters

dwVirtualOFFMode

This parameter is for specifying "Enable" or "Disable" for the Virtual OFF function selecting either of the values listed below.

- | | |
|--------------------|--|
| VIRTUALOFF_ENABLE | : Enables the Virtual OFF function. |
| VIRTUALOFF_DISABLE | : Disables the Virtual OFF function. (Default) |

Return Values

- | | |
|---------------------|---------------------|
| TRUE | : Normal end |
| FALSE | : Error |
| FUNCTION_UN SUPPORT | : Unsupported error |

Note:

Follow the steps below to set the terminal in the virtual OFF mode.

1. Set the virtual OFF mode effect with **SysSetVirtualOffMode** function.
2. Set "Turning off the power with Power key" effect with **SysDisablePowerOff** function or **SysSetVirtualOffModeEx** function.
3. Once the above settings have been made, the power driver will issue either one of the messages below depending on the terminal's condition when the Power key is pressed.

During the power is kept turned off;

- PBT_APMSPEND(0x0004) from WM_POWERBROADCAST(0x0218)
(In ordinary operation mode, the power driver will not issue this message, but only in the virtual OFF mode.)

During the power is kept turned on;

- PBT_APMRESUMESPEND(0x0007) from WM_POWERBROADCAST

4. In a process for the received messages above, be sure to include the settings below.

In case **SysDisablePowerOff** function is used;

Each setting below must be set up individually.

- Display power control
- Key input control (See step no. 5.)
- Sound control
- Buzzer control
- Vibrator control
- LED control

In case **SysSetVirtualOffModeEx** function is used:

Setting for "Enable" or "Disable" for the virtual OFF mode

5. For "Key input control", there are keys that cannot be set disabled. For those keys, use **SysDisablePowerOff** function and **OBRClose** function for the laser models (or **IMGDeinit** function for the imager models) to disable the Power key and Trigger keys respectively.
6. In case VDET1 occurs while the virtual OFF mode is active, the battery remaining capacity will run down quickly causing VDET2 to occur if the power to the WLAN module is kept on. To avoid battery's power consumption, it is recommended to turn off the power on the terminal with **SysPowerOff** function as soon as PBT_APMBATTERYLOW (0x0009) message of WM_POWERBROADCAST is issued at a time when VDET1 occurs.

3.30 SysSetVirtualOffModeEx

This function sets up "Enable" or "Disable" for the virtual turning off including touch panel, screen, key operation, prohibition on the power off, and setting the CPU speed. The function must be carried out along with **SysSetVirtualOffMode** function to control the virtual OFF state on the components.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetVirtualOffMode** function.

Calling Sequences

```
[C++]
DWORD SysSetVirtualOffModeEx(
    DWORD dwVirtualOFFMode
)
```

```
[Visual Basic]
Public Shared Function SysSetVirtualOffModeEx( _
    ByVal dwVirtualOFFMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetVirtualOffModeEx(
    Int32 dwVirtualOFFMode
);
```

Parameters

dwVirtualOFFMode

This parameter is for specifying "Enable" or "Disable" for the virtual off state selecting either of the values listed below.

VIRTUALOFF_ENABLE	: Enables the virtual off state.
VIRTUALOFF_DISABLE	: Disables the virtual off state. (Default)

If "VIRTUALOFF_ENABLE" is set in the parameter, the virtual off states on each component will be set up as follows. Or, if "VIRTUALOFF_DISABLE" is set the previous settings will be resumed.

Touch panel	: Prohibition
Screen	: OFF
Key operation	: Prohibition
APO prohibition	: Prohibition
Prohibition on turning off the power	: Prohibition
CPU speed	: "Low" enabled

Return Values

TRUE	: Normal end
FALSE	: Error
FUNCTION_UNSupport	: Unsupported error

3.31 SysGetVirtualOffMode

This function retrieves "Enable" or "Disable" status for the Virtual OFF function.

Calling Sequences

```
[C++]
DWORD SysGetVirtualOffMode(
    DWORD *pdwVirtualOFFMode
)
```

```
[Visual Basic]
Public Shared Function SysGetVirtualOffMode( _
    ByRef pdwVirtualOFFMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetVirtualOffMode(
    ref Int32 pdwVirtualOFFMode
);
```

Parameters

pdwVirtualOFFMode

This parameter is for retrieving "Enable" or "Disable" status for the Virtual OFF function.

Table 3.12

Returned Value	Status of SysSetVirtualOffMode function	Status of SysSetVirtualOffModeEx function
VIRTUALOFF_ENABLE	Enabled	Enabled
VIRTUALOFF_DISABLE	Disabled	Disabled
VIRTUALOFF_ENABLE_NOT_EX	Enabled	Disabled
VIRTUALOFF_EX_ENABLE	Disabled	Enabled

Return Values

TRUE : Normal end
FUNCTION_UNSUPPORTED : Unsupported error

3.32 SysSetSystemManagedVirtualOffMode

This function sets up "Enable" or "Disable" for the system management virtual OFF mode.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetSystemManagedVirtualOffMode** function.

Calling Sequences

```
[C++]
DWORD SysSetSystemManagedVirtualOffMode(
    DWORD dwSysVOffMode
)
```

```
[Visual Basic]
Public Shared Function SysSetSystemManagedVirtualOffMode( _
    ByVal dwSysVOffMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetSystemManagedVirtualOffMode(
    Int32 dwSysVOffMode
);
```

Parameter

dwSysVOffMode

This parameter is for specifying "Enable" or "Disable" for the system management virtual OFF mode selecting either of the values listed below.

VIRTUALLYOFF_ENABLE	: Enable the system management virtual OFF mode.
VIRTUALLYOFF_DISABLE	: Disable the system management virtual OFF mode. (Default)

Return Values

TRUE	: Normal end
FALSE	: Error
FUNCTION_UNSUPPORTED	: Unsupported error

3.33 SysGetSystemManagedVirtualOffMode

This function retrieves the status of "Enable" or "Disable" for the system management virtual OFF mode.

Calling Sequences

```
[C++]
DWORD SysGetSystemManagedVirtualOffMode(
    DWORD *pdwSysVOffMode
)
```

```
[Visual Basic]
Public Shared Function SysGetSystemManagedVirtualOffMode( _
    ByRef pdwSysVOffMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetSystemManagedVirtualOffMode(
    ref Int32 pdwSysVOffMode
);
```

Parameter

pdwSysVOffMode

This parameter is for retrieving the status of "Enable" or "Disable" for the system management virtual OFF mode.

Return Values

TRUE	: Normal end.
FALSE	: Internal error.
FUNCTION_UNSUPPORTED	: Unsupported error.

3.34 SysDisplayOff

This function turns off the screen on the terminal for the virtual OFF mode.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetDisplayPowerState** function.

Calling Sequences

```
[C++]  
DWORD SysDisplayOff( )
```

```
[Visual Basic]  
Public Shared Function SysDisplayOff() As Int32
```

```
[C#]  
public static Int32 SysDisplayOff()
```

Parameters

None

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.35 SysDisplayOn

This function turns on the screen on the terminal that is being in the virtual OFF mode.

Calling Sequences

```
[C++]  
DWORD SysDisplayOn( )
```

```
[Visual Basic]  
Public Shared Function SysDisplayOn() As Int32
```

```
[C#]  
public static Int32 SysDisplayOn()
```

Parameters

None

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

Follow the steps below to set the terminal in the virtual OFF mode.

1. Set the virtual OFF mode effect with **SysSetVirtualOffMode** function.
7. Set "Turning off the power with Power key" effect with **SysDisablePowerOff** function.
8. Once the above settings have been made, the power driver will issue either one of the messages below depending on the terminal's condition when the Power key is pressed (the same messages that are issued in ordinary power ON/OFF procedures.)

During the power is kept turned off;

- PBT_APMSPEND(0x0004) from WM_POWERBROADCAST(0x0218)

During the power is kept turned on;

- PBT_APMRESUMESPEND(0x0007) from WM_POWERBROADCAST

9. Using **SysSetVirtualOffMode** and **SysSetAllKeyLock** functions, turn off or on the screen and set all key inputs disabled when either one of the messages above (in step 3) is issued. The Multi key and Trigger keys cannot be set disabled with **SysSetAllKeyLock** function.
10. To set the Multi key disabled, set any key not used by the application to the Multi key with **SysSetNormalUserDefineKey** function and turn off the click sound. If a reset is performed during the virtual OFF mode, the setting of the Multi key is remained the same with the one set during the virtual OFF mode. To avoid this, set the Multi key, if the virtual OFF mode is used, so that the default setting must be set to it when the application is invoked. To set the trigger keys disabled, perform the close process for the laser scanner or C-MOS imager.

11. In case VDET1 occurs while the virtual OFF mode is active, the battery remaining capacity will run down quickly causing VDET2 to occur if the power to the WLAN module is kept on. To avoid the battery's power consumption, it is recommended to turn off the power on the terminal with **SysPowerOff** function as soon as PBT_APMBATTERYLOW (0x0009) message is issued from WM_POWERBROADCAST generated at a time when VDET1 occurs.
12. It is recommended to set the Vibrator, Buzzer, and LED disabled during the virtual OFF mode since each performance of these devices can be controlled in the application.

3.36 SysGetDisplayPowerState

This function retrieves the ON/OFF status for display power.

Calling Sequences

```
[C++]  
DWORD SysGetDisplayPowerState( )
```

```
[Visual Basic]  
Public Shared Function SysGetDisplayPowerState ( ) As Int32
```

```
[C#]  
public static Int32 SysGetDisplayPowerState ( )
```

Parameter

None

Return Values

Values indicating the status of the display power.

TRUE	: Display is on.
FALSE	: Display is off.
FUNCTION_UN SUPPORT	: Unsupported error

3.37 SysTouchPanelOn

This function turns on the touch panel on the terminal that is being in the virtual OFF mode.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetTouchPanelState** function.

Calling Sequences

```
[C++]  
DWORD SysTouchPanelOn( )
```

```
[Visual Basic]  
Public Shared Function SysTouchPanelOn ( ) As Int32
```

```
[C#]  
public static Int32 SysTouchPanelOn ( )
```

Parameter

None

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UNSupport	: Unsupported error

3.38 SysTouchPanelOff

This function turns off the touch panel on the terminal for the virtual OFF mode.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetTouchPanelState** function.

Calling Sequences

```
[C++]  
DWORD SysTouchPanelOff( )
```

```
[Visual Basic]  
Public Shared Function SysTouchPanelOff ( ) As Int32
```

```
[C#]  
public static Int32 SysTouchPanelOff ( )
```

Parameter

None

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UNSupport	: Unsupported error

3.39 SysGetTouchPanelState

This function retrieves the ON/OFF status of the touch panel.

Calling Sequences

```
[C++]  
DWORD SysGetTouchPanelState( )
```

```
[Visual Basic]  
Public Shared Function SysGetTouchPanelState ( ) As Int32
```

```
[C#]  
public static Int32 SysGetTouchPanelState ( )
```

Parameter

None

Return Values

Values indicating the ON/OFF status of the touch panel

TRUE	: Touch panel is on.
FALSE	: Touch panel is off.
FUNCTION_UNSupport	: Unsupported error

3.40 SysSetEmulateMouseState

This function sets up "Enable" or "Disable" for the Mouse Emulator.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetEmulateMouseState** function.

Calling Sequences

```
[C++]
DWORD SysSetEmulateMouseState(
    DWORD  dwEmulateMouseState
)
```

```
[Visual Basic]
Public Shared Function SysSetEmulateMouseState( _
    ByVal dwEmulateMouseState As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetEmulateMouseState(
    Int32 dwEmulateMouseState
);
```

Parameter

dwEmulateMouseState

This parameter is for specifying "Enable" or "Disable" for the Mouse Emulator selecting either of the values listed below.

EMULATE_ENABLE	: Enable the Mouse Emulator.
EMULATE_DISABLE	: Disable the Mouse Emulator (Default).

Return Values

TRUE	: Normal end.
FALSE	: Internal error.
FUNCTION_UNSUPPORTED	: Unsupported error.

3.41 SysGetEmulateMouseState

This function retrieves the status of "Enable" or "Disable" for the Mouse Emulator.

Calling Sequences

```
[C++]
DWORD SysSetEmulateMouseState(
    DWORD *pdwEmulateMouseState
)
```

```
[Visual Basic]
Public Shared Function SysGetEmulateMouseState( _
    ByRef pdwEmulateMouseState As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetEmulateMouseState(
    ref Int32 pdwEmulateMouseState
);
```

Parameter

pdwEmulateMouseState

This parameter is for retrieving the status of "Enable" or "Disable" for the Mouse Emulator.

Return Values

TRUE	: Normal end.
FALSE	: Internal error.
FUNCTION_UN SUPPORT	: Unsupported error.

3.42 SysAudioOff

This function turns off the audio for the virtual OFF mode.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetAudioPowerState** function.

Calling Sequences

```
[C++]  
DWORD SysAudioOff( )
```

```
[Visual Basic]  
Public Shared Function SysAudioOff ( ) As Int32
```

```
[C#]  
public static Int32 SysAudioOff ( )
```

Parameters

None

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UNSupport	: Unsupported error

3.43 SysAudioOn

This function turns on the audio integrated in the terminal that is being in the virtual OFF mode.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetAudioPowerState** function.

Calling Sequences

```
[C++]  
DWORD SysAudioOn( )
```

```
[Visual Basic]  
Public Shared Function SysAudioOn ( ) As Int32
```

```
[C#]  
public static Int32 SysAudioOn ( )
```

Parameter

None

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UN SUPPORT	: Unsupported error

3.44 SysGetAudioPowerState

This function retrieves the ON/OFF status of the audio.

Calling Sequences

```
[C++]  
DWORD SysGetAudioPowerState( )
```

```
[Visual Basic]  
Public Shared Function SysGetAudioPowerState ( ) As Int32
```

```
[C#]  
public static Int32 SysGetAudioPowerState ( )
```

Parameter

None

Return Values

Values indicating the ON/OFF status of the audio

TRUE	: Audio is on.
FALSE	: Audio is off.
FUNCTION_UNSupport	: Unsupported error

3.45 SysKeyBackLightOn

This function turns on the key backlight.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetKeyBackLightState** function.

Calling Sequences

```
[C++]  
DWORD SysKeyBackLightOn( )
```

```
[Visual Basic]  
Public Shared Function SysKeyBackLightOn() As Int32
```

```
[C#]  
public static Int32 SysKeyBackLightOn()
```

Parameter

None

Return Values

The function returns one of the values below.

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UN SUPPORT	: Unsupported error

3.46 SysKeyBackLightOff

This function turns off the key backlight.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetKeyBackLightState** function.

Calling Sequences

```
[C++]  
DWORD SysKeyBackLightOff()
```

```
[Visual Basic]  
Public Shared Function SysKeyBackLightOff() As Int32
```

```
[C#]  
public static Int32 SysKeyBackLightOff()
```

Parameter

None

Return Values

The function returns one of the values below.

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UNSupport	: Unsupported error

3.47 SysGetKeyBackLightState

This function retrieves the status of ON/OFF for the key backlight.

Calling Sequences

```
[C++]  
DWORD SysGetKeyBackLightState()
```

```
[Visual Basic]  
Public Shared Function SysGetKeyBackLightState() As Int32
```

```
[C#]  
public static Int32 SysGetKeyBackLightState()
```

Parameter

None

Return Values

The function returns one of the values below.

TRUE	: The key backlight is on.
FALSE	: The key backlight is off.
FUNCTION_UN SUPPORT	: Unsupported error

3.48 SysMultiTouchOn

This function enable multi touch mode.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetMultiTouchState** function.

Calling Sequences

```
[C++]  
DWORD SysMultiTouchOn( )
```

```
[Visual Basic]  
Public Shared Function SysMultiTouchOn ( ) As Int32
```

```
[C#]  
public static Int32 SysMultiTouchOn ( )
```

Parameter

None

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UNSupport	: Unsupported error

Note

This setting will be disable by reset operation.

3.49 SysMultiTouchOff

This function disable multi touch model.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetMultiTouchState** function.

Calling Sequences

```
[C++]
DWORD SysMultiTouchOff( )
```

```
[Visual Basic]
Public Shared Function SysMultiTouchOff ( ) As Int32
```

```
[C#]
public static Int32 SysMultiTouchOff ( )
```

Parameter

None

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UNSupport	: Unsupported error

3.50 SysGetMultiTouchState

This function retrieves the Enable / Disable status of the multi touch mode.

Calling Sequences

```
[C++]  
DWORD SysGetMultiTouchState( )
```

```
[Visual Basic]  
Public Shared Function SysGetMultiTouchState ( ) As Int32
```

```
[C#]  
public static Int32 SysGetMultiTouchState ( )
```

Parameter

None

Return Values

Values indicating the ON/OFF status of the touch panel

TRUE	: Multi touch is on.
FALSE	: Multi touch is off. (Default)
FUNCTION_UNSupport	: Unsupported error

3.51 SysSetLED

This function turns on or off the LED.

Calling Sequences

```
[C++]
DWORD SysSetLED(
    DWORD dwLEDMODE,
    DWORD dwNum,
    DWORD dwOnTime,
    DWORD dwOFFTime
)
```

```
[Visual Basic]
Public Shared Function SysSetLED( _
    ByVal dwLEDMODE As Int32, _
    ByVal dwNum As Int32, _
    ByVal dwOnTime As Int32, _
    ByVal dwOFFTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetLED(
    Int32 dwLEDMODE,
    Int32 dwNum,
    Int32 dwOnTime,
    Int32 dwOFFTime
);
```

Parameters

dwLEDMODE

This parameter is for specifying turning on or off the LED.

Table 3.13 Setting values

Setting Value	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
LED_OFF	Turns on the LED. (Default)	Y	Y	Y	Y	Y	Y	Y	Y
LED_GREEN	Turns on the LED in green.	Y	Y	Y	Y	Y	Y	Y	Y
LED_RED	Turns on the LED in red.	Y	Y	Y	Y	Y	Y	Y	Y
LED_ORANGE	Turns on the LED in orange.	Y	Y	Y	Y	Y	Y	Y	Y
LED_BLUE	Turns on the LED in blue.	-	Y	Y	Y	Y	Y	Y	Y
LED_CYAN	Turns on the LED in cyan.	-	Y	Y	Y	Y	Y	Y	Y

Continue.

LED_MAGENTA	Turns on the LED in magenta.	-	Y	Y	Y	Y	Y	Y	Y
LED_ACCESS	Turns on the access LED.	-	-	-	-	-	-	-	-
LED_NOTIFICATION	Turns on the notification LED.	-	-	-	-	-	-	-	-
LED_BOTH	Turns on both the LEDs (access and notification).	-	-	-	-	-	-	-	-
LED_BLINK	Turns on the specified LED in specified color. Specify a color using the parameters of LED_GREEN, LED_RED and LED_ORANGE in logical OR.	-	-	-	-	-	-	-	-

dwNum

This parameter is for specifying the number of times to turn on the LED. Specify the number of times that the LED has turned on and the number of times that the LED has turned off.

dwOnTime

This parameter is for specifying a period of time in 1/16 seconds for turning on the LED.

Table 3.14

DT-X11 IT-600 DT-X7 DT-X30 IT-3100 IT-800 IT-300 DT-X8	0 to 255
---	----------

dwOFFTime

This parameter is for specifying a period of time in 1/16 seconds for turning off the LED.

Table 3.15

DT-X11 IT-600 DT-X7 DT-X30 IT-3100 IT-800 IT-300 DT-X8	0 to 255
---	----------

Return Values

TRUE	: Normal end
FALSE	: Internal error
SYS_PARAMERR	: Parameter error
FUNCTION_UNSupport	: Unsupported error

Note:

If LED_BLINK (example; LED_BLINK|LED_RED) is specified in ***dwLedMode*** parameter, the interval of turning on and off the LED continuously will follow the default setting specified in the system. Set LED_OFF in the parameter to turn off the LED. This ignores any values set in the ***dwNum***, ***downtime***, and ***dwOFFTime*** parameters.

See "Examples" on the following.

Examples

- Setting for quickly turning on and off the LED in orange (Hardware specifications dictate that the following values set in the ***dwNum***, ***downtime***, and ***dwOFFTime*** parameters are the minimum values.)
 SysSetLED(LED_ORANGE, 1, 1, 0);
- Setting for blinking the LED in green three times (ON for one second and OFF for one second)
 SysSetLED(LED_GREEN, 3, 16, 16);
- Setting for turning on the LED in red for two seconds
 SysSetLED(LED_RED, 1, 32, 0);

3.52 SysGetLED

This function retrieves the status of the LED (Indicator #1 or #2, model dependant) being on or off.

Calling Sequences

```
[C++]  
DWORD SysGetLED( )
```

```
[Visual Basic]  
Public Shared Function SysGetLED() As Int32
```

```
[C#]  
public static Int32 SysGetLED()
```

Parameter

None

Return Values

Values indicating the status of the LED are returned. See **SysSetLED** function for the values to return.

Otherwise

FUNCTION_UNSUPPORTED : Unsupported error

The return values below are for DT-X11, IT-600, DT-X7, DT-X30., IT-800, IT-300, and DT-X8.

LED_NOTICE_ON	: The LED is lit for user notification.
LED_WLAN_ON	: The LED is lit for the terminal being in WLAN operation.
LED_BT_ON	: The LED is lit for the terminal being in Bluetooth operation.
LED_SCANOK_ON	: The LED is lit for completion in scanning bar code.
LED_SCANERR_ON	: The LED is lit for error in scanning bar code.
LED_DISKACCESS_ON	: The LED is lit for the disk bening accessed.
LED_WWAN_ON	: The LED is lit for the terminal being in WWAN operation.
LED_GPS_ON	: The LED is lit for the terminal being in GPS operation.

3.53 SysPrepareLED

This function prepares for turning on the LED. With **SysSetLED** function, it takes about 130 milliseconds before turning on the LED. This is due to the hardware specifications. However, using this function and **SysUpdateLED** function together will immediately turn it on. See also "Examples" on the next page.

Calling Sequences

```
[C++]
DWORD SysPrepareLED(
    DWORD dwLEDMode,
    DWORD dwNum,
    DWORD dwOnTime,
    DWORD dwOFFTime
)
```

```
[Visual Basic]
Public Shared Function SysPrepareLED( _
    ByVal dwLEDMode As Int32, _
    ByVal dwNum As Int32, _
    ByVal dwOnTime As Int32, _
    ByVal dwOFFTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysPrepareLED(
    Int32 dwLEDMode,
    Int32 dwNum,
    Int32 dwOnTime,
    Int32 dwOFFTime
);
```

Parameters

dwLEDMode

This parameter is for preparation to turn on the specified LED selecting one of the values listed below.

LED_GREEN	: Preparation for turning on in green
LED_RED	: Preparation for turning on in red
LED_ORANGE	: Preparation for turning on in orange

dwNum

This parameter is for specifying the number of times to repeat turning on the LED. Set a value in this parameter to repeat on and off.

dwOnTime

This parameter is for specifying a period of ON time in the range of 0 to 255 (in the unit of 1/16 seconds) for the LED.

dwOFFTime

This parameter is for specifying a period of OFF time in the range of 0 to 255 (in the unit of 1/16 seconds) for the LED.

Return Values

TRUE	: Normal end
FALSE	: Abnormal end
FUNCTION_UNSUPPORTED	: Unsupported error

Examples

- Setting for preparation to quickly turn on and off the LED in orange (Hardware specifications dictate that the following values set in the ***dwNum***, ***downtime***, and ***dwOFFTime*** parameters are the minimum values.)
SysPrepareLED(LED_ORANGE, 1, 1, 3);
- Setting for preparation to blink the LED in green three times (ON for one second and OFF for one second)
SysPrepareLED(LED_GREEN, 3, 16, 16);
- Setting for preparation to turn on the LED in red for two seconds
SysPrepareLED(LED_RED, 1, 32, 3);
- Setting for preparation to turn on the LED in red continuously (Set "0" in the ***dwOnTime*** and ***dwOFFTime*** parameters for continuous turning on the LED.)
SysPrepareLED(LED_RED, 1, 255, 0);

Note:

If **SysPrepareLED** function is set while the LED is being lit, the setting may be disturbed. To avoid this, the LED is forced to turn off when **SysPrepareLED** function is carried out.

3.54 SysUpdateLED

This function turns on the LED. With **SysSetLED** function, it takes about 130 milliseconds before turning on the LED. This is due to the hardware specifications. However, using the function and **SysPrepareLED** function together will immediately turn it on. See also "How to Use the Function" below.

Calling Sequences

```
[C++]
DWORD SysUpdateLED(
    DWORD dwLEDMode
)
```

```
[Visual Basic]
Public Shared Function SysUpdateLED( _
    ByVal dwLEDMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysUpdateLED(
    Int32 dwLEDMode
);
```

Parameters

dwLEDMode

This parameter is for specifying turning on or off the LED selecting one of the values listed below.

LED_OFF	: Turns off the LED.
LED_GREEN	: Turns on the LED in green with current value set in the parameter.
LED_RED	: Turns on the LED in red with current value set in the parameter.
LED_ORANGE	: Turns on the LED in orange with current value set in the parameter.

Return Values

TRUE	: Normal end
FALSE	: Abnormal end
FUNCTION_UN SUPPORT	: Unsupported error

How to Use the Function

Before turning on the LED, set the preparation by carrying out **SysPrepareLED** function. Then carry out **SysUpdateLED** function to turn it on under conditions preset with **SysPrepareLED** function. If **SysUpdateLED** function is carried out, allow a time period of 130 milliseconds or more to elapse after carrying out **SysPrepareLED** function and then proceed to turning on the LED.

Note:

If **SysUpdateLED** function is carried out to turn on the LED instead of **SysPrepareLED** function, it may be lit in an unexpected way following the existing settings.

3.55 SysSetLEDState

This function sets up "Enable" or "Disable" for turning on the system LED (Indicator #2).

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetLEDState** function.

Calling Sequences

```
[C++]
DWORD SysSetLEDState(
    DWORD dwType,
    BOOL bMute
)
```

```
[Visual Basic]
Public Shared Function SysSetLEDState( _
    ByVal dwType As Int32 _
    ByVal bMute As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetLEDState(
    Int32 dwType,
    Boolean bMute
);
```

Parameters

dwType

This parameter is for specifying an event for turning on the system LED (Indicator #2) selecting one of the values listed below.

LED_BT	: Connection in Bluetooth established.
LED_WLAN	: Connection in WLAN established.
LED_SCANOK	: Scanning bar code is complete.
LED_SCANERR	: Scanning bar code results in error.
LED_ALL	: All the states mentioned above.
LED_WWAN	: Connection in WAN established.
LED_GPS	: Connection in GPS established.

bMute

This parameter is for turning on or off the system LED (Indicator #2) selecting either of the values listed below.

LED_Disable	: Turns off the LED.
LED_Enable	: Turns on the LED.

Return Values

TRUE	: Normal end
------	--------------

FUNCTION_UN SUPPORT : Unsupported error

Note:

If either "All is set to off" has been set, or "an attribute that is to be turned on or blinked" has been individually set to on, the LED will not be turned on or blinked. This setting will become effect when the LED turning-on process is carried out a next time.

3.56 SysGetLEDState

This function retrieves the status of the system LED (Indicator #2) being on or off.

Calling Sequences

```
[C++]  
DWORD SysGetLEDState(  
    DWORD dwType  
)
```

```
[Visual Basic]  
Public Shared Function SysGetLEDState( _  
    ByVal dwType As Int32 _  
) As Int32
```

```
[C#]  
public static Int32 SysGetLEDState(  
    Int32 dwType  
) ;
```

Parameters

dwType

This parameter is for specifying an LED type. See **SysSetLEDState** function for LED types.

Return Values

LED_Disable	: Turns off the LED
LED_Enable	: Turns on the LED
FUNCTION_UNSupport	: Unsupported error

3.57 SysDisableCardDetect

This function turns off the power to the virtual card by disabling the card detect terminals and removes or inserts the virtual card.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetCardDetect** function.

Calling Sequences

```
[C++]
DWORD SysDisableCardDetect(
    DWORD socket
)
```

```
[Visual Basic]
Public Shared Function SysDisableCardDetect( _
    ByVal socket As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysDisableCardDetect(
    Int32 socket
);
```

Parameters

socket

This parameter is for specifying a card socket for removing or inserting the virtual card selecting either of the values listed in the table.

Table 3.16 Setting values

Setting Value	Card Type	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
TYPE_PCMCIA	PCMCIA card	Y	--	--	--	Y	Y	Y	Y
TYPE_CF	CF card	Y	--	--	Y	--	--	--	--
TYPE_SD	SD card	--	--	--	--	--	Y	Y	Y

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error
SYS_PARAMERR	: Parameter error

3.58 SysEnableCardDetect

This function turns on the power to the virtual card by enabling the card detect terminals after removing or inserting the virtual card.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetCardDetect** function.

Calling Sequences

```
[C++]
DWORD SysEnableCardDetect(
    DWORD socket
)
```

```
[Visual Basic]
Public Shared Function SysEnableCardDetect( _
    ByVal socket As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysEnableCardDetect(
    Int32 socket
);
```

Parameters

socket

This parameter is for specifying a card socket for removing or inserting the virtual card. See **SysDisableCardDetect** function for setting values.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error
SYS_PARAMERR	: Parameter error

3.59 SysGetCardDetect

This function retrieves the card detect terminals status.

Calling Sequences

```
[C++]
DWORD SysGetCardDetect(
    DWORD socket
)
```

```
[Visual Basic]
Public Shared Function SysGetCardDetect( _
    ByVal socket As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetCardDetect(
    Int32 socket
);
```

Parameters

Socket

This parameter is for specifying a card socket for removing or inserting the virtual card. See **SysDisableCardDetect** function for setting values.

Return Values

TRUE	: Card detect terminals are set enabled.
FALSE	: Card detect terminals are set disabled.
FUNCTION_UNSUPPORTED	: Unsupported error
SYS_PARAMERR	: Parameter error

3.60 SysDisableWLAN

This function turns off the power to the integrated WLAN module (operable only on models with WLAN module integrated).

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetWLAN** function.

Calling Sequences

```
[C++]  
DWORD SysDisableWLAN( )
```

```
[Visual Basic]  
Public Shared Function SysDisableWLAN() As Int32
```

```
[C#]  
public static Int32 SysDisableWLAN()
```

Parameter

None

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.61 SysEnableWLAN

This function turns on the power to the integrated WLAN module (operable only on models with WLAN module integrated).

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetWLAN** function.

Calling Sequences

```
[C++]  
DWORD SysEnableWLAN( )
```

```
[Visual Basic]  
Public Shared Function SysEnableWLAN() As Int32
```

```
[C#]  
public static Int32 SysEnableWLAN()
```

Parameter

None

Return Values

TRUE	: Normal end
FUNCTION_UN SUPPORT	: Unsupported error

3.62 SysGetWLAN

This function retrieves the status of the power to the integrated WLAN module (operable only on models with WLAN module integrated).

Calling Sequences

```
[C++]  
DWORD SysGetWLAN( )
```

```
[Visual Basic]  
Public Shared Function SysGetWLAN() As Int32
```

```
[C#]  
public static Int32 SysGetWLAN()
```

Parameter

None

Return Values

TRUE	: Power to the WLAN module is on.
FALSE	: Power to the WLAN module is off.
FUNCTION_UNSUPPORTED	: Unsupported error

3.63 SysRenewPartition

This function carries out "Dismount" → "Mount" on the partition of storage so that the system can recognize the storage again.

In the Device Emulator, the function does not perform.

Calling Sequences

```
[C++]
DWORD SysRenewPartition(
    DWORD    dwStorageType
)
```

```
[Visual Basic]
Public Shared Function SysRenewPartition( _
    ByVal dwStorageType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysRenewPartition(
    Int32 dwStorageType
);
```

Parameter

dwStorageType

This parameter is for specifying a kind of memory selecting either of the values listed below.

STRAGE_SD	: Recognize SD memory again.
STRAGE_CF	: Recognize CF memory again.

Return Values

TRUE	: Power to the WLAN module is on.
FUNCTION_UN SUPPORT	: Unsupported error

Note:

This function supports SD memory only (current as of March 2009).

3.64 SysPlayBuzzer

This function turns on the integrated buzzer.

Calling Sequences

```
[C++]
DWORD SysPlayBuzzer(
    DWORD dwType,
    DWORD dwHelz,
    DWORD dwTime
)
```

```
[Visual Basic]
Public Shared Function SysPlayBuzzer( _
    ByVal dwType As Int32, _
    ByVal dwHelz As Int32, _
    ByVal dwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysPlayBuzzer(
    Int32 dwType,
    Int32 dwHelz,
    Int32 dwTime
);
```

Parameters

dwType

This parameter is for specifying a buzzer sound selecting one of the values in the table below.

Table 3.17 Setting values

Setting Value	Sound	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-800	IT-800
B_USERDEF	User defined	Y	Y	Y	Y	--	Y	Y	Y
B_TAP	Tap (2600Hz for 25 milliseconds)	Y	Y	--	Y	--	Y	Y	Y
B_CLICK	Key click (2800Hz for 50 milliseconds)	Y	Y	Y	Y	--	Y	Y	Y
B_ALARM	Alarm (3500Hz for 150 milliseconds)	Y	Y	Y	Y	--	Y	Y	Y
B_WARNING	Warning (3000Hz for 100 milliseconds)	Y	Y	Y	Y	--	Y	Y	Y
B_SCANEND	Scan complete (3300Hz for 75 milliseconds)	Y	Y	Y	Y	--	Y	Y	Y
B_CARDRED	Card read/write (2900Hz for 100 milliseconds)	--	--	--	--	--	--	--	--
B_WIREREAD	Wireless in call (3200Hz for 100 milliseconds)	--	--	--	--	--	--	--	--

dwHelz

This parameter is for specifying frequency of buzzer sound which is effect only when user defined sound is specified. BUZ_DEFAULT is used in cases other than BUZ_USERDEF.

dwTime

This parameter is for specifying a period of ON time for sounding the buzzer which is effect only when user defined sound is specified. BUZ_DEFAULT is used in cases other than BUZ_USERDEF.

Return Values

TRUE	: Normal end
FUNCTION_UNSupport	: Unsupported error
SYS_PARAMERR	: Parameter error

Note:

If either "all the sound types" is set to mute, or "a specific sound type set enabled to sound" is set to mute, the buzzer will not sound.

3.65 SysStopBuzzer

This function turns off the buzzer.

Calling Sequences

```
[C++]  
DWORD SysStopBuzzer( )
```

```
[Visual Basic]  
Public Shared Function SysStopBuzzer() As Int32
```

```
[C#]  
public static Int32 SysStopBuzzer()
```

Parameter

None

Return Values

TRUE	: Normal end
FUNCTION_UN SUPPORT	: Unsupported error

3.66 SysSetBuzzerVolume

This function sets up a sound volume for a specific buzzer sound type.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetBuzzerVolume** function.

Calling Sequences

```
[C++]
DWORD SysSetBuzzerVolume(
    DWORD dwType,
    DWORD dwBuzzerVolume
)
```

```
[Visual Basic]
Public Shared Function SysSetBuzzerVolume( _
    ByVal dwType As Int32, _
    ByVal dwBuzzerVolume As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetBuzzerVolume(
    Int32 dwType,
    Int32 dwBuzzerVolume
);
```

Parameters

dwType

This parameter is for specifying a buzzer sound type. See **SysPlayBuzzer** function for buzzer sound types.

dwBuzzerVolume

This parameter is for specifying a sound volume selecting one from the values listed below.

BUZZERVOLUME_MIN	: Minimum
BUZZERVOLUME_MID	: Medium (Default)
BUZZERVOLUME_MAX	: Maximum

Return Values

TRUE	: Normal end
SYS_PARAMERR	: Parameter error
FUNCTION_UNSupport	: Unsupported error

3.67 SysGetBuzzerVolume

This function retrieves the buzzer's sound volume.

Calling Sequences

```
[C++]
DWORD SysGetBuzzerVolume(
    DWORD dwType
)
```

```
[Visual Basic]
Public Shared Function SysGetBuzzerVolume( _
    ByVal dwType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBuzzerVolume(
    Int32 dwType
);
```

Parameters

dwType

This parameter is for specifying a buzzer' sound type. See **SysPlayBuzzer** function for sound types.

Return Values

BUZZERVOLUME_MIN	: Minimum
BUZZERVOLUME_MID	: Medium (Default)
BUZZERVOLUME_MAX	: Maximum
SYS_PARAMERR	: Parameter error
FUNCTION_UNSupport	: Unsupported error

3.68 SysSetBuzzerMute

This function sets up sound mute for all the buzzer's sound types and a specific sound type.

Calling Sequences

```
[C++]
DWORD SysSetBuzzerMute(
    DWORD dwType,
    BOOL bMute
)
```

```
[Visual Basic]
Public Shared Function SysSetBuzzerMute( _
    ByVal dwType As Int32, _
    ByVal bMute As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetBuzzerMute(
    Int32 dwType,
    Boolean bMute
);
```

Parameters

dwType

This parameter is for specifying a buzzer's sound type. See **SysPlayBuzzer** function for the sound types to specify.

bMute

This parameter is for specifying mute status selecting either of the values listed below.

TRUE	: Mute ON (the buzzer does not sound.)
FALSE	: Mute OFF (the buzzer sounds.) Default

Return Values

TRUE	: Normal end
SYS_PARAMERR	: Parameter error
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

If either "all the sound types" is set to mute, or "a specific sound type set enabled to sound" is set to mute, the buzzer will not sound.

3.69 SysGetBuzzerMute

This function retrieves all the sound volumes of buzzer sound types and individual mute for the buzzer.

Calling Sequences

```
[C++]
DWORD SysGetBuzzerMute(
    DWORD dwType
)
```

```
[Visual Basic]
Public Shared Function SysGetBuzzerMute( _
    ByVal dwType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBuzzerMute(
    Int32 dwType
);
```

Parameters

dwType

This parameter is for specifying a buzzer's sound type. See **SysPlayBuzzer** function for the sound types to specify.

Return Values

TRUE	: Mute ON (the buzzer does not sound.)
FALSE	: Mute OFF (the buzzer sounds.) Default
SYS_PARAMERR	: Parameter error
FUNCTION_UN SUPPORT	: Unsupported error

Note:

If either "all the sound types" is set to mute, or "a specific sound type set enabled to sound" is set to mute, the buzzer will not sound.

3.70 SysSetCPUMode

This function sets up the CPU frequency control.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetCPUMode** function.

Calling Sequences

```
[C++]
DWORD SysSetCPUMode(
    DWORD dwMode
)
```

```
[Visual Basic]
Public Shared Function SysSetCPUMode( _
    ByVal dwMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetCPUMode(
    Int32 dwMode
);
```

Parameters

dwMode

This parameter is for setting up a CPU frequency control mode and its frequency selecting one of the values in the table below.

Table 3.18 Setting values

Setting Value	Description	DT-X11	IT-600	DT-X7	DT-X30	IT-3100	IT-800	IT-300	DT-X8
CPUMODE_LOW	Low speed (200MHz)	Y	Y	Y	Y	Y	Y	Y	Y
CPUMODE_MIDDLE	Normal (300MHz)	-	Y	Y	Y	-	Y	Y	Y
CPUMODE_HIGH	High speed (400MHz)	Y	Y	Y	Y	Y	Y	Y	Y
CPUMODE_AUTO	Auto power save	-	Y	Y	Y	-	Y	Y	Y

Return Values

TRUE : Normal end
FUNCTION_UNSUPPORTED : Unsupported error

3.71 SysGetCPUMode

This function retrieves the CPU frequency control.

Calling Sequences

```
[C++]
DWORD SysGetCPUMode(
    DWORD *pdwMode
)
```

```
[Visual Basic]
Public Shared Function SysGetCPUMode( _
    ByRef pdwMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetCPUMode(
    ref Int32 pdwMode
);
```

Parameters

pdwMode

This parameter is for retrieving the CPU frequency control mode. See **SysSetCPUMode** function for the control modes.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.72 SysSetDefaultCPUMode

This function resumes the CPU speed setting to the factory-default.

In the Device Emulator, the function does not perform any.

Calling Sequences

```
[C++]  
DWORD SysSetDefaultCPUMode( )
```

```
[Visual Basic]  
Public Shared Function SysSetDefaultCPUMode() As Int32
```

```
[C#]  
public static Int32 SysSetDefaultCPUMode()
```

Parameters

None

Return Values

TRUE	: Normal end
FUNCTION_UNSupport	: Unsupported error

3.73 SysSet180Rotate

This function flips the display screen 180 degree.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGet180Rotate** function.

Calling Sequences

```
[C++]
DWORD SysSet180Rotate(
    BOOL bRotate
)
```

```
[Visual Basic]
Public Shared Function SysSet180Rotate( _
    ByVal bRotate As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSet180Rotate(
    Boolean bRotate
);
```

Parameters

bRotate

This parameter is for specifying a display screen flip mode selecting either of the values listed below.

TRUE	: Flip the screen 180 degree (reverse)
FALSE	: Flip the screen 0 degree (ordinary)

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.74 SysGet180Rotate

This function retrieves the status of display screen flipped.

Calling Sequences

```
[C++]  
DWORD SysGet180Rotate( )
```

```
[Visual Basic]  
Public Shared Function SysGet180Rotate() As Int32
```

```
[C#]  
public static Int32 SysGet180Rotate()
```

Parameter

None

Return Values

TRUE	: Flip the screen 180 degree (reverse)
FALSE	: Flip the screen 0 degree (ordinary)
FUNCTION_UN SUPPORT	: Unsupported error

3.75 SysSetBLBattery

This function sets up a brightness of the backlight while the operation power is provided by the installed battery pack.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetBLBattery** function.

Calling Sequences

```
[C++]
DWORD SysSetBLBattery(
    DWORD setting
)
```

```
[Visual Basic]
Public Shared Function SysSetBLBattery( _
    ByVal setting As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetBLBattery(
    Int32 setting
);
```

Parameters

setting

This parameter is for setting a brightness of the backlight. The backlight is turned off if "0" is set in this parameter. The defaults for the respective models are as follows.

DT-X11	: 6
IT-600	: 6
DT-X7	: 6
DT-X30	: 6
IT-3100	: 6
IT-800	: 6
IT-300	: 6
DT-X8	: 6

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

For the maximum value to set in the *setting* parameter, call **SysGetBLMaximum** function to retrieve it.

3.76 SysGetBLBattery

This function retrieves the brightness value of the backlight effect while the operation power is provided by the installed battery pack.

Calling Sequences

```
[C++]
DWORD SysGetBLBattery(
    DWORD *setting
)
```

```
[Visual Basic]
Public Shared Function SysGetBLBattery( _
    ByRef setting As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBLBattery(
    ref Int32 setting
);
```

Parameters

setting

This parameter is for retrieving the brightness value of the backlight. See **SysSetBLBattery** function for the values to retrieve.

Return Values

TRUE	: Normal end
FUNCTION_UNSupport	: Unsupported error

3.77 SysSetBLExpower

This function sets up a brightness of the backlight effect while the operation power is provided by the dedicated AC Adaptor via cradle.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetBLExpower** function.

Calling Sequences

```
[C++]
DWORD SysSetBLExpower(
    DWORD setting
)
```

```
[Visual Basic]
Public Shared Function SysSetBLExpower( _
    ByVal setting As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetBLExpower(
    Int32 setting
);
```

Parameters

setting

This parameter is for specifying a brightness of the backlight. The backlight is turned off if "0" is set in this parameter. The defaults for the respective models are as follows.

DT-X11	: 8
IT-600	: 8
DT-X7	: 8
DT-X30	: 8
IT-3100	: 8
IT-800	: 8
IT-300	: 8
DT-X8	: 8

For the maximum value to set in the *setting* parameter, call **SysGetBLMaximum** function to retrieve it.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.78 SysGetBLExpower

This function retrieves the brightness value of the backlight effect while the operation power is provided by the dedicated AC Adaptor via cradle.

Calling Sequences

```
[C++]
DWORD SysGetBLExpower(
    DWORD *setting
)
```

```
[Visual Basic]
Public Shared Function SysGetBLExpower( _
    ByRef setting As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBLExpower(
    ref Int32 setting
);
```

Parameters

setting

This parameter is for retrieving the brightness value of the backlight. See **SysSetBLExpower** function for the values to retrieve.

Return Values

TRUE	: Normal end
FUNCTION_UN SUPPORT	: Unsupported error

3.79 SysGetBLMaximum

This function retrieves the maximum brightness values of the backlight for the operation powers provided by both the installed battery and the dedicated AC Adaptor via cradle.

Calling Sequences

```
[C++]
DWORD SysGetBLMaximum(
    DWORD *BAsetting,
    DWORD *ACsetting
)
```

```
[Visual Basic]
Public Shared Function SysGetBLMaximum( _
    ByRef BAsetting As Int32, _
    ByRef ACsetting As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBLMaximum(
    ref Int32 BAsetting,
    ref Int32 ACsetting
);
```

Parameters

BAsetting

This parameter is for retrieving the maximum brightness value of the backlight effect while the operation power is provided by the installed battery.

ACsetting

This parameter is for retrieving the maximum brightness value of the backlight effect while the operation power is provided by the dedicated AC Adaptor via cradle.

Return Values

TRUE	: Normal end
FUNCTION_UN SUPPORT	: Unsupported error

3.80 SysSetBLOffTime

This function sets up a time period for continuously pressing the Power key until when the backlight is turned off.

Calling Sequences

```
[C++]  
DWORD SysSetBLOffTime(  
    DWORD dwTime  
)
```

```
[Visual Basic]  
Public Shared Function SysSetBLOffTime( _  
    ByVal dwTime As Int32 _  
) As Int32
```

```
[C#]  
public static Int32 SysSetBLOffTime(  
    Int32 dwTime  
);
```

Parameters

dwTime

This parameter is for specifying a time period for continuously pressing the Power key. The default is 2,000 milliseconds.

Return Values

TRUE	: Normal end
FUNCTION_UNSupport	: Unsupported error

3.81 SysGetBLOffTime

This function retrieves the time period set for continuously pressing the Power key until when the backlight is turned off.

Calling Sequences

```
[C++]
DWORD SysGetBLOffTime(
    DWORD *pdwTime
)
```

```
[Visual Basic]
Public Shared Function SysGetBLOffTime( _
    ByRef pdwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBLOffTime(
    ref Int32 pdwTime
);
```

Parameters

pdwTime

This parameter is for retrieving the time period set for continuously pressing the Power key.
See **SysSetBLOffTime** function for the values to retrieve.

Return Values

TRUE	: Normal end
FUNCTION_UN SUPPORT	: Unsupported error

3.82 SysPlayVibrator

This function turns on the vibrator.

Calling Sequences

```
[C++]
DWORD SysPlayVibrator(
    DWORD dwType,
    DWORD dwCount,
    DWORD dwOnTime,
    DWORD dwOFFTime
)
```

```
[Visual Basic]
Public Shared Function SysPlayVibrator( _
    ByVal dwType As Int32, _
    ByVal dwCount As Int32, _
    ByVal dwOnTime As Int32, _
    ByVal dwOFFTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysPlayVibrator(
    Int32 dwType,
    Int32 dwCount,
    Int32 dwOnTime,
    Int32 dwOFFTime
);
```

Parameters

dwType

This parameter is for specifying a vibration pattern selecting one of the listed patterns in the table below.

Table 3.19 Setting values

Setting Value	Event	Time	ON Time	OFF Time
B_ALARM	Alarm	1	1000	1000
B_WARNING	Warning	1	1000	1000
B_SCANEND	Scanning complete	1	1000	1000
B_CARDREAD	Card read/write	1	1000	1000
B_WIREREAD	Wireless in call	1	1000	1000
B_USERDEF	User defined	-	-	-

dwCount

This parameter is for specifying the number of times in the range of 1 to 20 (effect only for the user defined mode) to turn on the vibrator. VIBRATOR_DEFAULT is used in other cases than B_USERDEF.

dwOnTime

This parameter is for specifying a time period in the range of 0 to 16,000 milliseconds for tuning on the vibrator (effect only for the user defined mode). VIBRATOR_DEFAULT is used in cases other than B_USERDEF.

dwOFFTime

This parameter is specifying for a time period in the range of 0 to 1,000 milliseconds for turning off the vibrator (effect only for the user defined mode). VIBRATOR_DEFAULT is used in cases other than B_USERDEF.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

If either "all the vibration types" is set to mute, or "a specific vibration type set enabled to vibrate" is set to mute, the vibrator will not be turned on.

3.83 SysStopVibrator

This function stops turning on the vibrator.

Calling Sequences

```
[C++]  
DWORD SysStopVibrator( )
```

```
[Visual Basic]  
Public Shared Function SysStopVibrator() As Int32
```

```
[C#]  
public static Int32 SysStopVibrator()
```

Parameters

None

Return Values

TRUE	: Normal end
FUNCTION_UN SUPPORT	: Unsupported error

3.84 SysSetVibratorMute

This function sets up all the vibration modes and individual mute for the vibrator.

Calling Sequences

```
[C++]
DWORD SysSetVibratorMute(
    DWORD dwType,
    BOOL bMute
)
```

```
[Visual Basic]
Public Shared Function SysSetVibratorMute( _
    ByVal dwType As Int32, _
    ByVal bMute As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetVibratorMute(
    Int32 dwType,
    Boolean bMute
);
```

Parameters

dwType

This parameter is for specifying vibration pattern selecting one of the values listed below.

B_ALARM	: Alarm
B_WARNING	: Warning
B_SCANEND	: Scanning complete
B_CARDREAD	: Card read/write
B_WIREREAD	: Wireless in call
B_USERDEF	: User defined
B_ALL	: All sounds

bMute

This parameter is for specifying the mute selecting either of the values listed below.

TRUE	: Mute ON (the vibrator will not be turned on.)
FALSE	: Mute OFF (the vibrator will be turned on.) Default

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

If either all the vibration types are set to mute, or a specific vibration type set enabled to vibrate is set to mute, the vibrator will not be turned on.

3.85 SysGetVibratorMute

This function retrieves all the vibration modes and individual mute for the vibrator.

Calling Sequences

```
[C++]
DWORD SysGetVibratorMute(
    DWORD dwType
)
```

```
[Visual Basic]
Public Shared Function SysGetVibratorMute( _
    ByVal dwType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetVibratorMute(
    Int32 dwType
);
```

Parameters

dwType

This parameter is for specifying a vibration mode. See **SysSetVibratorMute** function for the vibration modes.

Return Values

TRUE	: Mute ON (the vibrator does not turn on.)
FALSE	: Mute OFF (the vibrator turns on.) Default
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

If either all the vibration modes are set to mute, or a specific vibration mode set enabled to vibrate is set to mute, the vibrator will not be turned on.

3.86 SysSetFnKeyLock

This function sets up "Enable" or "Disable" for the Fn key to operate.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetFnKeyLock** function.

Calling Sequences

```
[C++]
DWORD SysSetFnKeyLock(
    BOOL bFnKey
)
```

```
[Visual Basic]
Public Shared Function SysSetFnKeyLock( _
    ByVal bFnKey As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetFnKeyLock(
    Boolean bFnKey
);
```

Parameters

bFnKey

This parameter is for specifying "Enable" or "Disable" for the Fn key selecting either of the values listed below.

TRUE	: Disable the Fn key.
FALSE	: Enable the Fn key (Default)

Return Values

TRUE	: Normal end
FUNCTION_UN SUPPORT	: Unsupported error

3.87 SysGetFnKeyLock

This function retrieves the status of "Enable" or "Disable" for the Fn key to operate.

Calling Sequences

```
[C++]  
DWORD SysGetFnKeyLock( )
```

```
[Visual Basic]  
Public Shared Function SysGetFnKeyLock() As Int32
```

```
[C#]  
public static Int32 SysGetFnKeyLock()
```

Parameter

None

Return Values

TRUE	: Fn key disabled.
FALSE	: Fn key enabled.
FUNCTION_UNSupport	: Unsupported error

3.88 SysSetOneKeyLock

This function sets up "Enable" or "Disable" for lock (not allow any key to be input) on each key except the Trigger keys, Multi key, and Power key.

Calling Sequences

```
[C++]
DWORD SysSetOneKeyLock(
    DWORD dwLockBits
)
```

```
[Visual Basic]
Public Shared Function SysSetOneKeyLock( _
    ByVal dwLockBits As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetOneKeyLock(
    Int32 dwLockBits
);
```

Parameters

dwLockBits

Specify a sum of values in logical OR for all the keys that are to be locked.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

Note:

Performing a reset on the terminal disables all "Disable" settings, and all key inputs become possible. If KEY_LOCK_ALL is set in the parameter, all key inputs except with Trigger keys, Multi key, and Multi key are disabled. Only way to reset this condition is performing a reset.

3.89 SysGetOneKeyLock

This function retrieves the status of "Enable" or "Disable" for individual key lock of all the keys except the Trigger keys, Multi key, and Power key.

Calling Sequences

```
[C++]  
DWORD SysGetOneKeyLock( )
```

```
[Visual Basic]  
Public Shared Function SysGetOneKeyLock() As Int32
```

```
[C#]  
public static Int32 SysGetOneKeyLock()
```

Parameter

None

Return Values

Value indicating status of "Enable" or "Disable" for individual key lock. See

SysSetOneKeyLock function for the values to retrieve.

Otherwise

FUNCTION_UNSUPPORTED : Unsupported error

3.90 SysSetAllKeyLock

This function sets up "Enable" or "Disable" for lock on all the keys except the Trigger keys, Multi key, and Power key.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetAllKeyLock** function.

Calling Sequences

```
[C++]
DWORD SysSetAllKeyLock(
    BOOL bKeyLock
)
```

```
[Visual Basic]
Public Shared Function SysSetAllKeyLock( _
    ByVal bKeyLock As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetAllKeyLock(
    Boolean bKeyLock
);
```

Parameters

bKeyLock

This parameter is for specifying "Enable" or "Disable" for key lock on all the keys.

Table 3.20 Setting values

TRUE	All keys are disabled except Trigger keys and Power key.
FALSE	All keys are enabled (Unlocked status). (Default)

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.91 SysGetAllKeyLock

This function retrieves the status of "Enable" or "Disable" for key lock on all the keys.

Calling Sequences

```
[C++]  
DWORD SysGetAllKeyLock( )
```

```
[Visual Basic]  
Public Shared Function SysGetAllKeyLock() As Int32
```

```
[C#]  
public static Int32 SysGetAllKeyLock()
```

Parameter

None

Return Values

Value indicating the status of "Enable" or "Disable" for key lock. See **SysSetAllKeyLock** function for the values.

Otherwise

FUNCTION_UN SUPPORT : Unsupported error

3.92 SysSetInputMode

This function sets up a key input change mode.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetInputMode** function.

Calling Sequences

```
[C++]
DWORD SysSetInputMode(
    DWORD dwInputMode
)
```

```
[Visual Basic]
Public Shared Function SysSetInputMode( _
    ByVal dwInputMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetInputMode(
    Int32 dwInputMode
);
```

Parameters

dwInputMode

This parameter is for specifying a key input change mode selecting one of the values listed below.

INPUT_NORMAL	: Ordinary input (Default)	Switch-able
INPUT_LOCK_NUM	: Fixed to numeric input	Not switch-able
INPUT_LOCK_ALPHA	: Fixed to alphanumeric (uppercase)	Not switch-able
INPUT_LOCK_ALPHAS	: Fixed to alphanumeric (lowercase)	Not switch-able
INPUT_LOCK_PHONE	: Fixed to Phone (for IT-600, DT-X7, DT-X30, IT-800, IT-300, DT-X8)	Not switch-able

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.93 SysGetInputMode

This function retrieves the key input change mode.

Calling Sequences

```
[C++]  
DWORD SysGetInputMode( )
```

```
[Visual Basic]  
Public Shared Function SysGetInputMode() As Int32
```

```
[C#]  
public static Int32 SysGetInputMode()
```

Parameter

None

Return Values

Values indicating the key input change modes.

INPUT_NORMAL	: Ordinary input
INPUT_LOCK_NUM	: Fixed to numeric
INPUT_LOCK_ALPHA	: Fixed to alphanumeric (uppercase)
INPUT_LOCK_ALPHAS	: Fixed to alphanumeric (lowercase)
INPUT_LOCK_PHONE	: Fixed to phone (for IT-600, DT-X7, DT-X30, IT-800, IT-300, and DT-X8)

Otherwise

FUNCTION_UNSupport	: Unsupported error
--------------------	---------------------

3.94 SysSetNormalUserDefineKey

This function sets up key codes for ordinary key input mode.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetNormalUserDefineKey** function.

Calling Sequences

```
[C++]
DWORD SysSetNormalUserDefineKey(
    DWORD dwKeyID,
    DWORD KeySetBuff[16]
)
```

```
[Visual Basic]
Public Shared Function SysSetNormalUserDefineKey( _
    ByVal dwKeyId As Int32, _
    ByVal KeySetBuff As Int32() _
) As Int32
```

```
[C#]
public static Int32 SysSetNormalUserDefineKey(
    Int32 dwKeyId,
    Int32[] KeySetBuff
);
```

Parameters

dwKeyID

This parameter is for specifying key IDs using the values in the table below.

Table 3.21 Setting values

Setting Value	Description	DT-X11	IT-600	DT-X7	DT-X30	DT-X30R-50	IT-3100	IT-800	IT-300	DT-X8
KEYID_LEFTTRIGGER	Left Trigger key Left Up key (IT-800xx-05 models)	Y	Y	Y	Y	Y	Y	Y	--	Y
KEYID_RIGHTTRIGGER	Right Trigger key Right Enter key (IT-800xx-05 models)	Y	Y	Y	Y	Y	Y	Y	--	Y
KEYID_LEFT	Left key	Y	Y	--	Y	Y	Y	Y	Y	Y
KEYID_RIGHT	Right key	Y	Y	--	Y	Y	Y	Y	Y	Y
KEYID_CLEAR	Clear key	Y	Y	Y	Y	Y	Y	Y	Y	Y
KEYID_ENTER	Enter key	Y	Y	Y	Y	Y	Y	Y	Y	Y
KEYID_0	0 key	Y	Y	Y	Y	Y	Y	Y	Y	Y

KEYID_1	1 key	Y	Y	Y	Y	Y	Y	Y	Y	Y
KEYID_2	2 key	Y	Y	Y	Y	Y	Y	Y	Y	Y
KEYID_3	3 key	Y	Y	Y	Y	Y	Y	Y	Y	Y
KEYID_4	4 key	Y	Y	Y	Y	Y	Y	Y	Y	Y
KEYID_5	5 key	Y	Y	Y	Y	Y	Y	Y	Y	Y
KEYID_6	6 key	Y	Y	Y	Y	Y	Y	Y	Y	Y
KEYID_7	7 key	Y	Y	Y	Y	Y	Y	Y	Y	Y
KEYID_8	8 key	Y	Y	Y	Y	Y	Y	Y	Y	Y
KEYID_9	9 key	Y	Y	Y	Y	Y	Y	Y	Y	Y
KEYID_MULTI	Multi key	Y	--	--	--	--	--	--	--	--
KEYID_FUNCTION	Function key	Y	--	--	--	--	--	--	--	--
KEYID_UP	Up key	Y	Y	Y	Y	Y	--	Y	Y	Y
KEYID_DOWN	Down key	Y	Y	Y	Y	Y	--	Y	Y	Y
KEYID_F1	F1 key	Y	--	Y	Y	Y	--	Y	Y	Y
KEYID_F2	F2 key	Y	--	Y	Y	Y	--	Y	Y	Y
KEYID_F3	F3 key	Y	--	Y	Y	Y	--	Y	Y	Y
KEYID_F4	F4 key	Y	--	Y	Y	Y	--	Y	Y	Y
KEYID_F5	F5 key	--	--	Y	Y	Y	--	--	--	Y
KEYID_F6	F6 key	--	--	Y	Y	Y	--	--	--	Y
KEYID_F7	F7 key	--	--	Y	Y	Y	--	--	--	Y
KEYID_F8	F8 key	--	--	Y	Y	Y	--	--	--	Y
KEYID_F9	F9 key	--	--	--	--	Y	--	--	--	--
KEYID_F10	F10 key	--	--	--	--	Y	--	--	--	--
KEYID_F11	F11 key	--	--	--	--	Y	--	--	--	--
KEYID_F12	F12 key	--	--	--	--	Y	--	--	--	--
KEYID_000	000 key	--	--	--	--	--	Y	--	--	--
KEYID_00	00 key	--	--	--	--	--	Y	--	--	--
KEYID_HYPHEN	Hyphen	--	Y	--	Y	Y	Y	Y	Y	--
KEYID_BS	BS key	--	--	--	--	--	Y	--	--	--
KEYID_PERIOD	Period (.) key	--	--	Y	--	--	--	--	--	Y
KEYID_INPUTMODE	Character key	--	Y	Y	Y	Y	--	Y	Y	Y
KEYID_CENTERTRIGGER	Center Trigger key Center Enter key (IT-800xx-05 models)	--	Y	Y	Y	Y	--	Y	Y	Y
KEYID_GUNTRIGGER	Trigger grip key	--	Y	--	--	--	--	Y	--	--
KEYID_LEFTDOWN	Left Down key (IT-800xx-05 models)	--	--	--	--	--	--	Y	--	--
KEYID_MENU	MENU key	--	--	--	--	--	--	--	Y	--
KEYID_LMULTI	L-Multi key	--	--	--	--	--	--	--	--	Y
KEYID_RMULTI	R-Multi key	--	--	--	--	--	--	--	--	Y

KeySetBuff[16]

This parameter is for specifying key codes.

See notes below for "Virtual Key Code and Option Flag".

Return Values

TRUE	: Normal end
FALSE	: Internal error
SYS_PARAMERR	: Parameter error
FUNCTION_UNSupport	: Unsupported error

Note:

Virtual Key Code and Option Flag

Specify any combination of the following values except for the combination of VF_PAGE and VF_NO_FIX_PAGE.

Table 3.22 Virtual key code and option flag

Definition	VF_NO_KEY_UP			
	Undesignated		Designated	
	When key down	When key up	When key down	When key up
Not designated	Virtual key code down	Inaction		
Only virtual key code designated	Virtual key code up			
VF_PAGE	Virtual key code down	Inaction	Virtual key code down	Virtual key code up
Designates Changeover key operation. Designates as an initial key.	Virtual key code up			
VF_NO_FIX_PAGE	Virtual key code down	Inaction	Virtual key code down	Virtual key code up
Does not confirm Changeover key. Designates to initial key. If VF_PAGE is concurrently designated, VF_NO_FIX_PAGE will be ignored. Normally, designates as a special key that does not output characters like a program key.	Virtual key code up			
VF_KANA	VK_KANA down	Inaction	VK_KANA down	Virtual key code up
Sends VK_KANA. Designates this if Japanese kana character is to be output	VK_KANA up Virtual key code down Virtual key code up VK_KANA down VK_KANA up		VK_KANA up Virtual key code down	VK_KANA down VK_KANA up
VF_SHIFT	VK_SHIFT down	Inaction	VK_SHIFT down	Virtual key code up
Combines VK_SHIFT and virtual key code. (Press SHIFT key simultaneously)	Virtual key code down Virtual key code up VK_SHIFT up		Virtual key code down	VK_SHIFT up
VF_CONTROL	VK_CONTROL down	Inaction	VK_CONTROL down	Virtual key code up
Combines VK_CONTROL and virtual key code. (Press CTRL key simultaneously)	Virtual key code down Virtual key code up VK_CONTROL up		Virtual key code down	VK_CONTROL up

Continue.

VF_MENU	VK_MENU down	Inaction	VK_MENU down	Virtual key code up
Combines VK_MENU and virtual key code. (Press ALT key simultaneously)	Virtual key code down Virtual key code up VK_MENU up		Virtual key code down	VK_MENU up
VF_NO_KEY_UP			Virtual key code down	Virtual key code up
Virtual key code will not be up when key is down. Virtual key code will be up when key is up. Normally, this is designated when key operation is repeated.				
VF_NOP	Virtual key code down	Inaction	Virtual key code down	Virtual key code up
Sends disable key code (0x00). Normally, combined with VF_NO_KEY_UP to prevent repeat of virtual key code, but is designated for virtual key code up when key is up. With an application like program key, the GetAsyncKeyState function is used, and it is designated for monitoring of key down and key up.	Disable key code down Disable key code up Virtual key code up		Disable key code down Disable key code up	
VF_NOP_CLICK	Virtual key code down	Inaction	Virtual key code down	Virtual key code up
Sends disable key code (0x00) sounds key click sound. Normally not used.	Disable key code down Disable key code up Virtual key code up		Disable key code down Disable key code up	
KEYBD_DEVICE_SILENT	Virtual key code down	Inaction	Virtual key code down	Virtual key code up
Will not sound initial key click sound or repeat key click sound.	Virtual key code up			
KEYBD_DEVICE_SILENT_REPEAT	Virtual key code down	Inaction	Virtual key code down	Virtual key code up
Will not sound repeat key click sound. Normally combined with VF_NO_KEY_UP.	Virtual key code up			

- The following values are specified for the virtual key code and option flag of the established key.

Table 3.23 Virtual key code and option flag (established key)

Key ID	UserDefineKeyBuf[16]
KEYID_000 (000key)	VK_0,VK_0 KEYBD_DEVICE_SILENT,VK_0 KEYBD_DEVICE_SILENT, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_00 (English OS 00key)	VK_0, VK_0 KEYBD_DEVICE_SILENT, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_0 (0 key)	VK_0 KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_1 (1 key)	VK_1 KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_2 (2 key)	VK_2 KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_3 (3 key)	VK_3 KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_4 (4 key)	VK_4 KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_5 (5 key)	VK_5 KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_6 (6 key)	VK_6 KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_7 (7 key)	VK_7 KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_8 (8 key)	VK_8 KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_9 (9 key)	VK_9 KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_PERIOD (Period (.)key)	VK_PERIOD, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_HYPHEN (Hyphen (-)key)	VK_HYPHEN, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_C (Cancel key)	VK_ESCAPE, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_BS (Backspace key)	VK_BACK, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_ENT (Input key)	VK_RETURN, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_LEFT (Left (<) key)	VK_LEFT KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_RIGHT (Right (>) key)	VK_RIGHT KEYBD_DEVICE_SILENT_REPEAT VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_LTR (Left program key)	VF_NO_FIX_PAGE VK_F24 KEYBD_DEVICE_SILENT VF_NO_KEY_UP VF_NOP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_RTR (Right program key)	VF_NO_FIX_PAGE VK_F21 KEYBD_DEVICE_SILENT VF_NO_KEY_UP VF_NOP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

- About VF_PAGE

VF_PAGE | VK_0, VK_1, VK_2, VK_3, VF_KANA | WCH_A, VF_KANA | WCH_I, VF_KANA | WCH_U, VF_KANA | WCH_E, VK_A, VK_B, VK_C, VK_D, VK_SHIFT | VK_A, VK_SHIFT | VK_B, VK_SHIFT | VK_C, VK_SHIFT | VK_D

When VF_PAGE is designated to UserDefineKeyBuf[16], and the above is established, the Changeover key will become operable, and each time it is pressed, the output will navigate to 0→1→2→3→4→ ア → イ → ウ → エ →a→b→c→d→A→B→C→D→repeat.

If VF_PAGE for the initial key (VF_PAGE | VK_0) is deleted, and KEYBD_DEVICE_SILENT is combined with all keys from the second key (VK_1) onward, the key click sound will sound once, and "01234 アイウエ abcdABCD" will be output once.

Note:

Once set in the usual way, calling **SysSetUserDefineKeyState**(TRUE) will notify the keyboard driver and the user define key can be enabled.

Even if the user define key is in the enabled state, it will not become enabled until either post-setting reset is carried out or **SysSetUserDefineKeyState**(TRUE) is called.

Key Codes (For DT-X7 series and DT-X30 Windows CE series):

Table 3.24

Name	Font	Code
INVERT_EXCLAMATION	¡	0x003A
CENT	¢	0x003B
POUND	£	0x003C
CURRENCY	¤	0x003D
YEN	¥	0x003E
BROKEN_BAR		0x00A6
SECTION	§	0x00A7
DIAERESIS	¨	0x00A8
COPYRIGHT	©	0x00A9
FEMININE_ORDINAL	ª	0x00AA
LEFT_DOUBLE_ANGLE	«	0x00AB
NOT	¬	0x00AC
SOFT_HYPHEN		0x00AD
REGISTER	®	0x00AE
MACRON	—	0x00AF
DEGREE	°	0x00B0
PLUS_MINUS	±	0x00B1
SUPERSCRIP2	²	0x00B2
SUPERSCRIP3	³	0x00B3
ACUTE_ACCENT	´	0x00B4
MICRO	μ	0x00B5
PILCROW	¶	0x00B6
MIDDLE_DOT	·	0x00B7
CEDILLA	¸	0x00B8
SUPERSCRIP1	¹	0x00B9
MASCULINE_ORDINAL	º	0x0092
RIGHT_DOUBLE_ANGLE	»	0x0093
QUARTER	¼	0x0094
HALF	½	0x0095
3QUARTER	¾	0x0096
INVERT_QUESTION	¿	0x0097
EURO	€	0x0098
GRAVE_A	À	0x0099
ACUTE_A	Á	0x00C1
CIRCUMFLEX_A	Â	0x00C2

Continue.

TILDE_A	Ã	0x00C3
DIAERESIS_A	Ä	0x00C4
RING_ABOVE_A	Å	0x00C5
AE	Æ	0x00C6
CEDILLA_C	Ç	0x00C7
GRAVE_E	È	0x00C8
ACUTE_E	É	0x00C9
CIRCUMFLEX_E	Ê	0x00CA
DIAERESIS_E	Ë	0x00CB
GRAVE_I	Ì	0x00CC
ACUTE_I	Í	0x00CD
CIRCUMFLEX_I	Î	0x00CE
DIAERESIS_I	Ï	0x00CF
ETH	Ð	0x00D0
TILDE_N	Ñ	0x00D1
GRAVE_O	Ò	0x00D2
ACUTE_O	Ó	0x00D3
CIRCUMFLEX_O	Ô	0x00D4
TILDE_O	Õ	0x00D5
DIAERESIS_O	Ö	0x00D6
MULTIPLICATION	×	0x00D7
STROKE_O	Ø	0x00D8
GRAVE_U	Ù	0x00D9
ACUTE_U	Ú	0x00DA
CIRCUMFLEX_U	Û	0x009A
DIAERESIS_U	Ü	0x009B
ACUTE_Y	Ý	0x009C
THORN	þ	0x009D
SHARP_S	ß	0x009E
GRAVE_A_SMALL	à	0x00E0
ACUTE_A_SMALL	á	0x00E1
CIRCUMFLEX_A_SMALL	â	0x00E2
TILDE_A_SMALL	ã	0x00E3
DIAERESIS_A_SMALL	ä	0x00E4
RING_ABOVE_A_SMALL	å	0x003F
AE_SMALL	æ	0x00E6
CEDILLA_C_SMALL	ç	0x00E7
GRAVE_E_SMALL	è	0x00E8

Continue.

ACUTE_E_SMALL	é	0x00E9
CIRCUMFLEX_E_SMALL	ê	0x00EA
DIAERESIS_E_SMALL	ë	0x00EB
GRAVE_I_SMALL	ì	0x00EC
ACUTE_I_SMALL	í	0x00ED
CIRCUMFLEX_I_SMALL	î	0x00EE
DIAERESIS_I_SMALL	ï	0x00EF
ETH_SMALL	ð	0x00F0
TILDE_N_SMALL	ñ	0x00F1
GRAVE_O_SMALL	ò	0x00F2
ACUTE_O_SMALL	ó	0x00F3
CIRCUMFLEX_O_SMALL	ô	0x00F4
TILDE_O_SMALL	õ	0x00F5
DIAERESIS_O_SMALL	ö	0x00F6
DIVISION	÷	0x00F7
STROKE_O_SMALL	ø	0x00F8
GRAVE_U_SMALL	ù	0x00F9
ACUTE_U_SMALL	ú	0x00FA
CIRCUMFLEX_U_SMALL	û	0x00FB
DIAERESIS_U_SMALL	ü	0x00FC
ACUTE_Y_SMALL	ý	0x00FD
THORN_SMALL	þ	0x009F
DIAERESIS_Y_SMALL	ÿ	0x00FF

Name	Font	Code
MACRON_A	Ā	0x083A
MACRON_A_SMALL	ā	0x083B
BREVE_A	Ă	0x083C
BREVE_A_SMALL	ă	0x083D
OGONEK_A	Ą	0x083E
OGONEK_A_SMALL	ą	0x08A6
ACUTE_C	Ć	0x08A7
ACUTE_C_SMALL	ć	0x08A8
CIRCUMFLEX_C	Ĉ	0x08A9
CIRCUMFLEX_C_SMALL	ĉ	0x08AA
DOT_ABOVE_C	Č	0x08AB
DOT_ABOVE_C_SMALL	č	0x08AC
CARON_C	Č	0x08AD

Continue.

CARON_C_SMALL	č	0x08AE
CARON_D	Ď	0x08AF
CARON_D_SMALL	d'	0x08B0
STROKE_D	Đ	0x08B1
STROKE_D_SMALL	đ	0x08B2
MACRON_E	Ē	0x08B3
MACRON_E_SMALL	ē	0x08B4
BREVE_E	Ė	0x08B5
BREVE_E_SMALL	ė	0x08B6
DOT_ABOVE_E	Ê	0x08B7
DOT_ABOVE_E_SMALL	ê	0x08B8
OGONEK_E	Ę	0x08B9
OGONEK_E_SMALL	ę	0x0892
CARON_E	Ě	0x0893
CARON_E_SMALL	ě	0x0894
CIRCUMFLEX_G	Ĝ	0x0895
CIRCUMFLEX_G_SMALL	ĝ	0x0896
BREVE_G	Ğ	0x0897
BREVE_G_SMALL	ğ	0x0898
DOT_ABOVE_G	Ġ	0x0899
DOT_ABOVE_G_SMALL	ġ	0x08C1
CEDILLA_G	Ģ	0x08C2
CEDILLA_G_SMALL	ģ	0x08C3
CIRCUMFLEX_H	Ĥ	0x08C4
CIRCUMFLEX_H_SMALL	ĥ	0x08C5
STROKE_H	Ħ	0x08C6
STROKE_H_SMALL	ħ	0x08C7
TILDE_I	Ĩ	0x08C8
TILDE_I_SMALL	ĩ	0x08C9
MACRON_I	Ī	0x08CA
MACRON_I_SMALL	ī	0x08CB
BREVE_I	İ	0x08CC
BREVE_I_SMALL	ı	0x08CD
OGONEK_I	Į	0x08CE
OGONEK_I_SMALL	į	0x08CF
DOT_ABOVE_I	İ	0x08D0
DOTLESS_I_SMALL	ı	0x08D1
LIGATURE_IJ	IJ	0x08D2

Continue.

LIGATURE_IJ_SMALL	ij	0x08D3
CIRCUMFLEX_J	Ĵ	0x08D4
CIRCUMFLEX_J_SMALL	ĵ	0x08D5
CEDILLA_K	Ƙ	0x08D6
CEDILLA_K_SMALL	ƙ	0x08D7
KRA_SMALL	κ	0x08D8
ACUTE_L	Ł	0x08D9
ACUTE_L_SMALL	ł	0x08DA
CEDILLA_L	ƚ	0x089A
CEDILLA_L_SMALL	ƚ	0x089B
CARON_L	Ľ	0x089C
CARON_L_SMALL	ĺ	0x089D
MIDDLE_DOT_L	Ĺ	0x089E
MIDDLE_DOT_L_SMALL	ĺ	0x08E0
STROKE_L	Ł	0x08E1
STROKE_L_SMALL	ł	0x08E2
ACUTE_N	Ń	0x08E3
ACUTE_N_SMALL	ń	0x08E4
CEDILLA_N	ƚ	0x083F
CEDILLA_N_SMALL	ƚ	0x08E6
CARON_N	Ň	0x08E7
CARON_N_SMALL	ň	0x08E8
APOSTROPHE_N_SMALL	’n	0x08E9
ENG	Ǝ	0x08EA
ENG_SMALL	ɱ	0x08EB
MACRON_O	Ō	0x08EC
MACRON_O_SMALL	ō	0x08ED
BREVE_O	Ȭ	0x08EE
BREVE_O_SMALL	ȭ	0x08EF
DOUBLE_ACUTE_O	Ő	0x08F0
DOUBLE_ACUTE_O_SMALL	ő	0x08F1
LIGATURE_OE	Œ	0x08F2
LIGATURE_OE_SMALL	œ	0x08F3
ACUTE_R	Ŕ	0x08F4
ACUTE_R_SMALL	ŕ	0x08F5
CEDILLA_R	ƚ	0x08F6
CEDILLA_R_SMALL	ƚ	0x08F7
CARON_R	Ř	0x08F8

Continue.

CARON_R_SMALL	ř	0x08F9
ACUTE_S	Ś	0x08FA
ACUTE_S_SMALL	ś	0x08FB
CIRCUMFLEX_S	Ŝ	0x08FC
CIRCUMFLEX_S_SMALL	ŝ	0x08FD
CEDILLA_S	Ş	0x089F
CEDILLA_S_SMALL	ş	0x08FF

Name	Font	Code
CARON_S	Š	0x103A
CARON_S_SMALL	š	0x103B
CEDILLA_T	Ț	0x103C
CEDILLA_T_SMALL	ț	0x103D
CARON_T	Ť	0x103E
CARON_T_SMALL	ť	0x10A6
STROKE_T	Ƨ	0x10A7
STROKE_T_SMALL	Ƨ	0x10A8
TILDE_U	Ũ	0x10A9
TILDE_U_SMALL	ũ	0x10AA
MACRON_U	Ū	0x10AB
MACRON_U_SMALL	ū	0x10AC
BREVE_U	Ŭ	0x10AD
BREVE_U_SMALL	ŭ	0x10AE
RING_ABOVE_U	Ů	0x10AF
RING_ABOVE_U_SMALL	ů	0x10B0
DOUBLE_ACUTE_U	Ű	0x10B1
DOUBLE_ACUTE_U_SMALL	ű	0x10B2
OGONEK_U	Ų	0x10B3
OGONEK_U_SMALL	ų	0x10B4
CIRCUMFLEX_W	Ŵ	0x10B5
CIRCUMFLEX_W_SMALL	ŵ	0x10B6
CIRCUMFLEX_Y	Ŷ	0x10B7
CIRCUMFLEX_Y_SMALL	ŷ	0x10B8
DIAERESIS_Y	ÿ	0x10B9
ACUTE_Z	Ž	0x1092
ACUTE_Z_SMALL	ž	0x1093
DOT_ABOVE_Z	Ẑ	0x1094
DOT_ABOVE_Z_SMALL	ẑ	0x1095

Continue.

CARON_Z	Ž	0x1096
CARON_Z_SMALL	ž	0x1097
LONG_S	ſ	0x1098
HOOK_F	ƒ	0x1099
RING_ABOVE_ACUTE_A	Ā	0x10C1
RING_ABOVE_ACUTE_A_SMALL	ā	0x10C2
ACUTE_AE	Æ	0x10C3
ACUTE_AE_SMALL	æ	0x10C4
STROKE_ACUTE_O	Ŏ	0x10C5
STROKE_ACUTE_O_SMALL	ŏ	0x10C6

3.95 SysGetNormalUserDefineKey

This function retrieves key codes for ordinary key input mode.

Calling Sequences

```
[C++]
DWORD SysGetNormalUserDefineKey(
    DWORD dwKeyID,
    DWORD KeySetBuff[16]
)
```

```
[Visual Basic]
Public Shared Function SysGetNormalUserDefineKey( _
    ByVal dwKeyId As Int32, _
    ByVal KeySetBuff As Int32() _
) As Int32
```

```
[C#]
public static Int32 SysGetNormalUserDefineKey(
    Int32 dwKeyId,
    Int32[] KeySetBuff
);
```

Parameters

dwKeyID

This parameter is for retrieving key ID. See **SysSetNormalUserDefineKey** function for the values to retrieve.

KeySetBuff[16]

This parameter is for retrieving the virtual key code and option flag that are set to key ID. See **SysSetNormalUserDefineKey** function for the virtual key code and option flag. To retrieve the virtual key code and option flag that are defined to key ID, use **SysGetDefaultKey** function.

Return Values

TRUE	: Normal end
FALSE	: Internal error or user define key is not defined yet. (for DT-X11, IT-600, DT-X7, DT-X30, IT-3100, IT-800, IT-300, and DT-X8)
SYS_PARAMERR	: Parameter error
FUNCTION_UNSupport	: Unsupported error

3.96 SysSetUserDefineKey

This function sets up user defined keys which are freely issued in either numeric, hiragana (Japanese fonts set), katakana (Japanese fonts set), alphabets in uppercase, or alphabets in lowercase mode.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetUserDefineKey** function.

Calling Sequences

```
[C++]
DWORD SysSetUserDefineKey(
    WORD        KeyMode,
    DWORD        KeyID,
    DWORD        UserDefineKeyBuff[16]
)
```

```
[Visual Basic]
Public Shared Function SysSetUserDefineKey( _
    ByVal KeyMode As Short, _
    ByVal KeyId As Int32, _
    ByVal UserDefineKeyBuff As Int32() _
) As Int32
```

```
[C#]
public static Int32 SysSetUserDefineKey(
    Short KeyMode,
    Int32 KeyId,
    Int32[] UserDefineKeyBuff
);
```

Parameters

KeyMode

This parameter is for specifying a key mode selecting one of the values listed below.

KEY_MODE_NUM	: Numeric
KEY_MODE_KANA	: Hiragana (Japanese fonts set) and katakana (Japanese fonts set)
KEY_MODE_ALPHA	: Alphabets in uppercase
KEY_MODE_ALPHAS	: Alphabets in lowercase

KeyID

This parameter is for specifying key ID. See **SysSetNormalUserDefineKey** function for the values to set.

UserDefineKeyBuff[16]

This parameter is for specifying key code. See **SysSetNormalUserDefineKey** function for the virtual key codes and option flags to set.

Return Values

TRUE	: Normal end.
FALSE	: Internal error.
SYS_PARAMERR	: Parameter error.
FUNCTION_UNSupport	: Unsupported error.

3.97 SysGetUserDefineKey

This function retrieves user defined keys which are freely issued in either numeric, hiragana (Japanese fonts set), katakana (Japanese fonts set), alphabets in uppercase, or alphabets in lowercase mode.

Calling Sequences

```
[C++]
DWORD SysSetUserDefineKey(
    WORD      KeyMode,
    DWORD     KeyID,
    DWORD     UserDefineKeyBuff[16]
)
```

```
[Visual Basic]
Public Shared Function SysGetUserDefineKey( _
    ByVal KeyMode As Short, _
    ByVal KeyId As Int32, _
    ByVal UserDefineKeyBuff As Int32() _
) As Int32
```

```
[C#]
public static Int32 SysGetUserDefineKey(
    Short KeyMode,
    Int32 KeyId,
    Int32[] UserDefineKeyBuff
);
```

Parameters

KeyMode

This parameter is for specifying a key mode selecting one of the values listed below.

KEY_MODE_NUM	: Numeric
KEY_MODE_KANA	: Hiragana (Japanese fonts set) and hiragana (Japanese fonts set)
KEY_MODE_ALPHA	: Alphabets in uppercase
KEY_MODE_ALPHAS	: Alphabets in lowercase

KeyID

This parameter is for specifying key ID. See **SysSetNormalUserDefineKey** function for the values to retrieve.

UserDefineKeyBuff[16]

This parameter is for specifying key code. See **SysSetNormalUserDefineKey** function for the virtual key codes and option flags to retrieve.

Return Values

TRUE	: Normal end.
FALSE	: Internal error.
SYS_PARAMERR	: Parameter error.
FUNCTION_UN SUPPORT	: Unsupported error.

3.98 SysSetFnUserDefineKey

This function sets up key code for the Fn key input mode.

Calling Sequences

```
[C++]
DWORD SysSetFnUserDefineKey(
    DWORD dwKeyID,
    DWORD dwKeyCode
)
```

```
[Visual Basic]
Public Shared Function SysSetFnUserDefineKey( _
    ByVal dwKeyId As Int32, _
    ByVal dwKeyCode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetFnUserDefineKey(
    Int32 dwKeyId,
    Int32 dwKeyCode
);
```

Parameters

dwKeyID

This parameter is for specifying key IDs selecting the values below.

KEYID_LEFTTRIGGER	: Left Trigger key
KEYID_RIGHTTRIGGER	: Right Trigger key
KEYID_LEFT	: Left key
KEYID_RIGHT	: Right key
KEYID_CLEAR	: Clear key
KEYID_ENTER	: Enter key
KEYID_0	: 0 key
KEYID_1	: 1 key
KEYID_2	: 2 key
KEYID_3	: 3 key
KEYID_4	: 4 key
KEYID_5	: 5 key
KEYID_6	: 6 key
KEYID_7	: 7 key
KEYID_8	: 8 key
KEYID_9	: 9 key
KEYID_MULTI	: Multi key
KEYID_FUNCTION	: Function key

Continue.

KEYID_UP	: Up key
KEYID_DOWN	: Down key
KEYID_F1	: F1 key
KEYID_F2	: F2 key
KEYID_F3	: F3 key
KEYID_F4	: F4 key
KEYID_F5	: F5 key
KEYID_F6	: F6 key
KEYID_F7	: F7 key
KEYID_F8	: F8 key

dwKeyCode

This parameter is for specifying key code.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.99 SysGetFnUserDefineKey

This function retrieves key codes for the Fn key input mode.

Calling Sequences

```
[C++]
DWORD SysGetFnUserDefineKey(
    DWORD dwKeyID,
    DWORD *pdwKeyCode
)
```

```
[Visual Basic]
Public Shared Function SysGetFnUserDefineKey( _
    ByVal dwKeyId As Int32, _
    ByRef pdwKeyCode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetFnUserDefineKey(
    Int32 dwKeyId,
    ref Int32 pdwKeyCode
);
```

Parameters

dwKeyID

This parameter is for specifying key ID to retrieve. See **SysSetFnUserDefineKey** function for the values.

pdwKeyCode

This parameter is for retrieving the specified key code.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.100 SysSetOtherUserDefineKey

This function sets up key codes for alphanumeric modes.

Calling Sequences

```
[C++]
DWORD SysSetOtherUserDefineKey(
    WORD wMode,
    WORD wKeyID,
    DWORD *pdwCodes
)
```

```
[Visual Basic]
Public Shared Function SysSetOtherUserDefineKey( _
    ByVal wMode As Short, _
    ByVal wKeyID As Short, _
    ByRef pdwCodes As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetOtherUserDefineKey(
    short wMode,
    short wKeyID,
    ref Int32 pdwCodes
);
```

Parameters

wMode

This parameter is for specifying a mode selecting either of the values listed below.

KEY_CONVERSION_MODE_CAPITALALPHA	: Alphanumeric in uppercase letters
KEY_CONVERSION_MODE_SMALLALPHA	: Alphanumeric in lowercase letters

wKeyID

This parameter is for specifying key IDs using the values below.

KEY ID_0
KEY ID_1
KEY ID_2
KEY ID_3
KEY ID_4
KEY ID_5
KEY ID_6
KEY ID_7
KEY ID_8
KEY ID_9

pdwCode[20]

This parameter is for specifying key codes (max. 20 key codes) that correspond to the modes and the key IDs.

The setting must start with "0" first. If 0x00000000 for "No Setting" is included in mid-course, the subsequent settings will become disabled. To initiate the setting, always carry out **SysGetOtherUserDefineKey** function first to retrieve the current key code status before changing the key code settings.

Return Values

TRUE	: Normal end
FUNCTION_UNSupport	: Unsupported error

3.101 SysGetOtherUserDefineKey

This function retrieves key codes for alphanumeric modes.

Calling Sequences

```
[C++]
DWORD SysGetOtherUserDefineKey(
    WORD wMode,
    WORD wKeyID,
    DWORD *pdwCode
)
```

```
[Visual Basic]
Public Shared Function SysGetOtherUserDefineKey( _
    ByVal wMode As Short, _
    ByVal wKeyID As Short, _
    ByRef pdwCode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetOtherUserDefineKey(
    short wMode,
    short wKeyID,
    ref Int32 pdwCode
);
```

Parameters

wMode

This parameter is for specifying a mode. See **SysSetOtherUserDefineKey** function for the values.

wKeyID

This parameter is for specifying a key ID. See **SysSetOtherUserDefineKey** function for the values.

pdwCode[20]

This parameter is for retrieving key codes (max. 20 key codes) that correspond to the modes and the key IDs.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.102 SysSetKeyRepeat

This function sets up key codes that perform key repeat.

Calling Sequences

```
[C++]
DWORD SysSetKeyRepeat(
    DWORD *pdwKeyRepeat
)
```

```
[Visual Basic]
Public Shared Function SysSetKeyRepeat( _
    ByRef pdwKeyRepeat As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetKeyRepeat(
    ref Int32 pdwKeyRepeat
);
```

Parameters

pdwKeyRepeat[16]

This parameter is for specifying array for key codes (max. 16) that perform key repeat. The following values in the table are set by default.

The setting must start with "0" first. If 0x00 for "No Setting" is included in mid-course, the subsequent settings will become disabled. To initiate the setting, always carry out **SysGetKeyRepeat** function first to retrieve the current key repeat status before changing the key repeat settings.

Table 3.25 Default key codes

Array	Virtual Key Code	Value of Virtual Key Code
0	VK_UP	0x26
1	VK_DOWN	0x28
2	VK_LEFT	0x25
3	VK_RIGHT	0x27
4	No setting	0x00
5	No setting	0x00
6	No setting	0x00
7	No setting	0x00
8	No setting	0x00
9	No setting	0x00
10	No setting	0x00
11	No setting	0x00

Continue.

12	No setting	0x00
13	No setting	0x00
14	No setting	0x00
15	No setting	0x00

Return Values

TRUE : Normal end
 FUNCTION_UN SUPPORT : Unsupported error

3.103 SysGetKeyRepeat

This function retrieves key codes that perform key repeat.

Calling Sequences

```
[C++]
DWORD SysGetKeyRepeat(
    DWORD *pdwKeyRepeat
)
```

```
[Visual Basic]
Public Shared Function SysGetKeyRepeat( _
    ByRef pdwKeyRepeat As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetKeyRepeat(
    ref Int32 pdwKeyRepeat
);
```

Parameters

pdwKeyRepeat[16]

The parameter is for specifying an array of key codes (max. 16) that perform key repeats.
See **SysSetKeyRepeat** function for the values to retrieve.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.104 SysSetFakeRepeat

This function sets up key codes that perform faked key repeat. Faked key repeat is used to prevent the KEY_DOWN event that often occurs when a key is pressed for a prolonged period. The key codes (VK_F24, VK_F21) of the Trigger keys registered by default are designed so that the integrated scanner does not repeat emitting laser beam time after time when it is pressed for a prolonged period.

Calling Sequences

```
[C++]
DWORD SysSetFakeRepeat(
    DWORD *pdwKeyRepeat
)
```

```
[Visual Basic]
Public Shared Function SysSetFakeRepeat( _
    ByRef pdwKeyRepeat As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetFakeRepeat(
    ref Int32 pdwKeyRepeat
);
```

Parameters

pdwKeyRepeat[4]

This parameter is for specifying key codes (max. 4) that perform faked key repeat. The values below are default setting.

The setting must start with "0" first. If 0x00 for "No Setting" is included in mid-course, the subsequent settings will become disabled. To initiate the setting, always carry out **SysGetFakeRepeat** function first to retrieve the current key repeat status before changing the faked key repeat settings.

Table 3.26 Default key codes

Array	Virtual Key Code	Value of Virtual Key Code
0	VK_F24	0x87
1	VK_F21	0x84
2	No setting	0x00
3	No setting	0x00

Return Values

TRUE : Normal end
FUNCTION_UNSUPPORTED : Unsupported error

3.105 SysGetFakeRepeat

This function retrieves key codes that perform faked key repeat.

Calling Sequences

```
[C++]
DWORD SysGetFakeRepeat(
    DWORD *pdwKeyRepeat
)
```

```
[Visual Basic]
Public Shared Function SysGetFakeRepeat( _
    ByRef pdwKeyRepeat As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetFakeRepeat(
    ref Int32 pdwKeyRepeat
);
```

Parameters

pdwKeyRepeat[4]

This parameter is for specifying an array to retrieve key codes (max. 4) that perform faked key repeat.

Return Values

TRUE	: Normal end
FUNCTION_UNSUPPORTED	: Unsupported error

3.106 SysSetAllKeyRepeat

This function sets up "Enable" or "Disable" for all individual keys for key repeat set with **SysSetKeyRepeat** function.

Calling Sequences

```
[C++]
DWORD SysSetAllKeyRepeat(
    BOOL bRepeat
)
```

```
[Visual Basic]
Public Shared Function SysSetAllKeyRepeat( _
    ByVal bRepeat As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetAllKeyRepeat(
    Boolean bRepeat
);
```

Parameters

bRepeat

This parameter is for specifying "Enable" or "Disable" for key repeat selecting either of the values listed below.

TRUE	: Enable key repeat.
FALSE	: Disable key repeat.

Return Values

TRUE	: Normal end
FUNCTION_UN SUPPORT	: Unsupported error

3.107 SysGetAllKeyRepeat

This function retrieves the status of "Enable" or "Disable" for all individual keys for key repeat set with **SysSetKeyRepeat** function.

Calling Sequences

```
[C++]  
DWORD SysGetAllKeyRepeat( )
```

```
[Visual Basic]  
Public Shared Function SysGetAllKeyRepeat() As Int32
```

```
[C#]  
public static Int32 SysGetAllKeyRepeat()
```

Parameters

None

Return Values

TRUE	: Key repeat enabled
FALSE	: Key repeat disabled
FUNCTION_UNSUPPORTED	: Unsupported error

3.108 SysDeleteUserDefineKey

This function deletes any user defined keys generated in the numeric key input mode. After the deletion, the default keys are used.

Calling Sequences

```
[C++]
DWORD SysDeleteUserDefineKey(
    int KeyId
)
```

```
[Visual Basic]
Public Shared Function SysDeleteUserDefineKey( _
    ByVal KeyId As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysDeleteUserDefineKey(
    Int32 KeyId
);
```

Parameters

KeyId

This parameter is for specifying key IDs of the user defined keys that are to be deleted. See **SysSetNormalUserDefineKey** function for the values.

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UNSUPPORTED	: Unsupported error

3.109 SysDeleteUserDefineKeyEx

This function deletes user defined keys that are freely issued in either numeric, hiragana (Japanese fonts set), katakana (Japanese fonts set), alphabets in uppercase, or alphabets in lowercase mode. After deleting the user defined keys, the default keys are used.

Calling Sequences

```
[C++]
DWORD SysDeleteUserDefineKey(
    WORD    KeyMode,
    DWORD    KeyID
)
```

```
[Visual Basic]
Public Shared Function SysDeleteUserDefineKeyEx( _
    ByVal KeyMode As Short, _
    ByVal KeyID As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysDeleteUserDefineKeyEx(
    Short KeyMode,
    Int32 KeyID
);
```

Parameters

KeyMode

This parameter is for specifying a key input mode selecting one of the values listed below.

KEY_MODE_NUM	: Numeric
KEY_MODE_KANA	: Hiragana (Japanese fonts set) and katakana (Japanese fonts set)
KEY_MODE_ALPHA	: Alphabets in uppercase
KEY_MODE_ALPHAS	: Alphabets in lowercase

KeyID

This parameter is for specifying key ID. See **SysSetNormalUserDefineKey** function for the values to retrieve.

Return Values

TRUE	: Normal end
FALSE	: Internal error
SYSPARAMERR	: Parameter error
FUNCTION_UNSUPPORTED	: Unsupported error

3.110 SysGetDefaultKey

This function retrieves the default keys generated in the numeric key input mode.

Calling Sequences

```
[C++]
DWORD SysGetDefaultKey(
    int KeyId,
    DWORD DefaultKeyBuf[16]
)
```

```
[Visual Basic]
Public Shared Function SysGetDefaultKey( _
    ByVal KeyId As Int32, _
    ByVal DefaultKeyBuf As Int32() _
) As Int32
```

```
[C#]
public static Int32 SysGetDefaultKey(
    Int32 KeyId,
    Int32[] DefaultKeyBuf
);
```

Parameters

KeyId

This parameter is for specifying key ID. See **SysSetNormalUserDefineKey** function for the values to retrieve.

DefaultKeyBuf[16]

This parameter is for retrieving the virtual key code and option flag that are set to key ID. See **SysSetNormalUserDefineKey** function for virtual key codes and option flags.

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UNSupport	: Unsupported error

3.111 SysSetUserDefineKeyState

This function sets up "Enable" or "Disable" for user defined keys and notifies the status to the keyboard driver. The setting can be made by navigating to **Settings** menu → **Control Panel** menu → **Keyboard** icon → **Others** tab and then **User definition key** field.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetUserDefineKeyState** function.

Calling Sequences

```
[C++]
DWORD SysSetUserDefineKeyState(
    BOOL State
)
```

```
[Visual Basic]
Public Shared Function SysSetUserDefineKeyState( _
    ByVal State As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetUserDefineKeyState(
    Boolean State
);
```

Parameters

State

This is for specifying "Enable" or "Disable" for user defined keys selecting either of the values listed below.

TRUE	: Enable the user defined keys.
FALSE	: Disable the user defined keys.

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UN SUPPORT	: Unsupported error

3.112 SysGetUserDefineKeyState

This function retrieves the status of "Enable" or "Disable" for the user defined keys that has been set. See also **SysSetUserDefineKeyState** function.

Calling Sequences

```
[C++]  
DWORD SysGetUserDefineKeyState( )
```

```
[Visual Basic]  
Public Shared Function SysGetUserDefineKeyState() As Int32
```

```
[C#]  
public static Int32 SysGetUserDefineKeyState()
```

Parameter

None

Return Values

TRUE	: User defined keys enabled.
FALSE	: User defined keys disabled.
FUNCTION_UNSUPPORTED	: Unsupported error

3.113 SysSetResetUserDefineKeyState

This function sets up "Disable" or "Enable" on the "Disable" for the user defined keys at a time of reset. The setting can be made by navigating to **Settings** menu → **Control Panel** menu → **Keyboard** icon → **Others** tab and then **Disable at reset** in **User definition key** field.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetResetUserDefineKeyState** function.

Calling Sequences

```
[C++]
DWORD SysSetResetUserDefineKeyState(
    BOOL State
)
```

```
[Visual Basic]
Public Shared Function SysSetResetUserDefineKeyState( _
    ByVal State As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetResetUserDefineKeyState(
    Boolean State
);
```

Parameters

State

This parameter is for specifying "Enable" or "Disable" for user defined keys selecting either of the values listed below.

TRUE	: Enable user defined keys.
FALSE:	: Disable user defined keys.

Return Values

TRUE	: Normal end
FALSE	: Internal error
FUNCTION_UN SUPPORT	: Unsupported error

3.114 SysGetResetUserDefineKeyState

This function retrieves the status of "Enable" or "Disable" on the "Disable" for user defined keys at time of reset. See also **SysSetResetUserDefineKeyState** function.

Calling Sequences

```
[C++]  
DWORD SysGetResetUserDefineKeyState( )
```

```
[Visual Basic]  
Public Shared Function SysGetResetUserDefineKeyState() As Int32
```

```
[C#]  
public static Int32 SysGetResetUserDefineKeyState()
```

Parameter

None

Return Values

TRUE	: "Disable" is set enabled.
FALSE	: "Disable" is set disabled.
FUNCTION_UNSUPPORTED	: Unsupported error

3.115 SysSetEnableKeyMode

This function sets up "Enable" or "Disable" for key mode transition when the key input mode is changed. For example, if disable [a] in the [1]→[A]→[a]→[P]→[N]→[1] key input transition, the key input transition will be changed to [1]→[A]→[P]→[N]→[1] order. However, **SysSetInputMode** function can change key input mode set disabled to active state.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetEnableKeyMode** function.

Calling Sequences

```
[C++]
DWORD SysSetEnableKeyMode(
    DWORD EnableState
)
```

```
[Visual Basic]
Public Shared Function SysSetEnableKeyMode( _
    ByVal EnableState As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetEnableKeyMode(
    Int32 EnableState
);
```

Parameters

EnableState

This parameter is to select key modes to be enabled by specifying a sum of selected values in logical OR from the list below.

SYS_ENABLE_KEYMODE_NUM	: Numeric
SYS_ENABLE_KEYMODE_ALPHA	: Alphanumeric in uppercase letters
SYS_ENABLE_KEYMODE_ALPHAS	: Alphanumeric in lowercase letters
SYS_ENABLE_KEYMODE_PHONE	: Phone (available on IT-600, DT-X7, DT-X30, IT-800, IT-300, and DT-X8)

Return Values

TRUE	: Success
FUNCTION_UNSUPPORTED	: Unsupported error

3.116 SysGetEnableKeyMode

This function retrieves the status of "Enable" or "Disable" for key mode transition when the key input mode is changed. It also retrieves key input modes available when the key input mode is changed.

Calling Sequences

```
[C++]  
DWORD SysGetEnableKeyMode( )
```

```
[Visual Basic]  
Public Shared Function SysGetEnableKeyMode() As Int32
```

```
[C#]  
public static Int32 SysGetEnableKeyMode()
```

Parameter

None

Return Values

A sum of the values in logical OR for key input modes set effect is retrieved. See **SysSetEnableKeyMode** function for the values.

3.117 SysSetEnableTriggerKey

This function sets up "Enable" or "Disable" for the Trigger keys. While the keys are set disabled, an input by other key can be accepted even if the Trigger keys are pressed.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetEnableTriggerKey** function.

Calling Sequences

```
[C++]
DWORD SysSetEnableTriggerKey(
    BOOL EnableTriggerKey
)
```

```
[Visual Basic]
Public Shared Function SysSetEnableTriggerKey( _
    ByVal EnableTriggerKey As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetEnableTriggerKey(
    Boolean EnableTriggerKey
);
```

Parameters

EnableTriggerKey

This parameter is for specifying "Enable" or "Disable" for the Trigger keys selecting either of the values listed below.

TRUE	: Enable the Trigger keys (Default)
FALSE	: Disable the Trigger keys.

Return Values

TRUE	: Success
FUNCTION_UNSUPPORTED	: Unsupported error

3.118 SysGetEnableTriggerKey

This function retrieves the status of "Enable" or "Disable" for input with the Trigger keys.

Calling Sequences

```
[C++]  
DWORD SysGetEnableTriggerKey( )
```

```
[Visual Basic]  
Public Shared Function SysGetEnableTriggerKey() As Int32
```

```
[C#]  
public static Int32 SysGetEnableTriggerKey()
```

Parameter

None

Return Values

TRUE	: Input by the Trigger keys set enabled (Default)
FALSE	: Input by the Trigger keys set disabled.
FUNCTION_UNSupport	: Unsupported error

3.119 SysSetFnKeyOperation

This function sets up "Enable" or "Disable" for individual Fn key operations.

In the Device Emulator, the function does not perform, but stores the preset value as internal variable. The value stored can be checked with **SysGetFnKeyOperation** function.

Calling Sequences

```
[C++]
DWORD SysSetFnKeyOperation (
    DWORD dwFnKey
)
```

```
[Visual Basic]
Public Shared Function SysSetFnKeyOperation( _
    ByVal dwFnKey As Int32_
) As Int32
```

```
[C#]
public static Int32 SysSetFnKeyOperation (
    Int32 dwFnKey
);
```

Parameters

dwFnKey

This parameter is to specify a sum of the values in logical OR selected from the list below for specific Fn key operations to disable.

STATE_FN_NON	: ALL enable the Fn key operation (Default)
STATE_FN_0	: Disable the Fn key + "0".
STATE_FN_1	: Disable the Fn key + "1".
STATE_FN_2	: Disable the Fn key + "2".
STATE_FN_3	: Disable the Fn key + "3".
STATE_FN_4	: Disable the Fn key + "4".
STATE_FN_5	: Disable the Fn key + "5".
STATE_FN_6	: Disable the Fn key + "6".
STATE_FN_7	: Disable the Fn key + "7".
STATE_FN_8	: Disable the Fn key + "8".
STATE_FN_9	: Disable the Fn key + "9".
STATE_FN_MENU	: Disable the Fn key + "MENU".
STATE_FN_RMULTI	: Disable the Fn key + "L-Multi".
STATE_FN_LMULTI	: Disable the Fn key + "R-Multi".
STATE_FN_ALL	: ALL disable the Fn key operation.

Return Values

TRUE	: Success
FUNCTION_UNSUPPORTED	: Unsupported error

3.120 SysGetFnKeyOperation

This function retrieves the status in a sum of the values for individual Fn key operations that have been set to "Disable".

Calling Sequences

```
[C++]  
DWORD SysGetFnKeyOperation ( )
```

```
[Visual Basic]  
Public Shared Function SysGetFnKeyOperation( ) As Int32
```

```
[C#]  
public static Int32 SysGetFnKeyOperation ( );
```

Parameters

None

Return Values

Status in a sum of the values in logical OR for specific Fn key operations being set disabled is returned. See **SysSetFnKeyOperation** function for the values to retrieve.

FUNCTION_UNSUPPORTED : Unsupported error

3.121 SysWaitForEvent

This function waits for an event that occurs in driver, etc. The function does not return until when an event occurs or when the preset period of the timeout elapses. To call the function, use another thread, not main thread used by application. See also note below.

To forcibly terminate the wait state by the function, call **SysTerminateWaitEvent** function from the main thread of the application. After forcibly terminating the wait state, the function will return one of the return values. But, it does not indicate whether the wait state has been reset for free. It is necessary to check the status of the event.

The function is good for cases that if you wish to acknowledge completion of an input/output via event notification. For example, if you wish to set up "Notification via Event" with **OBRSetScanningNotification** function of the Laser Scanner Library as the notification method for the completion of scanning bar code.

Calling Sequences

```
[Visual Basic]
Public Shared Function SysWaitForEvent( _
    ByVal EventHandle As IntPtr, _
    ByVal EventName As string, _
    ByVal TimeOut As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysWaitForEvent(
    IntPtr EventHandle,
    string EventName,
    Int32 TimeOut
)
```

Parameters

EventHandle

This parameter specifies the handle of an event that the function waits for. Specify "IntPtr.Zero" in this parameter if the specification is carried out in *EventName* parameter.

EventName

This parameter specifies the name of event that the function waits for. The parameter is ignored if other value not "IntPtr.Zero" is specified in the *EventHandle* parameter.

TimeOut

This parameter specifies a time period of the timeout in millisecond that the function waits for an event to occur. Specify "Def.INFINITE (-1)" in this parameter if the function infinitely waits for an event.

Return values

The function returns one of the values below.

FUNCTION_UN SUPPORT	: Unsupported error
WAIT_TIMEOUT	: The timeout elapses.
WAIT_OBJECT_0	: Event occurs.
WAIT_FAILED	: Failed to wait for an event.

Note:

This function with the capability equivalent to **WaitSingleObject** function of Windows API is made available only in Visual Basic and C#. Thus, it does not allow the function to run in C++.

3.122 SysTerminateWaitEvent

This function forcibly terminates the state in wait for an event to occur set with **SysWaitForEvent** function. The function must be carried out if **SysWaitForEvent** function has been called and the application is closed. See also note below.

Calling Sequences

```
[Visual Basic]
Public Shared Function SysTerminateWaitEvent() As Int32
```

```
[C#]
public static Int32 SysTerminateWaitEvent()
```

Return Values

The function returns one of the values below.

FUNCTION_UN SUPPORT	: Unsupported error
True	: Normal end
False	: Internal error

Note:

This function with the capability equivalent to **WaitSingleObject** function of Windows API is made available only in Visual Basic and C#. Thus, it does not allow the function to run in C++.