



# CASTLES TECHNOLOGY

*VEGA5000(S)/VEGA3000 EFT-POS Terminal*

---

*Book 4*

***CTOS Application Programming Interface***

**Confidential**

*Version 3.39 R21*

*June 2, 2016*

**Castles Technology Co., Ltd.**

2F, No. 205, Sec. 3, Beixin Rd., Xindian District,  
New Taipei City 23143, Taiwan R.O.C.

<http://www.castech.com.tw>

# **WARNING**

Information in this document is subject to change without prior notice.

No part of this publication may be reproduced, transmitted, stored in a retrieval system, nor translated into any human or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of **Castles Technology Co., Ltd.**

All trademarks mentioned are proprietary of their respective owners.

## Revision History

<b>Ver</b>	<b>Date</b>	<b>Descriptions</b>	<b>Author</b>
1.1	Mar 2, 2011	Initial creation.	
1.2	May 4, 2011	Added <b>CTOS_FileGetFreeSpace()</b> .	
z	May 13, 2011	Added <b>CTOS_FileOpenAttrib()</b> . Added <b>CTOS_CradleAttached()</b> . Added Base USB functions.	
1.4	Jul 25, 2011	Re-organize the document Added <b>CTOS_BaseUSBSetCDCMode()</b> . Added KBDBuf functions. Added PrinterBuffer functions .	
2.0	Aug 26, 2011	Revised all content.	
2.1	Sep 20, 2011	Remove <b>d_M_STATUS_MODEM_OPEN</b> in CTOS_ModemStatus().	
2.2	Nov 30, 2011	Added <b>CTOS_APFork()</b> . Added <b>CTOS_APRelease()</b> .	
2.2R1	Dec 7, 2011	Modify the <b>CTOS_EthernetURL2IP()</b> , parameter pbRspIPLen should be in/out	
2.3	Jun 6, 2012	<ul style="list-style-type: none"> <li>• System function, add  <b>CTOS_HWSettingGet()</b>  <b>CTOS_SystemMemoryStatus()</b>  <b>CTOS_GetFactorySN()</b></li> <li>• Power Management function, add  <b>CTOS_PowerMode()</b></li> <li>• Battery function, add  <b>CTOS_BatteryStatus()</b></li> <li>• Encryption/Decryption Functions, add  <b>CTOS_DES_CBC()</b></li> <li>• Ethernet Functions, add  <b>CTOS_EthernetOpenEx()</b></li> <li>• USB Functions, add  <b>CTOS_USBGetVidPid()</b>  <b>CTOS_USBGetStatus()</b></li> <li>• Cradle USB functions, add</li> </ul>	

		<p><b>CTOS_BaseUSBSetVidPid()</b>  <b>CTOS_BaseUSBGetVidPid()</b>  <b>CTOS_BaseUSBGetStatus()</b></p> <ul style="list-style-type: none"> <li>• Printer Functions, add</li> </ul> <p><b>CTOS_PrinterCode128Barcode()</b>  <b>CTOS_PrinterInterleaved2of5Barcode()</b></p> <ul style="list-style-type: none"> <li>• Printer Buffer Functions, add</li> </ul> <p><b>CTOS_PrinterBufferCode128Barcode()</b>  <b>CTOS_PrinterBufferInterleaved2of5Barcode()</b></p> <ul style="list-style-type: none"> <li>• GSM Functions, add</li> </ul> <p><b>CTOS_GSMSwitchToCmdMode_A()</b>  <b>CTOS_GSMSwitchToDataMode()</b>  <b>CTOS_GSMGetModuleOpMode()</b>  <b>CTOS_GSMPowerMode()</b>  <b>CTOS_MobileSetNetworkType()</b>  <b>CTOS_MobileGetNetworkTypeCurrent()</b>  <b>CTOS_MobileGetNetworkTypeSupport()</b>  <b>CTOS_SMSUnicodeSend()</b>  <b>stSMS_Unicode_Submit</b></p>	
2.31	Jun 11,2012	Revised the error code.	
3.0	Nov 18, 2012	Combined with TCP API Reference Manual..	
3.1	April.17.2013	Add Contactless Reader API.	
3.2	July.25.2013	Add <b>CTOS_UpdateGetResult()</b>	
3.3	Feb.11.2014	1. CTOS contactless API is supported only in V5S or V3. 2. Correct the description for the usage of parameter of <b>CTOS_CLLEDSet()</b> . 3. Fix the typo of <b>CTOS_MobileSetNetworkType()</b> . 4. Add VEGA3000 label.	
3.31	Feb.14.2014	1. Add <b>CTOS_DeviceModelGet()</b> 2. Add <b>CTOS_LCDAttributeGet()</b>	
3.32	Feb.26.2014	1. Exported PDF version has now links for “Contents table” and other linked references. 2. Updated “Contents” table. 3. Corrected the parameters definition of <b>CTOS_PowerSource()</b> .	

	<p>4. Changed the maximum threshold value of <b>CTOS_BatteryGetChargeThreshold()</b> and <b>CTOS_BatterySetChargeThreshold()</b> from 100 to 90</p> <p>5. Added d_AP_FLAG_DEF_UNSEL to <b>CTOS_APSet()</b>.</p> <p>6. Added (PM) abbreviation to <b>CTOS_APJump()</b> and <b>CTOS_APCall()</b>.</p> <p>7. Added a link to IPC information site in <b>CTOS_APFork()</b>.</p> <p>8. Chinese characters found in “User LoaderFunctions” list removed.</p> <p>9. Improved description for <b>CTOS_EnterDownloadMode()</b></p> <p>10. Improved description for <b>CTOS_UpdateFromMMCI()</b>. Added a note and graph.</p> <p>11. Added a note to <b>CTOS_LedSet()</b>.</p> <p>12. d_BKLIT_INVALID_PARA added to “LED and Backlight Error Codes”.</p> <p>13. d_RTC_TZ_OVERRANGE added to “Clock and Time Error Codes”</p> <p>14. d_CRYPTO_RSA_GEN_KEY_FAILED added to “Encryption and Decryption Error Codes”</p> <p>15. Added d_M_COUNTRY_SPAIN to <b>CTOS_ModemOpen()</b>, <b>CTOS_TCP_ModemOpen()</b> and <b>CTOS_TCP_ModemOpenEx()</b>.</p> <p>16. Improved description for <b>CTOS_TCP_ModemOpenEx()</b>.</p> <p>17. Removed Chinese characters in the “Ethernet Functions” list.</p> <p>18. d_ETHERNET_FTP_GETFILESIZE_FAILED added inside the “Ethernet Error Codes” list.</p> <p>19. Improved <b>CTOS_EthernetOpen()</b> documentation.</p> <p>20. Added a note for <b>CTOS_EthernetClose()</b>.</p> <p>21. Improved description for <b>CTOS_EthernetConnect()</b> and <b>CTOS_EthernetConnectURL()</b>.</p>	
--	---	--

		<p>22. Fixed typo in <b>CTOS_EthernetStatus()</b> and <b>CTOS_EthernetMSStatus()</b> parameters description.</p> <p>23. Improved documentation for <b>CTOS_LCDForeGndColor()</b> and <b>CTOS_LCDBackGndColor()</b>.</p> <p>24. Improved description for <b>CTOS_LCDTTFSelect(), CTOS_LCDFontSelectMode()</b> and <b>CTOS_LCDTSelectFontSize()</b>.</p> <p>25. d_PRINTER_PIC_xxx errors added inside the “Printer Error Codes” list,</p> <p>26. Improved description for <b>CTOS_PrinterSetHeatLevel()</b>.</p> <p>27. “USHORT CTOSx” string was replaced by “USORT CTOSx” in many function descriptions.</p> <p>28. d_PRTBUF_PIC_xxx errors added inside the “Printer Buffer Error Codes” list,</p> <p>29. Improved description for <b>CTOS_PrinterTTFSelect()</b>.</p> <p>30. Improved parameter descriptions for <b>CTOS_SCStatus()</b> and <b>CTOS_SCPowerOff()</b>.</p> <p>31. d_MENU_LCD_NOT_SUPPORT added to the “User Interface Error Codes” list.</p> <p>32. Added a note to “File System Functions” section.</p> <p>33. Removed a reference to Vega5000 in <b>CTOS_TCP_GPRSGetIP()</b>.</p> <p>34. Fixed pusType parameter definition in <b>CTOS_GSMGetCurrentSIM()</b>.</p>	
3.33	Aug.15.2014	1. Add Wifi Functions.	
3.34	Jan.8.2015	1. Modify the note of <b>CTOS_KBDHit()</b> .	
3.35	Jan.15.2015	1. Modify the value of <b>d_CAP_ESIGN_INCORRECT</b> to be 0007h	
3.36	Jan. 26.2015	1. Add note for <b>CTOS_TCP_GPRSSStatus</b> function 2. Add note for <b>CTOS_TCP_GPRSSocketStatus</b> function 3. Modify Contactless Reader return code	

		4. Add notation for <b>CTOS_SyncICCReset</b> function	
3.37	Jan. 27.2015	1. Modify Contactless Reader return code	
3.38	Jan. 29.2015	1. Modify Contactless Reader return code	
3.38R	Jan. 29.2015	1. Modify Contactless Reader return code	
3.39	Mar. 25.2015	1. Add new return code <b>d_SC_INSUFFICIENT_BUF</b> for SC. 2. Add Battery function return code <b>d_BATTERY_BUSY</b> , <b>d_BATTERY_INVALID</b> , and <b>d_BATTERY_NOT_SUPPORT</b> 3. Add notation for <b>CTOS_BatteryStatus()</b> . 4. Add notation for <b>CTOS_PowerOff()</b> . 5. Add notation for <b>CTOS_CLAPDU()</b> and <b>CTOS_TclTransparent()</b> 6. Add PowerAutoMode functions, and sample code in appendix 7. Add <b>CTOS_LCDGShowBMPPic()</b> , <b>d_LCD_PIC_FORMAT_NOT_SUPPORT</b> , and <b>d_LCD_PIC_OPEN_FAILED</b> 8. Add <b>CTOS_PrinterBMPPic()</b> and <b>CTOS_PrinterBufferBMPPic()</b>	
3.39R	Apr 06, 2016	R1 1. Add <b>CTOS_PowerModeOptSet()</b> and <b>CTOS_PowerModeOptGet()</b> R2 1. Correct type error of the output state of <b>CTOS_WifiStatus()</b> 2. Correct type error of the input parameters of <b>CTOS_WifiPing()</b> 3. Add suppot features for <b>CTOS_HWSettingGet()</b> , wifi and Bluetooth R3 1. Corrected some typos inside <b>CTOS_PowerMode()</b> and <b>CTOS_PowerModeEx()</b> 2. Now the date of the revision in the cover is an automatic field and will change each time this document is edited.	Jorge Jorge

	<p>3. Added a fourth column to this revision table so the author of each modification can be noted.</p> <p>4. Added a hyperlink that links the corporate webpage in the footer</p> <p>5. Added <a href="#">Font error codes</a> hyperlink inside the <b>CTOS_LCDTTFSelect()</b> and <b>CTOS_PrinterTTFSelect()</b> descriptions</p> <p>6. Added a note to <b>CTOS_LCDSetContrast()</b> remarking that it cannot be used for 3.5" displays</p> <p>7. Explain the relationship between d_FONT_TYPEEx and \ftx escape codes in <b>CTOS_LanguagePrinterFontSize()</b>.</p> <p>8. Re-format some text in <b>CTOS_LanguageConfig()</b></p> <p>9. Words "usedwhen" and "usFontSizeshould" have been splitted.</p> <p>R8</p> <p>10. d_PWRSRC_USB added to <b>CTOS_PowerSource()</b></p> <p>R9</p> <p>11. CTOS_RTC structure detailed explanation</p> <p>R10</p> <p>12. d_TCP_DEVICE_WIFI added to <b>CTOS_TCP_SelectDefaultDevice()</b></p> <p>R11</p> <p>13. USHORTCTOS and ULONGCTOS strings are now splitted.</p> <p>14. Improved the description of <b>CTOS_WifiConfigSet()</b></p> <p>R12</p> <p>15. Wrong explanation for parameter of <b>CTOS_WifiPing()</b></p> <p>16. Added supplement in notation for <b>CTOS_EthernetDisable()</b></p> <p>R13</p> <p>17. Added default values for PPP and TCP timeouts and retries</p> <p>18. Added an explanatory note to <b>CTOS_TCP_GPRSInit()</b></p>	Jorge
		Mario
		Jorge
		Shaun
		Shaun
		Jorge
		Jorge

	R14	19. Removed “Vega5000” from the page header in <b>CTOS_TCP_SetRxAckTO()</b> and <b>CTOS_TCP_GetRxAckTO()</b> since they are not supported by such device.	Jorge
	R15	20. <b>CTOS_PowerModeEx()</b> renamed to <b>CTOS_PowerModeEX()</b> to match the prototype inside ctosapi.h	Jorge
	R16	21. Add the return code of Wifi IO Error (0x60XX) in 2.30. 22. Revise the return code of <b>CTOS_PINGetAuthStatus()</b>	Jerry
	R17	23. Table of contents updated 24. Exported to PDF with embedded bookmarks	Jorge
	R18	25. Added a remark to the <b>CTOS_GSMGetBAND()</b>	Jorge
	R19	26. Added escape codes for <b>CTOS_LCDTPrintXY()</b> and <b>CTOS_PrinterBufferPutString()</b>	Jorge
		27. Call CTOS_WifiStatus() to check <b>CTOS_WifiOpen()</b> has finished.	Jorge
		28. Added a note to <b>CTOS_UpdateFromMMCIE()</b>	Jorge
	R20	29. Add Touch Signature Series API.	Jerry
		30. Add Note for CTOS_TCP_SelectDefaultDevice()	Jerry
	R.21	31. Add Bluetooth Series API.	Jerry

# Contents

<b>1. Introduction .....</b>	<b>30</b>
<b>2. CTOS Application Programming Interface .....</b>	<b>32</b>
<b>    2.1. System Functions.....</b>	<b>32</b>
CTOS_GetSerialNumber.....	33
CTOS_GetFactorySN .....	34
CTOS_GetSystemInfo.....	35
CTOS_FactoryReset .....	37
CTOS_GetKeyHash .....	38
CTOS_SystemWait .....	39
CTOS_CradleAttached.....	41
CTOS_HWSettingGet.....	42
CTOS_SystemMemoryStatus.....	44
CTOS_DeviceModelGet .....	45
<b>    2.2. Power Management Functions .....</b>	<b>46</b>
CTOS_PowerSource .....	47
CTOS_SystemReset .....	48
CTOS_PowerOff.....	49
CTOS_PowerMode .....	50
CTOS_PowerModeEX.....	51
CTOS_PowerModeOptSet .....	53
CTOS_PowerModeOptGet .....	55
CTOS_PowerAwakening .....	57
CTOS_PowerAutoModeEnable .....	58
CTOS_PowerAutoModeDisable .....	59
CTOS_PowerAutoModeTimeToStandby .....	60
CTOS_PowerAutoModeTimeToSleep .....	61
CTOS_PowerAutoModeRegisterCallback.....	62

CTOS_PowerAutoModeUNRegisterCallback .....	64
CTOS_PowerWakeupSrcGet .....	65
CTOS_PowerWakeupSrcSet.....	67
<b>2.3. Battery Functions .....</b>	<b>69</b>
CTOS_BatteryGetCapacity.....	70
CTOS_BatteryForceCharge.....	71
CTOS_BatterySetChargeThreshold.....	72
CTOS_BatteryGetChargeThreshold.....	73
CTOS_BatteryStatus .....	74
<b>2.4. Program Manager Functions .....</b>	<b>75</b>
CTOS_APFind.....	76
CTOS_APGet.....	77
CTOS_APSet .....	78
CTOS_APDel .....	79
CTOS_APJump .....	80
CTOS_APCall .....	81
CTOS_APFork.....	82
<b>2.5. User Loader Functions.....</b>	<b>83</b>
CTOS_EnterDownloadMode .....	84
CTOS_UpdateFromMMCI .....	85
CTOS_UpdateGetResult .....	87
CTOS_SetPMEnterPassword.....	88
CTOS_SetFunKeyPassword .....	89
CTOS_SetULDPassword .....	90
CTOS_SealULD .....	91
<b>2.6. LED and Backlight Functions .....</b>	<b>92</b>
CTOS_LEDSet .....	93
CTOS_BackLightSet .....	94
CTOS_BackLightSetEx .....	95
<b>2.7. Clock and Time Functions .....</b>	<b>96</b>

CTOS_RTCGet .....	97
CTOS_RTCSet.....	98
CTOS_Delay .....	99
CTOS_TickGet.....	100
CTOS_TimeOutSet .....	101
CTOS_TimeOutCheck.....	102
<b>2.8. Buzzer Functions.....</b>	<b>103</b>
CTOS_Beep.....	104
CTOS_Sound .....	105
<b>2.9. Encryption and Decryption Functions .....</b>	<b>106</b>
CTOS_RSA.....	107
CTOS_RNG .....	108
CTOS_DES.....	109
CTOS_SHA1Init .....	110
CTOS_SHA1Update.....	111
CTOS_SHA1Final .....	112
CTOS_AES .....	113
CTOS_MAC .....	114
CTOS_DES_CBC.....	115
<b>2.10. RS232 Communication Functions.....</b>	<b>116</b>
CTOS_RS232Open.....	117
CTOS_RS232TxReady .....	119
CTOS_RS232TxData .....	120
CTOS_RS232RxReady .....	121
CTOS_RS232RxData.....	122
CTOS_RS232SetRTS .....	123
CTOS_RS232GetCTS.....	124
CTOS_RS232FlushRxBuffer .....	125
CTOS_RS232FlushTxBuffer.....	126
CTOS_RS232Close .....	127

<b>2.11. Modem Functions.....</b>	<b>128</b>
CTOS_ModemOpen.....	129
CTOS_ModemClose .....	132
CTOS_ModemDialup.....	133
CTOS_ModemTxReady .....	134
CTOS_ModemTxData .....	135
CTOS_ModemRxReady .....	136
CTOS_ModemRxData.....	137
CTOS_ModemATCommand.....	138
CTOS_ModemStatus.....	139
CTOS_ModemHookOff.....	141
CTOS_ModemHookOn.....	142
CTOS_ModemSetDialPrecheck .....	143
CTOS_ModemSetConfig .....	144
CTOS_ModemReadConfig.....	147
CTOS_ModemFlushRxData .....	149
CTOS_ModemListen .....	150
CTOS_ModemSetCommParam .....	151
<b>2.12. Modem TCP Functions.....</b>	<b>152</b>
CTOS_TCP_ModemInit.....	156
CTOS_TCP_ModemOpen.....	157
CTOS_TCP_ModemOpenEx.....	160
CTOS_TCP_ModemClose .....	163
CTOS_TCP_ModemDialup .....	164
CTOS_TCP_ModemOnHook.....	165
CTOS_TCP_ModemGetIP .....	166
CTOS_TCP_ModemSetIP .....	167
CTOS_TCP_ModemConnectEx .....	168
CTOS_TCP_ModemConnectURL .....	169

CTOS_TCP_ModemDisconnect .....	170
CTOS_TCP_ModemTx.....	171
CTOS_TCP_ModemRx .....	172
CTOS_TCP_ModemStatus .....	173
CTOS_TCP_ModemGetDNSServer .....	174
CTOS_TCP_ModemSetDNSServer .....	175
CTOS_TCP_ModemURL2IP .....	176
CTOS_TCP_ModemPing .....	177
CTOS_UDP_ModemTx .....	178
CTOS_UDP_ModemRx .....	179
<b>2.13. Ethernet Functions .....</b>	<b>180</b>
CTOS_EthernetOpen .....	183
CTOS_EthernetOpenEx .....	184
CTOS_EthernetClose .....	185
CTOS_EthernetConfigGet .....	186
CTOS_EthernetConfigSet .....	188
CTOS_EthernetPing .....	191
CTOS_EthernetURL2IP.....	192
CTOS_EthernetIP2MAC.....	193
CTOS_EthernetEnable .....	194
CTOS_EthernetDisable .....	195
CTOS_EthernetConnect.....	196
CTOS_EthernetConnectEx.....	197
CTOS_EthernetConnectURL.....	198
CTOS_EthernetConnectURLEX.....	199
CTOS_EthernetConnectURLPort .....	200
CTOS_EthernetDisconnect.....	201
CTOS_EthernetTxReady.....	202
CTOS_EthernetTxData.....	203

CTOS_EthernetRxReady .....	204
CTOS_EthernetRxData .....	205
CTOS_EthernetStatus .....	206
CTOS_EthernetGetConnectionInfo.....	207
CTOS_EthernetFlushRxData.....	208
CTOS_EthernetMSConnect.....	209
CTOS_EthernetMSConnectEx .....	210
CTOS_EthernetMSDisconnect .....	211
CTOS_EthernetMSTxReady.....	212
CTOS_EthernetMSTxData .....	213
CTOS_EthernetMSRxReady .....	214
CTOS_EthernetMSRxData.....	215
CTOS_EthernetMSStatus.....	216
CTOS_EthernetMSGetConnectionInfo .....	217
CTOS_EthernetMSListen .....	218
CTOS_EthernetMSUnlisten.....	219
CTOS_EthernetMSRxbuSet .....	220
CTOS_EthernetFTPConnect .....	221
CTOS_EthernetFTPDisconnect.....	223
CTOS_EthernetFTPGetState .....	224
CTOS_EthernetFTPLList.....	225
CTOS_EthernetFTPChangeDir .....	226
CTOS_EthernetFTPDownload.....	227
CTOS_EthernetFTPDownloadFile .....	228
CTOS_EthernetFTPUpload .....	229
CTOS_EthernetFTPUploadFile .....	230
CTOS_EthernetFTPTranStatus .....	231
CTOS_EthernetFTPCancel .....	232
<b>2.14. USB Functions.....</b>	<b>233</b>

CTOS_USBSelectMode .....	234
CTOS_USBOpen .....	235
CTOS_USBClose .....	236
CTOS_USBTxReady .....	237
CTOS_USBTxData .....	238
CTOS_USBRxReady .....	239
CTOS_USBRxData .....	240
CTOS_USBTxFlush .....	241
CTOS_USBRxFlush .....	242
CTOS_USBSetCDCMode .....	243
CTOS_USBSetSTDMode .....	244
CTOS_USBSetVidPid .....	245
CTOS_USBGetVidPid .....	246
CTOS_USBGetStatus .....	247
CTOS_USBHostOpen .....	248
CTOS_USBHostClose .....	249
CTOS_USBHostTxData .....	250
CTOS_USBHostRxData .....	251
<b>2.15. Cradle USB Functions.....</b>	<b>252</b>
CTOS_BaseUSBOpen .....	253
CTOS_BaseUSBClose .....	254
CTOS_BaseUSBTxReady .....	255
CTOS_BaseUSBTxData .....	256
CTOS_BaseUSBRxReady .....	257
CTOS_BaseUSBRxData .....	258
CTOS_BaseUSBTxFlush .....	259
CTOS_BaseUSBRxFlush .....	260
CTOS_BaseUSBSetCDCMode .....	261
CTOS_BaseUSBSetSTDMode .....	262

CTOS_BaseUSBSetVidPid.....	263
CTOS_BaseUSBGetVidPid .....	264
CTOS_BaseUSBGetStatus .....	265
<b>2.16. LCD Display Functions .....</b>	<b>266</b>
CTOS_LCDSelectMode.....	268
CTOS_LCDSelectModeEx .....	270
CTOS_LCDSetContrast.....	272
CTOS_LCDCForeColor .....	273
CTOS_LCDCBackColor .....	274
CTOS_LCDDFontSelectMode.....	275
CTOS_LCDTTFSelect.....	276
CTOS_LCDTTFCheckLanguageSupport.....	277
CTOS_LCDGClearCanvas .....	278
CTOS_LCDGPixel.....	279
CTOS_LCDGTextOut.....	280
CTOS_LCDGSetBox .....	281
CTOS_LCDGShowPic.....	282
CTOS_LCDGShowPicEx.....	283
CTOS_LCDGShowBMPPic .....	284
CTOS_LCDGClearWindow .....	285
CTOS_LCDGMoveWindow .....	286
CTOS_LCDGGetWindowOffset.....	287
CTOS_LCDTClearDisplay .....	288
CTOS_LCDTGotoXY.....	289
CTOS_LCDTWhereX .....	290
CTOS_LCDTWhereY .....	291
CTOS_LCDTPrint.....	292
CTOS_LCDTPrintXY .....	293
CTOS_LCDTPutch .....	294

CTOS_LCDTPutchXY .....	295
CTOS_LCDTClear2EOL.....	296
CTOS_LCDTSetReverse.....	297
CTOS_LCDTSelectFontSize .....	298
CTOS_LCDTSetASCIIVerticalOffset .....	300
CTOS_LCDTSetASCIIEHorizontalOffset .....	301
CTOS_LCDUInit.....	302
CTOS_LCDUClearDisplay.....	303
CTOS_LCDUPrintXY.....	304
CTOS_LCDUPrint .....	305
CTOS_ LCDAttributeGet .....	306
<b>2.17. Keyboard Functions .....</b>	<b>307</b>
CTOS_KBDGet .....	308
CTOS_KBDHit.....	309
CTOS_KBDSetSound.....	310
CTOS_KBDSetFrequency.....	311
CTOS_KBDInKey .....	312
CTOS_KBDInKeyCheck .....	313
CTOS_KBDSetResetEnable.....	314
CTOS_KBDBufCheck.....	315
CTOS_KBDBufGet .....	316
CTOS_KBDBufPut .....	317
CTOS_KBDBufFlush .....	318
CTOS_KBDScan .....	319
<b>2.18. Printer Functions .....</b>	<b>320</b>
CTOS_PrinterPutString .....	321
CTOS_PrinterLogo .....	322
CTOS_PrinterBMPPic .....	323
CTOS_PrinterFline .....	324

CTOS_PrinterStatus .....	325
CTOS_PrinterSetWorkTime.....	326
CTOS_PrinterSetHeatLevel.....	327
CTOS_PrinterFontSelectMode .....	328
CTOS_PrinterTTFSelect.....	329
CTOS_PrinterTTFCheckLanguageSupport .....	330
CTOS_PrinterCodaBarBarcode .....	331
CTOS_PrinterCode39Barcode .....	334
CTOS_PrinterEAN13Barcode.....	336
CTOS_PrinterCode128Barcode .....	342
CTOS_PrinterInterleaved2of5Barcode .....	343
<b>2.19. Printer Buffer Functions .....</b>	<b>344</b>
CTOS_PrinterBufferEnable.....	345
CTOS_PrinterBufferInit.....	346
CTOS_PrinterBufferSelectActiveAddress .....	347
CTOS_PrinterBufferFill.....	348
CTOS_PrinterBufferHLine .....	349
CTOS_PrinterBufferVLine .....	350
CTOS_PrinterBufferLogo.....	351
CTOS_PrinterBufferBMPPic.....	352
CTOS_PrinterBufferPixel.....	353
CTOS_PrinterBufferPutString .....	354
CTOS_PrinterBufferOutput.....	356
CTOS_PrinterBufferCodaBarBarcode .....	357
CTOS_PrinterBufferCode39Barcode .....	360
CTOS_PrinterBufferEAN13Barcode .....	362
CTOS_PrinterBufferCode128Barcode .....	368
CTOS_PrinterBufferInterleaved2of5Barcode .....	369
<b>2.20. Smart Card Functions .....</b>	<b>370</b>

CTOS_SCStatus .....	371
CTOS_SCPowerOff.....	372
CTOS_SCResetEMV.....	373
CTOS_SCResetISO .....	375
CTOS_SCSendAPDU .....	377
CTOS_SCCommonReset.....	378
<b>2.21. Memory Card Functions.....</b>	<b>381</b>
CTOS_SyncICCReset .....	383
CTOS_44x2ReadMainMemory.....	384
CTOS_44x2WriteMainMemory .....	385
CTOS_44x2ReadProtectionMemory.....	386
CTOS_44x2SetProtectionBit .....	387
CTOS_44x2ReadSecurityMemory.....	388
CTOS_44x2ChangePSC .....	389
CTOS_44x2VerifyPSC .....	390
CTOS_44x8WriteMemoryWithoutPBit .....	391
CTOS_44x8WriteMemoryWithPBit .....	392
CTOS_44x8WritePBitWithDataCompare .....	393
CTOS_44x8ReadMemoryWithoutPBit .....	394
CTOS_44x8ReadMemoryWithPBit.....	395
CTOS_44x8VerifyPSC .....	396
CTOS_44x6ReadData.....	397
CTOS_44x6WriteData .....	398
CTOS_44x6Reload .....	399
CTOS_44x6VerifyTSC.....	400
CTOS_44x6Auth .....	401
CTOS_SLE44x4ReadMemory.....	402
CTOS_SLE44x4VerifyUserCode .....	403
CTOS_SLE44x4VerifyMemoryCode .....	404

CTOS_SLE44x4WriteMemory .....	405
CTOS_SLE44x4EraseMemory .....	406
CTOS_SLE44x4BlowFuse .....	407
CTOS_SLE44x4EnterTestMode .....	408
CTOS_SLE44x4ExitTestMode .....	409
CTOS_GPM896ReadMemory .....	410
CTOS_GPM896VerifySecurityCode .....	411
CTOS_GPM896WriteMemory .....	412
CTOS_GPM896EraseMemory .....	413
CTOS_GPM896EraseArea1 .....	414
CTOS_GPM896EraseArea2 .....	415
CTOS_I2CCReset .....	416
CTOS_I2CReadMemory .....	417
CTOS_I2CWriteMemory .....	419
<b>2.22. Magnetic Stripe Reader Functions .....</b>	<b>420</b>
CTOS_MSRRead .....	421
CTOS_MSRReadEx .....	423
CTOS_MSRGetLastErr .....	425
<b>2.23. Contactless Reader .....</b>	<b>427</b>
CTOS_CLInitComm .....	431
CTOS_CLCloseComm .....	432
CTOS_CLInit .....	433
CTOS_CLPowerOn .....	434
CTOS_CLPowerOff .....	435
CTOS_CLVerGet .....	436
CTOS_ReadRC .....	437
CTOS_WriteRC .....	438
CTOS_REQA .....	439
CTOS_WUPA .....	440

CTOS_ANTIA .....	441
CTOS_SELECTA .....	442
CTOS_CLTypeAHalt .....	443
CTOS_CLTypeAActiveFromIdle .....	444
CTOS_CLTypeAActiveFromHalt .....	445
CTOS_MifareREADE2 .....	446
CTOS_MifareWRITEE2 .....	447
CTOS_MifareLOADKEY .....	448
CTOS_MifareLOADKEYE2 .....	449
CTOS_MifareAUTHEx .....	450
CTOS_MifareAUTH2Ex .....	451
CTOS_MifareREADBLOCK .....	452
CTOS_MifareWRITEBLOCK .....	453
CTOS_MifareINCREASE .....	454
CTOS_MifareDECREASE .....	455
CTOS_MifareRESTORE .....	456
CTOS_MifareTRANSFER .....	457
CTOS_REQB .....	458
CTOS_WUPB .....	459
CTOS_ATTRIB .....	460
CTOS_HALTB .....	461
CTOS_CLTypeBActive .....	462
CTOS_CLTypeBActiveEx .....	463
CTOS_CLRATS .....	465
CTOS_CLRATSEx .....	466
CTOS_CLAPDU .....	468
CTOS_CLDESELECT .....	470
CTOS_CLNAKPOLL .....	471
CTOS_TclTransparent .....	472

CTOS_TclTransparentEx .....	473
CTOS_TclSetTimeout.....	474
CTOS_FelicaPolling .....	475
CTOS_FelicaReadWrite .....	476
CTOS_CLLEDSet.....	477
CTOS_CLSound.....	478
<b>2.24. FNT Font Functions.....</b>	<b>479</b>
CTOS_LanguageConfig .....	480
CTOS_LanguageLCDFontSize.....	484
CTOS_LanguagePrinterFontSize .....	486
CTOS_LanguageInfo.....	489
CTOS_LanguageNum .....	490
CTOS_LanguagePrinterSelectASCII .....	491
CTOS_LanguageLCDSelectASCII.....	492
CTOS_LanguagePrinterGetFontInfo.....	493
CTOS_LanguageLCDGetFontInfo .....	494
<b>2.25. GSM Functions .....</b>	<b>495</b>
CTOS_GSMOpen.....	498
CTOS_GSMClose .....	499
CTOS_GSMPowerOff.....	500
CTOS_GSMPowerOn.....	501
CTOS_GSMReset.....	502
CTOS_GSMQueryOperatorName .....	503
CTOS_GSMHangup .....	504
CTOS_GSMSendData.....	505
CTOS_GSMRecvData.....	506
CTOS_GSMSendATCmd .....	507
CTOS_GSMSignalQuality .....	508
CTOS_GSMGetIMEI .....	509

CTOS_GSMGetNetworkType.....	510
CTOS_GSMSetBAND .....	511
CTOS_GSMGetBAND.....	512
CTOS_GSMDial .....	513
CTOS_GSMSwitchToCmdMode .....	514
CTOS_GSMSwitchToCmdMode_A .....	515
CTOS_GSMSwitchToDataMode .....	516
CTOS_GSMGetModuleOpMode .....	517
CTOS_GSMPowerMode .....	518
CTOS_MobileSetNetworkType.....	519
CTOS_MobileGetNetworkTypeCurrent.....	520
CTOS_MobileGetNetworkTypeSupport .....	521
CTOS_GSMSelectSIM .....	522
CTOS_GSMGetCurrentSIM.....	523
CTOS_SIMGetIMSI .....	524
CTOS_SIMCheckReady.....	525
CTOS_PINGetAuthStatus .....	526
CTOS_PINVerify .....	527
CTOS_PINCheckLock .....	528
CTOS_PINLock .....	529
CTOS_PINUnLock .....	530
CTOS_PINUpdate .....	531
CTOS_GPRSAttach .....	532
CTOS_GPRSDetach .....	533
CTOS_GPRSGetRegState .....	534
CTOS_GPRSPPPConnect .....	535
CTOS_PBSelect.....	536
CTOS_PBRead .....	538
CTOS_PBWrite .....	539

CTOS_PBDelete .....	540
CTOS_SMSGetSCNumber.....	541
CTOS_SMSSetSCNumber.....	542
CTOS_SMSGetList .....	543
CTOS_SMSRead .....	544
CTOS_SMSDelete .....	545
CTOS_SMSSend .....	546
CTOS_SMSSendPDU .....	547
CTOS_SMSUnicodeSend .....	548
<b>2.26. GPRS Functions .....</b>	<b>549</b>
CTOS_TCP_GPRSInit.....	552
CTOS_TCP_GPRSOpen.....	553
CTOS_TCP_GPRSOpenEx.....	554
CTOS_TCP_GPRSClose .....	555
CTOS_TCP_GPRSClose_A.....	556
CTOS_TCP_GPRSGetIP .....	557
CTOS_TCP_GPRSConnectEx .....	558
CTOS_TCP_GPRSDisconnect.....	559
CTOS_TCP_GPRSTx .....	560
CTOS_TCP_GPRSRx .....	561
CTOS_TCP_GPRSStatus .....	562
CTOS_TCP_GPRSSocketStatus .....	563
CTOS_TCP_GPRSSwitchAPN.....	564
CTOS_TCP_GPRSSwitchAPN_A .....	565
CTOS_TCP_GPRSCancelTask.....	566
CTOS_TCP_GPRSGetDNSServer.....	567
CTOS_TCP_GPRSSetDNSServer .....	568
CTOS_TCP_GPRSConnectURL .....	569
CTOS_TCP_GPRSPing .....	570

CTOS_TCP_GPRSURL2IP .....	571
<b>2.27. File System Functions.....</b>	<b>572</b>
CTOS_FileOpen.....	574
CTOS_FileClose.....	575
CTOS_FileDelete .....	576
CTOS_FileGetSize .....	577
CTOS_FileSeek.....	578
CTOS_FileRead .....	579
CTOS_FileWrite .....	580
CTOS_FileDir .....	581
CTOS_FileCut .....	582
CTOS_FileRename .....	583
CTOS_FileGetPosition .....	584
CTOS_FileReopen .....	585
CTOS_FileSetAttrib .....	586
CTOS_FileDirAttrib .....	587
CTOS_FileGetAttrib.....	588
CTOS_FileFormat .....	589
CTOS_FileDirA.....	590
CTOS_FileGetFreeSpace.....	591
CTOS_FileOpenAttrib.....	592
<b>2.28. User Interface Functions.....</b>	<b>593</b>
CTOS_LCDGMenu.....	594
CTOS_LCDGMenuEx.....	597
CTOS_LCDTMenu .....	600
CTOS_UIKeypad.....	603
<b>2.29. TCP Setting Functions .....</b>	<b>608</b>
CTOS_TCP_SetConnectTO .....	611
CTOS_TCP_GetConnectTO.....	612
CTOS_TCP_SetRxAckTO .....	613

CTOS_TCP_GetRxAckTO .....	614
CTOS_TCP_SetRxTO.....	615
CTOS_TCP_GetRxTO .....	616
CTOS_TCP_SetRetryCounter.....	617
CTOS_TCP_GetRetryCounter.....	618
CTOS_TCP_BindToDevice .....	619
CTOS_TCP_SelectDefaultDevice.....	620
CTOS_PPP_SetTO .....	621
CTOS_PPP_GetTO.....	622
CTOS_PPP_SetRetryCounter .....	623
CTOS_PPP_GetRetryCounter.....	624
<b>2.30. Wifi Functions.....</b>	<b>625</b>
CTOS_WifiOpen.....	626
CTOS_WifiClose .....	627
CTOS_WifiScan .....	628
CTOS_WifiInfoGet.....	629
CTOS_WifiConnectAP.....	630
CTOS_WifiDisconnectAP .....	631
CTOS_WifiStatus .....	632
CTOS_WifiConfigSet.....	633
CTOS_WifiConfigGet.....	635
CTOS_WifiConnectAPEX .....	637
CTOS_WifiPing .....	639
<b>2.31. Touch Signature .....</b>	<b>640</b>
CTOS_TouchSignatureStart.....	641
CTOS_GetSignatureStatus .....	643
CTOS_ToucSignatureConfigSet.....	644
CTOS_TouchSignatureTerminate .....	645
CTOS_BMPConverter .....	646

<b>2.32. Bluetooth Functions.....</b>	<b>647</b>
CTOS_BluetoothConfigSet.....	648
CTOS_BluetoothConfigGet .....	649
CTOS_BluetoothTxReady .....	651
CTOS_BluetoothTxData .....	652
CTOS_BluetoothRxReady .....	653
CTOS_BluetoothRxData.....	654
CTOS_BluetoothDisconnect.....	655
CTOS_BluetoothStatus .....	656
CTOS_BluetoothListen.....	657
CTOS_BluetoothUnlisten.....	658
CTOS_BluetoothOpen.....	659
CTOS_BluetoothClose .....	660
CTOS_BluetoothPairedListGet.....	661
CTOS_BluetoothUnpair .....	662
<b>Appendix A: Languge Id.....</b>	<b>663</b>
<b>Appendix B:Examples .....</b>	<b>668</b>
B.1. <b>Encryption and Decryption.....</b>	<b>668</b>
B.2. <b>Ethernet.....</b>	<b>672</b>
B.3. <b>FileSystem .....</b>	<b>681</b>
B.4. <b>GSM.....</b>	<b>691</b>
B.5. <b>Hardware .....</b>	<b>705</b>
B.6. <b>Hello .....</b>	<b>712</b>
B.7. <b>Keyboard .....</b>	<b>713</b>
B.8. <b>Keypad .....</b>	<b>718</b>
B.9. <b>Language .....</b>	<b>721</b>
B.10. <b>LCD Display .....</b>	<b>723</b>
B.11. <b>Memory Card .....</b>	<b>728</b>
B.12. <b>Modem.....</b>	<b>734</b>
B.13. <b>MSReader.....</b>	<b>744</b>
B.14. <b>Printer.....</b>	<b>746</b>
B.15. <b>RS232 .....</b>	<b>750</b>

<b>B.16. SmartCard</b> .....	<b>752</b>
<b>B.17. System Mode</b> .....	<b>754</b>
<b>B.18. Timer Led</b> .....	<b>756</b>
<b>B.19. USB</b> .....	<b>757</b>
<b>B.20. TCP Setting</b> .....	<b>760</b>
<b>B.21. GPRS</b> .....	<b>764</b>
<b>B.22. Modem TCP</b> .....	<b>768</b>
<b>B.23. Mifare</b> .....	<b>773</b>
<b>B.24. Mifare EEPROM</b> .....	<b>775</b>
<b>B.25. T=CL</b> .....	<b>776</b>
<b>B.26. PowerAutoMode</b> .....	<b>778</b>

# 1. Introduction

The CTOS API is an implementation presenting a consistent interface for user to develop their software on Terminal.

## **Hungarian Notation Conventions used in CTOS**

As programs have become more complex both in terms of size and of the proliferation of data types, many programmers have adopted a variable-naming convention, which is commonly referred to as Hungarian notation.

Over the past several years, several "standard" versions of Hungarian notation have been proposed and/or published. The version given here is dictated in part by personal preferences and in part by conventions established by CTOS in naming constants, variable, and data structure definitions. Because all of these standards are intended to be mnemonic for your convenience, you may follow or alter them as desired.

Using Hungarian notation, variable names begin with one or more lowercase letters that denote the variable type, thus providing an inherent identification. For example, the prefix **us** is used to identify an **unsigned short**, it referring to a 16-bits **unsigned variable**, respectively.

Next Table summarizes the Hungarian notation conventions used in CTOS.

Prefix	Data Type
<b>bo</b>	BOOLEAN
<b>pbo</b>	BOOLEAN *, pointer to boolean
<b>b</b>	BYTE
<b>ba</b>	BYTE[ ], byte array
<b>pb</b>	BYTE *, pointer to byte (PBYTE)
<b>c</b>	CHAR
<b>ca</b>	CHAR[ ], character string
<b>pc</b>	CHAR *, pointer to character string
<b>dw</b>	DWORD, double word
<b>dwa</b>	DWORD[ ], double word array
<b>pdw</b>	DWORD *, pointer to double word
<b>st</b>	Structure
<b>pst</b>	Structure *, pointer to structure
<b>ul</b>	ULONG, unsigned long
<b>ula</b>	ULONG[ ], unsigned long array
<b>pul</b>	ULONG *, pointer to unsigned long
<b>us</b>	USHORT, unsigned short
<b>usa</b>	USHORT[ ], unsigned short array
<b>pus</b>	USHORT *, pointer to unsigned short
<b>w</b>	WORD, unsigned word
<b>wa</b>	WORD[ ], unsigned word array
<b>pw</b>	WORD *, pointer to unsigned word
<b>p</b>	PBYTE, a pointer variable containing the address of a

variable. This is a pointer to a byte, used for pass-by-reference calls.

## 2. CTOS Application Programming Interface

### 2.1. System Functions

- CTOS\_GetSerialNumber
- CTOS\_GetFactorySN
- CTOS\_GetSystemInfo
- CTOS\_FactoryReset
- CTOS\_GetKeyHash
- CTOS\_SystemWait
- CTOS\_CradleAttached
- CTOS\_HWSettingGet
- CTOS\_SystemMemoryStatus

#### System Error Codes

<b>Constants</b>	<b>Value</b>
d_SYSTEM_INVALID_PARA	0201h
d_SYSTEM_HALT_FAULT	0202h
d_SYSTEM_SYS_PARA_ABSENT	0203h

## CTOS\_GetSerialNumber

---

```
USHORT CTOS_GetSerialNumber(BYTE baBuf[16] );
```

**Description** Retrieve terminal CPU hard-coded unique serial number.

**Parameters** [OUT] *baBuf*  
Buffer to store the serial number.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_SYSTEM_INVALID_PARA	0201h
	d_SYSTEM_HALT_FAULT	0202h
	d_SYSTEM_SYS_PARA_ABSENT	0203h

**Example** Please refer to example in appendix [B.5 Hardware](#)

**Note** Minimum output buffer size is 16 bytes.

## CTOS\_GetFactorySN

---

```
USHORT CTOS_GetFactorySN(BYTE baFactorySN[16]);
```

**Description** Retrieve terminal factory serial number. It matches with the stick on the back part of the terminal.

**Parameters** [ OUT ] *baFactorySN*  
Buffer to store the factory serial number. E.g: 0 0 0 0 0 1 0 5  
0 0 1 1 1 8 0 =

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_SYSTEM_INVALID_PARA	0201h
	d_SYSTEM_HALT_FAULT	0202h
	d_SYSTEM_SYS_PARA_ABSENT	0203h

**Note** Minimum output buffer size is 16 bytes.  
The last digit is the check code of factory serial number. The check code is calculated by exclusive-OR (XOR) the first 15 digits.

## CTOS\_GetSystemInfo

---

```
USHORT CTOS_GetSystemInfo (BYTE bID, BYTE baBuf[17] );
```

**Description**      Retrieve version information of specific system component.

Parameters	[ IN ]	<u>bID</u>
		<ul style="list-style-type: none"><li>• <i>ID_BOOTSULD</i></li><li>• <i>ID_CRYPTO_HAL</i></li><li>• <i>ID_KMS</i></li><li>• <i>ID_LINUX_KERNEL</i></li><li>• <i>ID_SECURITY_KO</i></li><li>• <i>ID_SYSUPD_KO</i></li><li>• <i>ID_KMODEM_KO</i></li><li>• <i>ID_CADRV_KO</i></li><li>• <i>ID_CAUSB_KO</i></li><li>• <i>ID_LIBCAUART_SO</i></li><li>• <i>ID_LIBCAUSBH_SO</i></li><li>• <i>ID_LIBCAMODEM_SO</i></li><li>• <i>ID_LIBCAETHERNET_SO</i></li><li>• <i>ID_LIBCAFONT_SO</i></li><li>• <i>ID_LIBCALCD_SO</i></li><li>• <i>ID_LIBCAPRT_SO</i></li><li>• <i>ID_LIBCARTC_SO</i></li><li>• <i>ID_LIBCAULDPM_SO</i></li><li>• <i>ID_LIBCAPMODEM_SO</i></li><li>• <i>ID_LIBCAGSM_SO</i></li><li>• <i>ID_LIBCAEMVL2_SO</i></li><li>• <i>ID_LIBCAKMS_SO</i></li><li>• <i>ID_LIBCAFS_SO</i></li><li>• <i>ID_LIBCABARCODE_SO</i></li><li>• <i>ID_CRADLE_MP</i></li><li>• <i>ID_LIBTLS_SO</i></li></ul>

- *ID\_LIBCLVW\_SO*
- *ID\_LIBCTOSAPI\_SO*
- *ID\_CASC\_KO*
- *ID\_CLVWM\_MP*
- *ID\_ROOTFS*

[ OUT] *baBuf*

Buffer to store the version information.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_SYSTEM_INVALID_PARA	0201h
	d_SYSTEM_HALT_FAULT	0202h
	d_SYSTEM_SYS_PARA_ABSENT	0203h

**Example** Please refer to example in appendix [B.5 Hardware](#)

**Note** Minimum output buffer size is 17 bytes.

## CTOS\_FactoryReset

---

```
USHORT CTOS_FactoryReset(BOOL IsNeedConfirm);
```

**Description** Initialize the terminal to the default factory settings.

**Parameters** [ IN ] *IsNeedConfirm*  
Indicating whether it requires the terminal user to confirm before performing the factory reset.

- ***d\_TRUE***

A notification page will be displayed on LCD and requires user to confirm before performing the factory reset.

- ***d\_FALSE***

Perform the factory reset directly.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_SYSTEM_INVALID_PARA	0201h
d_SYSTEM_HALT_FAULT	0202h
d_SYSTEM_SYS_PARA_ABSENT	0203h

## CTOS\_GetKeyHash

---

```
USHORT CTOS_GetKeyHash (BYTE bKeyIndex, BYTE* baHash);
```

**Description** Retrieve hash value of a key.

**Parameters** [ IN ] bKeyIndex

Key index.

[ OUT ] baHash

Hash value of key (SHA-1, 20 bytes)

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_SYSTEM_INVALID_PARA	0201h
d_SYSTEM_HALTFAULT	0202h
d_SYSTEM_SYS_PARA_ABSENT	0203h

**Note**

Minimum output buffer size is 20 bytes.

## CTOS\_SystemWait

---

```
USHORT CTOS_SystemWait (DWORD dwTimeout, DWORD dwWaitEvent,  
DWORD* pdwWakeupEvent);
```

**Description** Halt the system and wait until any specific event is triggered or specified timeout occur.

- Parameters**
- [ IN ] *dwTimeout*  
The timeout value in 10 milliseconds to leave system halt state. Value of **d\_TIME\_INFINITE**(0xFFFFFFFF) means to wait infinitely.
- [ IN ] *dwWaitEvent*  
Specify the event which will wake up the system when the event is triggered. Value is bit mask.
- ***d\_EVENT\_KBD***  
Wait until any key is pressed.
  - ***d\_EVENT\_SC***  
Wait until the smart card is inserted or withdrawn.
  - ***d\_EVENT\_MSR***  
Wait until any magnetic card is swiped.
  - ***d\_EVENT\_MODEM***  
Wait until any data is received from modem.
  - ***d\_EVENT\_ETHERNET***  
Wait until any data is received from Ethernet.
  - ***d\_EVENT\_COM1***  
Wait until any data is received from COM1.

- ***d\_EVENT\_COM2***

Wait until any data is received from COM2.

[ OUT ] ***pdwWakeupEvent***

Indicating the triggered event which waked up the system.

- ***d\_EVENT\_KBD***

Key pressed event waked up the system.

- ***d\_EVENT\_SC***

Smart card inserted or withdrawn event waked up the system.

- ***d\_EVENT\_MSR***

Magnetic card swiped event waked up the system.

- ***d\_EVENT\_MODEM***

Modem data received event waked up the system.

- ***d\_EVENT\_ETHERNET***

Ethernet data received event waked up the system.

- ***d\_EVENT\_COM1***

COM1 data received event waked up the system.

- ***d\_EVENT\_COM2***

COM2 data received event waked up the system.

#### Return Value

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_SYSTEM_INVALID_PARA	0201h
d_SYSTEM_HALT_FAULT	0202h
d_SYSTEM_SYS_PARA_ABSENT	0203h

## CTOS\_CradleAttached

---

```
USHORT CTOS_CradleAttached( void );
```

**Description**      Detect whether the handset is attached on a cradle.

**Parameters**      None

**Return Value**

- ***d\_YES***  
Handset is attached on the cradle.
- ***d\_NO***  
Handset is not attached on the cradle.

## CTOS\_HWSettingGet

---

USHORT

```
CTOS_HWSettingGet (BOOL*fPortable, BOOL*fPCI, USHORT*mkHWSupport);
```

**Description** Get current hardware configuration.

**Parameters** [ OUT ] *fPortable*

Pointer to BOOL for storing whether the terminal is Portable type or Countertop type.

- **d\_TRUE**

Terminal is Portable type.

- **d\_FALSE**

Terminal is Countertop type

[ OUT ] *fPCI*

Pointer to BOOL for storing whether the terminal is PCI or non PCI type.

- **d\_TRUE**

Terminal is PCI type.

- **d\_FALSE**

Terminal is non PCI type.

[ OUT ] *mkHWSupport*

Pointer to USHORT for storing the hardware support by the terminal.

- **d\_MK\_HW\_MODEM**

Modem is supported.

- **d\_MK\_HW\_ETHERNET**

Ethernet is supported.

- ***d\_MK\_HW\_GPRS***

Mobile communication device (GPRS/CDMA/UMTS) is supported.

- ***d\_MK\_HW\_CONTACTLESS***

Contactless reader is supported.

- ***d\_MK\_HW\_ZIGBEE***

Zigbee is supported.

- ***d\_MK\_HW\_WIFI***

Wifi is supported.

- ***d\_MK\_HW\_BT***

Bluetooth is supported.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_SYSTEM_INVALID_PARA	0201h
d_SYSTEM_HALT_FAULT	0202h
d_SYSTEM_SYS_PARA_ABSENT	0203h

## CTOS\_SystemMemoryStatus

---

USHORT

```
CTOS_SystemMemoryStatus(ULONG*ulUsedDiskSize , ULONG*ulTotalDiskSize,
ULONG*ulUsedRamSize, ULONG*ulTotalRamSize);
```

**Description** Retrieve system memory information.

**Parameters**

- [ OUT] *ulUsedDiskSize*  
Pointer to ULONG for storing the size of used non-volatile disk memory..

- [ OUT ] *ulTotalDiskSize*  
Pointer to ULONG for storing the size of total non-volatile disk memory in the system.

- [ OUT ] *ulUsedRamSize*  
Pointer to ULONG for storing the size of used RAM.

- [ OUT ] *ulTotalRamSize*  
Pointer to ULONG for storing the size of total RAM in the system.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_SYSTEM_INVALID_PARA	0201h
d_SYSTEM_HALT_FAULT	0202h
d_SYSTEM_SYS_PARA_ABSENT	0203h

## CTOS\_DeviceModelGet

---

```
USHORT CTOS_DeviceModelGet(BYTE *bModel);
```

**Description** Retrieve the model of the terminal.

**Parameters** [ OUT ] bModel :  
d\_Model\_VEGA9000  
d\_Model\_VEGA7000  
d\_Model\_VEGA5000  
d\_Model\_VEGA5000S  
d\_Model\_VEGA3000  
d\_Model\_VEGA3000P

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_NOT_SUPPORT	FF70h

## 2.2. Power Management Functions

- CTOS\_PowerSource
- CTOS\_SystemReset
- CTOS\_PowerOff
- CTOS\_PowerMode
- CTOS\_PowerModeEX
- CTOS\_PowerAwakening
- CTOS\_PowerAutoModeEnable
- CTOS\_PowerAutoModeDisable
- CTOS\_PowerAutoModeTimeToStandby
- CTOS\_PowerAutoModeTimeToSleep
- CTOS\_PowerAutoModeRegisterCallback
- CTOS\_PowerAutoModeUNRegisterCallback
- CTOS\_PowerWakeupSrcGet
- CTOS\_PowerWakeupSrcSet

### Power Management Error Codes

<b>Constants</b>	<b>Value</b>
d_POWER_INVALID_PARA	3001h
d_POWER_NOT_SUPPORT	3002h
d_POWER_INSUFFICIENT_RESOURCE	3003h
d_POWER_ALREADY_REGISTERED	3004h

## CTOS\_PowerSource

---

```
USHORT CTOS_PowerSource (UCHAR*bSrc) ;
```

**Description** Get the power source of terminal.

**Parameters** [ OUT ] *bSrc*

Power sources:

- *d\_PWRSRC\_DCJACK*
- *d\_PWRSRC\_CRADLE*
- *d\_PWRSRC\_BATTERY*
- *d\_PWRSRC\_USB*

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PowerManagementErrorCodes</a>	30xxh

**Note**

*d\_PWRSRC\_USB* is available only for Vega3000

## CTOS\_SystemReset

---

```
void CTOS_SystemReset( void );
```

**Description** Soft reset the system. Calling this function will reset all the peripherals, close all opened files, and terminate all connected sessions.

**Parameters** None

**Return Value** None

**Example** Please refer to example in appendix [B.17 System Mode](#)

## CTOS\_PowerOff

---

```
void CTOS_PowerOff( void );
```

**Description** Power off the terminal.

**Parameters** None

**Return Value** None

**Note** Power off feature is not supported for countertop type terminal.

**Example** Please refer to example in appendix [B.17 System Mode](#)

## CTOS\_PowerMode

---

USHORT CTOS\_PowerMode (BYTE bMode) ;

**Description** Manage the power state of the system. The function is for power management.

<b>Parameters</b>	[ IN ] <i>bMode</i>
	Power mode to set.
	<ul style="list-style-type: none"> <li>• <b><i>d_PWR_STANDBY_MODE</i></b> Enter the standby mode. Under this mode, system will be put to power saving mode, but peripherals are all remain powered. System can be woken up by swiping MSR, inserting or removing the ICC or pressing any key.</li> </ul>
	<ul style="list-style-type: none"> <li>• <b><i>d_PWR_SLEEP_MODE</i></b> Enter the sleep mode. Under this mode, system will be put to power saving mode, and peripherals are all power off. All communication connections will be disconnected, and all transaction will be aborted. System can be woken up by pressing any key.</li> </ul>
	<ul style="list-style-type: none"> <li>• <b><i>d_PWR_REBOOT</i></b> Reboot the system.</li> </ul>
	<ul style="list-style-type: none"> <li>• <b><i>d_PWR_POWER_OFF</i></b> Power off the system.</li> </ul>

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PowerManagementErrorCodes</a>	30xxh

## CTOS\_PowerModeEX

---

USHORT CTOS\_PowerModeEX(BYTE bMode, USHORT \*usWakeSrc);

**Description** Manage the power state of the system. The function is for power management. Users can get the terminal wake up source when terminal wake up.

**Parameters** [ IN ] bMode

Power mode to set.

- **d\_PWR\_STANDBY\_MODE**

Enter the standby mode. Under this mode, system will be put to power saving mode, but peripherals are all remain powered.

System can be woken up by swiping MSR, inserting or removing the ICC or pressing any key.

- **d\_PWR\_SLEEP\_MODE**

Enter the sleep mode. Under this mode, system will be put to power saving mode, and peripherals are all power off. All communication connections will be disconnected, and all transaction will be aborted.

System can be woken up by pressing any key.

[ OUT ] usWakeSrc

Return the wake up source.

- **d\_MK\_WAKE\_KBD**

If terminal is woken up by pressing any key.

- **d\_MK\_WAKE\_SC**

If terminal is woken up by inserting smart card.

- **d\_MK\_WAKE\_MSR**

If terminal is woken up by swiping card.

- **d\_MK\_WAKE\_RTC**

If terminal is woken up by RTC. Users can set due time by **CTOS\_PowerAwakening()** for awaking terminal up from standby mode or sleep mode.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#"><u>PowerManagementErrorCodes</u></a>	30xxh

## CTOS\_PowerModeOptSet

---

```
USHORT CTOS_PowerModeOptSet (UCHAR bMode, ULONG ulTag, ULONG ulOption);
```

**Description** Set the peripheral behavior after entering power mode.

**Parameters** [ IN ] *bMode*

Power mode to set.

- ***d\_PWR\_STANDBY\_MODE***
- ***d\_PWR\_SLEEP\_MODE***

[ IN ] *ulTag*

Peripheral tag.

- ***d\_PWR\_TAG\_LED\_G***
- ***d\_PWR\_TAG\_LED\_B***
- ***d\_PWR\_TAG\_LCD\_BRIGHTNESS***
- ***d\_PWR\_TAG\_LCD\_CONTRAST***

[ IN ] *ulOption*

Peripheral power status options.

- ***d\_PWR\_OPT\_KEEP***
- ***d\_PWR\_OPT\_ON***
- ***d\_PWR\_OPT\_OFF***

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">PowerManagementErrorCodes</a>	30xxh

**Note** For LCD brightness and contrast, a value ranged from 0 to 0xFF can be set to *ulOption* except for d\_PWR\_OPT\_KEEP, d\_PWR\_OPT\_ON, and d\_PWR\_OPT\_OFF.

## CTOS\_PowerModeOptGet

---

```
USHORT CTOS_PowerModeOptSet (UCHAR bMode, ULONG ulTag, ULONG *ulOption);
```

**Description** Get the peripheral power status when being in power mode.

**Parameters** [ IN ] bMode

Power mode to set.

- **d\_PWR\_STANDBY\_MODE**
- **d\_PWR\_SLEEP\_MODE**

[ IN ] ulTag

Peripheral tag.

- **d\_PWR\_TAG\_LED\_G**
- **d\_PWR\_TAG\_LED\_B**
- **d\_PWR\_TAG\_LCD\_BRIGHTNESS**
- **d\_PWR\_TAG\_LCD\_CONTRAST**

[ OUT ] ulOption

Peripheral power status options.

- **d\_PWR\_OPT\_KEEP**
- **d\_PWR\_OPT\_ON**
- **d\_PWR\_OPT\_OFF**

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">PowerManagementErrorCodes</a>	30xxh

**Note** For LCD brightness and contrast, a value ranged from 0 to 0xFF can be gotten from *ulOption* except for d\_PWR\_OPT\_KEEP, d\_PWR\_OPT\_ON, and d\_PWR\_OPT\_OFF.

## CTOS\_PowerAwakening

---

```
USHORT CTOS_PowerAwakening(SHORT sSecond, SHORT sMinute, SHORT sHour,
SHORT sDay, SHORT sMonth, SHORT sYear);
```

**Description** Set the time and date information to wake the terminal up from standby mode or sleep mode by the system Real Time Clock.

<b>Parameters</b>	[ IN ] <u>sSecond</u> Second of RTC.
	[ IN ] <u>sMinute</u> Minute of RTC.
	[ IN ] <u>sHour</u> Hour of RTC.
	[ IN ] <u>sDay</u> Day of RTC.
	[ IN ] <u>sMonth</u> Month of RTC.
	[ IN ] <u>sYear</u> Year of RTC.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">PowerManagementErrorCodes</a>	30xxh

## CTOS\_PowerAutoModeEnable

---

```
USHORT CTOS_PowerAutoModeEnable(void);
```

**Description** Enable the Auto Power Management of the system. Developer have to call this API before using any other Auto Power Management API. The Auto Power Management is disabled in default. In case Auto Power Management is enabled, the system will get into Standby Mode in 30 seconds by default if there were no keyboard event, smart card event or MSR event. User can also enable the Auto Power Management in System Setting manually.

**Parameters** None

**Note** User must check the error code while using all Auto Power Management API. If any error is returned, the original setting for power mode will not be changed.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PowerManagementErrorCodes</a>	30xxh

**Example**

Please refer to example in appendix [PowerAutoMode](#)

## CTOS\_PowerAutoModeDisable

---

```
USHORT CTOS_PowerAutoModeDisable(void);
```

**Description** Disable the Auto Power Management of the system. User can also disable the Auto Power Management in System Setting manually.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PowerManagementErrorCodes</a>	30xxh

**Example** Please refer to example in appendix [PowerAutoMode](#)

## CTOS\_PowerAutoModeTimeToStandby

---

```
USHORT CTOS_PowerAutoModeTimeToStandby(ULONG ulSec);
```

**Description** Set the idle time to enter the Standby Mode of Auto Power Management.

**Parameters** [ IN ] ulSec Idle period (unit: second).

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PowerManagementErrorCodes</a>	30xxh

**Note** Currently only one of Standby Mode or Sleep Mode will be entered. If idle time to enter the Standby Mode and idle time to enter Sleep Mode are both not zero, then the mode with less idle time will be entered.

**Example** Please refer to example in appendix [PowerAutoMode](#)

## CTOS\_PowerAutoModeTimeToSleep

---

USHORT CTOS\_PowerAutoModeTimeToSleep(ULONG ulSec);

**Description** Set the idle time to enter Sleep Mode of Auto Power Management.

**Parameters** [ IN ] ulSec  
Idle period (unit: second).

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">PowerManagementErrorCodes</a>	30xxh

**Note** Currently only one of Standby Mode or Sleep Mode will be entered. If idle time to enter the Standby Mode and idle time to enter Sleep Mode are both not zero, then the mode with less idle time will be entered.

**Example** Please refer to example in appendix [PowerAutoMode](#)

## CTOS\_PowerAutoModeRegisterCallback

---

```
USHORT CTOS_PowerAutoModeRegisterCallback(CTOS_stPowerModeCallback
*stCallback);
```

**Description** Register for the Callback function of Auto Power Management. Arrange what the Callback function should be called before get into power saving mode or after exit power saving mode (Standby Mode / Sleep Mode).

**Parameters** [ IN ] stCallback

```
typedef struct
{
    eventEnterPowerMode OnEnterPowerMode;
    void *pEnterPowerModeData;

    eventExitPowerMode OnExitPowerMode;
    void *pExitPowerModeData;

} CTOS_stPowerModeCallback;
```

Initialize the Callback function with struct (CTOS\_stPowerModeCallback).

```
typedef BOOL (*eventEnterPowerMode)(void*
pData, BYTE bMode);
```

Set the events (eventEnterPowerMode) need to be done and the data (\*pEnterPowerModeData) need to transfer before enter Power Mode.

```
typedef BOOL (*eventExitPowerMode)(void*
pData, BYTE bMode, USHORT
*usWakeSrc);
```

Set the events (eventEnterPowerMode) need to be done and the data (\*pEnterPowerModeData) need to transfer after exit Power Mode.

.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#"><u>PowerManagementErrorCodes</u></a>	30xxh

**Example**

Please refer to example in appendix [PowerAutoMode](#)

## CTOS\_PowerAutoModeUNRegisterCallback

---

```
USHORT CTOS_PowerAutoModeUNRegisterCallback(void);
```

**Description** Unregister the registered Callback function of the Auto Power Management.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PowerManagementErrorCodes</a>	30xxh

**Example** Please refer to example in appendix [PowerAutoMode](#)

## CTOS\_PowerWakeupSrcGet

---

```
USHORT CTOS_PowerWakeupSrcGet (UCHAR bMode, UCHAR* baWakeupSrc, UCHAR*  
bWakeupSrcNum) ;
```

**Description** Get the allowable wake up source information form specific power mode.

**Parameters** [ IN ] *bMode*

Power mode to set.

- *d\_PWR\_STANDBY\_MODE*
- *d\_PWR\_SLEEP\_MODE*

[ OUT ] *baWakeupSrc*

- *d\_PWR\_WAKEUP\_SRC\_ALL*
- *d\_PWR\_WAKEUP\_SRC\_KBD\_ALL*
- *d\_PWR\_WAKEUP\_SRC\_SC*
- *d\_PWR\_WAKEUP\_SRC\_MSR*
- *d\_PWR\_WAKEUP\_SRC\_BLUETOOTH*

Only for the device type which support Bluetooth.

- *d\_PWR\_WAKUEP\_SRC\_KBD\_1*
- *d\_PWR\_WAKUEP\_SRC\_KBD\_2*
- *d\_PWR\_WAKUEP\_SRC\_KBD\_3*
- *d\_PWR\_WAKUEP\_SRC\_KBD\_4*
- *d\_PWR\_WAKUEP\_SRC\_KBD\_5*
- *d\_PWR\_WAKUEP\_SRC\_KBD\_6*
- *d\_PWR\_WAKUEP\_SRC\_KBD\_7*
- *d\_PWR\_WAKUEP\_SRC\_KBD\_8*
- *d\_PWR\_WAKUEP\_SRC\_KBD\_9*
- *d\_PWR\_WAKUEP\_SRC\_KBD\_0*

All keys can be set to wake up the device individually.

[ OUT ] *bWakeupSrcNum*

The number of wake up source  
( baWakeupSrc ).

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#"><u>PowerManagementErrorCodes</u></a>	30xxh

## CTOS\_PowerWakeupSrcSet

---

```
USHORT CTOS_PowerWakeupSrcSet( UCHAR bMode, UCHAR* baWakeupSrc, UCHAR  
bWakeupSrcNum);
```

**Description** Set the allowable wake up source to specific power saving mode.

**Parameters** [ IN ] *bMode*

Power mode to set.

- ***d\_PWR\_STANDBY\_MODE***
- ***d\_PWR\_SLEEP\_MODE***

[ IN ] *baWakeupSrc*

Wake up source to set.

- ***d\_PWR\_WAKEUP\_SRC\_ALL***
- ***d\_PWR\_WAKEUP\_SRC\_KBD\_ALL***
- ***d\_PWR\_WAKEUP\_SRC\_SC***
- ***d\_PWR\_WAKEUP\_SRC\_MSR***
- ***d\_PWR\_WAKEUP\_SRC\_BLUETOOTH***

Only for the device type which support Bluetooth.

- ***d\_PWR\_WAKUEP\_SRC\_KBD\_1***
- ***d\_PWR\_WAKUEP\_SRC\_KBD\_2***
- ***d\_PWR\_WAKUEP\_SRC\_KBD\_3***
- ***d\_PWR\_WAKUEP\_SRC\_KBD\_4***
- ***d\_PWR\_WAKUEP\_SRC\_KBD\_5***
- ***d\_PWR\_WAKUEP\_SRC\_KBD\_6***
- ***d\_PWR\_WAKUEP\_SRC\_KBD\_7***
- ***d\_PWR\_WAKUEP\_SRC\_KBD\_8***
- ***d\_PWR\_WAKUEP\_SRC\_KBD\_9***
- ***d\_PWR\_WAKUEP\_SRC\_KBD\_0***

All keys can be set to wake up the device individually.

[ IN ]    *bWakeupSrcNum*

The number of wake up source  
( baWakeupSrc ).

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#"><u>PowerManagementErrorCodes</u></a>	30xxh

**Note**

For Sleep Mode, only keypads can wake up the system, and user must set at least one key as the wake up source.

For Standby Mode, all defined source (keypad, smartcard, msr and Bluetooth) can be set.

## 2.3. Battery Functions

- CTOS\_BatteryGetCapacity
- CTOS\_BatteryForceCharge
- CTOS\_BatterySetChargeThreshold
- CTOS\_BatteryGetChargeThreshold
- CTOS\_BatteryStatus

### Battery Error Codes

<b>Constants</b>	<b>Value</b>
d_BATTERY_INVALID_PARA	5601h
d_BATTERY_NOT_EXIST	5602h
d_BATTERY_SN_NOT_INSTALL	5603h
d_BATTERY_NO_EXT_POWER	5604h
d_BATTERY_BUSY	5605h
d_BATTERY_INVALID	5606h
d_BATTERY_NOT_SUPPORT	5607h

## CTOS\_BatteryGetCapacity

---

```
USHORT CTOS_BatteryGetCapacity(BYTE*bPercentage);
```

**Description** Get the battery capacity.

**Parameters** [ OUT] *bPercentage*  
Battery capacity in percentage. The range is from 0 to 100.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#"><u>BatteryErrorCode</u></a>	56xxh

## CTOS\_BatteryForceCharge

---

```
USHORT CTOS_BatteryForceCharge( void );
```

**Description** Force to enable the charger to charge the battery.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">BettryErrorCode</a>	56xxh

## CTOS\_BatterySetChargeThreshold

---

```
USHORT CTOS_BatterySetChargeThreshold (UCHARbPercentage);
```

**Description** Set the threshold of battery capacity for enabling the charger. While the capacity of the battery is below the threshold, the charger will be enabled to charge the battery.

**Parameters** [IN] *bPercentage*  
Threshold to enable the charger in percentage. The range is from 0 to 90.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">BettryErrorCode</a>	56xxh

**Note** The threshold is default 80%, which means as long as the battery capacity is under 80%, the charger will be enabled to charge the battery.

## CTOS\_BatteryGetChargeThreshold

---

```
USHORT CTOS_BatteryGetChargeThreshold(UCHAR*bPercentage);
```

**Description** Get the current threshold of battery capacity for enabling the charger.

**Parameters** [ OUT ] *bPercentage*  
Current threshold to enable the charger in percentage. The range is from 0 to 90.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">BettryErrorCode</a>	56xxh

## CTOS\_BatteryStatus

---

USHORT CTOS\_BatteryStatus (DWORD\*pdwStatus) ;

**Description** Get the current status of battery.

**Parameters** [ OUT ] *pdwStatus*

Pointer of DWORD for storing the current status of battery. Status is returned in the bitwise-OR of following bitmask.

- **d\_MK\_BATTERY\_EXIST**

Whether the battery is present or not.

- **d\_MK\_BATTERY\_CHARGE**

Whether the battery is charging or not.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_BATTERY_NOT_SUPPORT	5607h

**Note**

The return value of CTOS\_BatteryStatus() will be d\_OK if the terminal type is portable type and it will return d\_BATTERY\_NOT\_SUPPORT while the terminal type is countertop type.

## 2.4. Program Manager Functions

- CTOS\_APFind
- CTOS\_APGet
- CTOS\_APSet
- CTOS\_APDel
- CTOS\_APJump
- CTOS\_APCall
- CTOS\_APFork

### Program Manager Error Codes

<b>Constants</b>	<b>Value</b>
d_PM_INVALID_PARA	2C01h
d_PM_INVALID_AP_TABLE	2C02h
d_PM_AP_NOT_FOUND	2C03h
d_PM_INVALID_AP	2C04h
d_PM_SPACE_NOT_ENOUGH	2C05h
d_PM_REACH_MAX_NO_AP	2C06h
d_PM_NO_AP_EXECUTED	2C07h
d_PM_AP_ZERO_LENGTH	2C08h
d_PM_INVALID_TYPE	2C09h
d_PM_SHARD_LIBRARY_OPEN	2C0Ah
d_PM_UNEXPECTED_ERROR	2C0Bh

## CTOS\_APFind

---

```
USHORT CTOS_APFind(BYTE *baAPName, USHORT*usAPIIndex);
```

**Description** Get index of AP by name.

**Parameters** [IN] *baAPName*

Name of AP.

[ OUT ] *usAPIIndex*

Index of AP.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Program Manager error codes</a>	2Cxxh

## CTOS\_APGet

---

```
USHORT CTOS_APGet (USHORT usAPIIndex, CTOS_stCAPInfo *stInfo);
```

**Description** Get information of AP.

**Parameters** [ IN ] [usAPIIndex](#)

Index of AP.

[ OUT ] [stInfo](#)

Pointer to **CTOS\_stCAPInfo** structure for storing the information of AP.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<u><a href="#">Program Manager error codes</a></u>	2Cxxh

**Note**

AP index can be retrieved from calling **CTOS\_APFind()** with input AP name.

## CTOS\_APSet

---

```
USHORT CTOS_APSet (USHORT usAPIIndex, BYTE bFlag);
```

**Description** Set AP attribute.

**Parameters** [ IN ] *usAPIIndex*

AP index.

[ IN ] *bFlag*

AP attribute to set.

- ***d\_AP\_FLAG\_DEF\_SEL***

Set the AP as default running AP on system startup.

- ***d\_AP\_FLAG\_DEF\_UNSEL***

Unset the AP as default running AP on system startup.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Program Manager error codes</a>	2Cxxh

## CTOS\_APDel

---

```
USHORT CTOS_APDel (USHORT usAPIndex) ;
```

**Description** Delete AP.

**Parameters** [ IN ] [usAPIndex](#)  
AP index.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Program Manager error codes</a>	2Cxxh

**Note** AP index can be retrieved from calling **CTOS\_APFind()** with input AP name.

## CTOS\_APJump

---

```
USHORT CTOS_APJump (BYTE *baAPName) ;
```

**Description** Set next AP by name to be executed by Program Manager (PM) after the current AP exits.

**Parameters** [ IN ] *baAPName*  
AP name to jump.

Return Value	<i>Constants</i>	<i>Value</i>
	d_OK	0000h
	<a href="#">Program Manager error codes</a>	2Cxxh

**Note** If AP1 calls CTOS\_APJump(AP2),  
and AP2 calls CTOS\_APJump(AP3),  
the execution flow will be AP1 -> PM -> AP2 -> PM -> AP3 -> PM

## CTOS\_APCall

---

```
USHORT CTOS_APCall (BYTE *baAPName) ;
```

**Description** Set next AP by name to be executed by Program Manager (PM) after the current AP exit. The current AP will be put to an "AP Stack", and once the next AP exits, the current AP will be "returned" to execution by Program Manager.

**Parameters** [ IN ] [baAPName](#)  
AP name to call.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<u><a href="#">Program Manager error codes</a></u>	2Cxxh

**Note** An AP stack is maintained, thus if AP1 calls CTOS\_APCall(AP2) and AP2 calls CTOS\_APCall(AP3), the execution flow will be  
AP1 -> PM -> AP2 -> PM -> AP3 -> PM -> AP2 -> PM -> AP1 -> PM

## CTOS\_APFork

---

```
USHORT CTOS_APFork (BYTE *baAPName);
```

**Description** Execute other applications that are existing on terminal without exiting the caller application itself. Two or more processes created by this API will run at the same time.

**Parameters** [ IN ] *baAPName*  
Application to be executed.

Return Value	<i>Constants</i>	<i>Value</i>
	d_OK	0000h
	<a href="#">Program Manager error codes</a>	2Cxh

**Note** Processes can use the [IPC](#) to coordinate the CTOS resource.

## 2.5. User Loader Functions

- CTOS\_EnterDownloadMode
- CTOS\_UpdateFromMMCI
- CTOS\_SetPMEEnterPassword
- CTOS\_SetFunKeyPassword
- CTOS\_SetULDPassword
- CTOS\_SealULD

### User Loader Error Codes

<b>Constants</b>	<b>Value</b>
d_ULD_INVALID_PARA	2A01h
d_ULD_PORT_OPEN_FAILED	2A02h
d_ULD_KEY_NOT_RECEIVED	2A03h
d_ULD_VERIFY_FAILED	2A04h
d_ULD_SIGN_NOT_RECEIVED	2A05h
d_ULD_HEADER_NOT_RECEIVED	2A06h
d_ULD_FILE_NOT_READY	2A07h
d_ULD_RECEIVE_ERROR	2A08h
d_ULD_RECEIVE_TIMEOUT	2A09h
d_ULD_TOTAL_TIMEOUT	2A0Ah
d_ULD_LOCKED	2A0Bh
d_ULD_INVALID	2A0Ch
d_ULD_PASSWORD_INCORRECT	2A0Dh
d_ULD_AP_NOT_EXIST	2A0Eh
d_ULD_TIMEOUT	2A40h

## CTOS\_EnterDownloadMode

---

```
voidCTOS_EnterDownloadMode (void);
```

<b>Description</b>	Enter the User Loader mode which allows the user to load files (applications, fonts, firmware, etc.) by USB/RS232 cable, external USB drive or uSD memory card.
<b>Parameters</b>	None
<b>Return Value</b>	None
<b>Example</b>	Please refer to example in appendix <a href="#">B.17 System Mode</a>

## CTOS\_UpdateFromMMCI

---

```
void CTOS_UpdateFromMMCI (BYTE *baMMCI, BYTE bShowUI);
```

**Description** Loadfiles (applications, firmware, fonts, etc.) by using MMCI format files.

**Parameters**

- [ IN ] *baMMCI*  
Full path of MMCI file.

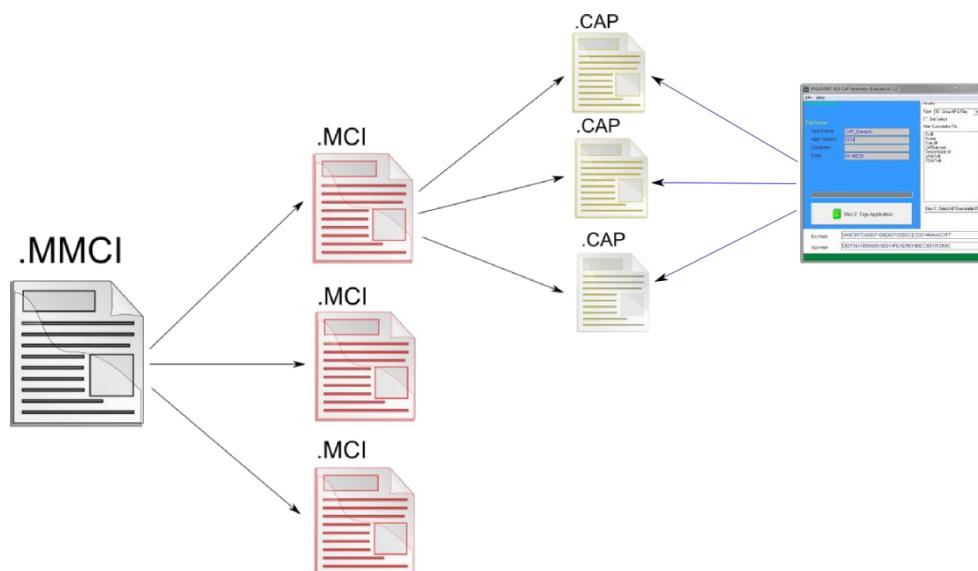
- [ IN ] *bShowUI*  
Whether to show default progress messages on LCD.  
= 1 : Display default progress messages on LCD.  
= 0 : Do not display anything on LCD.

**Return Value** None

**Note** MMCI stands for Multiple MCI. They are plain text files that can be created/modified with any text editor (e.g: Notepad++).

A MMCI file contains a list of MCI files that will be used during the loading procedure.

On the other hand, a MCI file includes a list of CAP files (files that have been signed using the CAP Generator tool).



**Note 2**

If Rootfs is included as part of the .mmci/mci it has to be at the end.  
Otherwise, the whole update will be interrupted after Rootfs is loaded.

**Correct:**

User\userapp.mci  
User\usersharedlib.mci  
User\userfont.mci  
Linux\_Rootfs\_A117\_ULD\update\_all\_1103.mci

**Incorrect:**

User\userapp.mci  
Linux\_Rootfs\_A117\_ULD\update\_all\_1103.mci  
User\usersharedlib.mci  
User\userfont.mci

## CTOS\_UpdateGetResult

---

`USHORT CTOS_UpdateGetResult( void );`

**Description** Check last update result status for **CTOS\_UpdateFromMMCI()**.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_PM_MMCI_NOT_FOUND	0001h
d_PM_MCI_NOT_FOUND	0002h
d_PM_COPY_FILE_ERROR	0003h
d_CAP_OPEN_FILE_FAILED	0004h
d_CAP_FILE_FORMAT_ERROR	0005h
d_CAP_EKEY_INCORRECT	0006h
d_CAP_ESIGN_INCORRECT	0007h
d_CAP_HEADER2_INCORRECT	0008h
d_CAP_NOT_SIGNED	0009h
d_CAP_CA_KEY_REQUIRED	000Ah
d_CAP_SIGN_KEY_INCORRECT	000Bh
d_CAP_CAP_HASH_INCORRECT	000Ch
d_CAP_DECOMPRESS_ERROR	000Dh
d_CAP_WRONG_KEY	000Eh
d_CAP_UPDATE_FINISHED	0064h
d_CAP_RESET_REQUIRED	0065h
d_CAP_REBOOT_REQUIRED	0066h
d_CAP_CONTINUE_DOWNLOAD	0067h
<a href="#">User Loader error codes</a>	2Axxh

## CTOS\_SetPMEnterPassword

---

```
USHORT CTOS_SetPMEnterPassword(BYTE *baPasswordString, BOOL  
IsEnable);
```

**Description** Set the password for entering Program Manager.

**Parameters**

[ IN ]	<i>baPasswordString</i>
	Password to set. The length of password should be between 4 to 8 bytes, and format should be in ASCII digit, range from '0' to '9'.
[ IN ]	<i>IsEnable</i>
	Enable/Disable the password for entering Program Manager.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">User Loader error codes</a>	2Axxh

**Note** Enabling the password for Program Manager will prompt for entering the password before entering the Program Manager.

## CTOS\_SetFunKeyPassword

```
USHORT CTOS_SetFunKeyPassword(BYTE *baPasswordString, BOOL IsEnable);
```

**Description** Set the password for function key in Program Manager.

**Parameters**

[ IN ]	<i>baPasswordString</i>
	Password to set. The length of password should be between 4 to 8 bytes, and format should be in ASCII digit, range from '0' to '9'.
[ IN ]	<i>IsEnable</i>
	Enable/Disable the password for function key.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">User Loader error codes</a>	2Axxh

**Note** Enabling the password for function key will prompt for entering the password after the function key is pressed in Program Manager

## CTOS\_SetULDPassword

---

```
USHORT CTOS_SetULDPassword(BYTE *baPasswordString);
```

**Description** Set the password for sealing/unsealing the ULD.

**Parameters** [ IN ] *baPasswordString*  
Password to set. The length of password should be between 4 to 8 bytes, and format should be in ASCII digit, range from '0' to '9'.

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">User Loader error codes</a>	2Axxh

## CTOS\_SealULD

---

USHORT CTOS\_SealULD(BYTE \*baPasswordString, BOOL IsEnable);

**Description** Seal the ULD to avoid the user to enter ULD by calling **CTOS\_EnterDownloadMode()**.

**Parameters** [ IN ] *baPasswordString*  
Password to verify.

[ IN ] *IsEnable*  
Whether to seal the ULD.  
= 1 : Seal the ULD.  
= 0 : Unseal the ULD.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">User Loader error codes</a>	2Axxh

**Note** To seal the ULD, a correct password must be input for verification, then the user will be blocked to enter the ULD when calling **CTOS\_EnterDownloadMode()**. To unseal the ULD, correct password still need to be input to allow the user to enter the ULD when calling **CTOS\_EnterDownloadMode()**. The password can be set by calling **CTOS\_SetULDPassword()**.

## 2.6. LED and Backlight Functions

- CTOS\_LEDSet
- CTOS\_BackLightSet
- CTOS\_BackLightSetEx

### LED and Backlight Error Codes

<b>Constants</b>	<b>Value</b>
d_LED_INVALID_PARA	0E01h
d_LED_HAL_FAULT	0E02h
d_BKLIT_NOT_SUPPORT	1F01h
d_BKLIT_INVALID_PARA	1F02h

## CTOS\_LEDSet

---

```
USHORT CTOS_LEDSet (BYTE bLED, BYTE bOnOff );
```

**Description** Turn on or off one specific LED.

**Parameters** [ IN ] bLED

Index of LED.

- **d\_LED1 or d\_LED\_RED**

Red LED

- **d\_LED2 or d\_LED\_ORANGE**

Orange LED

- **d\_LED3 or d\_LED\_GREEN**

Green LED

[ IN ] bOnOff

- **d\_ON**

- **d\_OFF**

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_LED_INVALID_PARA	0E01h
d_LED_HAL_FAULT	0E02h

**Example** Please refer to example in appendix [B.5 Hardware](#) or [B.18 Timer Led](#)

**Note** This function has no effect in Vega3000

## CTOS\_BackLightSet

---

```
USHORT CTOS_BackLightSet(BYTEbDevice, BYTE bOnOff );
```

**Description** Turn on / off the backlight of LCD or keypad.

**Parameters**

[ IN ]	<u>bDevice</u>
LCD or Keypad backlight.	
<ul style="list-style-type: none"> <li>• <u>d_BKLIT_LCD</u></li> <li>• <u>d_BKLIT_KBD</u></li> </ul>	

[ IN ]	<u>bOnOff</u>
<ul style="list-style-type: none"> <li>• <u>d_ON</u></li> <li>• <u>d_OFF</u></li> </ul>	

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_BKLIT_NOT_SUPPORT	1F01h
d_BKLIT_INVALID_PARA	1F02h

**Example** Please refer to example in appendix [B.2 Ethernet](#) or [B.3 File System](#) or [B.8 Keypad](#) or [B.10 LCD Display](#) or [B.11 Memory Card](#) or [B.13 MSReader](#) or [B.16 SmartCard](#)

## CTOS\_BackLightSetEx

---

```
USHORT CTOS_BackLightSetEx(BYTE bDevice, BYTE bOnOff, DWORD dwDuration);
```

**Description** Turn on the backlight of LCD or keypad for a specific duration.

**Parameters**

[ IN ]	<i>bDevice</i>
	LCD or Keypad backlight.
	<ul style="list-style-type: none"> <li>• <i>d_BKLIT_LCD</i></li> <li>• <i>d_BKLIT_KBD</i></li> </ul>

[ IN ]	<i>bOnOff</i>
	<ul style="list-style-type: none"> <li>• <i>d_ON</i></li> <li>• <i>d_OFF</i></li> </ul>

[ IN ]	<i>dwDuration</i>
	Duration in 10ms.

Return Value	<i>Constants</i>	<i>Value</i>
	d_OK	0000h
	d_BKLIT_NOT_SUPPORT	1F01h
	d_BKLIT_INVALID_PARA	1F02h

**Example** Please refer to example in appendix [B.13 MSReader](#)

## 2.7. Clock and Time Functions

- CTOS\_RTCGet
- CTOS\_RTCSet
- CTOS\_Delay
- CTOS\_TickGet
- CTOS\_TimeOutSet
- CTOS\_TimeOutCheck

### Clock and Time Error Codes

<b>Constants</b>	<b>Value</b>
d_TIMER_INVALID_PARA	0401h
d_TIMER_HAL_FAULT	0402h
d_RTC_INVALID_PARA	0601h
d_RTC_HAL_FAULT	0602h
d_RTC_TZ_OVERRANGE	0603h

## CTOS\_RTCGet

---

```
USHORT CTOS_RTCGet (CTOS_RTC *pstRTC) ;
```

**Description** Get the time and date information from the system Real Time Clock.

**Parameters**

[OUT] <i>pstRTC</i>	Pointer to a <b>CTOS_RTC</b> structure for storing the time and date information.
---------------------	---

```
typedef struct
{
    BYTE   bSecond;
    BYTE   bMinute;
    BYTE   bHour;
    BYTE   bDay;
    BYTE   bMonth;
    BYTE   bYear;
    BYTE   bDoW;

    //The day of the week (starting the week on Sunday)
} CTOS_RTC;
```

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_RTC_INVALID_PARA	0601h
	d_RTC_HAL_FAULT	0602h
	d_RTC_TZ_OVERRANGE	0603h

**Example** Please refer to example in appendix [B.5 Hardware](#) or [B.7 Keyboard](#)

**Note** Before calling this function, programmer should make sure that the time and date information are properly set.

## CTOS\_RTCSet

---

USHORT CTOS\_RTCSet (CTOS\_RTC \*pstRTC) ;

**Description** Set the time and date information to the system Real Time Clock.

**Parameters** [IN] *pstRTC*  
Pointer to a **CTOS\_RTC** structure storing the information to set.

```
typedef struct
{
    BYTE   bSecond;
    BYTE   bMinute;
    BYTE   bHour;
    BYTE   bDay;
    BYTE   bMonth;
    BYTE   bYear;
    BYTE   bDoW;
    //The day of the week (starting the week on Sunday)
} CTOS_RTC;
```

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_RTC_INVALID_PARA	0601h
	d_RTC_HAL_FAULT	0602h
	d_RTC_TZ_OVERRANGE	0603h

**Example** Please refer to example in appendix [B.7 Keyboard](#)

## CTOS\_Delay

---

```
voidCTOS_Delay (ULONG ulMSec);
```

**Description**      Delay the execution of the program for a specific period.

**Parameters**      [ IN ]      *ulMSec*  
                            Delay time in milliseconds.  
                            Example: input 3000 for ulMSec means to delay for 3  
                            seconds.

**Return Value**      None

**Example**      Please refer to example in appendix [B.2 Ethernet](#) or [B.3 File System](#)  
                            or [B.5 Hardware](#) or [B.7 Keyboard](#) or [B.10 LCD Display](#) or [B.12](#)  
                            [Modem](#) or [B.15 RS232](#) or [B.16 SmartCard](#) or [B.19 USB](#)

## CTOS\_TickGet

---

```
ULONG CTOS_TickGet(void);
```

**Description** Get the current system tick value. The system timer tick is incremented by one every 10 milliseconds.

**Parameters** None

**Return Value** Value of current system tick.

**Example** Please refer to example in appendix [B.5 Hardware](#)

## CTOS\_TimeOutSet

---

```
USHORT CTOS_TimeOutSet (BYTE bTID, ULONG ulMSec) ;
```

**Description** Set the timeout value for the specific system timer. There are 4 timers provided by the system, and every timer can be setup and run independently.

**Parameters** [ IN ] bTID

Timer ID.

- **TIMER\_ID\_1**
- **TIMER\_ID\_2**
- **TIMER\_ID\_3**
- **TIMER\_ID\_4**

[ IN ] ulMSec

Timeout value in 10 milliseconds to set to the system timer.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_TIMER_INVALID_PARA	0401h
d_TIMER_HAL_FAULT	0402h

**Example**

Please refer to example in appendix [B.5 Hardware](#) or [B.17 Timer Led](#)

**Note**

The timer will start running immediately after calling this function. Programmer should call **CTOS\_TimeOutCheck()** to check whether the timer is timeout.

## CTOS\_TimeOutCheck

---

```
USHORT CTOS_TimeOutCheck(BYTE bTID) ;
```

**Description** Check whether the specific timer is timeout.

**Parameters** [ IN ] bTID

Timer ID.

- **TIMER\_ID\_1**
- **TIMER\_ID\_2**
- **TIMER\_ID\_3**
- **TIMER\_ID\_4**

**Return Value**

- **d\_YES**

Timeout

- **d\_NO**

Not Timeout

**Example** Please refer to example in appendix [B.5 Hardware](#) or [B.17 Timer Led](#)

**Note** Programmer should call **CTOS\_TimeOutSet()** to setup the timer properly before calling this function.

## 2.8. Buzzer Functions

- CTOS\_Beep
- CTOS\_Sound

### Buzzer Error Codes

<b>Constants</b>	<b>Value</b>
d_BUZZER_INVALID_PARA	1001h
d_BUZZER_HAL_FAULT	1002h

## CTOS\_Beep

---

```
USHORT CTOS_Beep(void);
```

**Description** Play a simple beep sound on buzzer. The frequency and duration are predefined and cannot be changed.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_BUZZER_INVALID_PARA	1001h
d_BUZZER_HAL_FAULT	1002h

**Example** Please refer to example in appendix[B.5 Hardware](#)

## CTOS\_Sound

---

```
USHORT CTOS_Sound(USHORT usFreq, USHORT usDuration);
```

**Description** Play a sound with specific frequency and duration.

**Parameters** [ IN ] *usFreq*  
Frequency in Hertz.

[ IN ] *usDuration*  
Duration in 10 milliseconds.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_BUZZER_INVALID_PARA	1001h
	d_BUZZER_HAL_FAULT	1002h

**Example** Please refer to example in appendix[B.5 Hardware](#)

## 2.9. Encryption and Decryption Functions

- CTOS\_RSA
- CTOS\_RNG
- CTOS\_DES
- CTOS\_SHA1Init
- CTOS\_SHA1Update
- CTOS\_SHA1Final
- CTOS\_AES
- CTOS\_MAC
- CTOS\_DES\_CBC

### Encryption and Decryption Error Codes

<b>Constants</b>	<b>Value</b>
d_CRYPTO_INVALID_PARA	0801h
d_CRYPTO_HAL_FAULT	0802h
d_CRYPTO_RNG_TIMEOUT	0803h
d_CRYPTO_RSA_GEN_KEY_FAILED	0804h

## CTOS\_RSA

---

```
USHORT CTOS_RSA(BYTE *baModulus, USHORT usModulousLen, BYTE
*baExponent, USHORT usExponentLen, BYTE *baData, BYTE *baResult);
```

**Description** Perform RSA exponential operation for encryption or decryption.

**Parameters** [ IN ] *baModulus*  
Modulus parameter of the exponentiation.

[ IN ] *usModulousLen*  
Length of modulus, cipher and result.

[ IN ] *baExponent*  
Exponent parameter of exponentiation.

[ IN ] *usExponentLen*  
Length of exponent.

[ IN ] *baData*  
Input data to be encrypted or decrypted.

[ OUT ] *baResult*  
Buffer to store the result of operation. Its size should be equal to or larger than usModulousLen.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Encryption and Decryption error codes</a>	08xxh

**Example** Please refer to example in appendix [B.1 Encryption and Decryption](#)

## CTOS\_RNG

---

```
USHORT CTOS_RNG (BYTE *baResult);
```

**Description** Get 8 bytes of random number.

**Parameters** [ OUT ] *baResult*  
Buffer to store 8 bytes of random number.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Encryption and Decryption error codes</a>	08xxh

**Example** Please refer to example in appendix[B.1 Encryption and Decryption](#) or [B.5 Hardware](#)

## CTOS\_DES

---

```
USHORT CTOS_DES (BYTE bEncDec, BYTE *baKey, BYTE bKeyLen, BYTE
*baData, USHORT usDataLen, BYTE *baResult);
```

**Description** Perform the DES (Data Encryption Standard) encryption or decryption.

**Parameters** [ IN ] *bEncDec*

Encryption or Decryption type.

- *d\_ENCRYPTION*
- *d\_DECRIPTION*

[ IN ] *baKey*

Key for encryption or decryption.

[ IN ] *bKeyLen*

Length of the key. It should be in 8/16/24 bytes.

[ IN ] *baData*

Input data for DES encryption/decryption.

[ IN ] *usDataLen*

Length of Input data. It should be multiple of 8.

[ OUT ] *baResult*

Buffer to store the result for DES encryption or decryption. Its size should be equal to or larger than usDataLen.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Encryption and Decryption error codes</a>	08xxh

**Example**

Please refer to example in appendix[B.1 Encryption and Decryption](#)

## CTOS\_SHA1Init

---

```
void CTOS_SHA1Init (SHA_CTX *pstInfo);
```

**Description** Initialize the **SHA\_CTX** structure and prepare for the SHA\_1 operation.

**Parameters** [ IN ] *pstInfo*  
Pointer to a **SHA\_CTX** structure to be initialized.

**Return Value** None

**Example** Please refer to example in appendix[B.1 Encryption and Decryption](#)

## CTOS\_SHA1Update

---

```
void CTOS_SHA1Update (SHA_CTX *pstInfo, BYTE *baBuffer, USHORT usCount);
```

<b>Description</b>	Perform the SHA_1 algorithm with the input data.						
<b>Parameters</b>	<table><tr><td>[ IN ]</td><td><u><i>pstInfo</i></u> Pointer to a <b>SHA_CTX</b> structure which is initialized by <b>CTOS_SHA1Init()</b>.</td></tr><tr><td>[ IN ]</td><td><u><i>baBuffer</i></u> Input data for the SHA_1 operation.</td></tr><tr><td>[ IN ]</td><td><u><i>usCount</i></u> Number of the input data in byte.</td></tr></table>	[ IN ]	<u><i>pstInfo</i></u> Pointer to a <b>SHA_CTX</b> structure which is initialized by <b>CTOS_SHA1Init()</b> .	[ IN ]	<u><i>baBuffer</i></u> Input data for the SHA_1 operation.	[ IN ]	<u><i>usCount</i></u> Number of the input data in byte.
[ IN ]	<u><i>pstInfo</i></u> Pointer to a <b>SHA_CTX</b> structure which is initialized by <b>CTOS_SHA1Init()</b> .						
[ IN ]	<u><i>baBuffer</i></u> Input data for the SHA_1 operation.						
[ IN ]	<u><i>usCount</i></u> Number of the input data in byte.						
<b>Return Value</b>	None						
<b>Example</b>	Please refer to example in appendix <a href="#">B.1 Encryption and Decryption</a>						
<b>Note</b>	There is no limitation for the number of input data. You can call this function with 1 byte of input data or 1k bytes of input data. Please see also <b>CTOS_SHA1Init()</b> and <b>CTOS_SHA1Final()</b> .						

## **CTOS\_SHA1Final**

---

```
void CTOS_SHA1Final (BYTE *baOutput, SHA_CTX *pstInfo);
```

**Description**      Finalize the SHA\_1 operation and return the result.

**Parameters**      [ IN ]      *baOutput*  
                            Buffer to store the final result of SHA\_1 operation, 20 bytes.

                            [ IN ]      *pstInfo*  
                            Pointer to a **SHA\_CTX** structure which is initialized by  
**CTOS\_SHA1Init()**.

**Return Value**      None

**Example**      Please refer to example in appendix [B.1 Encryption and Decryption](#)

**Note**      Please see also **CTOS\_SHA1Init()** and **CTOS\_SHA1Update()**.

## **CTOS\_AES**

---

```
USHORT CTOS_AES (BYTE bEncDec, BYTE *baKey, BYTE *baData, USHORT
usDataLen, BYTE *baResult);
```

**Description**      Perform the AES (Advanced Encryption Standard) encryption/decryption.

**Parameters**      [ IN ]      *bEncDec*  
Encryption or Decryption type.

- *d\_ENCRYPTION*
- *d\_DECRYPTION*

[ IN ]      *baKey*  
Key for encryption or decryption.

[ IN ]      *baData*  
Input data for AES encryption/decryption.

[ IN ]      *usDataLen*  
Length of Input data. It should be multiple of 16.

[ OUT ]      *baResult*  
Buffer to store the result for AES encryption or decryption. Its size should be equal to or larger than usDataLen.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Encryption and Decryption error codes</a>	08xxh

**Example**      Please refer to example in appendix [B.1 Encryption and Decryption](#)

## CTOS\_MAC

---

```
USHORT CTOS_MAC(BYTE *baKey, BYTE bKeyLen, BYTE *baICV, BYTE *baData,
USHORT usDataLen, BYTE *baMAC);
```

**Description** Perform CBC MAC calculation.

**Parameters** [ IN ] *baKey*  
MAC Key.

[ IN ] *bKeyLen*  
Length of MAC key.

[ IN ] *baICV*  
Initial Chaining Vector.

[ IN ] *baData*  
Input data.

[ IN ] *usDataLen*  
Length of Input data.

[ OUT ] *baResult*  
Buffer to store the result for MAC operation. Its size should  
be 8 bytes.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Encryption and Decryption error codes</a>	08xxh

**Example** Please refer to example in appendix [B.1 Encryption and Decryption](#)

## CTOS\_DES\_CBC

---

```
USHORT CTOS_DES_CBC (BYTE bEncDec, BYTE *baKey, BYTE bKeyLen, BYTE
*baICV, BYTE *baData, USHORT usDataLen, BYTE *baResult);
```

<b>Description</b>	Perform the DES CBC encryption or decryption.	
<b>Parameters</b>	[ IN ]	<i>bEncDec</i> Encryption or Decryption type. <ul style="list-style-type: none"><li>• <i>d_ENCRYPTION</i></li><li>• <i>d_DECRIPTION</i></li></ul>
	[ IN ]	<i>baKey</i> Key for encryption or decryption.
	[ IN ]	<i>bKeyLen</i> Length of the key. It should be in 8/16/24 bytes.
	[ IN ]	<i>baICV</i> Initial Chaining Vector.
	[ IN ]	<i>baData</i> Input data for DES encryption/decryption.
	[ IN ]	<i>usDataLen</i> Length of Input data. It should be multiple of 8.
	[ OUT ]	<i>baResult</i> Buffer to store the result for DES encryption or decryption. Its size should be equal to or larger than usDataLen.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Encryption and Decryption error codes</a>	08xxh

## 2.10.RS232 Communication Functions

- CTOS\_RS232Open
- CTOS\_RS232TxReady
- CTOS\_RS232TxData
- CTOS\_RS232RxReady
- CTOS\_RS232RxData
- CTOS\_RS232SetRTS
- CTOS\_RS232GetCTS
- CTOS\_RS232FlushRxBuffer
- CTOS\_RS232FlushTxBuffer
- CTOS\_RS232Close

### RS232Communication Error Codes

<b>Constants</b>	<b>Value</b>
d_RS232_INVALID_PARA	0A01h
d_RS232_HAL_FAULT	0A02h
d_RS232_BUSY	0A03h
d_RS232_NOT_OPEN	0A04h
d_RS232_NOT_SUPPORT	0A05h

## CTOS\_RS232Open

---

```
USHORT CTOS_RS232Open(BYTE bPort, ULONG ulBaudRate, BYTE bParity,  
BYTE bDataBits, BYTE bStopBits);
```

**Description** Open a RS232 port with the specific parameters.

**Parameters** [ IN ] *bPort*  
Port number to open.

- *d\_COM1*
- *d\_COM2*
- *d\_COM3*

[ IN ] *ulBaudRate*  
Baud rate in bps (bits per second).  
= 115200 : 115200 bps  
= 57600 : 57600 bps  
= 38400 : 38400 bps  
= 19200 : 19200 bps  
= 9600 : 9600 bps

[ IN ] *bParity*  
Parity mode.  
= 'E' : Even parity  
= 'O' : Odd parity  
= 'N' : None parity

[ IN ] *bDataBits*  
Number of data bits for each data byte.  
= 7 : 7 data bits  
= 8 : 8 data bits

[ IN ]    *bStopBits*

Number of stop bits for each data byte.

= 1 : 1 stop bit

= 2 : 2 stop bits

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#"><u>RS232 error codes</u></a>	0Axxh

**Example**

Please refer to example in appendix [B.15 RS232](#)

**Note**

Number of ports supported may be varied for different type of terminal.

## CTOS\_RS232TxReady

---

```
USHORT CTOS_RS232TxReady(BYTE bPort);
```

**Description** Check if the specific RS232 port is ready for sending next data.

**Parameters** [ IN ] *bPort*

Port number.

- *d\_COM1*
- *d\_COM2*
- *d\_COM3*

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">RS232 error codes</a>	0Axxh

**Example**

Please refer to example in appendix [B.15 RS232](#)

## CTOS\_RS232TxData

---

USHORT CTOS\_RS232TxData (BYTE bPort, BYTE\* baBuf, USHORT usLen);

**Description** Send data through the specific RS232 port.

**Parameters** [ IN ] bPort

Port number.

- d\_COM1
- d\_COM2
- d\_COM3

[ IN ] baBuf

Data to send.

[ IN ] usLen

Length of data. The maximum value is 2048.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">RS232 error codes</a>	0Axxh

**Example**

Please refer to example in appendix [B.15 RS232](#)

**Note**

Before sending any data, programmer should call **CTOS\_RS232TxReady()** to check whether the port is ready.

## CTOS\_RS232RxReady

---

USHORT CTOS\_RS232RxReady(BYTE bPort, USHORT\* pusLen);

**Description** Get the number of data currently received from the specific RS232 port.

**Parameters**

[ IN ]	<i>bPort</i>
Port number.	
<ul style="list-style-type: none"> <li>• <i>d_COM1</i></li> <li>• <i>d_COM2</i></li> <li>• <i>d_COM3</i></li> </ul>	

[ OUT ]	<i>pusLen</i>
Length of data received.	

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">RS232 error codes</a>	0Axxh

**Example** Please refer to example in appendix [B.15 RS232](#)

## CTOS\_RS232RxData

---

```
USHORT CTOS_RS232RxData(BYTE bPort, BYTE* baBuf, USHORT* pusLen);
```

**Description** Get the data received from the specific RS232 port.

**Parameters** [ IN ] bPort

Port number.

- d\_COM1
- d\_COM2
- d\_COM3

[ OUT ] baBuf

Buffer to store the received data.

[ IN ] pusLen

Size of the buffer. Maximum number of data to get.

[ OUT ] pusLen

Length of the data actually returned in the buffer.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">RS232 error codes</a>	0Axxh

**Example**

Please refer to example in appendix [B.15 RS232](#)

**Note**

Programmer can call **CTOS\_RS232RxReady()** to check the number of data currently received, and then call this function with corresponding size of buffer get the data.

If the size input is smaller than the actual number of data received, then only the size number of data will be returned.

If the size input is bigger than the actual number of data, then the actual number of data returned will be stored in pusLen.

## CTOS\_RS232SetRTS

---

USHORT CTOS\_RS232SetRTS(BYTE bPort, BYTE bOnOff);

**Description** Set the status of RTS line (Ready To Send) for the specific RS232 port.

**Parameters** [ IN ] *bPort*  
Port number.

- *d\_COM1*
- *d\_COM2*
- *d\_COM3*

[ IN ] *bOnOff*  

- *d\_ON*  
Set RTS on (Logically LOW).
- *d\_OFF*  
Set RTS off (Logically HIGH).

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">RS232 error codes</a>	0Axxh

**Example** Please refer to example in appendix [B.15 RS232](#)

## CTOS\_RS232GetCTS

---

```
USHORT CUSHORT CTOS_RS232GetCTS (BYTE bPort) ;
```

**Description** Get current status of CTS line (Clear To Send) for the specific RS232 port.

**Parameters** [ IN ] *bPort*  
Port number.

- *d\_COM1*
- *d\_COM2*
- *d\_COM3*

**Return Value**

- *d\_ON*  
CTS on (Logically LOW).
- *d\_OFF*  
CTS off (Logically HIGH).
- *Others*  
[RS232 error codes](#)

**Example** Please refer to example in appendix [B.15 RS232](#)

## CTOS\_RS232FlushRxBuffer

---

```
USHORT CUSHORT CTOS_RS232FlushRxBuffer( BYTE bPort );
```

**Description** Flush the data already received from the specific RS232 port.

**Parameters** [ IN ] *bPort*

Port number.

- *d\_COM1*
- *d\_COM2*
- *d\_COM3*

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">RS232 error codes</a>	0Axxh

**Example** Please refer to example in appendix [B.15 RS232](#)

**Note** All the data received before calling this function will be discarded.

## CTOS\_RS232FlushTxBuffer

---

```
USHORT CUSHORT CTOS_RS232FlushTxBuffer ( BYTE bPort );
```

**Description** Flush the data to be sent to the specific RS232 port.

**Parameters** [ IN ] *bPort*

Port number.

- *d\_COM1*
- *d\_COM2*
- *d\_COM3*

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">RS232 error codes</a>	0Axxh

**Example** Please refer to example in appendix [B.15 RS232](#)

**Note** This function is normally used to cancel the current transmission.

## CTOS\_RS232Close

---

```
USHORT CUSHORT CTOS_RS232Close( BYTE bPort );
```

**Description** Close the RS232 port.

**Parameters** [ IN ] *bPort*

Port number.

- *d\_COM1*
- *d\_COM2*
- *d\_COM3*

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">RS232 error codes</a>	0Axxh

**Example** Please refer to example in appendix [B.15 RS232](#)

**Note**

All the data received and about to be transmitted will be discarded, and all hardware and software resource will be released.

## 2.11. Modem Functions

- CTOS\_ModemOpen
- CTOS\_ModemClose
- CTOS\_ModemDialup
- CTOS\_ModemTxReady
- CTOS\_ModemTxData
- CTOS\_ModemRxReady
- CTOS\_ModemRxData
- CTOS\_ModemATCommand
- CTOS\_ModemStatus
- CTOS\_ModemHookOff
- CTOS\_ModemHookOn
- CTOS\_ModemSetDialPrecheck
- CTOS\_ModemSetConfig
- CTOS\_ModemReadConfig
- CTOS\_ModemFlushRxData
- CTOS\_ModemListen
- CTOS\_ModemSetCommParam

### Modem Error Codes

<b>Constants</b>	<b>Value</b>
d_MODEM_INVALID_PARA	4001h
d_MODEM_NOT_OPEN	4002h
d_MODEM_NOT_ONLINE	4003h
d_MODEM_TIMEOUT	4004h
d_MODEM_INSUFFICIENT_BUF	4005h
d_MODEM_CMD_INVALID_RESPONSE	4006h
d_MODEM_CMD_NOT_SUCCESS	4007h
d_MODEM_TX_BUSY	4008h
d_MODEM_NOT_SUPPORTED	4009h
d_MODEM_DIALING	400Ah
d_MODEM_LISTENING	400Bh
d_MODEM_ALREADY_ONLINE	400Ch
d_MODEM_NOT_ASYN_MODE	400Dh
d_MODEM_SDLC_NOT_READY	400Eh

## CTOS\_ModemOpen

---

```
USHORT CUSHORT CTOS_ModemOpen(BYTE bMode, BYTE bHandShake, BYTE  
bCountryCode);
```

**Description** Open modem port with specific mode and parameters.

**Parameters** [ IN ] *bMode*  
Mode of Modem connection.  
• ***d\_M\_MODE\_ASYNC\_NORMAL***  
ASYNC normal connect, all handshake.

- ***d\_M\_MODE\_ASYNC\_FAST***  
ASYNC Fast connect, only handshake  
*d\_M\_HANDSHAKE\_V22\_ONLY*  
*d\_M\_HANDSHAKE\_BELL\_212*  
*d\_M\_HANDSHAKE\_BELL\_103*  
*d\_M\_HANDSHAKE\_V21\_ONLY*

- ***d\_M\_MODE\_SDLC\_NORMAL***  
SYNC SDLC normal connect, only handshake  
*d\_M\_HANDSHAKE\_V22BIS\_ONLY*

- ***d\_M\_MODE\_SDLC\_FAST***  
SYNC SDLC Fast connect, only handshake  
*d\_M\_HANDSHAKE\_V22\_ONLY*  
*d\_M\_HANDSHAKE\_BELL\_212*

[ IN ] *bHandShake*  
Handshakes of modem connection.

- ***d\_M\_HANDSHAKE\_V90\_AUTO\_FB***  
V.90 automatic fallback (56kbps to 300bps)

- ***d\_M\_HANDSHAKE\_V90\_ONLY***

V.90 only (56kbps to 28kbps)

- ***d\_M\_HANDSHAKE\_V34\_AUTO\_FB***

V.34 automatic fallback (33.6kbps to 300bps)

- ***d\_M\_HANDSHAKE\_V34\_ONLY***

V.34 only (33.6kbps to 2400 bps)

- ***d\_M\_HANDSHAKE\_V32BIS\_AUTO\_FB***

ITU-T V.32bis automatic fallback (14.4kbps to 300  
bps)

- ***d\_M\_HANDSHAKE\_V32BIS\_ONLY***

ITU-T V.32bis only (14.4kbps to 4800 bps)

- ***d\_M\_HANDSHAKE\_V22BIS\_ONLY***

ITU-T V.22bis only (2400bps or 1200bps)

- ***d\_M\_HANDSHAKE\_V22\_ONLY***

ITU-T V.22 only (1200bps)

- ***d\_M\_HANDSHAKE\_BELL\_212***

Bell 212 only (1200bps)

- ***d\_M\_HANDSHAKE\_BELL\_103***

Bell 103 only (300bps)

- ***d\_M\_HANDSHAKE\_V21\_ONLY***

ITU\_T V.21 only (300 bps)

- ***d\_M\_HANDSHAKE\_V23***

V.23 (1200/75 bps)

[ IN ]    *bCountryCode*

Country Code.

- ***d\_M\_COUNTRY\_SPAIN***

- `d_M_COUNTRY_TAIWAN`
- `d_M_COUNTRY_CHINA`
- `d_M_COUNTRY_USA`
- `d_M_COUNTRY_GERMANY`
- `d_M_COUNTRY_FRANCE`
- `d_M_COUNTRY_MALAYSIA`
- `d_M_COUNTRY_ENGLAND`
- `d_M_COUNTRY_THAILAND`
- `d_M_COUNTRY_SINGAPORE`
- `d_M_COUNTRY_JAPEN`
- `d_M_COUNTRY_TURKEY`
- `d_M_COUNTRY_RUSSIA`
- `d_M_COUNTRY_HONGKONG`
- `d_M_COUNTRY_INDONESIA`
- `d_M_COUNTRY_AUSTRALIA`
- `d_M_COUNTRY_BRAZIL`
- `d_M_COUNTRY_CANADA`
- `d_M_COUNTRY SOUTH_AFRICA`
- `d_M_COUNTRY SOUTH_KOREA`
- `d_M_COUNTRY_QATAR`

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem error codes</a>	40xxh

#### Example

Please refer to example in appendix [B.12 Modem](#)

## CTOS\_ModemClose

---

```
USHORT CUSHORT CTOS_ModemClose( void );
```

**Description** Close modem port.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Modem error codes</a>	40xxh

**Example** Please refer to example in appendix [B.12 Modem](#)

**Note**

All the data received and about to be transmitted will be discarded, the connection established will be lost, and all hardware and software resource will be released.

## CTOS\_ModemDialup

---

```
USHORT CUSHORT CTOS_ModemDialup(BYTE *baNumber, USHORT usLen);
```

**Description** Dial up to establish a connection to the host.

**Parameters**

[ IN ]	<u><i>baNumber</i></u>
Number to dial. Dialing number should be input in the form of ASCII character. If pulse dialing is used, a "P" character must be added before the dialing number.	

[ IN ]	<u><i>usLen</i></u>
Length of the number to dial.	

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem error codes</a>	40xxh

**Example** Please refer to example in appendix [B.12 Modem](#)

## CTOS\_ModemTxReady

---

```
USHORT CUSHORT CTOS_ModemTxReady( void );
```

**Description** Check if the modem port is ready for sending next data.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Modem error codes</a>	40xxh

**Example** Please refer to example in appendix [B.12 Modem](#)

## CTOS\_ModemTxData

---

```
USHORT CUSHORT CTOS_ModemTxData(BYTE *baData, USHORT usLen);
```

**Description** Send data through the modem port.

**Parameters** [ IN ] *baData*  
Data to send.

[ IN ] *usLen*  
Length of data. The maximum value is 2048.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem error codes</a>	40xxh

**Example** Please refer to example in appendix [B.12 Modem](#)

**Note** Before sending any data, programmer should call **CTOS\_ModemTxReady()** to check whether the modem is ready.

## CTOS\_ModemRxReady

---

```
USHORT CUSHORT CTOS_ModemRxReady (USHORT*pusLen) ;
```

**Description** Get the number of data currently received from the modem port.

**Parameters** [ IN ] *pusLen*  
Length of data received.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem error codes</a>	40xxh

**Example** Please refer to example in appendix [B.12 Modem](#)

## CTOS\_ModemRxData

---

```
USHORT CUSHORT CTOS_ModemRxData (BYTE*baData, USHORT*pusLen);
```

**Description** Get the data received from the modem port.

**Parameters**

- [ OUT] *baData*  
Buffer to store the received data.

- [ IN ] *usLen*  
Size of buffer. Maximum number of data to get.

- [ OUT ] *usLen*  
Length of the data actually returned in the buffer.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem error codes</a>	40xxh

**Example** Please refer to example in appendix [B.12 Modem](#)

**Note** Programmer can call **CTOS\_ModemRxReady()** to check the number of data currently received, and then call this function with corresponding size of buffer get the data.  
If the size input is smaller than the actual number of data received, then only the size number of data will be returned.  
If the size input is bigger than the actual number of data, then the actual number of data returned will be stored in pusLen.

## CTOS\_ModemATCommand

---

```
USHORT CUSHORT CTOS_ModemATCommand (BYTE*baCmd, USHORT usCmdLen,
BYTE*pbResponse, USHORT*pusRespLen) ;
```

**Description** Send an AT command to modem.

**Parameters** [ IN ] *baCmd*  
AT command to send.

[ IN ] *usCmdLen*  
Length of AT command.

[ OUT ] *pbResponse*  
Buffer to store the response of AT command.

[ IN ] *pusRespLen*  
Size of response buffer.

[ OUT ] *pusRespLen*  
Length of the response.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem error codes</a>	40xxh

**Example** Please refer to example in appendix [B.12 Modem](#)

## CTOS\_ModemStatus

---

```
USHORT CUSHORT CTOS_ModemStatus (DWORD* pdwStatus);
```

**Description** Get the status of modem.

**Parameters** [ OUT] *pdwStatus*

Point to a DWORD to store the current modem status.

- ***d\_M\_STATUS\_MODEM\_OPEN***

modem is open.

- ***d\_M\_STATUS\_MODEM\_ONLINE***

modem is online.

- ***d\_M\_STATUS\_SDLC\_MODE***

modem is in SDLC mode.

- ***d\_M\_STATUS\_SDLC\_ONLINE***

modem is online in SDLC mode.

- ***d\_M\_STATUS\_DIALING***

modem is dialing.

- ***d\_M\_STATUS\_NO\_DIAL\_TONE***

no dial tone while dialing.

- ***d\_M\_STATUS\_LINE\_BUSY***

phone line is busy.

- ***d\_M\_STATUS\_RING\_BACK***

modem has received the ring signal.

- ***d\_M\_STATUS\_TX\_BUSY***

modem is transmitting data.

- ***d\_M\_STATUS\_REMOTE\_NOT\_ANSWER***  
Remote side has no answer.
- ***d\_M\_STATUS\_NO\_CARRIER***  
Remote side has not answered the dialing or has hooked on.
- ***d\_M\_STATUS\_ALL\_DATA\_SENT***  
all the data has been sent.
- ***d\_M\_STATUS\_RX\_DATA\_VALID***  
while the modem is in
- ***d\_M\_STATUS\_LISTENING***  
modem is in listen mode.
- ***d\_M\_STATUS\_OTHER\_ERROR***  
unexpected error.
- ***d\_M\_STATUS\_DATA\_SENT\_ERROR***  
error while sending the data.
- ***d\_M\_STATUS\_DATA\_RECEIVE\_ERROR***  
error while receiving the data.
- ***d\_M\_STATUS\_TIMEOUT***  
modem chip do not respond the command.

#### Return Value

<b><i>Constants</i></b>	<b><i>Value</i></b>
d_OK	0000h
<a href="#">Modem error codes</a>	40xxh

#### Example

Please refer to example in appendix [B.12 Modem](#)

## CTOS\_ModemHookOff

---

```
USHORT CUSHORT CTOS_ModemHookOff( void );
```

**Description** Hook off the modem.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Modem error codes</a>	40xxh

**Example** Please refer to example in appendix [B.12 Modem](#)

## CTOS\_ModemHookOn

---

```
USHORT CUSHORT CTOS_ModemHookOn( void );
```

**Description** Hook on the modem.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Modem error codes</a>	40xxh

**Example** Please refer to example in appendix [B.12 Modem](#)

## CTOS\_ModemSetDialPrecheck

---

```
USHORT CUSHORT CTOS_ModemSetDialPrecheck(BYTE bMode);
```

**Description** Set precheck parameters for dialing up.

**Parameters** [ IN ] bMode

- **d\_M\_PRECHECK\_NO\_DETECT**  
Not detect dial tone or busy
- **d\_M\_PRECHECK\_DIAL\_TONE\_DETECT\_ONLY**  
Only detect dial tone
- **d\_M\_PRECHECK\_BUSY\_DETECT\_ONLY**  
Only detect busy
- **d\_M\_PRECHECK\_DETECT\_BOTH**  
Detect dial tone and busy (default)
- **d\_M\_PRECHECK\_DETECT\_ALL**  
Detect dial tone, busy, and ringback

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">Modem error codes</a>	40xxh

**Example**

Please refer to example in appendix [B.12 Modem](#)

## CTOS\_ModemSetConfig

---

```
USHORT CUSHORT CTOS_ModemSetConfig(BYTE bTag, USHORT usValue);
```

**Description** Set modem configuration.

**Parameters** [ IN ] bTag

- **d\_M\_CONFIG\_DIALING\_DURATION**

Dialing Maximum Total Duration (in second).

[Default: 30, range:0~255]

- **d\_M\_CONFIG\_MIN\_ONHOOK\_DURATION**

Minimum On-hook Duration (in second).

Any attempt to go off-hook remains on-hook for this duration delayed.

[Default: 3, range:0~255]

- **d\_M\_CONFIG\_PREDIAL\_DELAY\_TIME**

Pre-dial Delay Time (in ms).

[Default: 0]

- **d\_M\_CONFIG\_DIAL\_TONE\_DETECT\_DURATION**

Dial Tone Detect Duration (in ms).

[Default: 7000]

- **d\_M\_CONFIG\_DIAL\_TONE\_MIN\_ON\_TIME**

Dial Tone Minimum On Time (in 1 / 7200 second)

[Default: 14400]

- **d\_M\_CONFIG\_DTMF\_ON\_TIME**

DTMF On Time (in ms)

[Default: 100]

- **d\_M\_CONFIG\_DTMF\_OFF\_TIME**

DTMF Off Time (in ms)

[Default: 100]

- ***d\_M\_CONFIG\_BUSY\_TONE\_MIN\_TOTAL\_TIME***

Busy Tone Minimum Total Time (in 1 / 7200 second)

(=Busy Tone Minimum On Time + Busy Tone Minimum Off Time)

[Default: 1728]

- ***d\_M\_CONFIG\_BUSY\_TONE\_DELTA\_TIME***

Busy Tone Delta Time (in 1 / 7200 second)

(= Busy Tone Maximum Total Time Δ Busy ToneMinimum Total Time)

[Default: 7632]

- ***d\_M\_CONFIG\_BUSY\_TONE\_MIN\_ON\_TIME***

Busy Tone Minimum On Time (in 1 / 7200 second)

[Default: 864]

- ***d\_M\_CONFIG\_RINGBACK\_TONE\_MIN\_TOTAL\_TIME***

Ringback Tone Minimum Total Time (in 1 / 7200 second)

(=Ringback Tone Minimum On Time + Ringback Tone Minimum Off Time)

[Default: 18000]

- ***d\_M\_CONFIG\_RINGBACK\_TONE\_DELTA\_TIME***

Ringback Tone Delta Time (in 1 / 7200 second)

(=Ringback Tone Maximum Total Time Δ Ringback Tone Minimum Total Time)

[Default: 61200]

- ***d\_M\_CONFIG\_RINGBACK\_TONE\_MIN\_ON\_TIME***

Ringback Tone Minimum On Time (in 1 / 7200 second)

[Default: 4608]

- ***d\_M\_CONFIG\_ANSWER\_TONE\_WAIT\_DURATION***

Ringback Wait Duration (in second).

[Default: 50, range:0~255]

- ***d\_M\_CONFIG\_BLIND\_DIAL\_DELAY\_TIME***

Blind Dial (no detect dial tone) Delay Time (in second).

Note: This is only valid if Dialing Monitor is set to no dial tone detection.

[Default: 2, range:0~255]

- ***d\_M\_CONFIG\_CARRIER\_PRESENT\_TIME***

Carrier Presence Time (in 100ms)

[Default: 6, range:1~255]

[ IN ]     *usValue*

Value to set.

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">Modem error codes</a>	40xxh

**Example**

Please refer to example in appendix [B.12 Modem](#)

## CTOS\_ModemReadConfig

---

```
USHORT CUSHORT CTOS_ModemReadConfig(BYTE bTag, USHORT*pusValue);
```

**Description**      Read modem configuration.

- Parameters**      [ IN ]      bTag
- **d\_M\_CONFIG\_DIALING\_DURATION**  
Dialing Duration Time (in second).
  - **d\_M\_CONFIG\_MIN\_ONHOOK\_DURATION**  
Minimum On-hook Duration (in second)
  - **d\_M\_CONFIG\_PREDIAL\_DELAY\_TIME**  
Pre-dial Delay Time (in ms)
  - **d\_M\_CONFIG\_DIAL\_TONE\_DETECT\_DURATION**  
Dial Tone Detect Duration (in ms)
  - **d\_M\_CONFIG\_DIAL\_TONE\_MIN\_ON\_TIME**  
Dial Tone Minimum On Time (in 1 / 7200 second)
  - **d\_M\_CONFIG\_DTMF\_ON\_TIME**  
DTMF On Time (in ms)
  - **d\_M\_CONFIG\_DTMF\_OFF\_TIME**  
DTMF Off Time (in ms)
  - **d\_M\_CONFIG\_BUSY\_TONE\_MIN\_TOTAL\_TIME**  
Buy Tone Minimum Total Time (in 1 / 7200 second)
  - **d\_M\_CONFIG\_BUSY\_TONE\_DELTA\_TIME**  
Busy Tone Delta Time (in 1 / 7200 second)

- ***d\_M\_CONFIG\_BUSY\_TONE\_MIN\_ON\_TIME***  
Busy Tone Minimum On Time (in 1 / 7200 second)
- ***d\_M\_CONFIG\_RINGBACK\_TONE\_MIN\_TOTAL\_TIME***  
Ringback Tone Minimum Total Time (in 1/7200second)
- ***d\_M\_CONFIG\_RINGBACK\_TONE\_DELTA\_TIME***  
Ringback Tone Delta Time (in 1 / 7200 second)
- ***d\_M\_CONFIG\_RINGBACK\_TONE\_MIN\_ON\_TIME***  
Ringback Tone Minimum On Time (in 1 / 7200 second)
- ***d\_M\_CONFIG\_ANSWER\_TONE\_WAIT\_DURATION***  
Answer Tone Wait Duration (in second)
- ***d\_M\_CONFIG\_BLIND\_DIAL\_DELAY\_TIME***  
Blind Dial Delay Time (in second)
- ***d\_M\_CONFIG\_CARRIER\_PRESENT\_TIME***  
Carrier Presence Time (in 100ms)

[ OUT ] *pusValue*

Point to USHORT to store current value of specified tag.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem error codes</a>	40xxh

#### Example

Please refer to example in appendix [B.12 Modem](#)

## CTOS\_ModemFlushRxData

---

```
USHORT CUSHORT CTOS_ModemFlushRxData( void );
```

**Description** Flush the data already received from the modem port.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Modem error codes</a>	40xxh

**Example** Please refer to example in appendix [B.12 Modem](#)

**Note** All the data received before calling this function will be discarded.

## CTOS\_ModemListen

---

```
USHORT CUSHORT CTOS_ModemListen(BYTE bNumOfRing);
```

<b>Description</b>	Enable the listening mode of modem.						
<b>Parameters</b>	<p>[ IN ]    <u><i>bNumOfRing</i></u></p> <p>The number of the rings to wait before picking up the phone.</p>						
<b>Return Value</b>	<table border="1"><thead><tr><th><b>Constants</b></th><th><b>Value</b></th></tr></thead><tbody><tr><td>d_OK</td><td>0000h</td></tr><tr><td><a href="#">Modem error codes</a></td><td>40xxh</td></tr></tbody></table>	<b>Constants</b>	<b>Value</b>	d_OK	0000h	<a href="#">Modem error codes</a>	40xxh
<b>Constants</b>	<b>Value</b>						
d_OK	0000h						
<a href="#">Modem error codes</a>	40xxh						
<b>Example</b>	Please refer to example in appendix <a href="#">B.12 Modem</a>						

## CTOS\_ModemSetCommParam

---

```
USHORT CUSHORT CTOS_ModemSetCommParam(ULONG ulBaudRate, BYTE bParity,
BYTE bDataBits, BYTE bStopBits);
```

**Description** Set modem communication paramters.

**Parameters** [ IN ] *ulBaudRate*  
Baud rate in bps (bits per second).

- = 115200 : 115200 bps
- = 57600 : 57600 bps
- = 38400 : 38400 bps
- = 19200 : 19200 bps
- = 9600 : 9600 bps

[ IN ] *bParity*  
Parity mode.  
= 'E' : Even parity  
= 'O' : Odd parity  
= 'N' : None parity

[ IN ] *bDataBits*  
Number of data bits for each data byte.  
= 7 : 7 data bits  
= 8 : 8 data bits

[ IN ] *bStopBits*  
Number of stop bits for each data byte.  
= 1 : 1 stop bit  
= 2 : 2 stop bits

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Modem error codes</a>	40xxh

## 2.12.Modem TCP Functions

- CTOS\_TCP\_ModemInit
- CTOS\_TCP\_ModemOpen
- CTOS\_TCP\_ModemOpenEx
- CTOS\_TCP\_ModemClose
- CTOS\_TCP\_ModemDialup
- CTOS\_TCP\_ModemOnHook
- CTOS\_TCP\_ModemGetIP
- CTOS\_TCP\_ModemSetIP
- CTOS\_TCP\_ModemConnectEx
- CTOS\_TCP\_ModemConnectURL
- CTOS\_TCP\_ModemDisconnect
- CTOS\_TCP\_ModemTx
- CTOS\_TCP\_ModemRx
- CTOS\_TCP\_ModemStatus
- CTOS\_TCP\_ModemGetDNSServer
- CTOS\_TCP\_ModemSetDNSServer
- CTOS\_TCP\_ModemURL2IP
- CTOS\_TCP\_ModemPing
- CTOS\_UDP\_ModemTx
- CTOS\_UDP\_ModemRx

### Modem TCP Error Codes

<b>Constants</b>	<b>Value</b>
d_TCP_IO_PROCESSING	2321h
d_TCP_IO_BUSY	2322h
d_TCP_URL_NOT_FOUND	2370h
d_TCP_PING_FAILED	2371h
d_TCP_PPP_CONNECTION_TERMINATED	2372h
d_TCP_SOCKET_FULL	2380h
d_TCP_CONNECTION_NOT_ESTABLISHED	2381h
d_TCP_BAD_FCS	2382h
d_TCP_PPP_TIMEOUT	2383h

d_TCP_PROTOCOL_ERROR	2384h
d_TCP_LENGTH_ERROR	2385h
d_TCP_PPP_SEND_TIMEOUT	2386h
d_TCP_PPP_SEND_ERROR	2387h
d_TCP_SOCKET_FAILED	2388h
d_TCP_SOCKET_IO_MODE_FAILED	2389h
d_TCP_RX_RECEIVE_TIMEOUT	238Ah
d_TCP_INVALID_PARA	238Bh
d_TCP_CONNECT_TIMEOUT	2390h
d_TCP_DISCONNECT_FAIL	2391h
d_TCP_RESET	2392h
d_TCP_SEQNENCE_INCORRECT	2393h
d_TCP_NO_SERVER	2394h
d_TCP_RETRANSMISSION	2395h
d_TCP_PROTOCOL	2396h
d_TCP_IP_FORMAT_WRONG	2397h
d_TCP_FORMAT_WRONG	2398h
d_TCP_SEND_TIMEOUT	2399h
d_TCP_NO_ACK	239Ah
d_TCP_BUF_FULL	239Bh
d_TCP_RECEIVE_NON_UDP_PACKAGE	239Ch
d_TCP_IP_ADDRESS_NOT_EXIST	239Dh
d_TCP_RECEIVE_DATA_FAILED	239Eh
d_TCP_SEND_DATA_FAILED	239Fh
d_TCP_LCP_TIMEOUT	23A0h
d_TCP_LCP_ACK	23A1h
d_TCP_LCP_NAK	23A2h
d_TCP_LCP_REJECT	23A3h
d_TCP_LCP_CODE_REJECT	23A4h
d_TCP_LCP_PROTO_REJECT	23A5h
d_TCP_LCP_TERM_REJECT	23A6h
d_TCP_PEER_LCP_ACK	23A7h
d_TCP_PEER_LCP_NAK	23A8h
d_TCP_PEER_LCP_REJ	23A9h
d_TCP_PEER_LCP_CODE_REJECT	23AAh

d_TCP_PEER_LCP_PROTO_REJECT	23ABh
d_TCP_PEER_LCP_TERM_REQUEST	23ACh
d_TCP_PEER_LCP_TERM_ACK	23ADh
d_TCP_SEND_EPIPE_FAILED	23AEh
d_TCP_RECEIVE_EPIPE_FAILED	23AFh
d_TCP_CHAP_TIMEOUT	23B0h
d_TCP_CHAP_RESPONSE	23B1h
d_TCP_CHAP_CODE_REJECT	23B2h
d_TCP_PEER_CHAP_SUCCESS	23B3h
d_TCP_PEER_CHAP_AUTH_FAIL	23B4h
d_TCP_IPCP_TIMEOUT	23C0h
d_TCP_IPCP_ACK	23C1h
d_TCP_IPCP_REJECT	23C2h
d_TCP_IPCP_CODE_REJECT	23C3h
d_TCP_IPCP_PROTO_REJECT	23C4h
d_TCP_PEER_IPCP_ACK	23C5h
d_TCP_PEER_IPCP_NAK	23C6h
d_TCP_IPCP_TOSS	23C7h
d_TCP_PEER_IPCP_REJ	23C8h
d_TCP_CCP_REJECT	23CAh
d_TCP_BIND_FAILED	23CBh
d_TCP_MODEM_DIALUP_TIMEOUT	23D0h
d_TCP_MODEM_OPEN_FAILED	23D1h
d_TCP_MODEM_ATCMD_FAILED	23D2h
d_TCP_MODEM_DIALUP_FAILED	23D3h
d_TCP_MODEM_CLOSE_FAILED	23D4h
d_TCP_MODEM_HOOKON_FAILED	23D5h
d_TCP_MODEM_STATUS_FAILED	23D6h
d_TCP_MODEM_CONNECTION_FAILED	23D7h
d_TCP_MODEM_NO_DIAL_TONE	23D8h
d_TCP_MODEM_LINE_BUSY	23D9h
d_TCP_MODEM_RING_BACK	23DAh
d_TCP_MODEM_REMOTE_NOT_ANSWER	23DBh
d_TCP_MODEM_NO_CARRIER	23DCh
d_TCP_MODEM_OTHER_ERROR	23DDh

d_TCP_MODEM_ALREADY_OPENED	23DEh
d_TCP_MODEM_AUTHENTICATION_FAILED	23DFh
d_TCP_CHANNEL_TYPE_FAILED	23E0h
d_TCP_CHANNEL_OFFLINE	23E1h
d_TCP_USER_INTERRUPT	23FFh

## **CTOS\_TCP\_ModemInit**

---

```
void CTOS_TCP_ModemInit(void);
```

**Description**      The 1st API to call when using the Modem APIs.

**Parameters**      None

**Return Value**      None

**Example**      Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemOpen

---

```
USHORT CUSHORT CTOS_TCP_ModemOpen(BYTE bMode , BYTE bHandShake ,  
BYTE bCountryCode );
```

**Description** Open the Modem device.

- Parameters** [ IN ] *bMode*
- The mode of Modem connection.
- ***d\_M\_MODE\_ASYNC\_NORMAL***  
ASYNC normal connect, all handshake.
  - ***d\_M\_HANDSHAKE\_V90\_ONLY***  
V.90 only (56kbps to 28kbps)
  - ***d\_M\_MODE\_ASYNC\_FAST***  
ASYNC Fast connect. If you can not dial up by using  
this setting, please use  
*d\_M\_MODE\_SDLC\_NORMAL* instead.
  - ***d\_M\_MODE\_SDLC\_NORMAL***  
SYNC SDLC normal connect, used in below  
handshake:  
*d\_M\_HANDSHAKE\_V22BIS\_ONLY*  
*d\_M\_HANDSHAKE\_V22\_ONLY*
  - ***d\_M\_MODE\_SDLC\_FAST***  
SYNC SDLC Fast connect, used in below  
handshakes:  
*d\_M\_HANDSHAKE\_V22\_ONLY*

[ IN ] *bHandShake*

The handshakes of modem connection.

- ***d\_M\_HANDSHAKE\_V90\_AUTO\_FB***

V.90 automatic fallback (56kbps to 300bps)

- **d\_M\_HANDSHAKE\_V90\_ONLY**

V.90 only (56kbps to 28kbps)

- **d\_M\_HANDSHAKE\_V34\_AUTO\_FB**

V.34 automatic fallback (33.6kbps to 300bps)

- **d\_M\_HANDSHAKE\_V34\_ONLY**

V.34 only (33.6kbps to 2400 bps)

- **d\_M\_HANDSHAKE\_V32BIS\_AUTO\_FB**

ITU-T V.32bis automatic fallback (14.4kbps to 300 bps)

- **d\_M\_HANDSHAKE\_V32BIS\_ONLY**

ITU-T V.32bis only (14.4kbps to 4800 bps)

- **d\_M\_HANDSHAKE\_V22BIS\_ONLY**

ITU-T V.22bis only (2400bps or 1200bps)

- **d\_M\_HANDSHAKE\_V22\_ONLY**

ITU-T V.22 only (1200bps)

- **d\_M\_HANDSHAKE\_BELL\_212**

Bell 212 only (1200bps)

- **d\_M\_HANDSHAKE\_BELL\_103**

Bell 103 only (300bps)

- **d\_M\_HANDSHAKE\_V21\_ONLY**

ITU\_T V.21 only (300 bps)

- **d\_M\_HANDSHAKE\_V23**

V.23 (1200/75 bps)

[ IN ]    *bCountryCode*

Country Code.

- *d\_M\_COUNTRY\_SPAIN*
- *d\_M\_COUNTRY\_TAIWAN*
- *d\_M\_COUNTRY\_CHINA*
- *d\_M\_COUNTRY\_USA*
- *d\_M\_COUNTRY\_GERMANY*
- *d\_M\_COUNTRY\_FRANCE*
- *d\_M\_COUNTRY\_MALAYSIA*
- *d\_M\_COUNTRY\_ENGLAND*
- *d\_M\_COUNTRY\_THAILAND*
- *d\_M\_COUNTRY\_SINGAPORE*
- *d\_M\_COUNTRY\_JAPEN*
- *d\_M\_COUNTRY\_TURKEY*
- *d\_M\_COUNTRY\_RUSSIA*
- *d\_M\_COUNTRY\_HONGKONG*
- *d\_M\_COUNTRY\_INDONESIA*
- *d\_M\_COUNTRY\_AUSTRALIA*
- *d\_M\_COUNTRY\_BRAZIL*
- *d\_M\_COUNTRY\_CANADA*
- *d\_M\_COUNTRY\_SOUTH\_AFRICA*
- *d\_M\_COUNTRY\_SOUTH\_KOREA*
- *d\_M\_COUNTRY\_QATAR*

**Return Value**

<b>Constants</b>	<b>Value</b>
<i>d_OK</i>	0000h
<a href="#">Modem TCP error codes</a>	23xxh

**Example**

Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemOpenEx

---

```
USHORT CUSHORT CTOS_TCP_ModemOpenEx (BYTE bMode , BYTE bHandShake ,  
BYTE bCountryCode );
```

**Description** Open the Modem device in a separated thread. It will return immediately and do the open operation in background.

- Parameters**
- |        |                     |
|--------|---------------------|
| [ IN ] | <b><u>bMode</u></b> |
|--------|---------------------|
- The mode of Modem connection.
- **d\_M\_MODE\_ASYNC\_NORMAL**  
ASYNC normal connect, all handshake.
  - **d\_M\_HANDSHAKE\_V90\_ONLY**  
V.90 only (56kbps to 28kbps)
  - **d\_M\_MODE\_ASYNC\_FAST**  
ASYNC Fast connect. If you can not dial up by using this setting, please use d\_M\_MODE\_SDLC\_NORMAL instead.
  - **d\_M\_MODE\_SDLC\_NORMAL**  
SYNC SDLC normal connect, used in below handshake:  
d\_M\_HANDSHAKE\_V22BIS\_ONLY  
d\_M\_HANDSHAKE\_V22\_ONLY
  - **d\_M\_MODE\_SDLC\_FAST**  
SYNC SDLC Fast connect, used in below handshakes:  
d\_M\_HANDSHAKE\_V22\_ONLY

[ IN ]	<b><u>bHandShake</u></b>
--------	--------------------------

The handshakes of modem connection.

- ***d\_M\_HANDSHAKE\_V90\_AUTO\_FB***  
V.90 automatic fallback (56kbps to 300bps)
- ***d\_M\_HANDSHAKE\_V90\_ONLY***  
V.90 only (56kbps to 28kbps)
  
- ***d\_M\_HANDSHAKE\_V34\_AUTO\_FB***  
V.34 automatic fallback (33.6kbps to 300bps)
  
- ***d\_M\_HANDSHAKE\_V34\_ONLY***  
V.34 only (33.6kbps to 2400 bps)
  
- ***d\_M\_HANDSHAKE\_V32BIS\_AUTO\_FB***  
ITU-T V.32bis automatic fallback (14.4kbps to 300 bps)
  
- ***d\_M\_HANDSHAKE\_V32BIS\_ONLY***  
ITU-T V.32bis only (14.4kbps to 4800 bps)
  
- ***d\_M\_HANDSHAKE\_V22BIS\_ONLY***  
ITU-T V.22bis only (2400bps or 1200bps)
  
- ***d\_M\_HANDSHAKE\_V22\_ONLY***  
ITU-T V.22 only (1200bps)
  
- ***d\_M\_HANDSHAKE\_BELL\_212***  
Bell 212 only (1200bps)
  
- ***d\_M\_HANDSHAKE\_BELL\_103***  
Bell 103 only (300bps)
  
- ***d\_M\_HANDSHAKE\_V21\_ONLY***  
ITU\_T V.21 only (300 bps)
  
- ***d\_M\_HANDSHAKE\_V23***  
V.23 (1200/75 bps)

[ IN ]    *bCountryCode*

Country Code.

- *d\_M\_COUNTRY\_SPAIN*
- *d\_M\_COUNTRY\_TAIWAN*
- *d\_M\_COUNTRY\_CHINA*
- *d\_M\_COUNTRY\_USA*
- *d\_M\_COUNTRY\_GERMANY*
- *d\_M\_COUNTRY\_FRANCE*
- *d\_M\_COUNTRY\_MALAYSIA*
- *d\_M\_COUNTRY\_ENGLAND*
- *d\_M\_COUNTRY\_THAILAND*
- *d\_M\_COUNTRY\_SINGAPORE*
- *d\_M\_COUNTRY\_JAPEN*
- *d\_M\_COUNTRY\_TURKEY*
- *d\_M\_COUNTRY\_RUSSIA*
- *d\_M\_COUNTRY\_HONGKONG*
- *d\_M\_COUNTRY\_INDONESIA*
- *d\_M\_COUNTRY\_AUSTRALIA*
- *d\_M\_COUNTRY\_BRAZIL*
- *d\_M\_COUNTRY\_CANADA*
- *d\_M\_COUNTRY\_SOUTH\_AFRICA*
- *d\_M\_COUNTRY\_SOUTH\_KOREA*
- *d\_M\_COUNTRY\_QATAR*

**Return Value**

<i>Constants</i>	<i>Value</i>
<i>d_OK</i>	0000h
<a href="#"><u>Modem TCP error codes</u></a>	23xxh

## CTOS\_TCP\_ModemClose

---

```
USHORT CUSHORT CTOS_TCP_ModemClose(void);
```

**Description** Close the modem device.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Modem TCP error codes</a>	23xxh

**Example** Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemDialup

---

```
USHORT CUSHORT CTOS_TCP_ModemDialup(BYTE *baPhone, BYTE *baID, BYTE
*baPW, ULONG ulTimeout);
```

**Description** Dial the number of ISP and get the IP address of this connection.

Please call **CTOS\_TCP\_ModemStatus()** to check whether this action is finished.

**Parameters** [ IN ] *baPhone*

The phone number of the ISP.

[ IN ] *baID*

The ID of this account.

[ IN ] *baPW*

The password of this account.

[ IN ] *ulTimeout*

The dialup timeout value. The base unit of this timeout is 10ms.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Modem TCP error codes</a>	23xxh

**Example**

Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemOnHook

---

```
USHORT CUSHORT CTOS_TCP_ModemOnHook(void);
```

**Description** Hook on this connection.  
Please call **CTOS\_TCP\_ModemStatus()** to check whether this action is finished.

**Parameters** None

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem TCP error codes</a>	23xxh

**Example** Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemGetIP

---

```
USHORT CUSHORT CTOS_TCP_ModemGetIP(BYTE *baIP);
```

**Description** Get the local IP address of the current connection.

**Parameters** [OUT] *baIP*  
The local IP address (4 bytes).

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem TCP error codes</a>	23xxh

**Example** Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemSetIP

---

```
USHORT CUSHORT CTOS_TCP_ModemSetIP(BYTE *baIP);
```

**Description** Set the local IP address of the connection.

**Parameters** [ IN ] *baIP*  
The local IP address, 4 bytes.  
When *baIP* is "\x00\x00\x00\x00", the local IP will be assigned by the ISP.

Return Value	<i>Constants</i>	<i>Value</i>
	d_OK	0000h
	<a href="#">Modem TCP error codes</a>	23xxh

**Example** Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemConnectEx

---

```
USHORT CUSHORT CTOS_TCP_ModemConnectEx(BYTE *bSocket, BYTE *baIP,
USHORT usPort );
```

**Description** Connect to the remote server.  
Please call **CTOS\_TCP\_ModemStatus()** to check whether this action is finished.

**Parameters** [OUT] bSocket  
The socket descriptor of this connection.

[ IN ] baIP  
The pointer of the remote IP address (4 bytes).

[ IN ] usPort  
The pointer of the remote port number.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem TCP error codes</a>	23xxh

**Example** Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemConnectURL

---

```
USHORT CUSHORT CTOS_TCP_ModemConnectURL(BYTE *pbSocket, BYTE
*baIPURL, USHORT usPort );
```

<b>Description</b>	Connect to the remote server using URL.  Please call <b>CTOS_TCP_ModemStatus()</b> to check whether this action is finished.						
<b>Parameters</b>	<p>[ OUT ] <u><i>pbSocket</i></u> The socket descriptor of this connection.</p> <p>[ IN ] <u><i>baIPURL</i></u> URL or IP address in ASCII character. Example: "www.google.com", "61.219.144.201"</p> <p>[ IN ] <u><i>usPort</i></u> The remote port number.</p>						
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Constants</b></th><th><b>Value</b></th></tr> </thead> <tbody> <tr> <td>d_OK</td><td>0000h</td></tr> <tr> <td><a href="#">Modem TCP error codes</a></td><td>23xxh</td></tr> </tbody> </table>	<b>Constants</b>	<b>Value</b>	d_OK	0000h	<a href="#">Modem TCP error codes</a>	23xxh
<b>Constants</b>	<b>Value</b>						
d_OK	0000h						
<a href="#">Modem TCP error codes</a>	23xxh						

## CTOS\_TCP\_ModemDisconnect

---

```
USHORT CUSHORT CTOS_TCP_ModemDisconnect (BYTE bSocket) ;
```

**Description** Disconnect to the remote server.

Please call **CTOS\_TCP\_ModemStatus()** to check whether this action is finished.

**Parameters** [ IN ] *bSocket*

The socket descriptor to be closed.

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">Modem TCP error codes</a>	23xxh

**Example**

Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemTx

---

```
USHORT CUSHORT CTOS_TCP_ModemTx(BYTE bSocket, BYTE *baBuffer, USHORT usLen);
```

**Description** Send data out.  
Please call **CTOS\_TCP\_ModemStatus()** to check whether this action is finished.

**Parameters** [ IN ] *bSocket*  
The socket descriptor of the connection.

[ IN ] *baBuffer*  
The pointer address of sending data.

[ IN ] *usLen*  
The length of the sending data. The maximum is 1024.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem TCP error codes</a>	23xxh

**Example** Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemRx

---

```
USHORT CUSHORT CTOS_TCP_ModemRx(BYTE bSocket, BYTE *baBuffer, USHORT
*usLen);
```

**Description** Receive data.  
Please call **CTOS\_TCP\_ModemStatus()** to check whether this action is finished.

**Parameters** [ IN ] bSocket  
The socket descriptor of the connection.

[ OUT ] baBuffer  
The pointer address of received buffer.

[ IN ] usLen  
Buffer size of baBuffer.

[ OUT ] usLen  
The length of the sending data. The maximum is 1024.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem TCP error codes</a>	23xxh

**Example** Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemStatus

---

```
USHORT CUSHORT CTOS_TCP_ModemStatus (DWORD *pdwState);
```

**Description** Get the current processing status when calling the Modem.

**Parameters** [ OUT] *pdwState*

The current processing status.

- **TCP\_MODEM\_STATE\_ONLINE**
- **TCP\_MODEM\_STATE\_DIALING**
- **TCP\_MODEM\_STATE\_CONNECTING**
- **TCP\_MODEM\_STATE\_SENDING**
- **TCP\_MODEM\_STATE RECEIVING**
- **TCP\_MODEM\_STATE\_DISCONNECTING**
- **TCP\_MODEM\_STATE\_ONHOOKING**

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Modem TCP error codes</a>	23xxh

**Example**

Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_TCP\_ModemGetDNSServer

---

```
USHORT CUSHORT CTOS_TCP_ModemGetDNSServer (BYTE *baIP) ;
```

**Description** Get the DNS IP address of the current connection.

**Parameters** [ OUT ] *baIP*  
DNS IP address (4 bytes).

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem TCP error codes</a>	23xxh

## CTOS\_TCP\_ModemSetDNSServer

---

```
USHORT CUSHORT CTOS_TCP_ModemSetDNSServer (BYTE *baIP) ;
```

**Description** Set the DNS IP address of the current connection.

**Parameters** [ IN ] *baIP*  
DNS IP address, 4 bytes.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem TCP error codes</a>	23xxh

## CTOS\_TCP\_ModemURL2IP

---

USHORT CUSHORT CTOS\_TCP\_ModemURL2IP(STR\* strURL, BYTE\* baIP);

**Description**      Query IP address for specific URL from DNS server.

**Parameters**      [ IN ]      *strURL*  
                          URL to query in ASCII character.  
                          Example: "www.google.com".

[ OUT ]      *baIP*  
                          Response IP address, 4 bytes.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem TCP error codes</a>	23xxh

## CTOS\_TCP\_ModemPing

---

```
USHORT CUSHORT CTOS_TCP_ModemPing(STR *strIPURL, BYTE bTimeout);
```

**Description** Test the reachability of a destination IP address on the network.

**Parameters** [ IN ] *strIPURL*

URL or IP address in ASCII character.

Example: "www.google.com", "61.219.144.201"

[ IN ] *bTimeout*

Timeout in second.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Modem TCP error codes</a>	23xxh

## CTOS\_UDP\_ModemTx

---

```
USHORT CUSHORT CTOS_UDP_ModemTx(BYTE *baIP, USHORT usPort, BYTE
*baBuffer, USHORT usLen);
```

**Description** Send UDP packages to remote IP and specified port.  
Using this function is not necessary to connect to remotes by the **CTOS\_TCP\_ModemConnectEx()**.

**Parameters** [ IN ] *baIP*  
Remote IP address to send the UDP package.

[ IN ] *usPort*  
Remote UDP port.

[ IN ] *baBuffer*  
Sending data.

[ IN ] *usLen*  
Sending length of data.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Modem TCP error codes</a>	23xxh

**Example** Please refer to example in appendix [B.22 Modem TCP](#)

## CTOS\_UDP\_ModemRx

---

```
USHORT CUSHORT CTOS_UDP_ModemRx(BYTE *bSrcDestIP, USHORT *usPort,
BYTE *baBuffer, USHORT *usLen);
```

<b>Description</b>	Receive UDP packages.  Using this function is not necessary to connect to remotes by the <b>CTOS_TCP_ModemConnectEx()</b> .  Please note this function will remove any TCP package received.						
<b>Parameters</b>	<p>[OUT] <u><i>bSrcDestIP</i></u> Source IP address and Destination IP address (8 bytes).</p> <p>[ OUT ] <u><i>usPort</i></u> Destination Port number.</p> <p>[ OUT ] <u><i>baBuffer</i></u> Receive data buffer.</p> <p>[ IN ] <u><i>usLen</i></u> Buffer length of baBuffer.</p> <p>[ OUT ] <u><i>usLen</i></u> Receive data length.</p>						
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Constants</b></th><th><b>Value</b></th></tr> </thead> <tbody> <tr> <td>d_OK</td><td>0000h</td></tr> <tr> <td><a href="#">Modem TCP error codes</a></td><td>23xxh</td></tr> </tbody> </table>	<b>Constants</b>	<b>Value</b>	d_OK	0000h	<a href="#">Modem TCP error codes</a>	23xxh
<b>Constants</b>	<b>Value</b>						
d_OK	0000h						
<a href="#">Modem TCP error codes</a>	23xxh						
<b>Example</b>	Please refer to example in appendix <a href="#">B.22 Modem TCP</a>						

## 2.13.Ethernet Functions

### General

- CTOS\_EthernetOpen
- CTOS\_EthernetOpenEx
- CTOS\_EthernetClose
- CTOS\_EthernetConfigGet
- CTOS\_EthernetConfigSet
- CTOS\_EthernetPing
- CTOS\_EthernetIP2MAC
- CTOS\_EthernetEnable
- CTOS\_EthernetDisable

### Single-socket

- CTOS\_EthernetConnect
- CTOS\_EthernetConnectEx
- CTOS\_EthernetConnectURL
- CTOS\_EthernetConnectURLEx
- CTOS\_EthernetConnectURLPort
- CTOS\_EthernetDisconnect
- CTOS\_EthernetTxReady
- CTOS\_EthernetTxData
- CTOS\_EthernetRxReady
- CTOS\_EthernetRxData
- CTOS\_EthernetStatus
- CTOS\_EthernetGetConnectionInfo
- CTOS\_EthernetFlushRxData
- CTOS\_EthernetURL2IP

### Multi-socket

- CTOS\_EthernetMSConnect
- CTOS\_EthernetMSConnectEx
- CTOS\_EthernetMSDisconnect
- CTOS\_EthernetMSTxReady

- CTOS\_EthernetMSTxData
- CTOS\_EthernetMSRxReady
- CTOS\_EthernetMSRxData
- CTOS\_EthernetMSStatus
- CTOS\_EthernetMSGetConnectionInfo
- CTOS\_EthernetMSListen
- CTOS\_EthernetMSUnlisten
- CTOS\_EthernetMSRxbufSet

## **FTP**

**(alternatively you can use cURL API to communicate with a FTP server)**

- CTOS\_EthernetFTPConnect
- CTOS\_EthernetFTPDDisconnect
- CTOS\_EthernetFTPGetState
- CTOS\_EthernetFTPList
- CTOS\_EthernetFTPChangeDir
- CTOS\_EthernetFTPDownload
- CTOS\_EthernetFTPDownloadFile
- CTOS\_EthernetFTPUpload
- CTOS\_EthernetFTPUploadFile
- CTOS\_EthernetFTPTranStatus
- CTOS\_EthernetFTPCancel

## **Ethernet Error Codes**

<b>Constants</b>	<b>Value</b>
d_ETHERNET_INVALID_PARA	4201h
d_ETHERNET_NOT_OPEN	4202h
d_ETHERNET_NOT_ONLINE	4203h
d_ETHERNET_TIMEOUT	4204h
d_ETHERNET_INSUFFICIENT_CMD_BUF	4205h
d_ETHERNET_CMD_INVALID_RESPONSE	4206h
d_ETHERNET_CMD_NOT_SUCCESS	4207h
d_ETHERNET_TX_BUSY	4208h
d_ETHERNET_NOT_SUPPORTED	4209h

d_ETHERNET_NOT_DATA_MODE	420Ah
d_ETHERNET_SOCKET_ALREADYOPENED	420Bh
d_ETHERNET_ALREADY_OPENED	420Ch
d_ETHERNET_HARDWARE_ERROR	420Dh
d_ETHERNET_GDB_RUNNING	420Eh
d_ETHERNET_DNS_INVALID_FORMAT	4214h
d_ETHERNET_DNS_SERVER_FAILURE	4215h
d_ETHERNET_DNS_REQUEST_NOT_EXIST	4216h
d_ETHERNET_DNS_OPCODE_NOT_SUPPORT	4217h
d_ETHERNET_DNS_SERVER_REFUSED	4218h
d_ETHERNET_FTP_NOT_OPEN	4219h
d_ETHERNET_FTP_GETDIR_FAILED	421Ah
d_ETHERNET_FTP_CHANGEDIR_FAILED	421Bh
d_ETHERNET_FTP_LIST_NOT_READY	421Ch
d_ETHERNET_FTP_BUSY	421Dh
d_ETHERNET_FTP_REQUEST_PASSWORD	421Eh
d_ETHERNET_FTP_LOGIN_FAILED	421Fh
d_ETHERNET_FTP_RESPONSE_TIMEOUT	4220h
d_ETHERNET_FTP_DATACONN_FAILED	4222h
d_ETHERNET_FTP_RECONNECTION_FAILED	4223h
d_ETHERNET_FTP_DOWNLOADFILE_FAILED	4224h
d_ETHERNET_FTP_UPLOADFILE_FAILED	4225h
d_ETHERNET_NOT_ENABLE	4226h
d_ETHERNET_FTP_GETFILESIZE_FAILED	4228h

## CTOS\_EthernetOpen

---

```
USHORT CUSHORT CTOS_EthernetOpen(void);
```

**Description** Open the Ethernet port using Castles proprietary protocol stack.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Example** Please refer to example in appendix[B.2 Ethernet](#)

**Note** To use native Linux's protocol stack, refer to CTOS\_EthernetOpenEx.

## CTOS\_EthernetOpenEx

---

```
USHORT CUSHORT CTOS_EthernetOpenEx( void );
```

**Description** Open the Ethernet port to using Linux socket function.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Note** Before using Linux socket function, **CTOS\_EthernetOpenEx()** must be called for initializing the system environment.

## CTOS\_EthernetClose

---

```
USHORT CUSHORT CTOS_EthernetClose( void );
```

**Description** Close the Ethernet port.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Example** Please refer to example in appendix [B.2 Ethernet](#)

**Notes** All the data received and about to be transmitted will be discarded, and all hardware and software resource will be released.

This function works for communications opened with both CTOS\_EthernetOpen() and CTOS\_EthernetOpenEx().

## CTOS\_EthernetConfigGet

---

```
USHORT CUSHORT CTOS_EthernetConfigGet (BYTE bTag, BYTE*baValue,  
BYTE*pbLen);
```

**Description** Get the configuration value of the Ethernet port.

**Parameters** [ IN ] *bTag*

- ***d\_ETHERNET\_CONFIG\_IP***

Get IP Address.

- ***d\_ETHERNET\_CONFIG\_MASK***

Get Network Mask.

- ***d\_ETHERNET\_CONFIG\_GATEWAY***

Get Gateway Address.

- ***d\_ETHERNET\_CONFIG\_HOSTIP***

Get Host IP Address.

- ***d\_ETHERNET\_CONFIG\_HOSTPORT***

Get Host Port Number.

- ***d\_ETHERNET\_CONFIG\_VERSION***

Get Firmware Version.

- ***d\_ETHERNET\_CONFIG\_MAC***

Get MAC Address.

- ***d\_ETHERNET\_CONFIG\_DHCP***

Get IP configuration.

= 0 : Static. Use the static IP set in the Ethernet configuration.

= 1 : DHCP. Retrieve the dynamic IP from the DHCP

server.

- **d\_Ethernet\_Config\_DNSIP**

Get DNS Server IP Address

- **d\_Ethernet\_Config\_HostUrl**

Get Host Domain Name.

- **d\_Ethernet\_Config\_AutoCon**

Get Connection Mode.

= 0 : Auto-connect. When Auto-connect is set, the Ethernet module will automatically try to connect to the host every 5 seconds.

= 1 : Not support

= 2 : Manual. The connection must be established manually by calling CTOS\_EthernetConnectxxx() function.

[ OUT ] *baValue*

Buffer to store the configuration data.

[ IN ] *pbLen*

Size of buffer.

[ OUT ] *pbLen*

Length of configuration data returned.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Example**

Please refer to example in appendix [B.2 Ethernet](#)

## CTOS\_EthernetConfigSet

---

```
USHORT CUSHORT CTOS_EthernetConfigSet (BYTE bTag, BYTE*baValue,  
BYTE bLen) ;
```

**Description** Set the configuration of the Ethernet port.

**Parameters** [ IN ] *bTag*

- ***d\_ETHERNET\_CONFIG\_IP***

Set IP Address.

- ***d\_ETHERNET\_CONFIG\_MASK***

Set Network Mask.

- ***d\_ETHERNET\_CONFIG\_GATEWAY***

Set Gateway Address.

- ***d\_ETHERNET\_CONFIG\_HOSTIP***

Set Host IP Address.

- ***d\_ETHERNET\_CONFIG\_HOSTPORT***

Set Host Port Number.

- ***d\_ETHERNET\_CONFIG\_DHCP***

Set IP configuration.

= 0 : Static. Use the static IP set in the Ethernet configuration.

= 1 : DHCP. Retrieve the dynamic IP from the DHCP server.

- ***d\_ETHERNET\_CONFIG\_DNSIP***

Set DNS Server IP Address

- **d\_Ethernet\_Config\_HostUrl**

Set Host address with its Domain Name. The IP Address will be resolved by current DNS Server directly.

- **d\_Ethernet\_Config\_AutoCon**

Set Connection Mode.

= 0 : Auto-connect. When Auto-connect is set, the Ethernet module will automatically try to connect to the host every 5 seconds.

= 1 : Not support

= 2 : Manual. The connection must be established manually by calling CTOS\_EthernetConnectxxx() function.

- **d\_Ethernet\_Config\_Update\_Exit**

Save the settings to the non-volatile memory in Ethernet module.

[ IN ] *baValue*

Configuration data to set. Data should be in the form of ASCII characters, i.e. "192.120.100.235".

[ IN ] *bLen*

Length of configuration data.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Example**

Please refer to example in appendix [B.2 Ethernet](#)

**Note**

The configuration is effective immediately after successfully configured, but will only be written into the non-volatile memory after calling the function with tag:

**d\_ETHERNET\_CONFIG\_UPDATE\_EXIT**.Upon the Ethernet module reset, the initial configuration will be retrieved from the non-volatile memory. Therefore, programmers should call this function with tag **d\_ETHERNET\_CONFIG\_UPDATE\_EXIT** after all the configuration is set if the setting is considered to be effective after reboot.

## CTOS\_EthernetPing

---

```
USHORT CUSHORT CTOS_EthernetPing(BYTE*baDestIP, BYTE bLen);
```

**Description** Test the reachability of a destination IP address on the network.

**Parameters**

[ IN ]	<u><i>baDestIP</i></u>
	Destination IP address in the form of ASCII character.
	Example: "192.120.100.168".

[ IN ]	<u><i>bLen</i></u>
	Length of the destination IP address.
	Example: IP="192.120.100.168", bLen =15.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Ethernet error codes</a>	42xxh

**Example** Please refer to example in appendix [B.2 Ethernet](#)

## CTOS\_EthernetURL2IP

---

```
USHORT CTOS_EthernetURL2IP(BYTE*baDestURL, BYTE bURLLen,
BYTE*baRspIP, BYTE*pbRspIPLen);
```

**Description** Query IP address for specific URL from DNS server.

**Parameters**

[ IN ]	<u><i>baDestURL</i></u>
	URL to query in ASCII character.
	Example: "www.google.com".

[ IN ]	<u><i>bURLLen</i></u>
	Length of URL in byte.
	Example: URL="www.google.com", bURLLen =14.

[ OUT ]	<u><i>baRspIP</i></u>
	Response IP address in ASCII character.
	Example: "192.120.100.168".

[ IN ]	<u><i>pbRspIPLen</i></u>
	Buffer size of baRspIP.

[ OUT ]	<u><i>pbRspIPLen</i></u>
	Length of response IP address in byte.
	Example: IP="192.120.100.168", baRspIPLen =15.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetIP2MAC

---

```
USHORT CUSHORT CTOS_EthernetIP2MAC(BYTE* baDestIP, BYTE bIPLen,
BYTE* baRspMAC, BYTE* pbRspMACLen);
```

**Description** Search MAC for specific IP address.

**Parameters**

[ IN ]	<i>baDestIP</i>
	IP address to search in ASCII character.
	Example: "192.120.100.168".

[ IN ]	<i>bIPLen</i>
	Length of IP address in byte.
	Example: IP="192.120.100.168", bIPLen =15.

[ OUT ]	<i>baRspMAC</i>
	Response MAC address in ASCII character.
	Example: "00-10-12-13-14-18".

[ OUT ]	<i>pbRspMACLen</i>
	Length of response MAC address in byte.
	Example: MAC="00-10-12-13-14-18", baRspMACLen =17.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Ethernet error codes</a>	42xxh

**Note** This function is not supported, if the connection is opened using **CTOS\_EthernetOpenEx()**.

## CTOS\_EthernetEnable

---

```
USHORT CUSHORT CTOS_EthernetEnable( void );
```

**Description** Enable the Ethernet module.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Note** Ethernet module is default enabled upon reboot.

## CTOS\_EthernetDisable

---

```
USHORT CUSHORT CTOS_EthernetDisable( void );
```

**Description** Disable Ethernet module.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Note** When the Ethernet module is disabled, it will not answer to any TCP/IP request from others. For example, ARP request received will not be answered. Before calling CTOS\_EthernetDisable(), be sure to call CTOS\_EthernetClose() first to terminate all the services.

## CTOS\_EthernetConnect

---

```
USHORT CUSHORT CTOS_EthernetConnect(void);
```

**Description** Connect to the host with default host IP address and port number stored in the configuration of Ethernet module after calling CTOS\_EthernetConfigSet()

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Example** Please refer to example in appendix [B.2 Ethernet](#)

## CTOS\_EthernetConnectEx

---

```
USHORT CUSHORT CTOS_EthernetConnectEx(BYTE*baDestIP, BYTE bIPLen,
BYTE*baPort, BYTE bPortLen);
```

**Description** Connect to the host with specific host IP address and port number.

**Parameters**

[ IN ]	<i>baDestIP</i>
	Host IP address in the form of ASCII character.
	Example: "192.120.100.168".

[ IN ]	<i>bIPLen</i>
	Length of host IP address.
	Example: IP="192.120.100.168", bIPLen =15.

[ IN ]	<i>baPort</i>
	Host port number in the form of ASCII character.
	Example: "5000".

[ IN ]	<i>bPortLen</i>
	Length of host port number.
	Example: "5000", bPortLen =4.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Ethernet error codes</a>	42xxh

**Example** Please refer to example in appendix [B.2 Ethernet](#)

## CTOS\_EthernetConnectURL

---

```
USHORT CUSHORT CTOS_EthernetConnectURL( void );
```

**Description** Connect to the host with default URL stored in the Ethernet module after calling CTOS\_EthernetConfigSet()

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetConnectURLEx

---

```
USHORT CUSHORT CTOS_EthernetConnectURLEx(BYTE*baDestURL, BYTE  
bURLLen);
```

**Description** Connect to the host with specific URL.

**Parameters** [ IN ] *baDestURL*  
URL to connect in ASCII character.  
Example: "www.google.com" ..

[ IN ] *bURLLen*  
Length of URL in byte.  
Example: URL="www.google.com", bURLLen = 14.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetConnectURLPort

---

```
USHORT CUSHORT CTOS_EthernetConnectURLEx(BYTE*baDestURL, BYTE bURLLen, BYTE* baPort, BYTE bPortLen);
```

**Description** Connect to the host with specific URL and port number.

**Parameters**

[ IN ]	<u><i>baDestURL</i></u>
	URL to connect in ASCII character.
	Example: "www.google.com".

[ IN ]	<u><i>bURLLen</i></u>
	Length of URL in byte.
	Example: URL="www.google.com", bURLLen = 14.

[ IN ]	<u><i>baPort</i></u>
	Port number in ASCII character.
	Example: "443".

[ IN ]	<u><i>bPortLen</i></u>
	Length of port number string in byte.
	Example: PortNum="443", bPortNumLen = 3.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetDisconnect

---

```
USHORT CUSHORT CTOS_EthernetDisconnect(void);
```

**Description** Disconnect from the host.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Example** Please refer to example in appendix [B.2 Ethernet](#)

## CTOS\_EthernetTxReady

---

```
USHORT CUSHORT CTOS_EthernetTxReady( void );
```

**Description** Check if the Ethernet port is ready for sending next data.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Example** Please refer to example in appendix [B.2 Ethernet](#)

## CTOS\_EthernetTxData

---

```
USHORT CUSHORT CTOS_EthernetTxData (BYTE*baData, USHORT usLen);
```

**Description** Send data through the Ethernet port.

**Parameters** [ IN ] *baData*  
Data to send.

[ IN ] *usLen*  
Length of data. The maximum value is 2048.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Ethernet error codes</a>	42xxh

**Example** Please refer to example in appendix [B.2 Ethernet](#)

## CTOS\_EthernetRxReady

---

```
USHORT CUSHORT CTOS_EthernetRxReady(USHORT*pusLen) ;
```

**Description** Get the number of data currently received from the Ethernet port.

**Parameters** [ OUT ] *pusLen*  
Length of data received.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Ethernet error codes</a>	42xxh

**Example** Please refer to example in appendix [B.2 Ethernet](#)

## **CTOS\_EthernetRxData**

---

USHORT CUSHORT CTOS\_EthernetRxData (BYTE\*baData, USHORT\*pusLen) ;

**Description** Get the data received from the Ethernet port.

**Parameters** [ OUT ] *baData*  
Buffer to store the received data.

[ IN ] *pusLen*  
Size of buffer. Maximum number of data to get.

[ OUT ] *pusLen*  
Length of the data actually returned in the buffer.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Ethernet error codes</a>	42xxh

**Example** Please refer to example in appendix [B.2 Ethernet](#)

**Note** Programmer can call **CTOS\_EthernetRxReady()** to check the number of data currently received, and then call this function with corresponding size of buffer get the data.  
If the size input is smaller than the actual number of data received, then only the size number of data will be returned.  
If the size input is bigger than the actual number of data, then the actual number of data returned will be stored in pusLen.

## CTOS\_EthernetStatus

---

USHORT CTOS\_EthernetStatus (DWORD\* pdwStatus) ;

**Description** Get the status of the Ethernet.

**Parameters** [ IN ] *pdwStatus*

Point to a DWORD to store the current Ethernet status. Status is returned in the bitwise-OR of following bitmask.

- ***d\_STATUS\_ETHERNET\_CONNECTED***

Connection to the host has been established.

- ***d\_STATUS\_ETHERNET\_COMMAND\_MODE***

Ethernet module is currently in command mode.

- ***d\_STATUS\_ETHERNET\_PHYSICAL\_ONLINE***

Physical network line is connected.

- ***d\_STATUS\_ETHERNET\_RX\_READY***

Data available in receive buffer.

- ***d\_STATUS\_ETHERNET\_TX\_BUSY***

Ethernet module is not ready to send data.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Example**

Please refer to example in appendix [B.2 Ethernet](#)

## CTOS\_EthernetGetConnectionInfo

---

```
USHORT CUSHORT CTOS_EthernetGetConnectionInfo (BYTE*baIP,
BYTE*pbIPLen, BYTE*baPort, BYTE*pbPortLen);
```

**Description** Get the IP and Port of current established connection.

- Parameters**
- [ OUT ] *baIP*  
Buffer to store the IP address of remote side. The IP address is stored in ASCII character.
  - [ IN ] *pbIPLen*  
Size of the buffer.
  - [ OUT ] *pbIPLen*  
Length of IP address returned in the buffer.
  - [ OUT ] *baPort*  
Buffer to store the port number of remote side. The port number is stored in ASCII character.
  - [ IN ] *pbPortLen*  
Size of the buffer.
  - [ OUT ] *pbPortLen*  
Length of port number returned in the buffer.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Example**

Please refer to example in appendix [B.2 Ethernet](#)

## CTOS\_EthernetFlushRxData

---

```
USHORT CUSHORT CTOS_EthernetFlushRxData(void);
```

**Description** Flush the data already received from the Ethernet port.

**Parameters** None

Return Value	<b>Constants</b>	<b>Value</b>
d_OK		0000h
<a href="#">Ethernet error codes</a>		42xxh

**Example** Please refer to example in appendix [B.2 Ethernet](#)

**Note** All the data received before calling this function will be discarded.

## CTOS\_EthernetMSConnect

---

```
USHORT CUSHORT CTOS_EthernetMSConnect (BYTE bMSSocket) ;
```

**Description** Connect to the host with default host IP address and port number stored in the configuration of Ethernet module.

**Parameters** [ IN ] *bMSSocket*  
Socket number. The range is from 0 to 2.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetMSConnectEx

---

```
USHORT CUSHORT CTOS_EthernetMSConnectEx(BYTE bMSsocket,
BYTE*baDestIP, BYTE bIPLen, BYTE*baPort, BYTE bPortLen);
```

**Description** Connect to the host with specific host IP address and port number.

**Parameters** [ IN ] *bMSsocket*  
Socket number. The range is from 0 to 2.

[ IN ] *baDestIP*  
Host IP address in the form of ASCII character.  
Example: "192.120.100.168".

[ IN ] *bIPLen*  
Length of host IP address.  
Example: IP="192.120.100.168", bIPLen =15.

[ IN ] *baPort*  
Host port number in the form of ASCII character.  
Example: "5000".

[ IN ] *bPortLen*  
Length of host port number.  
Example: "5000", bPortLen =4.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Ethernet error codes</a>	42xxh

**Note** Currently maximum 3 sockets can be connected concurrently.

## CTOS\_EthernetMSDisconnect

---

```
USHORT CUSHORT CTOS_EthernetMSDisconnect (BYTE bMSSocket) ;
```

**Description** Disconnect from the host.

**Parameters** [ IN ] *bMSSocket*  
Socket number. The range is from 0 to 2.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetMSTxReady

---

```
USHORT CUSHORT CTOS_EthernetMSTxReady (BYTE bMSSocket) ;
```

**Description** Check if the Ethernet port is ready for sending next data.

**Parameters** [ IN ] bMSSocket  
Socket number. The range is from 0 to 2.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetMSTxData

---

```
USHORT CUSHORT CTOS_EthernetMSTxData(BYTE bMSocket, BYTE *baData,  
USHORT usLen);
```

**Description** Send data through the Ethernet port.

**Parameters** [ IN ] *bMSocket*  
Socket number. The range is from 0 to 2.

[ IN ] *baData*  
Data to send.

[ IN ] *usLen*  
Length of data. The maximum value is 2048.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetMSRxReady

---

```
USHORT CTOS_EthernetMSRxReady (BYTE bMSSocket, DWORD* pdwLen);
```

**Description** Get the number of data currently received from the Ethernet port

**Parameters** [ IN ] bMSSocket  
Socket number. The range is from 0 to 2.

[ OUT ] pdwLen  
Length of data received.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetMSRxData

---

```
USHORT CTOS_EthernetMSRxData(BYTE bMSSocket, BYTE *baData, DWORD*
pdwLen);
```

**Description** Get the data received from the Ethernet port.

**Parameters**

[ IN ]	<i>bMSSocket</i>
Socket number. The range is from 0 to 2.	

[ OUT ]	<i>baData</i>
Buffer to store the received data.	

[ IN ]	<i>pusLen</i>
Size of buffer. Maximum number of data to get.	

[ OUT ]	<i>pusLen</i>
Length of the data actually returned in the buffer.	

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetMSStatus

---

```
USHORT CTOS_EthernetMSStatus (DWORD* pdwMSStatus);
```

**Description** Get the status of Ethernet.

**Parameters** [ OUT ] *pdwMSStatus*

Point to a DWORD to store the current Ethernet status. Status is returned in the bitwise-OR of following bitmask.

- ***d\_STATUS\_ETHERNET\_CONNECTED***

Connection is established on SOCKET0.

- ***d\_STATUS\_ETHERNET\_COMMAND\_MODE***

Ethernet module is in command mode.

- ***d\_STATUS\_ETHERNET\_PHYSICAL\_ONLINE***

Physical network line is connected.

- ***d\_STATUS\_ETHERNET\_RX\_READY***

Data is received and stored in receive buffer.

- ***d\_STATUS\_ETHERNET\_TX\_BUSY***

Ethernet module is busy and not able to transmit any data.

- ***d\_STATUS\_ETHERNET\_MS\_CONNECTED\_SOCKET0***

Connection is established on SOCKET0.

- ***d\_STATUS\_ETHERNET\_MS\_CONNECTED\_SOCKET1***

Connection is established on SOCKET1.

- ***d\_STATUS\_ETHERNET\_MS\_CONNECTED\_SOCKET2***

Connection is established on SOCKET2.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetMSGGetConnectionInfo

---

```
USHORT CTOS_EthernetMSGGetConnectionInfo (BYTE bMSSocket, BYTE*baIP,
BYTE*pbIPLen, BYTE*baPort, BYTE*pbPortLen);
```

**Description** Get the IP and Port of current established connection.

**Parameters**

- [ IN ] *bMSSocket*  
Socket number. The range is from 0 to 2.
- [ OUT ] *baIP*  
Buffer to store the IP address of remote side. The IP address is stored in ASCII character..
- [ IN ] *pbIPLen*  
Size of the buffer..
- [ OUT ] *pbIPLen*  
Length of IP address returned in the buffer.
- [ OUT ] *baPort*  
Buffer to store the port number of remote side. The port number is stored in ASCII character.
- [ IN ] *pbPortLen*  
Size of the buffer.
- OUT ] *pbPortLen*  
Length of port number returned in the buffer.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetMSListen

---

```
USHORT CTOS_EthernetMSListen (BYTE bMSSocket, BYTE*baListenPort, BYTE
bListenPortLen);
```

**Description** Enter the listen mode and wait for connecting.

**Parameters** [ IN ] *bMSSocket*  
Socket number. The range is from 0 to 2.

[ IN ] *baListenPort*  
Port to listen in ASCII character.  
Example: "5000".

[ IN ] *bListenPortLen*  
Length of port in byte.  
Example: "5000", bListenPortLen =4.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetMSUnlisten

---

```
USHORT CTOS_EthernetMSUnlisten(BYTE bMSSocket);
```

**Description** Leave the listen mode

**Parameters** [ IN ] bMSSocket  
Socket number. The range is from 0 to 2.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetMSRxbufSet

---

```
USHORT CTOS_EthernetMSRxbufSet (BYTE bMSSocket, BYTE *baBuf, DWORD dwBufsize);
```

**Description** Set the user-defined RX buffer for storing the data received from Ethernet port. If no user-defined RX buffer is set, a default 2K buffer will be used.

**Parameters** [ IN ] *bMSSocket*  
Socket number. The range is from 0 to 2.

[ IN ] *baBuf*  
Buffer to set.

[ IN ] *dwBufsize*  
Size of buffer.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
d_OK		0000h
<a href="#">Ethernet error codes</a>		42xxh

**Note** Programmer can call this function to provide larger buffer for storing data from Ethernet port. The maximum size of user-buffer is limited to 10K.

## CTOS\_EthernetFTPConnect

---

```
USHORT CTOS_EthernetFTPConnect (BYTE* baHost, BYTE bHostLen,
BYTE* baPort, BYTE bPortLen, BYTE* baUserName, BYTE bUserNameLen,
BYTE* baPassword, BYTE bPasswordLen, BOOL fActiveMode);
```

**Description** Connect to the FTP host with specific host IP address and port number.

**Parameters** [ IN ] *baHost*  
The host IP address which is in ASCII character.  
Example: "192.120.100.168"

[ IN ] *bHostLen*  
The number of bytes of host IP address.  
Example: IP="192.120.100.168", bIPLen =15

[ IN ] *baPort*  
The host Port number which is in ASCII character.  
Example: "5000", bPortLen =4

[ IN ] *bPortLen*  
The number of bytes of host port.

[ IN ] *baUserName*  
The input login username which is in ASCII character.  
Example: "user123", bUserNameLen =7

[ IN ] *bUserNameLen*  
The number of bytes of baUserName.

[ IN ] *baPassword*  
The input login password which is in ASCII character.  
Example: "password123", bPasswordLen =11.

[ IN ] *bPasswordLen*

The number of bytes of baPassword.

[ IN ] *fActiveMode*

The connecting mode for FTP.

= 0 : Passive Mode(Recommended)

= 1 : Active Mode

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Note**

When using FTP, socket1 and socket2 are unable to use.

## CTOS\_EthernetFTPDisconnect

---

```
USHORT CTOS_EthernetFTPDisconnect ( void );
```

**Description** Disconnect from the FTP host.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetFTPGetState

---

```
USHORT CTOS_EthernetFTPGetState (BYTE* pbFTPState);
```

**Description**      Query the FTP state.

**Parameters**      [ OUT] *pbFTPState*  
The FTP state which is in integer.  
= 0 : FTP is offline.  
= 1 : FTP is already online.  
= 2 : FTP is already online and busy in downloading file.  
= 3 : FTP is already online and busy in uploading file.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetFTPList

---

```
USHORT CTOS_EthernetFTPList (BYTE* baDataDir, DWORD* pdwDataDirLen,
BYTE* baDataFile, DWORD* pdwDataFileLen);
```

**Description** Get the DirectoryName and FileName list from FTP host.

**Parameters**

[ OUT] *baDataDir*  
The DirectoryName data which is in ASCII character. Each  
DirectoryName is seprated with "\x0D\x0A".

[ IN ] *pdwDataDirLen*  
Length of DirectoryName data to receive.

[ OUT ] *pdwDataDirLen*  
Actual length of DirectoryName data stored in buffer.

[ OUT] *baDataFile*  
The FileName data which is in ASCII character. Each  
FileName is seprated with "\x0D\x0A".

[ IN ] *pdwDataFileLen*  
Length of FileName data to receive.

[ OUT ] *pdwDataFileLen*  
Actual length of FileName data stored in buffer.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetFTPChangeDir

---

```
USHORT CTOS_EthernetFTPChangeDir (BYTE* baChangeDirName, USHORT
usChangeDirLen, BYTE* baDataDir, DWORD* pdwDataDirLen, BYTE*
baDataFile, DWORD* pdwDataFileLen);
```

**Description** Change to specific directory, then get the DirectoryName and FileName list from FTP host.

**Parameters** [ IN ] *baChangeDirName*  
The directory name which is in ASCII character.

[ IN ] *usChangeDirLen*  
The number of bytes of baChangeDirName.

[ OUT ] *baDataDir*  
The DirectoryName data which is in ASCII character. Each  
DirectoryName is seprated with "\x0D\x0A".

[ IN ] *pdwDataDirLen*  
Length of DirectoryName data to receive.

[ OUT ] *pdwDataDirLen*  
Actual length of DirectoryName data stored in buffer.

[ OUT ] *baDataFile*  
The FileName data which is in ASCII character. Each  
FileName is seprated with "\x0D\x0A".

[ IN ] *pdwDataFileLen*  
Length of FileName data to receive.

[ OUT ] *pdwDataFileLen*  
Actual length of FileName data stored in buffer.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h

<i>ETHERNET</i>	<a href="#">Ethernet error codes</a>	42xxh
		VEGA5000   VEGA5000S   VEGA3000

## CTOS\_EthernetFTPDownload

---

```
USHORT CTOS_EthernetFTPDownload (BYTE* baFileName, USHORT
usFileNameLen, BYTE* baFileData, DWORD* pdwFileDataLen);
```

**Description** Download the file with specific input file name which is in ASCII character.

**Parameters** [ IN ] *baFileName*  
The file name which is in ASCII character.

[ IN ] *usFileNameLen*  
The number of bytes of baFileName.

[ OUT] *baFileData*  
Received file data buffer.

[ IN ] *pdwFileDataLen*  
Length of file data to receive.

[ OUT ] *pdwFileDataLen*  
Actual length of data stored in buffer.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Ethernet error codes</a>	42xxh

**Note** If user does not execute **CTOS\_EthernetFTPList** command yet, execute **CTOS\_EthernetFTPDownload** will return error code.

## CTOS\_EthernetFTPDownloadFile

---

```
USHORT CTOS_EthernetFTPDownloadFile (BYTE* baFileName);
```

**Description** Download the file with specific input file name which is in ASCII character in to application data folder.

**Parameters** [ IN ] *baFileName*  
The file name which is in ASCII character and null terminated.

Return Value	<i>Constants</i>	<i>Value</i>
d_OK		0000h
<a href="#">Ethernet error codes</a>		42xxh

**Note** Please use CTOS File functions to access to file downloaded.

## CTOS\_EthernetFTPUpload

---

```
USHORT CTOS_EthernetFTPUpload (BYTE* baFileName, USHORT
usFileNameLen, BYTE* baFileData, DWORD dwFileDataLen);
```

**Description** Upload the file with specific input file name which is in ASCII character.

**Parameters**

[ IN ]	<i>baFileName</i>
The file name which is in ASCII character.	

[ IN ]	<i>usFileNameLen</i>
The number of bytes of baFileName.	

[ IN ]	<i>baFileData</i>
Data buffer storing the file data to send.	

[ IN ]	<i>dwFileDataLen</i>
Length of file data to send.	

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Ethernet error codes</a>	42xxh

**Note** If user does not execute **CTOS\_EthernetFTPList** command yet, execute **CTOS\_EthernetFTPUpload** will return error code.

## CTOS\_EthernetFTPUploadFile

---

```
USHORT CTOS_EthernetFTPUploadFile (BYTE* baFileName);
```

**Description** Upload the file with specific input file name which is in ASCII character from application data folder.

**Parameters** [ IN ] *baFileName*  
The file name which is in ASCII character and null terminated.

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetFTPTranStatus

---

```
USHORT CTOS_EthernetFTPTranStatus (DWORD* pdwFileDataLen);
```

**Description** Check the transmit status when downloading or uploading.

**Parameters** [ OUT] *pdwFileDataLen*  
The file data length which is already downloaded or uploaded.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_ETHERNET_FTP_BUSY	421Dh
	<a href="#">Ethernet error codes</a>	42xxh

## CTOS\_EthernetFTPCancel

---

```
USHORT CTOS_EthernetFTPCancel ( void );
```

**Description** Force FTP server to cancel downloading or uploading during transmit data.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Ethernet error codes</a>	42xxh

**Note** This command may cause FTP server unexpected failure.

## 2.14.USB Functions

- CTOS\_USBSelectMode
- CTOS\_USBOpen
- CTOS\_USBClose
- CTOS\_USBTxReady
- CTOS\_USBTxData
- CTOS\_USBRxReady
- CTOS\_USBRxData
- CTOS\_USBTxFlush
- CTOS\_USBRxFlush
- CTOS\_USBSetCDCMode
- CTOS\_USBSetSTDMode
- CTOS\_USBSetVidPid
- CTOS\_USBGetVidPid
- CTOS\_USBGetStatus
- CTOS\_USBHostOpen
- CTOS\_USBHostClose
- CTOS\_USBHostTxData
- CTOS\_USBHostRxData

### USB Error Codes

<b>Constants</b>	<b>Value</b>
d_USB_INVALID_PARA	1C01h
d_USB_NOT_OPEN	1C02h
d_USB_ALREADY_OPEN	1C03h
d_USB_FAILED	1C04h
d_USB_BUSY	1C05h
d_USB_NOT_READY	1C06h
d_USB_NOT_EXIST	1C07h
d_USB_TIMEOUT	1C08h
d_USB_INCORRECT_MODE	1C09h
d_USB_ID_CONFLICT	1C0Ah

## CTOS\_USBSelectMode

---

```
USHORT CTOS_USBSelectMode(BYTE bMode);
```

**Description**      Switch the USB module to operate as device mode or host mode.

**Parameters**      [ IN ]      bMode

- **d\_USB\_DEVICE\_MODE**

Switch the USB module to operate as an USB device.

- **d\_USB\_HOST\_MODE**

Switch the USB module to operate as an USB host.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">USB error codes</a>	1Cxxh

**Note**

The operation mode will be saved and still remains after the system reboot.

## CTOS\_USBOpen

---

```
USHORT CTOS_USBOpen (void);
```

**Description** Open USB port.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">USB error codes</a>	1Cxh

**Example** Please refer to example in appendix [B.19 USB](#)

## CTOS\_USBClose

---

```
USHORT CTOS_USBClose(void);
```

**Description** Close USB port.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">USB error codes</a>	1Cxxh

**Example** Please refer to example in appendix [B.19 USB](#)

## CTOS\_USBTxReady

---

```
USHORT CTOS_USBTxReady( void );
```

**Description** Check if the USB port is ready for sending next data.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">USB error codes</a>	1Cxh

**Example** Please refer to example in appendix [B.19 USB](#)

## CTOS\_USBTxData

---

```
USHORT CTOS_USBTxData (BYTE*baSBuf, USHORT usSLen);
```

**Description** Send data through the USB port.

**Parameters** [ IN ] *baSBuf*  
Data to send.

[ IN ] *usSLen*  
Length of data. The maximum value is 2048.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">USB error codes</a>	1Cxxh

**Example** Please refer to example in appendix [B.19 USB](#)

**Note** Before sending any data, programmer should call **CTOS\_USBTxReady()** to check whether the port is ready.

## CTOS\_USBRxReady

---

```
USHORT CTOS_USBRxReady(USHORT* pusRLen);
```

**Description** Get the number of data currently received from the USB port.

**Parameters** [ OUT ] *pusRLen*  
Length of data received.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">USB error codes</a>	1Cxh

**Example** Please refer to example in appendix [B.19 USB](#)

## CTOS\_USBRxData

---

`USHORT CTOS_USBRxData (BYTE*baRBuf, USHORT*pusRLen);`

**Description** Get the data received from the USB port.

**Parameters** [ OUT ] *baRBuf*  
Buffer to store the received data.

[ IN ] *pusRLen*  
Size of buffer. Maximum number of data to get.

[ OUT ] *pusRLen*  
Length of the data actually returned in the buffer.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">USB error codes</a>	1Cxh

**Example** Please refer to example in appendix [B.19 USB](#)

**Note** Programmer can call **CTOS\_USBRxReady()** to check the number of data currently received, and then call this function with corresponding size of buffer get the data.  
If the size input is smaller than the actual number of data received, then only the size number of data will be returned.  
If the size input is bigger than the actual number of data, then the actual number of data returned will be stored in pusLen.

## CTOS\_USBTxFlush

---

```
USHORT CTOS_USBTxFlush( void );
```

**Description** Flush the data to be sent to the USB port.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">USB error codes</a>	1Cxh

**Example** Please refer to example in appendix [B.19 USB](#)

**Note** This function is normally used to cancel the current transmission.

## CTOS\_USBRxFlush

---

USHORT CTOS\_USBRxFlush( void );

**Description** Flush the data already received from the USB port.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">USB error codes</a>	1Cxxh

**Example** Please refer to example in appendix [B.19 USB](#)

**Note** All the data received before calling this function will be discarded.

## CTOS\_USBSetCDCMode

---

```
USHORT CTOS_USBSetCDCMode( void );
```

**Description** Select USB to operate as a CDC (Communication Device Class) USB device.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">USB error codes</a>	1Cxh

**Note** The operation mode will be saved and still remains after the system reboot.

## CTOS\_USBSetSTDMode

---

USHORT CTOS\_USBSetSTDMode( void );

**Description** Select USB to operate as a standard USB device.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">USB error codes</a>	1Cxh

**Note** The operation mode will be saved and still remains after the system reboot.

## CTOS\_USBSetVidPid

---

USHORT CTOS\_USBSetVidPid(DWORD dwVidPid);

**Description**      Customize the USB Vendor ID and Product ID of the terminal.

**Parameters**      [ IN ]      *dwVidPid*  
                            Vendor ID and Product ID to set.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">USB error codes</a>	1Cxh

**Note**      Store the Vendor ID in the higher 2 bytes of dwVidPid, and Product ID in the lower 2 bytes of dwVidPid.

Example: dwVidPid = 0x0CA6A050, Vendor ID is 0x0CA6 and Product ID is 0xA050.

## CTOS\_USBGetVidPid

---

```
USHORT CTOS_USBGetVidPid(DWORD*dwVidPid);
```

**Description** Get the current USB Vendor ID and Product ID of the terminal.

**Parameters** [ OUT] *dwVidPid*  
Pointer to DWORD for storing the USB Vendor ID and Product ID.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">USB error codes</a>	1Cxh

**Note** Vendor ID will be stored in the higher 2 bytes of dwVidPid, and Product ID in the lower 2 bytes of dwVidPid.  
Example: dwVidPid = 0x0CA6A050, Vendor ID is 0x0CA6 and Product ID is 0xA050.

## CTOS\_USBGetStatus

---

USHORT CTOS\_USBGetStatus (DWORD\*pdwUSBStatus) ;

**Description** Get the current USB operation mode of the terminal.

**Parameters** [ OUT] *pdwUSBStatus*  
Pointer to DWORD for storing the current USB operation mode. Status is returned in the bitwise-OR of following bitmask.

- ***d\_MK\_USB\_STATUS\_CDCMODE***  
USB is operating in the CDC mode / standard mode.
- ***d\_MK\_USB\_STATUS\_HOSTMODE***  
USB is operating in host mode / device mode.

**Return Value**

<b><i>Constants</i></b>	<b><i>Value</i></b>
d_OK	0000h
<a href="#">USB error codes</a>	1Cxh

## CTOS\_USBHostOpen

---

```
USHORT CTOS_USBHostOpen(USHORT usVendorID, USHORT usProductID);
```

**Description** Open a channel to USB device with specific Vendor ID and Product ID.

**Parameters** [ IN ] *usVendorID*  
Vendor ID of the device connected to the terminal.

[ IN ] *usProductID*  
Product ID of the device connected to the terminal.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">USB error codes</a>	1Cxxh

## CTOS\_USBHostClose

---

```
USHORT CTOS_USBHostClose( void );
```

**Description** Close the channel to USB device.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">USB error codes</a>	1Cxh

## CTOS\_USBHostTxData

---

USHORT CTOS\_USBHostTxData(BYTE\*baSBuf, ULONG ulTxLen, ULONG ulMSec);

**Description** Send data to USB device.

**Parameters** [ IN ] *baSBuf*  
Data to send.

[ IN ] *ulTxLen*  
Length of data.

[ IN ] *ulMSec*  
Timeout value for the transmission.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">USB error codes</a>	1Cxh

**Note** For USB protocol, device might temporarily reject to receive the data sent from terminal(host), thus a timeout must be given to indicate how long should terminal try to send the data.

## CTOS\_USBHostRxData

---

USHORT CTOS\_USBHostRxData(BYTE\*baRBuf, ULONG\*ulRxLen, ULONG ulMSec);

**Description** Receive the data from USB device.

**Parameters**

[OUT]	<u><i>baRBuf</i></u>
	Buffer to store the data received from USB device.

[ IN ]	<u><i>ulRxLen</i></u>
	Size of buffer. Maximum number of data to receive.

[ OUT ]	<u><i>ulRxLen</i></u>
	Length of data received from USB device.

[ IN ]	<u><i>ulMSec</i></u>
	Timeout value for receiving the data from USB device.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">USB error codes</a>	1Cxxh

**Note** Calling this function will trigger the terminal to receive the data from USB device. Unlike other port, such as RS232, modem and Ethernet port, which always receive the data and store the data in internal buffer, USB host module will only start to receive the data from USB device when programmer calls this function.

## 2.15.Cradle USB Functions

- CTOS\_BaseUSBOpen
- CTOS\_BaseUSBClose
- CTOS\_BaseUSBTxReady
- CTOS\_BaseUSBTxData
- CTOS\_BaseUSBRxReady
- CTOS\_BaseUSBRxData
- CTOS\_BaseUSBTxFlush
- CTOS\_BaseUSBRxFlush
- CTOS\_BaseUSBSetCDCMode
- CTOS\_BaseUSBSetSTDMode
- CTOS\_BaseUSBSetVidPid
- CTOS\_BaseUSBGetVidPid
- CTOS\_BaseUSBGetStatus

### Cradle USB Error Codes

<b>Constants</b>	<b>Value</b>
d_CRADLEUSB_INVALID_PARA	1D01h
d_CRADLEUSB_NOT_OPEN	1D02h
d_CRADLEUSB_ALREADY_OPEN	1D03h
d_CRADLEUSB_FAILED	1D04h
d_CRADLEUSB_BUSY	1D05h
d_CRADLEUSB_NOT_READY	1D06h

## CTOS\_BaseUSBOpen

---

```
USHORT CTOS_BaseUSBOpen( void );
```

**Description** Open the base USB port.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Cradle USB error codes</a>	1Dxxh

**Note** Only the type terminal with base supports this function.

## CTOS\_BaseUSBClose

---

```
USHORT CTOS_BaseUSBClose( void );
```

**Description** Close the base USB port.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Cradle USB error codes</a>	1Dxxh

**Note** Only the type terminal with base supports this function.

## CTOS\_BaseUSBTxReady

---

```
USHORT CTOS_BaseUSBTxReady( void );
```

**Description** Check if the base USB port is ready for sending next data.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Cradle USB error codes</a>	1Dxxh

**Note** Only the type terminal with base supports this function.

## CTOS\_BaseUSBTxData

---

```
USHORT CTOS_BaseUSBTxData(BYTE*baSBuf, USHORT usSLen);
```

**Description** Send data through the base USB port.

**Parameters** [ IN ] *baSBuf*  
Data to send.

[ IN ] *usSLen*  
Length of data. The maximum value is 2048.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Cradle USB error codes</a>	1Dxxh

**Note** Before sending any data, programmer should call **CTOS\_BaseUSBTxReady()** to check whether the port is ready.  
Only the type terminal with base supports this function.

## CTOS\_BaseUSBRxReady

---

```
USHORT CTOS_BaseUSBRxReady (USHORT*pusRLen) ;
```

**Description** Get the number of data currently received from the base USB port.

**Parameters** [OUT] *pusRLen*  
Length of data received.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Cradle USB error codes</a>	1Dxxh

**Note** Only the type terminal with base supports this function.

## CTOS\_BaseUSBRxData

---

USHORT CTOS\_BaseUSBRxData(BYTE\*baRBuf, USHORT\*pusRLen);

**Description** Get the data received from the base USB port.

**Parameters**

[OUT] <i>baRBuf</i>	Buffer to store the received data.
---------------------	------------------------------------

[ IN ] <i>pusRLen</i>	Size of buffer. Maximum number of data to get.
-----------------------	--

[ OUT ] <i>pusRLen</i>	Length of the data actually returned in the buffer.
------------------------	---

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Cradle USB error codes</a>	1Dxxh

**Note**

Programmer can call **CTOS\_BaseUSBRxReady()** to check the number of data currently received, and then call this function with corresponding size of buffer get the data.

If the size input is smaller than the actual number of data received, then only the size number of data will be returned.

If the size input is bigger than the actual number of data, then the actual number of data returned will be stored in pusLen.

Only the type terminal with base supports this function.

## CTOS\_BaseUSBTxFlush

---

```
USHORT CTOS_BaseUSBTxFlush( void );
```

**Description** Flush the data to be sent to the base USB port.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Cradle USB error codes</a>	1Dxxh

**Note** This function is normally used to cancel the current transmission.

Only the type terminal with base supports this function.

## CTOS\_BaseUSBRxFlush

---

```
USHORT CTOS_BaseUSBRxFlush( void );
```

**Description** Flush the data already received from the base USB port.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Cradle USB error codes</a>	1Dxxh

**Note** All the data received before calling this function will be discarded.

Only the type terminal with base supports this function.

## CTOS\_BaseUSBSetCDCMode

---

USHORT CTOS\_BaseUSBSetCDCMode( void );

**Description** Select base USB to operate as a CDC (Communication Device Class) USB device.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Cradle USB error codes</a>	1Dxxh

**Note** The operation mode will be saved and still remains after the base reboot. Only the type terminal with base supports this function.

## CTOS\_BaseUSBSetSTDMode

---

```
USHORT CTOS_BaseUSBSetSTDMode( void );
```

**Description** Select base USB to operate as a standard USB device.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Cradle USB error codes</a>	1Dxxh

**Note** The operation mode will be saved and still remains after the base reboot. Only the type terminal with base supports this function.

## CTOS\_BaseUSBSetVidPid

---

USHORT CTOS\_BaseUSBSetVidPid (DWORD dwVidPid);

**Description**      Customize the USB Vendor ID and Product ID of the base.

**Parameters**      [ IN ]      *dwVidPid*  
                            Vendor ID and Product ID to set.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Cradle USB error codes</a>	1Dxxh

**Note**      Store the Vendor ID in the higher 2 bytes of dwVidPid, and Product ID in the lower 2 bytes of dwVidPid.

Example: dwVidPid = 0x0CA6A050, Vendor ID is 0x0CA6 and Product ID is 0xA050.

## CTOS\_BaseUSBGetVidPid

---

```
USHORT CTOS_BaseUSBGetVidPid(DWORD *pdwVidPid);
```

**Description** Get the current USB Vendor ID and Product ID of the base.

**Parameters** [ OUT ] *dwVidPid*  
Pointer to DWORD for storing the USB Vendor ID and Product ID.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Cradle USB error codes</a>	1Dxxh

**Note** Vendor ID will be stored in the higher 2 bytes of dwVidPid, and Product ID in the lower 2 bytes of dwVidPid.  
Example: dwVidPid = 0x0CA6A050, Vendor ID is 0x0CA6 and Product ID is 0xA050.

## CTOS\_BaseUSBGetStatus

---

USHORT CTOS\_BaseUSBGetStatus (DWORD\*pdwUSBStatus) ;

<b>Description</b>	Get the current USB operation mode of the base.						
<b>Parameters</b>	<p>[OUT] <u><i>pdwUSBStatus</i></u>            Pointer to DWORD for storing the current USB operation mode.            Status is returned in the bitwise-OR of following bitmask.</p> <ul style="list-style-type: none"> <li>• <b><i>d_MK_USB_STATUS_CDCMODE</i></b>            USB is operating in the CDC mode / standard mode.</li>   <li>• <b><i>d_MK_USB_STATUS_HOSTMODE</i></b>            USB is operating in host mode / device mode.</li> </ul>						
<b>Return Value</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;"><b><i>Constants</i></b></th> <th style="text-align: center;"><b><i>Value</i></b></th> </tr> </thead> <tbody> <tr> <td style="text-align: left;">d_OK</td> <td style="text-align: center;">0000h</td> </tr> <tr> <td style="text-align: left;"><a href="#">Cradle USB error codes</a></td> <td style="text-align: center;">1Dxxh</td> </tr> </tbody> </table>	<b><i>Constants</i></b>	<b><i>Value</i></b>	d_OK	0000h	<a href="#">Cradle USB error codes</a>	1Dxxh
<b><i>Constants</i></b>	<b><i>Value</i></b>						
d_OK	0000h						
<a href="#">Cradle USB error codes</a>	1Dxxh						

## 2.16.LCD Display Functions

### General

- CTOS\_LCDSelectMode
- CTOS\_LCDSelectModeEx
- CTOS\_LCDSetContrast
- CTOS\_LCDForeGndColor
- CTOS\_LCDCBackGndColor
- CTOS\_LCDDFontSelectMode
- CTOS\_LCDTTFSelect
- CTOS\_LCDTTFCheckLanguageSupport

### Graphic Mode

- CTOS\_LCDGClearCanvas
- CTOS\_LCDGPixel
- CTOS\_LCDGTextOut
- CTOS\_LCDGSetBox
- CTOS\_LCDGShowPic
- CTOS\_LCDGShowPicEx
- CTOS\_LCDGClearWindow
- CTOS\_LCDGMoveWindow
- CTOS\_LCDGGetWindowOffset
- CTOS\_LCDGShowBMPPic

### Text Mode

- CTOS\_LCDTClearDisplay
- CTOS\_LCDTGotoXY
- CTOS\_LCDTWhereX
- CTOS\_LCDTWhereY
- CTOS\_LCDTPrint
- CTOS\_LCDTPrintXY
- CTOS\_LCDTPutch
- CTOS\_LCDTPutchXY
- CTOS\_LCDTClear2EOL

- CTOS\_LCDTSetReverse
- CTOS\_LCDTSelectFontSize
- CTOS\_LCDTSetASCIIVerticalOffset
- CTOS\_LCDTSetASCIIHorizontalOffset

#### **User Defined Mode**

- CTOS\_LCDUIInit
- CTOS\_LCDClearDisplay
- CTOS\_LCDUPrintXY
- CTOS\_LCDUPrint

#### **LCD Display Error Codes**

<b><i>Constants</i></b>	<b><i>Value</i></b>
d_LCD_OUT_OF_RANGE	1A01h
d_LCD_MODE_NOT_SUPPORT	1A02h
d_LCD_LANGUAGE_NOT_SUPPORT	1A03h
d_LCD_FONT_SIZE_NOT_SUPPORT	1A04h
d_LCD_PARAMETER_NOT_SUPPORT	1A05h
d_LCD_WINDOW_REACH_BOTTOM	1A06h
d_LCD_PIC_FORMAT_NOT_SUPPORT	1A07h
d_LCD_PIC_OPEN_FAILED	1A08h

## CTOS\_LCDSelectMode

---

USHORT CTOS\_LCDSelectMode (BYTE bMode) ;

**Description** Set the LCD display mode.

**Parameters** [ IN ] bMode  
 Graphic mode or Text mode.  
 • **d\_LCD\_GRAPHIC\_MODE**  
 Set the LCD into graphics mode  
 • **d\_LCD\_TEXT\_MODE**  
 Set the LCD into text mode

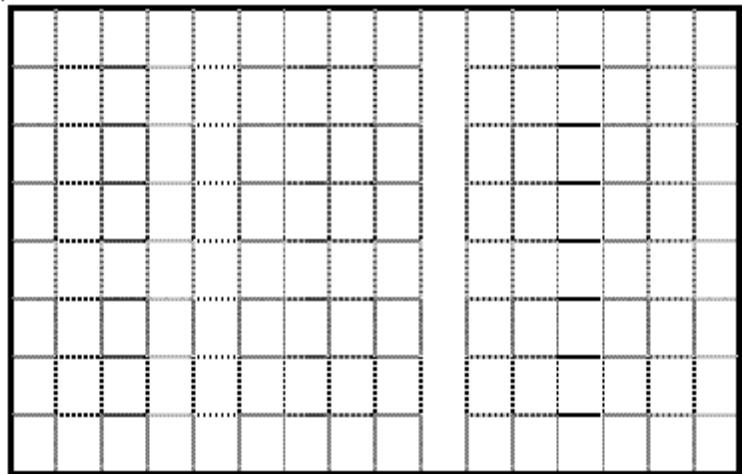
Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix[B.10 LCD Display](#)

**Note** The default mode is **d\_LCD\_TEXT\_MODE**.  
 For **Graphic** mode, programmer should consider there is a fix-sized 1x1 pixel-base canvas, and draw anything desire onto the canvas. The coordinate of most left-up pixel is (0,0). A window of resolution of LCD hardware is the display area of the canvas, programmer can move the window to display the different part of canvas on the LCD.  
 For **Text** mode, cursor is introduced and the font size is considered to decide the size of x and y coordinate. The most left-up cursor position is (1,1). It is like a one-way text-pad, so that an programmer can print text of selected font size on the screen, move the cursor to the desire position. If a change line character "\n" is encountered, and the cursor is currently at the maximum y coordinate, an automatic scroll down of the screen is performed. Care must be taken that it is not possible to scroll up the screen.

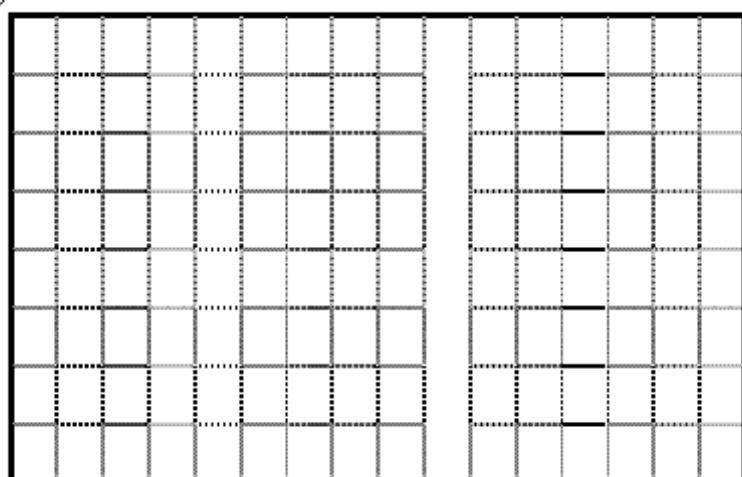
**Graphic Mode:**

(0,0)



**Text Mode:**

(1,1)



## CTOS\_LCDSelectModeEx

---

USHORT CTOS\_LCDSelectModeEx(BYTE bMode, BOOL fClear);

**Description** Set the LCD display mode.

**Parameters** [ IN ] bMode

Graphic mode or Text mode.

- **d\_LCD\_GRAPHIC\_MODE**

Set the LCD into graphics mode

- **d\_LCD\_TEXT\_MODE**

Set the LCD into text mode

[ IN ] fClear

Whether to clear the display.

- **d\_TRUE**

Clear the display.

- **d\_FALSE**

Don't clear the display.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

**Note**

The default mode is **d\_LCD\_TEXT\_MODE**.

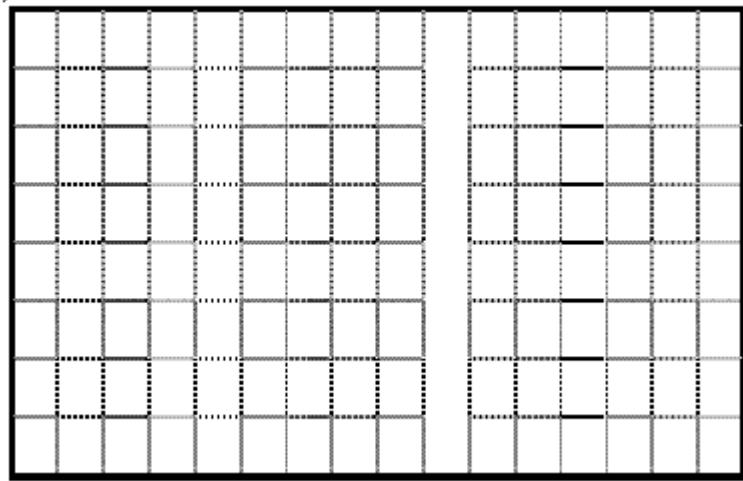
For **Graphic** mode, programmer should consider there is a fix-sized 1x1 pixel-base canvas, and draw anything desire onto the canvas. The coordinate of most left-up pixel is (0,0). A window of resolution of LCD hardware is the display area of the canvas, programmer can move the window to display the different part of canvas on the LCD.

For **Text** mode, cursor is introduced and the font size is considered to decide the size of x and y coordinate. The most left-up cursor position

is (1,1). It is like a one-way text-pad, so that a programmer can print text of selected font size on the screen, move the cursor to the desire position. If a change line character "\n" is encountered, and the cursor is currently at the maximum y coordinate, an automatic scroll down of the screen is performed. Care must be taken that it is not possible to scroll up the screen.

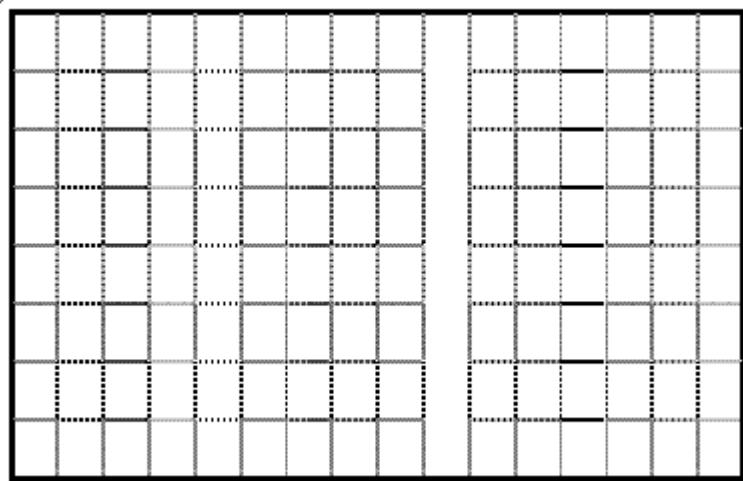
**Graphic Mode:**

(0,0)



**Text Mode:**

(1,1)



## CTOS\_LCDSetContrast

---

```
USHORT CTOS_LCDSetContrast (BYTE bValue);
```

**Description** Set the contrast of LCD.

**Parameters** [ IN ] *bValue*  
The contrast value from 0x00 to 0xFF of the LCD.

Return Value	<b>Constants</b>	<b>Value</b>
d_OK		0000h
<a href="#">LCD Display error codes</a>		1Axxh

**Example** Please refer to example in appendix[B.17 System Mode](#)

**Note** This function is only valid for Vega5000(S) and Vega3000(P) with a 2.4" display. For 3.5" please use CTOS\_BackLightSetBrightness() instead.

## CTOS\_LCDForeGndColor

---

`USHORT CTOS_LCDForeGndColor(ULONG ulColor);`

**Description** Set LCD foreground color when using functions such as `CTOS_LCDTPrintXY()`, etc.

**Parameters** [ IN ] *ulColor*  
 Color value. Encoding 00 BB GG RR, where  
 - RR is red color value,  
 - GG is green color value,  
 - BB is blue color value.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Note** If `CTOS_LCDForeGndColor(0x00B16D38)` is called, then [EXAMPLE](#) will be shown inside the LCD.

## CTOS\_LCDBackGndColor

---

```
USHORT CTOS_LCDBackGndColor(ULONG ulColor);
```

**Description** Set LCD background color when using functions such as CTOS\_LCDTPrintXY(), etc.

**Parameters** [ IN ] *ulColor*  
Color value. Encoding 00 BB GG RR, where  
- RR is red color value,  
- GG is green color value,  
- BB is blue color value.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Note** If CTOS\_LCDBackGndColor(0x00B16D38) is called, then **EXAMPLE** will be shown inside the LCD.

## CTOS\_LCDDFontSelectMode

---

USHORT CTOS\_LCDDFontSelectMode (BYTE bMode) ;

**Description** Set LCD display font mode.

**Parameters** [ IN ] bMode

Font mode.

- **d\_FONT\_FNT\_MODE:** use FNT fonts (legacy fonts with limited sizes)
- **d\_FONT\_TTF\_MODE:** use TTF fonts (vectorial fonts that can set to any size)

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

**Notes**

TTF fonts are recommended since they are much more flexible than FNT fonts (which are maintained for backwards compatibility only with Vega5000).

TTF fonts can be downloaded from the Internet but pay attention to the terms of use of each one.

OTF fonts can be used too if you convert them first to TTF  
(<http://www.freefontconverter.com>)

FNT fonts can be used in Vega5000, Vega5000S and Vega3000. TTF can be used in Vega5000S and Vega3000 only.

## CTOS\_LCDTTFSelect

---

USHORT CTOS\_LCDTTFSelect (BYTE \*baFilename, BYTE bIndex);

**Description** Select the TrueType font to be used in LCD display.

**Parameters** [ IN ] *baFilename*

Filename of the TrueType font.

[ IN ] *bIndex*

Index of font style. Default value is 0.

Some TTF fonts include several styles per each .ttf file and organize them by different index. If so, *bIndex* parameter can be used for selecting the TTF style..

For example, Times New Roman font has font style of:

0 – Regular

1 – Italic

2 – Bold

3 – Bold italic

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh
<a href="#">Font error codes</a>	1Exxh

## CTOS\_LCDTTFCheckLanguageSupport

---

```
USHORT CTOS_LCDTTFCheckLanguageSupport (ULONG ulLanguage, BOOL
*fSupported);
```

**Description** Check if specific language character set is supported in selected TrueType font file.

**Parameters**

[ IN ]	<u><i>ulLanguage</i></u>
	Language Id. Please refer to Appendix B.

[ OUT ]	<u><i>fSupported</i></u>
	<ul style="list-style-type: none"> <li>• <b><i>d_TRUE</i></b> Specific language character set supported.</li> </ul>

[ OUT ]	<u><i>fSupported</i></u>
	<ul style="list-style-type: none"> <li>• <b><i>d_FALSE</i></b> Specific language character set not supported.</li> </ul>

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

## CTOS\_LCDClearCanvas

---

```
USHORT CTOS_LCDClearCanvas( void );
```

**Description** Clear Canvas.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDGPixel

---

```
USHORT CTOS_LCDGPixel(USHORT usX, USHORT usY, BOOL boPat);
```

**Description** Draw a pixel on the canvas.

**Parameters** [ IN ] *usX*

X coordinate, the range is from 0 to  
d\_LCD\_CANVAS\_X\_SIZE-1.

[ IN ] *usY*

Y coordinate, the range is from 0 to  
d\_LCD\_CANVAS\_Y\_SIZE-1.

[ IN ] *boPat*

- *d\_TRUE*

Set the pixel.

- *d\_FALSE*

Clear the pixel.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

**Example**

Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDGTextOut

---

```
USHORT CTOS_LCDGTextOut(USHORT usX, USHORT usY, UCHAR*pusBuf, USHORT
usFontSize, BOOL boReverse);
```

**Description** Draw a text line on the canvas.

**Parameters**

- [ IN ] *usX*  
X coordinate, the range is from 0 to  
d\_LCD\_CANVAS\_X\_SIZE-1.

- [ IN ] *usY*  
Y coordinate, the range is from 0 to  
d\_LCD\_CANVAS\_Y\_SIZE-1.

- [ IN ] *pusBuf*  
String to draw on the canvas.

- [ IN ] *usFontSize*  
Font size of the string.

- [ IN ] *boReverse*  
Whether to draw a reversed string or normal string
  - ***d\_TRUE***  
Draw a reversed string.

- ***d\_FALSE***  
Draw a normal string.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

**Example**

Please refer to example in appendix [B.10 LCD Display](#)

## **CTOS\_LCDGSetBox**

---

```
USHORT CTOS_LCDGSetBox(USHORT usX, USHORT usY, USHORT usXSize,
USHORT usYSize, BYTE bFill);
```

**Description** Fill a rectangle box on the canvas.

- Parameters**
- [ IN ] *usX*  
X coordinate, the range is from 0 to  
d\_LCD\_CANVAS\_X\_SIZE-1.
  - [ IN ] *usY*  
Y coordinate, the range is from 0 to  
d\_LCD\_CANVAS\_Y\_SIZE-1.
  - [ IN ] *usXSize*  
Width of the box.
  - [ IN ] *usYSize*  
Height of the box.
  - [ IN ] *bFill*  
Style of filling the box.  
= 1 : Set all pixel in the box  
= 0 : Clear all pixel in the box  
= XOR : Reverse all pixel in the box

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

**Example**

Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDGShowPic

---

```
USHORT CTOS_LCDGShowPic(USHORT usX, USHORT usY, BYTE*baPat,
ULONGulPatLen, USHORT usXSize);
```

**Description** Draw a pattern on the canvas.

**Parameters**

- [ IN ] *usX*  
X coordinate, the range is from 0 to  
d\_LCD\_CANVAS\_X\_SIZE-1.

- [ IN ] *usY*  
Y coordinate, the range is from 0 to  
d\_LCD\_CANVAS\_Y\_SIZE-1.

- [ IN ] *baPat*  
Pattern to draw on canvas.

- [ IN ] *ulPatLen*  
Length of the baPat in byte. (1x8 pixel).

- [ IN ] *usXSize*  
Width of the pattern.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDGShowPicEx

---

```
USHORT CTOS_LCDGShowPicEx ( UCHAR bMode, USHORT usX, USHORT usY,
BYTE*baPat, ULONG ulPatLen, USHORT usXSize);
```

**Description** Draw a pattern on the canvas.

**Parameters** [ IN ] bMode

Display mode.

- d\_LCD\_SHOWPIC\_MONO
- d\_LCD\_SHOWPIC\_RGB

[ IN ] usX

X coordinate, the range is from 0 to  
d\_LCD\_CANVAS\_X\_SIZE-1.

[ IN ] usY

Y coordinate, the range is from 0 to  
d\_LCD\_CANVAS\_Y\_SIZE-1.

[ IN ] baPat

Pattern to draw on canvas.

[ IN ] ulPatLen

Length of the baPat in byte. (1x8 pixel).

[ IN ] usXSize

Width of the pattern.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

## CTOS\_LCDGShowBMPPic

---

```
USHORT CTOS_LCDGShowBMPPic (USHORT usX, USHORT usY, BYTE
*baFilename);
```

**Description** Display BMP file on the screen

**Parameters**

[ IN ]	<u>usX</u>
	X coordinate on the canvas or screen to start drawing picture

[ IN ]	<u>usY</u>
	Y coordinate on the canvas or screen to start drawing picture

[ IN ]	<u>baFilename</u>
	Path and filename of the BMP to be displayed.
	Ex: "./myfile/myBMP.bmp"

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

## CTOS\_LCDClearWindow

---

```
USHORT CTOS_LCDClearWindow( void );
```

**Description** Clear all the pixels in the display window on the canvas.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDGMoveWindow

---

```
USHORT CTOS_LCDGMoveWindow(USHORT usOffset);
```

**Description** Move the display window to display desire position of canvas.

**Parameters** [ IN ] *usOffset*  
Offset of the window to locate. Range from 0~  
(d\_LCD\_CANVAS\_Y\_SIZE - d\_LCD\_WINDOW\_Y\_SIZE).

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

**Note** If the windows specified by usOffset is over the bottom the canvas (ie. Return code is d\_LCD\_WINDOW\_REACH\_BOTTOM), the window will be located at bottom of canvas . ie. window offset = d\_LCD\_CANVAS\_Y\_SIZE-d\_LCD\_WINDOW\_Y\_SIZE.

## **CTOS\_LCDCGetWindowOffset**

---

```
USHORT CTOS_LCDCGetWindowOffset( void );
```

**Description**      Get the current position of window.

**Parameters**      None

**Return Value**      Offset of the current window.

**Example**      Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDTClearDisplay

---

```
USHORT CTOS_LCDTClearDisplay( void );
```

**Description** Clear LCD display and reset the cursor.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDTGotoXY

---

```
USHORT CTOS_LCDTGotoXY(USHORT usX, USHORT usY);
```

**Description** Locate the cursor to specific coordinate.

**Parameters** [ IN ] usX  
Horizontal cursor position.

[ IN ] usY  
Vertical cursor position.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDTWhereX

---

USHORT CTOS\_LCDTWhereX( void );

**Description**      Return the current X cursor position.

**Parameters**      None

**Return Value**      Horizontal cursor position

**Example**      Please refer to example in appendix [B.10 LCD Display](#)

## **CTOS\_LCDTWhereY**

---

USHORT CTOS\_LCDTWhereY( void );

**Description**      Return the current Y cursor position.

**Parameters**      None

**Return Value**      Vertical cursor position.

**Example**      Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDTPrint

---

```
USHORT CTOS_LCDTPrint (UCHAR*sBuf) ;
```

**Description** Display a string at current cursor position.

**Parameters** [ IN ] sBuf  
String to display.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

**Note** Escape codes:  
"\n": line feed.  
"\fc": clear window, cursor goto (1,1).  
"\fn": normal mode.  
"\fu": under line mode.  
"\fr": reverse mode.

## CTOS\_LCDTPrintXY

---

`USHORT CTOS_LCDTPrintXY(USHORT usX, USHORT usY, UCHAR* pbBuf);`

**Description** Display a string at specific cursor position.

**Parameters** [ IN ] *usX*  
Horizontal cursor position.

[ IN ] *usY*  
Vertical cursor position.

[ IN ] *pbBuf*  
String to display.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

**Note** Escape codes:  
 "\n": line feed.  
 "\fc": clear window, cursor goto (1,1).  
 "\fn": normal mode.  
 "\fu": under line mode.  
 "\fr": reverse mode.

## CTOS\_LCDTPutCh

---

```
USHORT CTOS_LCDTPutCh (UCHAR ch) ;
```

**Description** Display a character at current cursor position.

**Parameters** [ IN ] *ch*  
Character to display.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDTPutChXY

---

```
USHORT CTOS_LCDTPutChXY (USHORT usX, USHORT usY, UCHAR bChar);
```

**Description** Display a character at specific cursor position.

**Parameters** [ IN ] usX  
Horizontal cursor position.

[ IN ] usY  
Vertical cursor position.

[ IN ] bChar  
Character to display.

Return Value	Constants	Value
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDTClear2EOL

---

```
USHORT CTOS_LCDTClear2EOL( void );
```

**Description** Clear all characters displayed from the cursor position to the end of line.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

## CTOS\_LCDTSetReverse

---

```
USHORT CTOS_LCDTSetReverse(BOOL boReverse);
```

**Description** Set to display in reverse mode or normal mode.

- Parameters** [ IN ] *boReverse*
- ***d\_TRUE***  
Set to display in reverse mode.
  - ***d\_FALSE***  
Set to display in normal mode.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

**Note** All subsequent calling to CTOS\_LCDTxxx() functions will display the content according to the mode set.

## CTOS\_LCDTSelectFontSize

---

```
USHORT CTOS_LCDTSelectFontSize(USHORT usFontSize);
```

**Description** Set the font size.

- Parameters** [ IN ] *usFontSize*
- ***d\_FONT\_8x8***  
Select 8x8 font size
  - ***d\_FONT\_8x16***  
Select 8x16 font size
  - ***d\_FONT\_12x24***  
Select 12x24 font size
  - ***d\_FONT\_9x9***  
Select 9x9 font size
  - ***d\_FONT\_9x18***  
Select 9x18 font size
  - ***d\_FONT\_16x16***  
Select 16x16 font size
  - ***d\_FONT\_16x30***  
Select 16x30 font size
  - ***d\_FONT\_24x24***  
Select 24x24 font size

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Example** Please refer to example in appendix [B.10 LCD Display](#)

**Notes** ForFNT fonts, this function affects to ASCII characters which will be displayed according to the font size selected. For TTF fonts however, this function affects to both ASCII and non-ASCII characters.

If user's custom FNT fonts are loaded, *usFontSize* should indicate the proper size of the new font. E.g: for a 10x14 font, *usFontSize* shall be 0xA0E

TTF fonts can be set at any size. E.g: to set an "Arial" font to 20x34 size, set *usFontSize* to 0x1422.

FNT fonts can be used in Vega5000, Vega5000S and Vega3000. TTF can be used in Vega5000S and Vega3000 only.

## CTOS\_LCDTSetASCIIVerticalOffset

---

```
USHORT CTOS_LCDTSetASCIIVerticalOffset(BOOL fVDirection, BYTE bVOffset);
```

**Description** Set the vertical offset for shifting the ASCII font displayed on LCD.

**Parameters** [ IN ] *fVDirection*  
Vertical direction to shift.

- *d\_LCD\_SHIFTUP*

Shift up.

- *d\_LCD\_SHIFTDOWN*

Shift down.

[ IN ] *bVOffset*  
Vertical offset to shift in pixel.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Note** Use this function to adjust the vertical position of ASCII font displayed on LCD.

## CTOS\_LCDTSetASCIIHorizontalOffset

---

```
USHORT CTOS_LCDTSetASCIIHorizontalOffset(BOOL fHDirection, BYTE bOffset);
```

**Description** Set the horizontal offset for shifting the ASCII font displayed on LCD.

**Parameters** [ IN ] *fHDirection*  
Horizontal direction to shift.

- *d\_LCD\_SHIFTLEFT*

Shift left.

- *d\_LCD\_SHIFTRIGHT*

Shift right.

[ IN ] *bOffset*  
Horizontal offset to shift in pixel.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Note** Use this function to adjust the horizontal position of ASCII font displayed on LCD.

## CTOS\_LCDUInit

---

```
USHORT CTOS_LCDUInit(BYTE bCXSize, BYTE bCYSize, BYTE bTWXSize, BYTE
bTWYSIZE, BYTE *baPattern);
```

**Description** Initialize the User Font System.

<b>Parameters</b>	[ IN ] <u><i>bCXSize</i></u> Horizontal size of font in pixel.
	[ IN ] <u><i>bCYSize</i></u> Vertical size of font in pixel.
	[ IN ] <u><i>bTWXSize</i></u> Number of character can be displayed on X coordinate.
	[ IN ] <u><i>bTWYSIZE</i></u> Number of character can be displayed on Y coordinate.
	[ IN ] <u><i>baPattern</i></u> User-defined font pattern. If baPattern is NULL, the build-in 6x8 ASCII font pattern will be used, and bCXSize/bCYSize/bTWXSize/bTWYSIZE input will be discarded.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

**Note** The User Font System is designed for using the font with user-defined size and pattern in text mode (please refer to CTOS\_LCDSelectMode for description of text mode). User can specify the horizontal/vertical size of font, specify the limitation of number of character can be displayed on the LCD, and input an user-defined font pattern table to be used in the following CTOS\_LCDUxxx() functions.

## CTOS\_LCDUClearDisplay

---

```
USHORT CTOS_LCDUClearDisplay( void );
```

**Description** Clear LCD display and reset the cursor.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">LCD Display error codes</a>	1Axxh

## CTOS\_LCDUPrintXY

---

```
USHORT CTOS_LCDUPrintXY(USHORT usX, USHORT usY, BYTE *pbBuf);
```

**Description** Display a string at specific cursor position with user-defined font.

**Parameters** [ IN ] usX  
Horizontal cursor position.

[ IN ] usY  
Vertical cursor position.

[ IN ] pbBuf  
String to print.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

## CTOS\_LCDUPrint

---

```
USHORT CTOS_LCDUPrint(BYTE *pbBuf);
```

**Description**      Display a string at current cursor position with user-defined font.

**Parameters**      [ IN ]      *pbBuf*  
                            String to print.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">LCD Display error codes</a>	1Axxh

## **CTOS\_ LCDAttributeGet**

---

```
USHORT CTOS_ LCDAttributeGet(ULONG *pulResolution, BYTE *pbColor,
BYTE *pbTouch);
```

**Description**      Retrieve terminal LCD attribution.

**Parameters**      [ OUT ] pulResolution : LCD resolution parameter

5 resolution type supported:

d\_RESOLUTION\_480x320

d\_RESOLUTION\_320x480

d\_RESOLUTION\_320x240

d\_RESOLUTION\_128x64

d\_RESOLUTION\_128x32

EX: d\_RESOLUTION\_480x320 = ((480 << 16) | 320)

[ OUT ] pbColor :    LCD color type parameter

3 color type supported:

d\_COLOR\_MONO = 1

d\_COLOR\_262K = 2

d\_COLOR\_16M = 3

[ OUT ] pbTouch :    LCD touch support parameter

3 touch panel type supported:

d\_TOUCH\_NONE = 0

d\_TOUCH\_RESISTOR = 1

d\_TOUCH\_CAPACITOR\_1P = 2

**Return Value**

	<b>Constants</b>	<b>Value</b>
	d_OK	0000h

## 2.17.Keyboard Functions

- CTOS\_KBDGet
- CTOS\_KBDHit
- CTOS\_KBDSetSound
- CTOS\_KBDSetFrequency
- CTOS\_KBDInKey
- CTOS\_KBDInKeyCheck
- CTOS\_KBDSetResetEnable
- CTOS\_KBDBufCheck
- CTOS\_KBDBufGet
- CTOS\_KBDBufPut
- CTOS\_KBDBufFlush
- CTOS\_KBDScan

### Keyboard Error Codes

<b><i>Constants</i></b>	<b><i>Value</i></b>
d_KBD_INVALID_KEY	1301h

## CTOS\_KBDGet

---

```
USHORT CTOS_KBDGet (BYTE*pbKey) ;
```

**Description** Wait for a key is pressed, and return the key or the value of the key pressed.

**Parameters** [OUT] *pbKey*  
Pointer to BYTE to store the key.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_KBD_INVALID_KEY	1301h

**Example** Please refer to example in appendix[B.7 Keyboard](#)

**Note** This function checks whether there is any key in the internal key buffer. If there are several keys, it will return with the last key and clear the internal keybuffer. If there is no key in the internal key buffer, it will wait for a key and will not return until a key is pressed.

## CTOS\_KBDHit

---

```
USHORT CTOS_KBDHit (BYTE*pbKey) ;
```

**Description** Detect whether a key is pressed, and return the key of pressed key.

**Parameters** [OUT] *pbKey*  
Pointer to BYTE to store the key.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_KBD_INVALID_KEY	1301h

**Example** Please refer to example in appendix [B.7 Keyboard](#)

**Note** This function checks whether there is any key in the internal key buffer, if there is already keys, it will return with the last key and clear the internal keybuffer. If no key is pressed it returns d\_OK and sets \*pbKey to value 255 (d\_KBD\_INVALID).

## CTOS\_KBDSetSound

---

```
USHORT CTOS_KBDSetSound(BYTE bOnOff);
```

**Description** Turn on/off the keyboard sound.

**Parameters** [ IN ] bOnOff

- **d\_ON**

Turn the keyboard sound on.

- **d\_OFF**

Turn the keyboard sound off.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_KBD_INVALID_KEY	1301h

**Example**

Please refer to example in appendix [B.7 Keyboard](#)

## CTOS\_KBDSetFrequency

---

```
USHORT CTOS_KBDSetFrequency(USHORT usFreq, USHORT usDuration);
```

**Description** Set the frequency and duration for keyboard sound.

**Parameters** [ IN ] *usFreq*  
Frequency of keyboard sound in Hertz.

[ IN ] *usDuration*  
Duration of keyboard sound in 10 milliseconds.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_KBD_INVALID_KEY	1301h

**Example** Please refer to example in appendix [B.7 Keyboard](#)

## CTOS\_KBDInKey

---

```
USHORT CTOS_KBDInKey(BOOL*fKeyIn);
```

**Description** Detect if any key is pressed.

**Parameters** [OUT] *fKeyIn*  
If any key is pressed.

- *d\_TRUE*  
Key pressed is detected.

- *d\_FALSE*  
No key pressed is detected.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_KBD_INVALID_KEY	1301h

**Example** Please refer to example in appendix [B.7 Keyboard](#)

## CTOS\_KBDInKeyCheck

---

```
USHORT CTOS_KBDInKeyCheck(BYTE *pbKey);
```

**Description** Detect if any key is pressed and return the key of pressed key. The key will still remain in the key buffer.

**Parameters** [OUT] *pbKey*  
Pointer to BYTE to store the key.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_KBD_INVALID_KEY	1301h

**Note** For **CTOS\_KBDHit()**, it will get the key from the key buffer, but **CTOS\_KBDInkey()** will keep the key in the key buffer.

## CTOS\_KBDSetResetEnable

---

```
USHORT CTOS_KBDSetResetEnable(BOOL boIsEnable);
```

**Description** Enable/Disable the reset ability of key "F1".

**Parameters** [ IN ] *bolsEnable*  
Enable/Disable the reset ability of key "F1".

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_KBD_INVALID_KEY	1301h

## CTOS\_KDBufCheck

---

USHORT CTOS\_KDBufCheck(BOOL \*fKeyPressed) ;

**Description** Check if there any key in the key buffer.

**Parameters** [OUT] fKeyPressed

- **d\_TRUE**

At least one key in the buffer.

- **d\_FALSE**

No key is in the buffer.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_KBD_INVALID_KEY	1301h

## CTOS\_KDBufGet

---

```
USHORT CTOS_KDBufGet (UCHAR *bKey) ;
```

**Description** Get one key from the key buffer.

**Parameters** [ OUT] bKey  
Pointer to BYTE to store the key.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_KBD_INVALID_KEY	1301h

## CTOS\_KDBufPut

---

```
USHORT CTOS_KDBufPut (UCHAR bKey) ;
```

**Description** Put a key into the key buffer.

**Parameters** [ IN ] bKey  
Key to put into the key buffer.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_KBD_INVALID_KEY	1301h

## CTOS\_KDBBufFlush

---

```
USHORT CTOS_KDBBufFlush(void);
```

**Description** Clear the key buffer.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_KBD_INVALID_KEY	1301h

## CTOS\_KBDScan

---

```
USHORT CTOS_KBDScan(BYTE *pbKey);
```

**Description** Directly perform a scan to detect whether any key is currently pressed.

**Parameters** [OUT] *pbKey*  
Pointer to BYTE to store the key currently pressed.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_KBD_INVALID_KEY	1301h

**Note** This function directly performs a scan and return the key currently pressed immediately.

## 2.18. Printer Functions

- CTOS\_PrinterPutString
- CTOS\_PrinterLogo
- CTOS\_PrinterBMPPic
- CTOS\_PrinterFline
- CTOS\_PrinterStatus
- CTOS\_PrinterSetWorkTime
- CTOS\_PrinterSetHeatLevel
- CTOS\_PrinterFontSelectMode
- CTOS\_PrinterTTFSelect
- CTOS\_PrinterTTFCheckLanguageSupport
- CTOS\_PrinterCodaBarBarcode
- CTOS\_PrinterCode39Barcode
- CTOS\_PrinterEAN13Barcode
- CTOS\_PrinterCode128Barcode
- CTOS\_PrinterInterleaved2of5Barcode

### Printer Error Codes

<b>Constants</b>	<b>Value</b>
d_PRINTER_HEAD_OVERHEAT	1602h
d_PRINTER_PAPER_OUT	1603h
d_PRINTER_BARCODE_GENERATE_ERR	1604h
d_PRINTER_BARCODE_CONTENT_ERR	1605h
d_PRINTER_BARCODE_CONTENT_LEN_ERR	1606h
d_PRINTER_BARCODE_OUTSIDE_PAPER	1607h
d_PRINTER_PARA_ERR	1608h
d_PRINTER_PIC_FORMAT_NOT_SUPPORT	1609h
d_PRINTER_PIC_OPEN_FAILED	160Ah
d_PRINTER_PIC_OVER_SIZE	160Bh

## CTOS\_PrinterPutString

---

```
USHORT CTOS_PrinterPutString (UCHAR*baBuf) ;
```

**Description** Print out a string.

**Parameters** [ IN ] *baBuf*  
Buffer storing the string to print.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Printer error codes</a>	16xxh

**Example** Please refer to example in appendix[B.14 Printer](#)

**Note** Escape codes:  
"\n": line feed.  
"\fn": normal mode.  
"\fu": under line mode.  
"\fr": reverse mode.  
"\ft1": print type1 font.  
"\ft2": print type2 font.  
"\ft3": print type3 font.

## CTOS\_PrinterLogo

---

```
USHORT CTOS_PrinterLogo(BYTE*baLogo, USHORT usXstart, USHORT usXsize,
USHORT usY8Size);
```

**Description** Print out a pattern.

**Parameters** [ IN ] *baLogo*  
Buffer storing the pattern to print.

[ IN ] *usXstart*  
Horizontal position to start to print. Range from 0~383.

[ IN ] *usXsize*  
Horizontal size of the pattern in dot.

[ IN ] *usY8Size*  
Vertical size of the pattern in byte.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Printer error codes</a>	16xxh

**Example** Please refer to example in appendix [B.14 Printer](#)

## CTOS\_PrinterBMPPic

---

```
USHORT CTOS_PrinterBMPPic(USHORT usX, BYTE *baFilename);
```

**Description** Print out a BMP file. It has to be black and white.(indexed, 1 bit)

**Parameters** [ IN ] *usX*  
X coordinate on paper to start printing

[ IN ] *baFilename*  
Path and filename of the BMP to be printed. Ex:  
“./myfile/myBMP.bmp”

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Printer error codes</a>	16xxh

## CTOS\_PrinterFline

---

```
USHORT CTOS_PrinterFline(USHORT usLines);
```

**Description** Roll the paper of printer without printing.

**Parameters** [ IN ] *usLines*  
Number of dot line to roll.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Printer error codes</a>	16xxh

**Example** Please refer to example in appendix [B.14 Printer](#)

## CTOS\_PrinterStatus

---

```
USHORT CTOS_PrinterStatus( void );
```

**Description** Get the current status of printer.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_PRINTER_HEAD_OVERHEAT	1602h
d_PRINTER_PAPER_OUT	1603h

**Example** Please refer to example in appendix [B.14 Printer](#)

## CTOS\_PrinterSetWorkTime

---

USHORT CTOS\_PrinterSetWorkTime(USHORT usWorkTime, USHORT usCoolTime);

**Description** Set the working time and cool down time for continuous printing. To avoid the overheating of printer motor, after continuous printing for a period, system has to stop printing for printer motor to cool down.

**Parameters**

[ IN ]	<u><i>usWorkTime</i></u>
	Time for continuous printing in 10 milliseconds. Default value is 4000, ie. 40 seconds.

[ IN ]	<u><i>usCoolTime</i></u>
	Time for cool down in 10 milliseconds, after continuous printing for usWorkTime time. Default value is 6000, ie. 60 seconds

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Printer error codes</a>	16xxh

**Example** Please refer to example in appendix [B.14 Printer](#)

**Note** Default work time 40 seconds and cool down time 60 seconds means after continuous printing for 40 seconds, the system will automatically stop printing for 60 seconds, and then resume the printing after the printer motor is cool down.

## CTOS\_PrinterSetHeatLevel

---

`USHORT CTOS_PrinterSetHeatLevel (UCHAR bHeatLevel);`

**Description** Set the heat level for printing. The higher the level, the darker the printing quality.

**Parameters** [ IN ] *bHeatLevel*  
 Heat level for printing. The value ranges from 0 to 6. The default value is 2.  
 = 0 : Ultra light.  
 = 1 : Very light.  
 = 2 : Light.  
 = 3 : Medium.  
 = 4 : Dark.  
 = 5 : Very dark.  
 = 6 : Ultra dark.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Printer error codes</a>	16xxh

**Example** Please refer to example in appendix [B.14 Printer](#)

**Note** The higher the level, the higher power consumption (important if the terminal is a portable type)

## CTOS\_PrinterFontSelectMode

---

```
USHORT CTOS_PrinterFontSelectMode (BYTE bMode);
```

**Description** Set printer font mode.

**Parameters** [ IN ] bMode

Font mode.

- d\_FONT\_FNT\_MODE
- d\_FONT\_TTF\_MODE

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Printer error codes</a>	16xxh

## CTOS\_PrinterTTFSelect

---

```
USHORT CTOS_PrinterTTFSelect (BYTE *baFilename, BYTE bIndex);
```

**Description** Select TrueType font to be used when printing.

**Parameters** [ IN ] *baFilename*

Filename of the TrueType font.

[ IN ] *bIndex*

Index of font style. Default value is 0.

Some TTF fonts include several styles per each .ttf file and organize them by different index. If so, *bIndex* parameter can be used for selecting the TTF style. For example, Times New Roman font has font style of:

0 – Regular

1 – Italic

2 – Bold

3 – Bold italic

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Printer error codes</a>	16xxh
<a href="#">Font error codes</a>	1Exxh

## CTOS\_PrinterTTFCheckLanguageSupport

---

```
USHORT CTOS_PrinterTTFCheckLanguageSupport (ULONG ulLanguage,
BOOL *fSupported);
```

**Description** Check if specific language character set is supported in selected TrueType font file.

**Parameters**

[ IN ]	<u><i>ulLanguage</i></u> Language Id. Please refer to Appendix B.
--------	--

[ OUT ]	<u><i>fSupported</i></u>
---------	--------------------------

- ***d\_TRUE***  
Specific language character set supported.

- ***d\_FALSE***  
Specific language character set not supported.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Printer error codes</a>	16xxh

## CTOS\_PrinterCodaBarBarcode

---

```
USHORT CTOS_PrinterCodaBarBarcode(USHORT x, USHORT y, BYTE  
*baCodeContent, BYTE bContentLen, BYTE bXExtend, BYTE bY8Extend,  
BOOL bShowChar);
```

**Description** Print a barcode according to Codabar rule. Codabar was developed in 1972. It is a discrete, self-checking symbology that may encode 16 different characters, plus an additional 4 start/stop characters.

A Code 11 Barcode has the following structure:

1. One of four possible start characters (A, B, C, or D), encoded from the table below.
2. A narrow, inter-character space.
3. The data of the message, encoded from the table below, with a narrow inter-character space between each character.
4. One of four possible stop characters (A, B, C, or D), encoded from the table below.

This table indicates how to encode each digit of a Codabar barcode.

Note that the "Width Encoding" column is expressed as "0" (narrow bar or space) or "1" (wide bar or space) while the "Barcode Encoding" column represents how the barcode will actually be encoded as described above in "Encoding the Symbol."

For example, the character 0 is defined as "0000011" by Codabar. This means a "Narrow bar, narrow space, narrow bar, narrow space, narrow bar, wide space, wide bar". We convert this to 101010011 in the "Barcode Encoding" column which is consistent with the method we've used to express barcode formats in other documents on this site.

ASCII CHARACTER	WIDTH ENCODING	BARCODE ENCODING
0	0000011	101010011
1	0000110	101011001
2	0001001	101001011
3	1100000	110010101
4	0010010	101101001
5	1000010	110101001
6	0100001	100101011
7	0100100	100101101
8	0110000	100110101
9	1001000	110100101
- (Dash)	0001100	101001101
\$	0011000	101100101
:	1000101	1101011011
/ (Slash)	1010001	1101101011
.	1010100	1101101101
+	0011111	101100110011
Start/Stop A	0011010	1011001001
Start/Stop B	0001011	1010010011
Start/Stop C	0101001	1001001011
Start/Stop D	0001110	1010011001

**NOTE 1:** Since the first and last element of every character is always a bar, a narrow space is appended at the end of each character to separate the last bar of a character from the first bar of the character that follows.

**NOTE 2:** Codabar is interesting in that there are always 4 bars and 3 spaces of varying sizes, but the total size (width) of a character varies depending on the character being encoded.

#### Parameters

[ IN ] *usX*

Horizontal position of paper.

[ IN ] *usY*

Vertical position of paper.

[ IN ] *baCodeContent*

A typical Codabar barcode is:



Since Codabar is self-checking, there is no established checksum digit. Should a specific application wish to

implement a checksum digit for additional security, it is up to the implementer to define and handle same. However, keep in mind that other applications that read your barcode will interpret your checksum digit as part of the message itself.

Use ASCII character:

0~9

A,B,C,D

+ (Plus)

- (Dash)

/ (Slash)

\$

: (Colon)

. (Point)

Note: A,B,C,D is the Start/Stop character.

[ IN ] *bContentLen*

Length of the baCodeContent.

[ IN ] *bXExtend*

Horizontal extend size. The range is from 1 to 2.

[ IN ] *bY8Extend*

Vertical extend size. The range is from 1 to 3.

[ IN ] *boShowChar*

Print the barcode content or not.

#### Return Value

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Printer error codes</a>	16xxh

#### Note

The recommend extend size:

bXExtend = 2, bY8Extend = 3

## CTOS\_PrinterCode39Barcode

---

```
USHORT CTOS_PrinterCode39Barcode(USHORT x, USHORT y, BYTE  
*baCodeContent, BYTE bContentLen, BYTE bXExtend, BYTE bY8Extend,  
BOOL bShowChar);
```

**Description** Print out a barcode according to Code 39 rule. Code 39, which is the first alpha-numeric symbology to be developed, is still widely used, especially in non-retail environments. It is the standard bar code used by the United States Department of Defense, and is also used by the Health Industry Bar Code Council (HIBCC). Code 39 is also known as "3 of 9 Code" and "USD-3".

**Parameters** [ IN ] *usX*  
Horizontal position of paper.

[ IN ] *usY*  
Vertical position of paper.

[ IN ] *baCodeContent*  
A typical Code 39 bar code is:



Code 39 is a discrete, variable-length symbology. It is self-checking in that a single print defect cannot transpose one character into another valid character.

Use ASCII character:

0~9

A~Z

+ (Plus)

- (Dash)

\*

/ (Slash)

\$

%

. (Point)

SPACE

**Note:** "\*" is the first and end character.

[ IN ] *bContentLen*

Length of the baCodeContent.

[ IN ] *bXExtend*

Horizontal extend size. The range is from 1 to 2.

[ IN ] *bY8Extend*

Vertical extend size. The range is from 1 to 3.

[ IN ] *boShowChar*

Print the barcode content or not.

#### Return Value

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Printer error codes</a>	16xxh

#### Note

The recommend extend size:

- i. bXExtend = 1, bY8Extend = 3
- ii. bXExtend = 2, bY8Extend = 3

## CTOS\_PrinterEAN13Barcode

---

```
USHORT CTOS_PrinterEAN13Barcode(USHORT x, USHORT y, BYTE
*baCodeContent, BYTE bXExtend, BYTE bY8Extend, BOOL bShowChar);
```

**Description** Print out a barcode according to EAN-13 rule. EAN-13, which is based upon the UPC-A standard, was implemented by the International Article Numbering Association (EAN) in Europe. This standard was implemented mostly because the UPC-A standard was considered not a well design for international use.

**Parameters** [ IN ] *usX*  
Horizontal position of paper.

[ IN ] *usY*  
Vertical position of paper.

[ IN ] *baCodeContent*  
Input length is 12, and use '0'~'9' only.  
A typical EAN-13 bar code looks something like this:



The check digit is automatically calculated by this function.  
Please see the note below.

[ IN ] *bXExtend*  
Horizontal extend size. The range is from 1 to 3.

[ IN ] *bY8Extend*  
Vertical extend size. The range is from 1 to 3.

[ IN ]    *boShowChar*

Print the barcode content or not.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Printer error codes</a>	16xxh

**Note**

**Number System:** The number system consists of two digits (sometimes three digits) which identify the country (or economic region) numbering authority which assigned the manufacturer code. Any number system which starts with the digit **0** is a UPC-A bar code. The valid number system codes are presented in the following table:

00-13: USA & Canada  
20-29: In-Store Functions  
30-37: France  
40-44: Germany  
45: Japan (also 49)  
46: Russian Federation

471: Taiwan  
474: Estonia  
475: Latvia

477: Lithuania  
479: Sri Lanka  
480: Philippines

482: Ukraine  
484: Moldova  
485: Armenia

486: Georgia  
487: Kazakhstan  
489: Hong Kong

49: Japan (JAN-13)

50: United Kingdom

520: Greece

528: Lebanon

529: Cyprus

531: Macedonia

535: Malta

539: Ireland

54: Belgium & Luxembourg

560: Portugal

569: Iceland

57: Denmark

590: Poland

594: Romania

599: Hungary

600 & 601: South Africa

609: Mauritius

611: Morocco

613: Algeria

619: Tunisia

622: Egypt

625: Jordan

626: Iran

64: Finland

690-692: China

70: Norway

729: Israel

73: Sweden  
740: Guatemala  
741: El Salvador

742: Honduras  
743: Nicaragua  
744: Costa Rica

746: Dominican Republic  
750: Mexico  
759: Venezuela

76: Switzerland  
770: Colombia  
773: Uruguay

775: Peru  
777: Bolivia  
779: Argentina

780: Chile  
784: Paraguay  
785: Peru

786: Ecuador  
789: Brazil  
80 - 83: Italy

84: Spain  
850: Cuba  
858: Slovakia

859: Czech Republic  
860: Yugoslavia  
869: Turkey

87: Netherlands  
880: South Korea  
885: Thailand

888: Singapore  
890: India  
893: Vietnam

899: Indonesia  
90 & 91: Austria  
93: Australia

94: New Zealand  
955: Malaysia  
977: International Standard Serial Number for Periodicals (ISSN)

978: International Standard Book Numbering (ISBN)  
979: International Standard Music Number (ISMN)  
980: Refund receipts

981 & 982: Common Currency Coupons  
99: Coupons

**Manufacturer Code:** The manufacturer code is a unique code assigned to each manufacturer by the numbering authority indicated by the number system code. All products produced by a given company will use the same manufacturer code.

EAN uses what is called "variable-length manufacturer codes." Assigning fixed-length 5-digit manufacturer codes, as the UCC has done until recently, means that each manufacturer can have up to 99,999 product codes--and many manufacturers don't have that many products, which means hundreds or even thousands of potential product codes are being wasted on manufacturers that only have a few products. Thus if a potential manufacturer knows that it is only going to produce a few products, EAN may issue it a longer manufacturer code, leaving less space for the product code. This

results in more efficient use of the available manufacturer and product codes.

**Product Code:** The product code is a unique code assigned by the manufacturer. Unlike the manufacturer code, which must be assigned by the UCC, the manufacturer is free to assign product codes to each of their products without consulting any other organization. Since the UCC will already have guaranteed that the manufacturer code is unique, the manufacturer need only make sure that they do not repeat their own product codes.

**Check Digit:** The check digit is an additional digit used to verify that a bar code has been scanned correctly. Since a scan can produce incorrect data due to inconsistent scanning speed, print imperfections, or a host of other problems, it is useful to verify that the rest of the data in the bar code has been correctly interpreted. The check digit is calculated based on the rest of the digits of the bar code. Normally, if the check digit is the same as the value of the check digit based on the data that has been scanned, there is a high level of confidence that the bar code was scanned correctly. The method of calculating the check digit will be discussed later in this page.

The recommend extend size:

- i. bXExtend = 1,bY8Extend = 2
- ii. bXExtend = 2,bY8Extend = 3
- iii. bXExtend = 3,bY8Extend = 3

## CTOS\_PrinterCode128Barcode

---

```
USHORT CTOS_PrinterCode128Barcode(USHORT x, USHORT y, BYTE
*baCodeContent, BYTE bContentLen, BYTE bXExtend, BYTE bY8Extend, IN
BOOL bShowChar);
```

**Description** Print out a barcode according to Code 128 rule.

**Parameters** [ IN ] *usX*  
Horizontal position of paper.

[ IN ] *usY*  
Vertical position of paper.

[ IN ] *baCodeContent*  
Content of the barcode.

[ IN ] *bContentLen*  
Length of the baCodeContent.

[ IN ] *bXExtend*  
Horizontal extend size. The range is from 1 to 2.

[ IN ] *bY8Extend*  
Vertical extend size. The range is from 1 to 3.

[ IN ] *boShowChar*  
Print the barcode content or not.

Return Value	<b>Constants</b>	<b>Value</b>
d_OK		0000h
<a href="#">Printer error codes</a>		16xxh

## CTOS\_PrinterInterleaved2of5Barcode

---

```
USHORT CTOS_PrinterInterleaved2of5Barcode(USHORT x, USHORT y, BYTE
*baCodeContent, BYTE bContentLen, BYTE bXExtend, BYTE bY8Extend,
BOOL bShowChar);
```

**Description** Print out a barcode according to Interleave 2 of 5 rule.

**Parameters** [ IN ] *usX*

Horizontal position of paper.

[ IN ] *usY*

Vertical position of paper.

[ IN ] *baCodeContent*

Content of the barcode.

[ IN ] *bContentLen*

Length of the baCodeContent.

[ IN ] *bXExtend*

Horizontal extend size. The range is from 1 to 2.

[ IN ] *bY8Extend*

Vertical extend size. The range is from 1 to 3.

[ IN ] *bShowChar*

Print the barcode content or not.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Printer error codes</a>	16xxh

## 2.19. Printer Buffer Functions

- CTOS\_PrinterBufferEnable
- CTOS\_PrinterBufferInit
- CTOS\_PrinterBufferSelectActiveAddress

CTOS\_PrinterBufferFill

PRINTER BUFFER

VEGA5000 | VEGA5000S | VEGA3000

---

- CTOS\_PrinterBufferHLine
- CTOS\_PrinterBufferVLine
- CTOS\_PrinterBufferLogo
- CTOS\_PrinterBufferBMPPic
- CTOS\_PrinterBufferPixel
- CTOS\_PrinterBufferPutString
- CTOS\_PrinterBufferOutput
- CTOS\_PrinterBufferCodaBarBarcode
- CTOS\_PrinterBufferCode39Barcode
- CTOS\_PrinterBufferEAN13Barcode
- CTOS\_PrinterBufferCode128Barcode
- CTOS\_PrinterBufferInterleaved2of5Barcode

### Printer Buffer Error Codes

<b>Constants</b>	<b>Value</b>
d_PRTBUF_NOT_INIT	1701h
d_PRTBUF_PARA_ERR	1702h
d_PRTBUF_CANVAS_OVERFLOW	1703h
d_PRTBUF_PIC_FORMAT_NOT_SUPPORT	1704h
d_PRTBUF_PIC_OPEN_FAILED	1705h
d_PRTBUF_PIC_OVER_SIZE	1706h

## CTOS\_PrinterBufferEnable

---

```
voidCTOS_PrinterBufferEnable(void);
```

**Description** Initialize the printer buffer and the default buffer is registered as the printer buffer. Programmer can regard the printer buffer as a canvas, and draw anything desired to the canvas before outputting the canvas to the printer.

**Parameters** None

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">PrinterBufferErrorCodes</a>	17xxh

**Note** The width and height of default buffer is defined as **PB\_CANVAS\_X\_SIZE** and **PB\_CANVAS\_Y\_SIZE**.

## **CTOS\_PrinterBufferInit**

---

USHORT CTOS\_PrinterBufferInit(BYTE\*pPtr, USHORT usHeight);

**Description** Initialize the printer buffer and register the input buffer as the printer buffer. Programmer can regard the printer buffer as a canvas, and draw anything desired to the canvas before outputting the canvas to the printer. Register the user's buffer into the list of the printer canvas.

**Parameters** [ IN ] *pPtr*  
Buffer to register as printer buffer.

[ IN ] *usHeight*  
Height of buffer in pixel. Must be byte-aligned (multiple of 8).

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">PrinterBufferErrorCodes</a>	17xxh

**Note** The width of printer buffer is fixed to the definition **PB\_CANVAS\_X\_SIZE**.  
The max number of buffer can be registered is defined as **MAX\_PB\_CANVAS\_NUM**.

## CTOS\_PrinterBufferSelectActiveAddress

---

USHORT CTOS\_PrinterBufferSelectActiveAddress (BYTE \*pPtr);

**Description** Select between several printer buffers registered with **CTOS\_PrinterBufferInit()**.

**Parameters** [ IN ] *pPtr*  
Pointer to the buffer previously registered as printer buffer.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">PrinterBufferErrorCodes</a>	17xxh

## **CTOS\_PrinterBufferFill**

---

```
USHORT CTOS_PrinterBufferFill(USHORT usXStart, USHORT usYStart,
USHORT usXEnd, USHORT usYEnd, BOOL fPat);
```

**Description** Fill a rectangle box in the printer buffer.

- Parameters**
- [ IN ] *usXStart*  
X coordinate of the start point (up-left point of the rectangle) to draw on printer buffer.
  - [ IN ] *usYStart*  
Y coordinate of the start point to draw on printer buffer.
  - [ IN ] *usXEnd*  
X coordinate of the end point (down-right point of the rectangle) to draw on printer buffer.
  - [ IN ] *usYEnd*  
Y coordinate of the end point to draw on printer buffer.
  - [ IN ] *fPat*
    - = 1 : Set all pixel in the rectangle.
    - = 0 : Clear all pixel in the rectangle.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PrinterBufferErrorCodes</a>	17xxh

## CTOS\_PrinterBufferHLine

---

```
USHORT CTOS_PrinterBufferHLine(USHORT usXStart, USHORT usYStart,
USHORT usXEnd, BOOL fPat);
```

**Description** Draw a horizontal line in the printer buffer.

**Parameters**

[ IN ]	<u><i>usXStart</i></u>
	X coordinate of the start point to draw on printer buffer.

[ IN ]	<u><i>usYStart</i></u>
	Y coordinate of the start point to draw on printer buffer.

[ IN ]	<u><i>usXEnd</i></u>
	X coordinate of the end point on printer buffer.

[ IN ]	<u><i>fPat</i></u>
	= 1 : Set all pixel on the line. = 0 : Clear all pixel on the line.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PrinterBufferErrorCodes</a>	17xxh

## CTOS\_PrinterBufferVLine

---

```
USHORT CTOS_PrinterBufferVLine(USHORT usXStart, USHORT usYStart,
USHORT usYEnd, BOOL fPat);
```

**Description** Draw a vertical line in the printer buffer.

**Parameters** [ IN ] *usXStart*  
X coordinate of the start point to draw on printer buffer.

[ IN ] *usYStart*  
Y coordinate of the start point to draw on printer buffer.

[ IN ] *usYEnd*  
Y coordinate of the end point.

[ IN ] *fPat*  
= 1 : Set all pixel on the line.  
= 0 : Clear all pixel on the line.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PrinterBufferErrorCodes</a>	17xxh

## CTOS\_PrinterBufferLogo

---

```
USHORT CTOS_PrinterBufferLogo(USHORT usXPos, USHORT usYPos, USHORT
usWidth, USHORT usHeight, BYTE *baPat);
```

**Description** Draw a picture in the printer buffer.

**Parameters** [ IN ] *usXPos*  
X coordinate of the start point to draw on printer buffer.

[ IN ] *usYPos*  
Y coordinate of the start point to draw on printer buffer.

[ IN ] *usWidth*  
Width of the pattern.

[ IN ] *usHeight*  
Height of the pattern.

[ IN ] *baPat*  
Buffer storing the pattern.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PrinterBufferErrorCodes</a>	17xxh

## CTOS\_PrinterBufferBMPPic

---

```
USHORT CTOS_PrinterBufferBMPPic (BYTE* pbPtr, USHORT usXPos, USHORT
usYPos, BYTE *baFilename);
```

**Description** Draw a BMP format picture in the printer buffer.

**Parameters** [ IN ] *pbPtr*  
Buffer storing the picture.

[ IN ] *usXPos*  
X coordinate of the start point to draw on printer buffer.

[ IN ] *usYPos*  
Y coordinate of the start point to draw on printer buffer.

[ IN ] *baFilename*  
Path and filename of the BMP to be printed.  
Ex: "./myfile/myBMP.bmp"

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PrinterBufferErrorCodes</a>	17xxh

## CTOS\_PrinterBufferPixel

---

```
USHORT CTOS_PrinterBufferPixel (USHORT usXPos, USHORT usYPos, BOOL fPat);
```

**Description** Draw a pixel in the printer buffer.

**Parameters** [ IN ] *usXPos*  
X coordinate of the point to draw on printer buffer.

[ IN ] *usYPos*  
Y coordinate of the point to draw on printer buffer.

[ IN ] *fPat*  
= 1 : Set the pixel.  
= 0 : Clear the pixel.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PrinterBufferErrorCodes</a>	17xxh

## CTOS\_PrinterBufferPutString

---

```
USHORT CTOS_PrinterBufferPutString(BYTE*pbPtr, USHORT usXPos, USHORT
usYPos, BYTE *baStr, CTOS_FONT_ATTRIB*ATTRIB);
```

**Description** Draw a string into the printer buffer.

**Parameters**

[ IN ]	<u><i>pbPtr</i></u> Printer buffer. Its width is PAPER_X_SIZE per line.
--------	--

[ IN ]	<u><i>usXPos</i></u> X coordinate in the printer buffer to draw the string.
--------	--

[ IN ]	<u><i>usYPos</i></u> Y coordinate in the printer buffer to draw the string.
--------	--

[ IN ]	<u><i>baStr</i></u> String to print.
--------	---

[ IN ]	<u><i>ATTRIB</i></u> Font attribute of the string.
--------	---

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">PrinterBufferErrorCodes</a>	17xxh

**Example** Please refer to example in appendix [B.14 Printer](#)

**Note**

CTOS\_FONT\_ATTRIB:

FontSize: The font size for string. For non-ASCII string, Language must be selected by calling CTOS\_LanguageConfig().

X\_Zoom: The size to zoom in X coordinate. For example, if X\_Zoom is 1, it means normal size, and 2 means double size. 0 means print nothing .

Y\_Zoom: The size to zoom in Y coordinate. For example, if Y\_Zoom

is 1, it means normal size, and 2 means double size. 0 means print nothing.

X\_Space: The space in dot to insert between each character in x coordinate.

Y\_Space: The space in dot to insert between each character in y coordinate.

## Note 2

Escape codes:

"\n": line feed.

"\fn": normal mode.

"\fu": under line mode.

"\fr": reverse mode.

"\ft1": print type1 font.

"\ft2": print type2 font.

"\ft3": print type3 font.

## CTOS\_PrinterBufferOutput

---

```
USHORT CTOS_PrinterBufferOutput(BYTE*pbPtr, USHORT usY8Len);
```

**Description** Print out the content of the printer buffer.

**Parameters** [ IN ] *pbPtr*  
Printer buffer.

[ IN ] *usY8Len*  
Height(in byte) of printer buffer to print out.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">PrinterBufferErrorCodes</a>	17xxh

**Example** Please refer to example in appendix [B.14 Printer](#)

## CTOS\_PrinterBufferCodaBarBarcode

---

```
USHORT CTOS_PrinterBufferCodaBarBarcode(USHORT x, USHORT y, BYTE  
*baCodeContent, BYTE bContentLen, BYTE bXExtend, BYTE bY8Extend,  
BOOL bShowChar);
```

**Description** Draw a barcode in the printer buffer according to Codabar rule. Codabar was developed in 1972. It is a discrete, self-checking symbology that may encode 16 different characters, plus an additional 4 start/stop characters.

A Code 11 Barcode has the following structure:

1. One of four possible start characters (A, B, C, or D), encoded from the table below.
2. A narrow, inter-character space.
3. The data of the message, encoded from the table below, with a narrow inter-character space between each character.
4. One of four possible stop characters (A, B, C, or D), encoded from the table below.

This table indicates how to encode each digit of a Codabar barcode. Note that the "Width Encoding" column is expressed as "0" (narrow bar or space) or "1" (wide bar or space) while the "Barcode Encoding" column represents how the barcode will actually be encoded as described above in "Encoding the Symbol."

For example, the character 0 is defined as "0000011" by Codabar. This means a "Narrow bar, narrow space, narrow bar, narrow space, narrow bar, wide space, wide bar". We convert this to 101010011 in the "Barcode Encoding" column which is consistent with the method we've used to express barcode formats in other documents on this site.

ASCII CHARACTER	WIDTH ENCODING	BARCODE ENCODING
0	0000011	101010011
1	0000110	101011001
2	0001001	101001011
3	1100000	110010101
4	0010010	101101001
5	1000010	110101001
6	0100001	100101011
7	0100100	100101101
8	0110000	100110101
9	1001000	110100101
- (Dash)	0001100	101001101
\$	0011000	101100101
:	1000101	1101011011
/ (Slash)	1010001	1101101011
.	1010100	1101101101
+	0011111	101100110011
Start/Stop A	0011010	1011001001
Start/Stop B	0001011	1010010011
Start/Stop C	0101001	1001001011
Start/Stop D	0001110	1010011001

**NOTE 1:** Since the first and last element of every character is always a bar, a narrow space is appended at the end of each character to separate the last bar of a character from the first bar of the character that follows.

**NOTE 2:** Codabar is interesting in that there are always 4 bars and 3 spaces of varying sizes, but the total size (width) of a character varies depending on the character being encoded.

#### Parameters

[ IN ] *usX*

Horizontal position of paper.

[ IN ] *usY*

Vertical position of paper.

[ IN ] *baCodeContent*

A typical Codabar barcode is:



Since Codabar is self-checking, there is no established checksum digit. Should a specific application wish to

implement a checksum digit for additional security, it is up to the implementer to define and handle same. However, keep in mind that other applications that read your barcode will interpret your checksum digit as part of the message itself.

Use ASCII character:

0~9

A,B,C,D

+ (Plus)

- (Dash)

/ (Slash)

\$

: (Colon)

. (Point)

Note: A,B,C,D is the Start/Stop character.

[ IN ] *bContentLen*

Length of the baCodeContent.

[ IN ] *bXExtend*

Horizontal extend size. The range is from 1 to 2.

[ IN ] *bY8Extend*

Vertical extend size. The range is from 1 to 3.

[ IN ] *boShowChar*

Print the barcode content or not.

#### Return Value

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#"><u>PrinterBufferErrorCodes</u></a>	17xxh

#### Note

The recommend extend size:

bXExtend = 2, bY8Extend = 3

## CTOS\_PrinterBufferCode39Barcode

---

```
USHORT CTOS_PrinterBufferCode39Barcode(USHORT x, USHORT y, BYTE  
*baCodeContent, BYTE bContentLen, BYTE bXExtend, BYTE bY8Extend,  
BOOL bShowChar);
```

**Description** Draw a barcode in the printer buffer according to Code 39 rule. Code 39, which is the first alpha-numeric symbology to be developed, is still widely used, especially in non-retail environments. It is the standard bar code used by the United States Department of Defense, and is also used by the Health Industry Bar Code Council (HIBCC). Code 39 is also known as "3 of 9 Code" and "USD-3".

**Parameters** [ IN ] *usX*  
Horizontal position of paper.

[ IN ] *usY*  
Vertical position of paper.

[ IN ] *baCodeContent*  
A typical Code 39 bar code is:



Code 39 is a discrete, variable-length symbology. It is self-checking in that a single print defect cannot transpose one character into another valid character.

Use ASCII character:

0~9

A~Z

+ (Plus)

- (Dash)

\*

/ (Slash)

\$

%

. (Point)

SPACE

Note: "\*" is the first and end character.

[ IN ] *bContentLen*

Length of the baCodeContent.

[ IN ] *bXExtend*

Horizontal extend size. The range is from 1 to 2.

[ IN ] *bY8Extend*

Vertical extend size. The range is from 1 to 3.

[ IN ] *boShowChar*

Print the barcode content or not.

#### Return Value

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PrinterBufferErrorCodes</a>	17xxh

#### Note

The recommend extend size:

bXExtend :1, bY8Extend:3

bXExtend :2, bY8Extend:3

## **CTOS\_PrinterBufferEAN13Barcode**

---

```
USHORT CTOS_PrinterBufferEAN13Barcode(USHORT x, USHORT y, BYTE
*baCodeContent, BYTE bXExtend, BYTE bY8Extend, BOOL bShowChar);
```

**Description** Draw a barcode in the printer buffer according to EAN-13 rule. EAN-13, which is based upon the UPC-A standard, was implemented by the International Article Numbering Association (EAN) in Europe. This standard was implemented mostly because the UPC-A standard was considered not a well design for international use.

**Parameters** [ IN ] *usX*  
Horizontal position of paper.

[ IN ] *usY*  
Vertical position of paper.

[ IN ] *baCodeContent*  
Input length is 12, and use '0'~'9' only.  
A typical EAN-13 bar code looks something like this:



The check digit is automatically calculated by this function.  
Please see the note below.

[ IN ] *bXExtend*  
Horizontal extend size. The range is from 1 to 3.

[ IN ] *bY8Extend*  
Vertical extend size. The range is from 1 to 3.

[ IN ]    *boShowChar*

Print the barcode content or not.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PrinterBufferErrorCodes</a>	17xxh

**Note**

**Number System:** The number system consists of two digits (sometimes three digits) which identify the country (or economic region) numbering authority which assigned the manufacturer code. Any number system which starts with the digit 0 is a UPC-A bar code.

The valid number system codes are presented in the following table:

00-13: USA & Canada

20-29: In-Store Functions

30-37: France

40-44: Germany

45: Japan (also 49)

46: Russian Federation

471: Taiwan

474: Estonia

475: Latvia

477: Lithuania

479: Sri Lanka

480: Philippines

482: Ukraine

484: Moldova

485: Armenia

486: Georgia

487: Kazakhstan

489: Hong Kong

49: Japan (JAN-13)

50: United Kingdom

520: Greece

528: Lebanon

529: Cyprus

531: Macedonia

535: Malta

539: Ireland

54: Belgium & Luxembourg

560: Portugal

569: Iceland

57: Denmark

590: Poland

594: Romania

599: Hungary

600 & 601: South Africa

609: Mauritius

611: Morocco

613: Algeria

619: Tunisia

622: Egypt

625: Jordan

626: Iran

64: Finland

690-692: China

70: Norway

729: Israel

73: Sweden  
740: Guatemala  
741: El Salvador

742: Honduras  
743: Nicaragua  
744: Costa Rica

746: Dominican Republic  
750: Mexico  
759: Venezuela

76: Switzerland  
770: Colombia  
773: Uruguay

775: Peru  
777: Bolivia  
779: Argentina

780: Chile  
784: Paraguay  
785: Peru

786: Ecuador  
789: Brazil  
80 - 83: Italy

84: Spain  
850: Cuba  
858: Slovakia

859: Czech Republic  
860: Yugoslavia  
869: Turkey

87: Netherlands  
880: South Korea  
885: Thailand

888: Singapore  
890: India  
893: Vietnam

899: Indonesia  
90 & 91: Austria  
93: Australia

94: New Zealand  
955: Malaysia  
977: International Standard Serial Number for Periodicals (ISSN)

978: International Standard Book Numbering (ISBN)  
979: International Standard Music Number (ISMN)  
980: Refund receipts

981 & 982: Common Currency Coupons  
99: Coupons

**Manufacturer Code:** The manufacturer code is a unique code assigned to each manufacturer by the numbering authority indicated by the number system code. All products produced by a given company will use the same manufacturer code.

EAN uses what is called "variable-length manufacturer codes." Assigning fixed-length 5-digit manufacturer codes, as the UCC has done until recently, means that each manufacturer can have up to 99,999 product codes--and many manufacturers don't have that many products, which means hundreds or even thousands of potential product codes are being wasted on manufacturers that only have a few products. Thus if a potential manufacturer knows that it is only going to produce a few products, EAN may issue it a longer

manufacturer code, leaving less space for the product code. This results in more efficient use of the available manufacturer and product codes.

**Product Code:** The product code is a unique code assigned by the manufacturer. Unlike the manufacturer code, which must be assigned by the UCC, the manufacturer is free to assign product codes to each of their products without consulting any other organization. Since the UCC will already have guaranteed that the manufacturer code is unique, the manufacturer need only make sure that they do not repeat their own product codes.

**Check Digit:** The check digit is an additional digit used to verify that a bar code has been scanned correctly. Since a scan can produce incorrect data due to inconsistent scanning speed, print imperfections, or a host of other problems, it is useful to verify that the rest of the data in the bar code has been correctly interpreted. The check digit is calculated based on the rest of the digits of the bar code. Normally, if the check digit is the same as the value of the check digit based on the data that has been scanned, there is a high level of confidence that the bar code was scanned correctly. The method of calculating the check digit will be discussed later in this page.

The recommend extend size:

bXExtend :1 ,bY8Extend:2

bXExtend :2 ,bY8Extend:3

bXExtend :3 ,bY8Extend:3

## CTOS\_PrinterBufferCode128Barcode

---

```
USHORT CTOS_PrinterBufferCode128Barcode(USHORT x, USHORT y, BYTE
*baCodeContent, BYTE bContentLen, BYTE bXExtend, BYTE bY8Extend,
BOOL bShowChar);
```

**Description** Draw a barcode in the printer buffer according to Code 128 rule.

<b>Parameters</b>	[ IN ] <u><i>usX</i></u> Horizontal position of paper.
	[ IN ] <u><i>usY</i></u> Vertical position of paper.
	[ IN ] <u><i>baCodeContent</i></u> Content of the barcode.
	[ IN ] <u><i>bContentLen</i></u> Length of the baCodeContent.
	[ IN ] <u><i>bXExtend</i></u> Horizontal extend size. The range is from 1 to 2.
	[ IN ] <u><i>bY8Extend</i></u> Vertical extend size. The range is from 1 to 3.
	[ IN ] <u><i>bShowChar</i></u> Print the barcode content or not.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PrinterBufferErrorCodes</a>	17xxh

## CTOS\_PrinterBufferInterleaved2of5Barcode

---

```
USHORT CTOS_PrinterBufferInterleaved2of5Barcode(USHORT x, USHORT y,
BYTE *baCodeContent, BYTE bContentLen, BYTE bXExtend, BYTE bY8Extend,
BOOL bShowChar);
```

**Description** Draw a barcode in the printer buffer according to Interleave 2 of 5 rule.

<b>Parameters</b>	[ IN ] <u><i>usX</i></u> Horizontal position of paper.
	[ IN ] <u><i>usY</i></u> Vertical position of paper.
	[ IN ] <u><i>baCodeContent</i></u> Content of the barcode.
	[ IN ] <u><i>bContentLen</i></u> Length of the baCodeContent.
	[ IN ] <u><i>bXExtend</i></u> Horizontal extend size. The range is from 1 to 2.
	[ IN ] <u><i>bY8Extend</i></u> Vertical extend size. The range is from 1 to 3.
	[ IN ] <u><i>bShowChar</i></u> Print the barcode content or not.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">PrinterBufferErrorCodes</a>	17xxh

## 2.20.Smart Card Functions

- CTOS\_SCStatus
- CTOS\_SCPowerOff
- CTOS\_SCResetEMV
- CTOS\_SCResetISO
- CTOS\_SCSendAPDU
- CTOS\_SCCommonReset

### Smart Card Error Codes

<b>Constants</b>	<b>Value</b>
d_SC_NOT_PRESENT	1401h
d_SC_NOT_ACTIVATED	1402h
d_SC_COMM_ERROR	1403h
d_SC_MUTE	1404h
d_SC_PARITY_ERROR	1405h
d_SC_EDC_ERROR	1406h
d_SC_BUFFER_OVERRUN	1407h
d_SC_HW_ERROR	1408h
d_SC_DEACTIVATED_PROTOCOL	1409h
d_SC_BAD_TS	140Ah
d_SC_ATR_TOO_LONG	140Bh
d_SC_ATR_INVALID	140Ch
d_SC_TA1_NOT_SUPPORTED	140Dh
d_SC PTS_RESPONSE_ERROR	140Eh
d_SC PROCEDURE_BYTE_CONFLICT	140Fh
d_SC_POWER_FAILED	1410h
d_SC_INVALID_PARAMETER	1411h
d_SC_INSUFFICIENT_BUF	1420h
d_SC_SOCKET_NOT_EXIST	1430h
d_SC_BUF_TOO_SMALL	1431h

## CTOS\_SCStatus

---

USHORT CTOS\_SCStatus(BYTE bID, BYTE\*baStatus);

**Description** Check the ICC status.

**Parameters** [ IN ] bID  
Socket number of ICC module.

- **d\_SC\_USER**
- **d\_SC\_SAM1**
- **d\_SC\_SAM2**
- **d\_SC\_SAM3**
- **d\_SC\_SAM4**

[ OUT ] baStatus  
Bitmask of the ICC status.

- **d\_MK\_SC\_PRESENT**  
ICC insert / not insert

- **d\_MK\_SC\_ACTIVE**  
ICC active / not active

Return Value	Constants	Value
	d_OK	0000h
	<a href="#">Smart Card error codes</a>	14xxh

**Example** Please refer to example in appendix [B.16 SmartCard](#)

**Note** Call this function to check whether the ICC is present or absent and whether the ICC is active or de-active. For all SAM card, it is assumed always exist, so that the **d\_MK\_SC\_PRESENT** is always set.

**d\_SC\_USER** refers to the chip (ISO7816) reader of the terminal.

## CTOS\_SCPowerOff

---

```
USHORT CTOS_SCPowerOff (BYTE bID);
```

**Description** Turn off the power of ICC.

**Parameters** [ IN ] bID  
Socket number of ICC module.

- d\_SC\_USER
- d\_SC\_SAM1
- d\_SC\_SAM2
- d\_SC\_SAM3
- d\_SC\_SAM4

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Smart Card error codes</a>	14xxh

**Example** Please refer to example in appendix [B.16 SmartCard](#)

**Note** d\_SC\_USER refers to the chip (ISO7816) reader of the terminal.

## CTOS\_SCResetEMV

---

```
USHORT CTOS_SCResetEMV(BYTE bID, BYTE bVolt, BYTE*baATR,
BYTE*baATRLen, BYTE*baCardType);
```

**Description** Power on the ICC and return the ATR which meets the requirement of EMV specification.

**Parameters** [ IN ] bID  
Socket number of ICC module.

- d\_SC\_USER
- d\_SC\_SAM1
- d\_SC\_SAM2
- d\_SC\_SAM3
- d\_SC\_SAM4

[ IN ] bVolt  
Voltage level to apply to ICC.

- d\_SC\_5V
- d\_SC\_3V
- d\_SC\_1\_8V

[ OUT ] baATR  
Buffer to store ATR return from ICC.

[ IN ] baATRLen  
Size of baATR.

[ OUT ] baATRLen  
Length of ATR returned from ICC.

[ OUT ] baCardType  
Type of the ICC in the socket.

- d\_SC\_TYPE\_T0

T=0 asynchronous ICC

- *d\_SC\_TYPE\_T1*

T=1 asynchronous ICC

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Smart Card error codes</a>	14xxh

**Example**

Please refer to example in appendix [B.16 SmartCard](#)

**Note**

This function performs the ICC reset and parse the ATR according to the EMV specification. If the ATR do not meet the requirement of EMV specification, the function will return with error.

## CTOS\_SCResetISO

---

```
USHORT CTOS_SCResetISO(BYTE bID, BYTE bVolt, BYTE*baATR,
BYTE*baATRLen, BYTE*baCardType);
```

**Description** Power on the ICC and return the ATR which meets ISO-7816 specification.

**Parameters** [ IN ] *bID*  
Socket number of ICC module.

- *d\_SC\_USER*
- *d\_SC\_SAM1*
- *d\_SC\_SAM2*
- *d\_SC\_SAM3*
- *d\_SC\_SAM4*

[ IN ] *bVolt*  
Voltage level to apply to ICC.

- *d\_SC\_5V*
- *d\_SC\_3V*
- *d\_SC\_1\_8V*

[ OUT ] *baATR*  
Buffer to store ATR return from ICC.

[ IN ] *baATRLen*  
Size of baATR.

[ OUT ] *baATRLen*  
Length of ATR returned from ICC.

[ OUT ] *baCardType*  
Type of the ICC in the socket.

- *d\_SC\_TYPE\_T0*

T=0 asynchronous ICC

- *d\_SC\_TYPE\_T1*

T=1 asynchronous ICC

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Smart Card error codes</a>	14xxh

**Example**

Please refer to example in appendix [B.16 SmartCard](#)

**Note**

This function performs the ICC reset and parse the ATR according to the ISO-7816 specification. Generally ISO-7816 specification has less restriction than EMV specification.

## CTOS\_SCSendAPDU

---

```
USHORT CTOS_SCSendAPDU(BYTE bID, BYTE*baSBuf, USHORT usSLen,
BYTE*baRBuf, USHORT*pusRLen);
```

**Description** Send an APDU command to ICC and get the response from ICC.

**Parameters** [ IN ] bID

Socket number of ICC module.

- d\_SC\_USER
- d\_SC\_SAM1
- d\_SC\_SAM2
- d\_SC\_SAM3
- d\_SC\_SAM4

[ IN ] baSBuf

Buffer storing APDU command to send to ICC.

[ IN ] usSLen

Length of APDU command in baSBuf.

[ OUT ] baRBuf

Buffer to store the response from ICC.

[ IN ] pusRLen

Size of baRBuf.

[ OUT ] pusRLen

Length of response from ICC.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Smart Card error codes</a>	14xxh

**Example**

Please refer to example in appendix [B.16 SmartCard](#)

**Note**

This function take care of all the detail of T=0/T=1 protocol.

## CTOS\_SCCommonReset

---

```
USHORT CTOS_SCCommonReset (BYTE bID, BYTE bTA1, BOOL fColdReset,  
BOOL fEMV, BOOL fPTS, BOOL fIFSD, BYTE bVolt, BYTE* baATR,  
BYTE* bATRLen, BYTE* bCardType);
```

**Description** Power on the ICC and return the ATR.

**Parameters** [ IN ] *bID*  
Socket number of ICC module.

- ***d\_SC\_USER***
- ***d\_SC\_SAM1***
- ***d\_SC\_SAM2***
- ***d\_SC\_SAM3***
- ***d\_SC\_SAM4***

[ IN ] *bTA1*  
FI DI value for transmission speed.

[ IN ] *fColdReset*  

- ***d\_TRUE***  
Perform cold reset.
- ***d\_FALSE***  
Perform warm reset.

[ IN ] *fEMV*  

- ***d\_TRUE***  
Perform ICC reset and parse the ATR according to  
the EMV specification.
- ***d\_FALSE***  
Perform ICC reset and parse the ATR according to  
the ISO-7816 specification.

[ IN ] *fPTS*

- ***d\_TRUE***

Perform Protocol Type Selection (PTS).

- ***d\_FALSE***

No PTS execution.

[ IN ] *fIFSD*

- ***d\_TRUE***

Indicate IFSD supported.

- ***d\_FALSE***

Indicate IFSD not supported.

[ IN ] *bVolt*

Voltage level to apply to ICC.

- ***d\_SC\_5V***
- ***d\_SC\_3V***
- ***d\_SC\_1\_8V***

[ OUT ] *baATR*

Buffer to store ATR return from ICC.

[ IN ] *baATRLen*

Size of baATR.

[ OUT ] *baATRLen*

Length of ATR returned from ICC.

[ OUT ] *baCardType*

Type of the ICC in the socket.

- ***d\_SC\_TYPE\_T0***

T=0 asynchronous ICC

- ***d\_SC\_TYPE\_T1***

T=1 asynchronous ICC

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Smart Card error codes</a>	14xxh

**Example** Please refer to example in appendix [B.16 SmartCard](#)

**Note** This function performs the ICC reset and parse the ATR according to the ISO-7816 specification. Generally ISO-7816 specification has less restriction than EMV specification.

## 2.21. Memory Card Functions

### General

- CTOS\_SyncICCReset

### SLE44x2

- CTOS\_44x2ReadMainMemory
- CTOS\_44x2WriteMainMemory
- CTOS\_44x2ReadProtectionMemory
- CTOS\_44x2SetProtectionBit
- CTOS\_44x2ReadSecurityMemory
- CTOS\_44x2ChangePSC
- CTOS\_44x2VerifyPSC

### SLE44x8

- CTOS\_44x8WriteMemoryWithoutPBit
- CTOS\_44x8WriteMemoryWithPBit
- CTOS\_44x8WritePBitWithDataCompare
- CTOS\_44x8ReadMemoryWithoutPBit
- CTOS\_44x8ReadMemoryWithPBit
- CTOS\_44x8VerifyPSC

### SLE44x6

- CTOS\_44x6ReadData
- CTOS\_44x6WriteData
- CTOS\_44x6Reload
- CTOS\_44x6VerifyTSC
- CTOS\_44x6Auth

### SLE44x4

- CTOS\_SLE44x4ReadMemory
- CTOS\_SLE44x4VerifyUserCode
- CTOS\_SLE44x4VerifyMemoryCode
- CTOS\_SLE44x4WriteMemory
- CTOS\_SLE44x4EraseMemory
- CTOS\_SLE44x4BlowFuse

- CTOS\_SLE44x4EnterTestMode
- CTOS\_SLE44x4ExitTestMode

### **GPM896**

- CTOS\_GPM896ReadMemory
- CTOS\_GPM896VerifySecurityCode
- CTOS\_GPM896WriteMemory
- CTOS\_GPM896EraseMemory
- CTOS\_GPM896EraseArea1
- CTOS\_GPM896EraseArea2

### **I2C**

- CTOS\_I2CCReset
- CTOS\_I2CReadMemory
- CTOS\_I2CWriteMemory

### **Memory Card Error Codes**

<b>Constants</b>	<b>Value</b>
d_MC_NOT_PRESENT	1501h
d_MC_NOT_ACTIVATED	1502h
d_MC_MUTE	1504h
d_MC_PARITY_ERROR	1505h
d_MC_EDC_ERROR	1506h
d_MC_BUFFER_OVERRUN	1507h
d_MC_HW_ERROR	1508h
d_MC_DEACTIVATED_PROTOCOL	1509h
d_MC_POWER_FAILED	1510h
d_MC_INVALID_PARAMETER	1511h
d_MC_SOCKET_NOT_EXIST	1530h
d_MC_BUF_TOO_SMALL	1531h
d_MC_COMM_ERROR	15A3h
d_MC_COMMAND_ERROR	15A4h
d_MC_LOCKED_Memory	15A5h
d_MC_VERIFY_FAIL	15A6h
d_MC_WRITE_ERROR	15A7h
d_MC_NOT_SUPPORTED	15ABh

## **CTOS\_SyncICCReset**

---

USHORT CTOS\_SyncICCReset (UCHAR\*baBuf, BYTE\*bLen) ;

**Description** Reset a SLE44x2, SLE44x4 SLE44x6 or SLE44x8 memory card.

**Parameters** [ OUT ] *baBuf*  
Buffer to store the response data from memory card..

[ IN ] *bLen*  
Size of baBuf.

[ OUT ] *bLen*  
Length of response data.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

**Note** After calling this function successfully, the card type information will be put in the first byte of baBuf.

0x02 : SLE memory card  
0x03 : I2C card

## CTOS\_44x2ReadMainMemory

---

USHORT CTOS\_44x2ReadMainMemory (BYTE bAddr, UCHAR\*baBuf, USHORT\*usLen) ;

**Description** Read out the data of main memory.

**Parameters** [ IN ] bAddr

Start address of main memory to read. The range is from 0x00 to 0xFF.

[ OUT ] baBuf

Buffer to store the data of main memory.

[ IN ] usLen

Size of buffer. Maximum number of data to read.

[ OUT ] usLen

Length of data returned actually in the buffer.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

**Example**

Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x2WriteMainMemory

---

USHORT CTOS\_44x2WriteMainMemory (BYTE bAddr, UCHAR\*baBuf, USHORT usLen) ;

**Description** Write data to main memory.

**Parameters**

[ IN ]	<u>bAddr</u>
Start address of main memory to write. The range is from 0x00 to 0xFF.	

[ IN ]	<u>baBuf</u>
Data to write.	

[ IN ]	<u>usLen</u>
Length of data. The range is from 1 to 256.	

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x2ReadProtectionMemory

---

```
USHORT CTOS_44x2ReadProtectionMemory(UCHAR*baBuf);
```

**Description** Read out the data of protection memory. The function read out all four bytes at once.

**Parameters** [ OUT ] *baBuf*  
Buffer to store the data of protection memory, 4 bytes.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## **CTOS\_44x2SetProtectionBit**

---

USHORT CTOS\_44x2SetProtectionBit( UCHAR bAddr );

**Description** Set the protection bit in protection memory.

**Parameters** [ IN ] *bAddr*  
Address of protection bit to set. The range is from 0x00 to 0x1F.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

**Note** Once a bit in the protection memory is set, the corresponding bytes in the main memory will be protected.  
For example, if programmer calls **CTOS\_44x2SetProtectionBit(3)**, the data in address 3 of main memory is protected.

## CTOS\_44x2ReadSecurityMemory

---

```
USHORT CTOS_44x2ReadSecurityMemory (UCHAR*baBuf) ;
```

**Description**      Read out the data of security memory. The function read out all four bytes at once.

**Parameters**      [ OUT ] *baBuf*  
                        Buffer to store the data of security memory, 4 bytes.  
                        Byte1:Error counter  
                        Byte2~Byte4:PSC

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example**      Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x2ChangePSC

---

```
USHORT CTOS_44x2ChangePSC( UCHAR*baBuf );
```

**Description** Change the PSC. User have to successfully verify the PSC before changing the PSC.

**Parameters** [ IN ] *baBuf*  
New PSC value to change, 3 bytes.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x2VerifyPSC

---

```
USHORT CTOS_44x2VerifyPSC( UCHAR*baBuf );
```

**Description** Verify the PSC.

**Parameters** [ IN ] *baBuf*  
PSC value to verify, 3 bytes.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## **CTOS\_44x8WriteMemoryWithoutPBit**

---

```
USHORT CTOS_44x8WriteMemoryWithoutPBit(USHORT
usAddr, UCHAR*baBuf, USHORT usLen);
```

**Description** Write data to memory. The associated protect bits are not changed.

**Parameters** [ IN ] *usAddr*  
Start address to write. The range is from 0x00 to 0x3FF.

[ IN ] *baBuf*  
Data to write.

[ IN ] *usLen*  
Length of data to write.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x8WriteMemoryWithPBit

---

```
USHORT CTOS_44x8WriteMemoryWithPBit(USHORT usAddr, UCHAR*baBuf, USHORT usLen);
```

**Description** Write data to memory and change the associated protect bits from 0x01 to 0x00. Then the data in memory is protected which means it is locked and can not be changed by further write command.

**Parameters** [ IN ] *usAddr*  
Start address to write. The range is from 0x00 to 0x3FF.

[ IN ] *baBuf*  
Data to write.

[ IN ] *usLen*  
Length of data to write.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x8WritePBitWithDataCompare

---

```
USHORT CTOS_44x8WritePBitWithDataCompare(USHORT
usAddr, UCHAR*baBuf, USHORT usLen);
```

**Description** Write protected bit with data comparison. Comparing the input data and the content of memory, if they have the same value then the value of associated protect bit will be changed from 0x01 to 0x00, and the content of the memory is protected. Otherwise, the value of protected bit will be keep the same.

**Parameters** [ IN ] *usAddr*  
Start address to compare. The range is from 0x00 to 0x3FF.

[ IN ] *baBuf*  
Data to compare.

[ IN ] *usLen*  
Length of data to compare.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x8ReadMemoryWithoutPBit

---

```
USHORT CTOS_44x8ReadMemoryWithoutPBit (USHORT
usAddr, UCHAR*baBuf, USHORT*usLen) ;
```

**Description** Read out the data of memory.

**Parameters**

[ IN ]	<u><i>usAddr</i></u>
	Start address to read. The range is from 0x00 to 0x3FF.

[ OUT ]	<u><i>baBuf</i></u>
	Buffer to store the data of memory.

[ IN ]	<u><i>usLen</i></u>
	Size of buffer. Maximum number of data to read.

[ OUT ]	<u><i>usLen</i></u>
	Length of data actually returned in the buffer.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x8ReadMemoryWithPBit

---

```
USHORT CTOS_44x8ReadMemoryWithPBit (USHORT
usAddr, UCHAR*baBuf, UCHAR*baPBit, USHORT*usLen) ;
```

**Description** Read out the data of memory with its associated protect bit.

**Parameters**

[ IN ]	<u><i>usAddr</i></u>
	Start address to read. The range is from 0x00 to 0x3FF.

[ OUT ]	<u><i>baBuf</i></u>
	Buffer to store the datat.

[ OUT ]	<u><i>baPBit</i></u>
	Buffer to store the protect bits.

[ IN ]	<u><i>usLen</i></u>
	Size of buffer. Maximum number of data to read.

[ OUT ]	<u><i>usLen</i></u>
	Length of data actually returned in the buffer.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

**Example**

Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x8VerifyPSC

---

USHORT CTOS\_44x8VerifyPSC( UCHAR\*baBuf );

**Description** Verify PSC.

**Parameters** [ IN ] *baBuf*  
PSC value to verify, 2 bytes.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x6ReadData

---

USHORT CTOS\_44x6ReadData (UCHAR bAddr, UCHAR\*baBuf, USHORT\*usLen) ;

**Description** Read out the data of memory.

**Parameters**

[ IN ]	<u>bAddr</u>
	Start address to read.

[ OUT ]	<u>baBuf</u>
	Buffer to store the data of memory.

[ IN ]	<u>usLen</u>
	Size of buffer. Maximum number of data to read.

[ OUT ]	<u>usLen</u>
	Length of data actually returned in the buffer.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x6WriteData

---

```
USHORT CTOS_44x6WriteData (UCHAR bAddr, UCHAR*baBuf, USHORT usLen);
```

**Description** Write data to memory.

**Parameters** [ IN ] *bAddr*  
Start address to write.

[ IN ] *baBuf*  
Data to write.

[ IN ] *usLen*  
Length of data to write.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x6Reload

---

USHORT CTOS\_44x6Reload(BOOL fCounterBak, BYTE bBitNum) ;

**Description** Write one bit in higher counter stage and reload the lower counter stage.

**Parameters** [ IN ] *fCounterBak*  
Whether to operate in counter backup bit mode.

- *d\_TRUE*

Enable.

- *d\_FALSE*

Disable.

[ IN ] *bBitNum*  
Bit address in counter stage to write.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x6VerifyTSC

---

USHORT CTOS\_44x6VerifyTSC( UCHAR\*baBuf ) ;

**Description** Verify TSC.

**Parameters** [ IN ] *baBuf*  
TSC value to verify, 3 bytes.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_44x6Auth

---

```
USHORT CTOS_44x6Auth(BOOL fExtAuth, BYTE bKey, BYTE
bNumClk, BYTE*baChallenge, BYTE*baRepBits);
```

**Description** Perform the card authentication.

**Parameters** [ IN ] *fExtAuth*

Whether to perform extend authentication.

- *d\_TRUE*

Extend authentication.

- *d\_FALSE*

Standard authentication.

[ IN ] *bKey*

Key to be used.

= 0x6E : key 1

= 0x6F : key 2

[ IN ] *bNumClk*

Number of clock pulse for computing each response bit.

[ IN ] *baChallenge*

Challenge data, 6 bytes.

[ OUT ] *baRepBits*

Response data of authentication.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

**Example**

Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_SLE44x4ReadMemory

---

USHORT CTOS\_SLE44x4ReadMemory(BYTE bAddr, UCHAR\*baBuf, BYTE\*bLen);

**Description** Read out the data of memory.

**Parameters**

[ IN ]	<u>bAddr</u>
Start address to read. The range is from 0x00 to 0x33.	

[ OUT ]	<u>baBuf</u>
Buffer to store the data of memory.	

[ IN ]	<u>bLen</u>
Size of buffer. Maximum number of data to read.	

[ OUT ]	<u>bLen</u>
Length of data actually returned in the buffer.	

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

## CTOS\_SLE44x4VerifyUserCode

---

USHORT CTOS\_SLE44x4VerifyUserCode (UCHAR\*baBuf) ;

**Description** Verify User Code.

**Parameters** [ IN ] *baBuf*  
User Code value to verify, 2 bytes.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

## CTOS\_SLE44x4VerifyMemoryCode

---

USHORT CTOS\_SLE44x4VerifyMemoryCode (UCHAR\**baBuf*) ;

**Description** Verify Memory Code.

**Parameters** [ IN ] *baBuf*  
Memory Code value to verify, 4 bytes.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

## CTOS\_SLE44x4WriteMemory

---

USHORT CTOS\_SLE44x4WriteMemory(BYTE bAddr, UCHAR\*baBuf, BYTE bLen);

**Description** Write the data of memory to zero. If the input bit is 0, the corresponding bit in the memory will be written to 0, otherwise the value of bit will be left unchanged.

**Parameters**

[ IN ]	<i>bAddr</i>
	Start address to write. The range is from 0x00 to 0x33.

[ IN ]	<i>baBuf</i>
	Data to write.

[ IN ]	<i>bLen</i>
	Length of data to write.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

## CTOS\_SLE44x4EraseMemory

---

```
USHORT CTOS_SLE44x4EraseMemory(BYTE bAddr, BYTE bLen);
```

**Description** Erase the data of memory to 0xFF.

**Parameters**

[ IN ]	<u>bAddr</u>
	Start address to erase. The range is from 0x00 to 0x33.
[ IN ]	<u>bLen</u>
	Length of data to erase.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

## CTOS\_SLE44x4BlowFuse

---

```
USHORT CTOS_SLE44x4BlowFuse(void);
```

**Description**      Blow the fuse to change security level.

**Parameters**      None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

## CTOS\_SLE44x4EnterTestMode

---

```
USHORT CTOS_SLE44x4EnterTestMode(void);
```

**Description** Enter test mode. This function is only valid before blowing the fuse.

**Parameters** None

Return Value	Constants	Value
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

## CTOS\_SLE44x4ExitTestMode

---

```
USHORT CTOS_SLE44x4ExitTestMode (void);
```

**Description** Exit test mode. This function is only valid prior before blowing the fuse.

**Parameters** None

Return Value	Constants	Value
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

## **CTOS\_GPM896ReadMemory**

---

USHORT CTOS\_GPM896ReadMemory (BYTE *bAddr*, UCHAR\**baBuf*, BYTE\**bLen*) ;

**Description**      Read out the data of memory.

**Parameters**      [ IN ]      *bAddr*  
                           usAddr Start address to read. The range is from 0x00 to  
                           0x6F.

                          [ OUT ]    *baBuf*  
                           Buffer to store the data of memory.

                          [ IN ]     *bLen*  
                           Size of buffer. Maximum number of data to read.

                          [ OUT ]    *bLen*  
                           Length of data actually returned in the buffer.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

## **CTOS\_GPM896VerifySecurityCode**

---

USHORT CTOS\_GPM896VerifySecurityCode (UCHAR\**baKey*) ;

**Description** Verify Security Code.

**Parameters** [ IN ] *baKey*  
Security Code value to verify, 2 bytes.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

## **CTOS\_GPM896WriteMemory**

---

USHORT CTOS\_GPM896WriteMemory(BYTE bAddr, UCHAR\*baBuf, BYTE bLen);

**Description** Write the data of memory to zero. If the input bit is 0, the corresponding bit in the memory will be written to 0, otherwise the value of bit will be left unchanged.

**Parameters** [ IN ] *bAddr*  
Start address to write. The range is from 0x00 to 0x6F.

[ IN ] *baBuf*  
Data to write.

[ IN ] *bLen*  
Length of data to write.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

## CTOS\_GPM896EraseMemory

---

```
USHORT CTOS_GPM896EraseMemory(BYTE bAddr, BYTE bLen);
```

**Description** Erase the data of memory to 0xFF.

**Parameters**

[ IN ]	<u>bAddr</u>
	Start address to erase. The range is from 0x00 to 0x6F.
[ IN ]	<u>bLen</u>
	Length of data to erase.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

## CTOS\_GPM896EraseArea1

---

```
USHORT CTOS_GPM896EraseArea1 (UCHAR*baKey) ;
```

**Description** Verify the Erase Key of Application Zone 1(AZ1) and erase AZ1.

**Parameters** [ IN ] *baKey*  
Erase Key value to verify, 6 bytes.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

## CTOS\_GPM896EraseArea2

---

```
USHORT CTOS_GPM896EraseArea2 (UCHAR*baKey) ;
```

**Description** Verify the Erase Key of Application Zone 2(AZ2) and erase AZ2.

**Parameters** [ IN ] *baKey*  
Erase Key value to verify, 4 bytes.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

## CTOS\_I2CCReset

---

```
USHORT CTOS_I2CCReset (UCHAR*baBuf, BYTE*bLen) ;
```

**Description** Reset I2C card.

**Parameters** [ OUT ] *baBuf*  
Buffer to store the response data from I2C card.

[ IN ] *bLen*  
Size of baBuf.

[ OUT ] *bLen*  
Length of response data.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_I2CReadMemory

---

```
USHORT CTOS_I2CReadMemory(BYTE bCmd, BYTE bNumAddr, USHORT usAddr, BOOL fDummyRead, UCHAR*baBuf, USHORT*usLen);
```

**Description** Read out the data of I2C card memory.

**Parameters** [ IN ] *bCmd*  
Reading command of the I2C card.

[ IN ] *bNumAddr*  
Number of the address bytes. The range is from 0 to 2.

[ IN ] *usAddr*  
Start address to read.

[ IN ] *fDummyRead*  
Whether dummy write is required for reading the memory.

- *d\_FALSE*  
Dummy write is not required.

• *d\_TRUE*  
Dummy write is required.

[ OUT ] *baBuf*  
Buffer to store the data of memory.

[ IN ] *usLen*  
Size of buffer. Maximum number of data to read.

[ OUT ] *usLen*  
Length of data actually returned in the buffer.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Memory Card error codes</a>	15xxh

**Example**

Please refer to example in appendix [B.11 Memory Card](#)

## CTOS\_I2CWriteMemory

---

```
USHORT CTOS_I2CWriteMemory(BYTE bCmd, BYTE bNumAddr, USHORT
usAddr, UCHAR*baBuf, USHORT usLen);
```

**Description** Write data to I2C card memory.

**Parameters** [ IN ] *bCmd*  
Writing command of the I2C card.

[ IN ] *bNumAddr*  
Number of the address bytes. The range is from 0 to 2.

[ IN ] *usAddr*  
Start address to write.

[ OUT ] *baBuf*  
Data to write.

[ IN ] *usLen*  
Length of data to write.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Memory Card error codes</a>	15xxh

**Example** Please refer to example in appendix [B.11 Memory Card](#)

## 2.22. Magnetic Stripe Reader Functions

- CTOS\_MSRead
- CTOS\_MSReadEx
- CTOS\_MSRGetLastErr

### Magnetic Stripe Reader Error Codes

<b>Constants</b>	<b>Value</b>
d_MSR_NO_SWIPE	1201h
d_MSR_TRACK_ERROR	1202h

## CTOS\_MSRead

---

```
USHORT CTOS_MSRead(BYTE*baTk1Buf, USHORT*pusTk1Len, BYTE*baTk2Buf,  
USHORT*pusTk2Len, BYTE*baTk3Buf, USHORT*pusTk3Len);
```

**Description** Read the content of magnetic stripe card.

**Parameters** [ OUT ] *baTk1Buf*  
Buffer to store data of track 1.

[ IN ] *pusTk1Len*  
Size of baTk1Buf.

[ OUT ] *pusTk1Len*  
Length of data of track 1.

[ OUT ] *baTk2Buf*  
Buffer to store data of track 2.

[ IN ] *pusTk2Len*  
Size of baTk2Buf.

[ OUT ] *pusTk2Len*  
Length of data of track 2.

[ OUT ] *baTk3Buf*  
Buffer to store data of track 3.

[ IN ] *pusTk3Len*  
Size of baTk3Buf.

[ OUT ] *pusTk3Len*  
Length of data of track 3.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d MSR_NO_SWIPE	1201h
d MSR_TRACK_ERROR	1202h

**Example**

Please refer to example in appendix [B.13 MSReader](#)

## CTOS\_MSReadEx

---

```
USHORT CTOS_MSReadEx(BYTE bTkMask, BYTE*baTk1Buf, USHORT*pusTk1Len,  
BYTE*baTk2Buf, USHORT*pusTk2Len, BYTE*baTk3Buf, USHORT*pusTk3Len);
```

**Description** Read the content of desire track of magnetic stripe card.

**Parameters** [ IN ] bTkMask  
Bitwise-OR of the track wish to read.

- d\_MK\_MSR\_TK1
- d\_MK\_MSR\_TK2
- d\_MK\_MSR\_TK3

[ OUT ] baTk1Buf  
Buffer to store data of track 1.

[ IN ] pusTk1Len  
Size of baTk1Buf.

[ OUT ] pusTk1Len  
Length of data of track 1.

[ OUT ] baTk2Buf  
Buffer to store data of track 2.

[ IN ] pusTk2Len  
Size of baTk2Buf.

[ OUT ] pusTk2Len  
Length of data of track 2.

[ OUT ] baTk3Buf  
Buffer to store data of track 3.

[ IN ] *pusTk3Len*

Size of baTk3Buf.

[ OUT ] *pusTk3Len*

Length of data of track 3.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_MSR_NO_SWIPE	1201h
d_MSR_TRACK_ERROR	1202h

**Example**

Please refer to example in appendix [B.13 MSReader](#)

**Note**

This function will return the data of track according to the bTkMask.  
When the track is not indicated in the bTkMask,  
1. If there is data already read from track, the content will not be returned.  
2. if there is error during decoding the track, it will not be taken into consideration for return value, which means the return code will be still d\_OK if the error happens in the track which is not indicated by bTkMask.

## CTOS\_MSRGetLastErr

---

```
USHORT CTOS_MSRGetLastErr (BYTE*baTk1Err, BYTE*baTk2Err,
BYTE*baTk3Err);
```

**Description** Get the status of each track of last swipe.

**Parameters**

[ OUT ]	<u><i>baTk1Err</i></u>
	Status of track 1 of last swipe.

[ OUT ]	<u><i>baTk2Err</i></u>
	Status of track 2 of last swipe.

[ OUT ]	<u><i>baTk3Err</i></u>
	Status of track 3 of last swipe.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_MSR_NO_SWIPE	1201h
	d_MSR_TRACK_ERROR	1202h

**Example** Please refer to example in appendix [B.13 MSReader](#)

**Note** The error code returned for the last swipe for each track could be one of the following value. The error code will not be updated until next swipe,

- ***d\_MSR\_SUCCESS***  
MSR data is successfully decoded.
  
- ***d\_MSR\_STX\_NOT\_FOUND***  
STX is not found.
  
- ***d\_MSR\_NO\_DATA***  
No data is detected.

- ***d\_MSR\_BUF\_OVERFLOW***  
Buffer passed in is not sufficient to store the data decoded.
- ***d\_MSR\_ETX\_NOT\_FOUND***  
ETX is not found.
- ***d\_MSR\_LRC\_NOT\_FOUND***  
LRC is not found.
- ***d\_MSR\_LRC\_ERR***  
LRC is incorrect.
- ***d\_MSR\_UNKNOWN\_CHAR***  
An unknown character is decoded.

## 2.23.Contactless Reader

### General

- CTOS\_CLInitComm
- CTOS\_CLCloseComm
- CTOS\_CLInit
- CTOS\_CLPowerOn
- CTOS\_CLPowerOff
- CTOS\_CLVerGet
- CTOS\_ReadRC
- CTOS\_WriteRC

### ISO 14443-3 Type A

- CTOS\_REQA
- CTOS\_WUPA
- CTOS\_ANTIA
- CTOS\_SELECTA
- CTOS\_CLTypeAHalt
- CTOS\_CLTypeAActiveFromIdle
- CTOS\_CLTypeAActiveFromHalt

### Mifare

- CTOS\_MifareREADE2
- CTOS\_MifareWRITEE2
- CTOS\_MifareLOADKEY
- CTOS\_MifareLOADKEYE2
- CTOS\_MifareAUTHEx
- CTOS\_MifareAUTH2Ex
- CTOS\_MifareREADBLOCK
- CTOS\_MifareWRITEBLOCK
- CTOS\_MifareINCREASE
- CTOS\_MifareDECREASE
- CTOS\_MifareRESTORE
- CTOS\_MifareTRANSFER

### ISO 14443-3 Type B

- CTOS\_REQB
- CTOS\_WUPB
- CTOS\_ATTRIB
- CTOS\_HALTB
- CTOS\_CLTypeBActive
- CTOS\_CLTypeBActiveEx

### **ISO 14443-4 T=CL**

- CTOS\_CLRATS
- CTOS\_CLRATSEx
- CTOS\_CLAPDU
- CTOS\_CLDESELECT
- CTOS\_CLNAKPOLL

### **Transparent CL Access**

- CTOS\_TclTransparent
- CTOS\_TclTransparentEx
- CTOS\_TclSetTimeout

### **ISO 18092 Felica**

- CTOS\_FelicaPolling
- CTOS\_FelicaReadWrite

### **Peripheral**

- CTOS\_CLLEDSet
- CTOS\_CLSound

### **Contactless Reader Error Codes**

<b><i>Constants</i></b>	<b><i>Value</i></b>
d_CL_OK	0000h
d_CL_HAL_ERROR	8301h
d_CL_TX_TIMEOUT	8302h
d_CL_RX_TIMEOUT	8303h
d_CL_CRC_INCORRECT	8304h
d_CL_BUF_NOT_ENOUGH	8305h
d_CL_INIT_COM_ERROR	8331h
d_CL_INVALID_HANDLE	8332h
d_CL_INVALID_PARA	8333h
d_CL_COMM_FAIL	8334h
d_CL_RSP_LEN_ERROR	8335h
d_CL_RSP_CMD_ERROR	8336h
d_CL_OUT_OF_DATA_LEN	8337h
d_CL_SN_INCORRECT	8338h
d_CL_RSP_DATA_LEN_ERROR	8339h
d_CL_RSP_ID_ERROR	833Ah
d_CL_BASE_INDEX_ERROR	833Bh

d_CL_COM_ALREADY_OPEN	833Ch
d_CL_CMD_ID_INCORRECT	8345h
d_CL_HOST_ID_INCORRECT	8346h
d_CL_READER_ID_INCORRECT	8347h
d_CL_CBN_ERROR	8348h
d_CL_UNKNOWN_PROTOCOL_MODE	8349h
d_CL_SET_DATETIME_ERROR	834Ah
d_CL_PENDING	8350h
d_CL_TX_CANCEL	8351h
d_CL_TCL_ABORT	8381h
d_CL_TCL_TIMEOUT	8382h
d_CL_TCL_ERR	8383h
d_CL_NAK	8384h
d_CL_NOT_IMPLEMENT	839Ch
d_CL_WRONG_PARAMETER_VALUE	83C4h
d_CL RESPOND_TOO_EARLY	83C8h
d_CL SAME_BAUDRATE_REQUIRED	83C9h
d_CL BAUDRATE_NOT_SUPPORTED	83CAh
d_CL_CASC_LEV_EX	83CCh
d_CL_CARD_CODING_ERR	83E1h
d_CL_CODINGERR	83E2h
d_CL_NO_BITWISE_ANTICOLL	83E4h
d_CL_ACCESS_TIMEOUT	83E5h
d_CL_INTERFACE_ERR	83E6h
d_CL_INIT_ERR	83E7h
d_CL_COLL_ERR	83E8h
d_CL_UNKNOWN_COMMAND	83E9h
d_CL_ACCESS_ERR	83EAh
d_CL_FRAMING_ERR	83EBh
d_CL_OVFL_ERR	83EDh
d_CL_WRITE_ERR	83F1h
d_CL_BYTECOUNT_ERR	83F4h
d_CL_BITCOUNT_ERR	83F5h
d_CL_NOTAUTH_ERR	83F6h
d_CL_KEY_ERR	83F7h

d_CL_SERNR_ERR	83F8h
d_CL_CODE_ERR	83FAh
d_CL_PARITY_ERR	83FBh
d_CL_AUTH_ERR	83FCh
d_CL_CRC_ERR	83FEh
d_CL_UNDEFINE_ERR	83FFh

## CTOS\_CLInitComm

---

```
USHORT CTOS_CLInitComm(ULONG uiBaudRate);
```

**Description** Initialize contactless reader communication channel.

**Parameters** [IN] *uiBaudRate*

Baudrate.

- *d\_COMM\_DEFAULT\_BAUDRATE*

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_CLCloseComm

---

```
USHORT CTOS_CLCloseComm(void);
```

**Description** Close contactless reader communication channel.

**Parameters** None.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_CLInit

---

```
USHORT CTOS_CLInit(void);
```

**Description** Initial or reset the contactless reader.

**Parameters** None.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_CLPowerOn

---

```
USHORT CTOS_CLPowerOn(void);
```

**Description** Turn on the power of contactless reader.

**Parameters** None.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_CLPowerOff

---

```
USHORT CTOS_CLPowerOff(void);
```

**Description** Turn off the power of contactless reader.

**Parameters** None.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_CLVerGet

---

```
USHORT CTOS_CLVerGet(ULONG* ulVer);
```

**Description** Get the version of contactless module.

**Parameters** [ OUT ] *ulVer*  
Buffer to store the version of contactless module.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_ReadRC

---

```
USHORT CTOS_ReadRC(UCHAR bAddr, UCHAR* bValue);
```

**Description**      Read register value from the contactless interface chip.

**Parameters**      [IN]      *bAddr*  
                            Address of register.

    [ OUT ] *bValue*  
                    Value of the register.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_WriteRC

---

```
USHORT CTOS_WriteRC( UCHAR bAddr, UCHAR bValue );
```

**Description** Write value to the register of contactless interface chip.

**Parameters** [ IN ] *bAddr*  
Address of register.

[ IN ] *bValue*  
Value to write to register.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_REQA

---

```
USHORT CTOS_REQA (UCHAR* baATQA) ;
```

**Description** Execute Type A Request command.

**Parameters** [ OUT ] [baATQA](#)  
ATQA of the selected PICC.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<u><a href="#">Contactless reader error codes</a></u>	83xxh

## CTOS\_WUPA

---

```
USHORT CTOS_WUPA (UCHAR* baATQA) ;
```

**Description** Execute Type A Wake Up command.

**Parameters** [ OUT ] [baATQA](#)  
ATQA of the selected PICC.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<u><a href="#">Contactless reader error codes</a></u>	83xxh

## CTOS\_ANTIA

---

USHORT CTOS\_ANTIA( UCHAR bSelCode, UCHAR\* baSNR, UCHAR bCnt );

**Description** Issue a Type A ANTICOLLISION command to the PICC.

<b>Parameters</b>	[ IN ] <u>bSelCode</u> SEL code which defines the cascade level of UID to issue. (Can be one of the following value, 0x93.0x95 or 0x97)
	[ IN ] <u>baSNR</u> Preference UID sequence to perform the ANTICOLLISION.
	[ OUT ] <u>baSNR</u> UID CLn, 4 bytes of ANTICOLLISION result.(excluding BCC).
	[ IN ] <u>bCnt</u> Number of preference UID in bits. Normally set to 0.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_SELECTA

---

USHORT CTOS\_SELECTA(UCHAR bSelCode, UCHAR\* baSNR, UCHAR\* bSAK) ;

**Description** Issue a type A SELECT command to the PICC.

**Parameters**

[ IN ]	<u>bSelCode</u>
	SEL code which defines the cascade level of UID to issue. (Can be one of the following value, 0x93, 0x95 or 0x97)

[ OUT ]	<u>baSNR</u>
	CSN of the selected PICC. The maximum value is 10 bytes. 4 bytes of UID CLn to select. Normally input with the result of ANTICOLLISION.

[ OUT ]	<u>bSAK</u>
	Select Acknowledge of SELECT command.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_CLTypeAHalt

---

```
USHORT CTOS_CLTypeAHalt(void);
```

**Description** Put a type A PICC into HALT state. This function will send out HALTA command as defined in ISO14443.

**Parameters** None.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Example** Please refer to example in appendix [B.25. T=CL](#)

## CTOS\_CLTypeAActiveFromIdle

---

```
USHORT CTOS_CLTypeAActiveFromIdle(BYTE bBaudRate, BYTE* baATQA, BYTE
*bSAK, BYTE* baCSN, BYTE* bCSNLen);
```

**Description** Activate a PICC from its IDLE state. This command executes REQA, anti-collision loop, and SELECTA sequence for one of PICC in the field. After execution of Active\_IDLE command, the PICC goes from IDLE state to ACTIVE state, and is ready to enter ISO-14443-4 area.

<b>Parameters</b>	[ IN ] <u><i>bBaudRate</i></u> RFU (set to 0).
	[OUT] <u><i>baATQA</i></u> ATQA of the selected PICC.
	[ OUT ] <u><i>bSAK</i></u> SAK of the selected PICC.
	[ OUT] <u><i>baCSN</i></u> CSN of the selected PICC. The maximum value is 10 bytes.
	[OUT] <u><i>bCSNLen</i></u> The length of the CSN of the selected PICC, the length value should be 4, 7 or 10.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Example** Please refer to example in appendix [B.23. Mifare](#) or [B.25. T=CL](#)

## CTOS\_CLTypeAActiveFromHalt

---

```
USHORT CTOS_CLTypeAActiveFromHalt(BYTE bBaudRate, BYTE* baATQA, BYTE
*bSAK, BYTE* baCSN, BYTE bCSNLen);
```

<b>Description</b>	Activate a PICC from its HALT state. This command executes WUPA, and SELECTA sequence for the PICC with the same serial number comparing to the input CSN. After execution of Active_HALT command, the PICC goes from HALT state to ACTIVE state, and is ready to enter ISO-14443-4 area.
<b>Parameters</b>	<p>[ IN ] <u><i>bBaudRate</i></u> RFU (set to 0).</p> <p>[OUT] <u><i>baATQA</i></u> ATQA of the selected PICC.</p> <p>[OUT] <u><i>bSAK</i></u> SAK of the selected PICC.</p> <p>[ IN ] <u><i>baCSN</i></u> CSN of the PICC about to activate.</p> <p>[ IN ] <u><i>bCSNLen</i></u> The length of the CSN of the selected PICC, the length value should be 4, 7 or 10.</p>

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

<b>Example</b>	Please refer to example in appendix <a href="#">B.25. T=CL</a>
----------------	--

## CTOS\_MifareREADE2

---

```
USHORT CTOS_MifareREADE2 (UCHAR bAddrMSB, UCHAR bAddrLSB, UCHAR
bNumByte, UCHAR* baBuf);
```

**Description** Read the contents from the EEPROM of RC531.

**Parameters** [ IN ] bAddrMSB  
The MSB address of data in EEPROM.

[ IN ] bAddrLSB  
The LSB address of data in EEPROM.

[ IN ] bNumByte  
The length of data to read.

[ OUT ] bAddrMSB  
The data read from EEPROM of RC531.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Example** Please refer to example in appendix [B.24. Mifare EEPROM](#)

## CTOS\_MifareWRITEE2

---

```
USHORT CTOS_MifareWRITEE2( UCHAR bAddrMSB, UCHAR bAddrLSB, UCHAR
bNumByte, UCHAR* baBuf );
```

**Description** Write the data into the EEPROM of RC531.

**Parameters** [ IN ] bAddrMSB  
The MSB address of data in EEPROM.

[ IN ] bAddrLSB  
The LSB address of data in EEPROM.

[ IN ] bNumByte  
The length of data to write.

[ IN ] baBuf  
The data will be written to EEPROM of RC531.

Return Value	<b>Constants</b>	<b>Value</b>
d_OK		0000h
<a href="#">Contactless reader error codes</a>		83xxh

**Example** Please refer to example in appendix [B.24. Mifare EEPROM](#)

## CTOS\_MifareLOADKEY

---

```
USHORT CTOS_MifareLOADKEY (UCHAR* baKey) ;
```

**Description** Put key value into the internal key buffer of RC531.

**Parameters** [ IN ] *baKey*

Key value put into the internal key buffer of RC531.

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

**Example** Please refer to example in appendix [B.23. Mifare](#)

## CTOS\_MifareLOADKEYE2

```
USHORT CTOS_MifareLOADKEYE2 (UCHAR bAddrMSB, UCHAR bAddrLSB) ;
```

**Description**      Read the key from the EEPROM of RC531 and put it into the internal key buffer of RC531.

**Parameters**      [ IN ]      bAddrMSB  
                          The MSB address of key in EEPROM.

                          [ IN ]      bAddrLSB  
                          The LSB address of key in EEPROM.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Example**      Please refer to example in appendix [B.24. Mifare EEPROM](#)

## CTOS\_MifareAUTHEx

---

```
USHORT CTOS_MifareAUTHEx(UCHAR baKeyType, UCHAR bBlockNr, UCHAR*
baCardSN, UCHAR bCardSNLen);
```

**Description** Perform the CRYPTO1 (Mifare Classic) card authentication with the key in the internal key buffer. If authentication is successful, RC531 could operate the blocks in the same sector of the authenticated block.

<b>Parameters</b>	[ IN ] <u><i>baKeyType</i></u>
	The type off kye. = 0x60: Key A = 0x61: Key B
	[ IN ] <u><i>bBlockNr</i></u>
	The number of the block to be authenticated. (From #0 To #63) 16 sectors with 4 blocks of 16 bytes each. Blocks in the same sector would be authenticated with the same key stored in sector trailer (Block#3, #7, ..., #63).
	[ IN ] <u><i>baCardSN</i></u>
	Card serial number.
	[ IN ] <u><i>bCardSNLen</i></u>
	The length of the card serial number.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_MifareAUTH2Ex

---

```
USHORT CTOS_MifareAUTH2Ex (UCHAR baKeyType, UCHAR bBlockNr, UCHAR*
baKey, UCHAR* baCardSN, UCHAR bCardSNLen);
```

**Description** Perform the CRYPTO1 (Mifare Classic) card authentication with input key. If authentication is successful, user could operate the blocks in the same sector of the authenticated block.

**Parameters**

[ IN ]	<u><i>baKeyType</i></u> The type off kye. = 0x60: Key A = 0x61: Key B
--------	--

[ IN ]	<u><i>bBlockNr</i></u> The number of the block to be authenticated. (From #0 To #63) 16 sectors with 4 blocks of 16 bytes each. Blocks in the same sector would be authenticated with the same key stored in sector trailer (Block#3, #7, ..., #63).
--------	---

[ IN ]	<u><i>baKey</i></u> 6 bytes of Mifare key to authenticate.
--------	---

[ IN ]	<u><i>baCardSN</i></u> Card serial number.
--------	---

[ IN ]	<u><i>bCardSNLen</i></u> Number of bytes of card serial number. Can be 4, 7 , or 10 bytes.
--------	---

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_MifareREADBLOCK

---

```
USHORT CTOS_MifareREADBLOCK(UCHAR bBlockNr,UCHAR* baBuf);
```

**Description**      Read the content of the specific block (16 bytes) from the PICC.

**Parameters**      [ IN ]      *bBlockNr*  
                          The block number.

[OUT]      *baBuf*  
                          The data read from the block of PICC.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Example**      Please refer to example in appendix [B.23. Mifare](#)

## CTOS\_MifareWRITEBLOCK

---

```
USHORT CTOS_MifareWRITEBLOCK( UCHAR bBlockNr, UCHAR* baBuf );
```

**Description** Write data to the specific block (16 bytes) of the PICC.

**Parameters** [ IN ] *bBlockNr*  
The block number.

[ IN ] *baBuf*  
The data will be written to the block of PICC.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Example** Please refer to example in appendix [B.23. Mifare](#)

## CTOS\_MifareINCREASE

---

```
USHORT CTOS_MifareINCREASE (UCHAR bBlockNr, UCHAR* bValue);
```

**Description** Increases the value of the block and stores the result into the internal data register.

**Parameters** [ IN ] *bBlockNr*  
The block number.

[ IN ] *bValue*  
The amount to be added in hexadecimal.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Example** Please refer to example in appendix [B.23. Mifare](#)

## CTOS\_MifareDECREASE

---

```
USHORT CTOS_MifareDECREASE (UCHAR bBlockNr, UCHAR* bValue);
```

**Description** Decreases the value of the block and stores the result into the internal data register.

**Parameters** [ IN ] *bBlockNr*  
The block number.

[ IN ] *bValue*  
The amount to be reduced in hexadecimal.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Example** Please refer to example in appendix [B.23. Mifare](#)

## CTOS\_MifareRESTORE

---

```
USHORT CTOS_MifareRESTORE( UCHAR bBlockNr );
```

**Description**      Read the contents of the block and stores into the internal data register.

**Parameters**      [ IN ]      *bBlockNr*  
                            The block number.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_MifareTRANSFER

---

```
USHORT CTOS_MifareTRANSFER (UCHAR bBlockNr) ;
```

**Description** Write the contents of the internal data register into the block.

**Parameters** [ IN ] *bBlockNr*  
The block number.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Example** Please refer to example in appendix [B.23. Mifare](#)

## CTOS\_REQB

---

USHORT CTOS\_REQB (UCHAR bAFI, UCHAR\* baATQB, UCHAR\* bATQBLen);

**Description** Execute Type B Request command.

<b>Parameters</b>	[ IN ] <u>bAFI</u>
	Select AFI (Application Family Identifier) to pre-select PICC. Only the PICC's matching the AFI will answer the REQB/WUPB command. Normally we use 0 for selecting All families and sub-families.
	[OUT] <u>baATQB</u>
	ATQB of the selected PICC.
	[OUT] <u>bATQBLen</u>
	Length of the ATQB in byte.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_WUPB

---

USHORT CTOS\_WUPB (UCHAR bAFI, UCHAR\* baATQB, UCHAR\* bATQBLen);

**Description** Execute Type B Wake Up command.

**Parameters**

[ IN ]	<u>bAFI</u>
Select AFI (Application Family Identifier) to pre-select PICC. Only the PICC's matching the AFI will answer the REQB/WUPB command. Normally we use 0 for selecting All families and sub-families.	

[ OUT ]	<u>baATQB</u>
ATQB of the selected PICC.	

[ OUT ]	<u>bATQBLen</u>
Length of the ATQB in byte.	

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_ATTRIB

---

```
USHORT CTOS_ATTRIB (UCHAR* baPUPI, UCHAR* bRep) ;
```

**Description** Select the PICC. PICC goes to the ACTIVE state after performing the ATTRIBcommand.

**Parameters** [ OUT ] *baPUPI*  
Pseudo-Unique PICC Identifier from PICC.

[OUT] *bRep*  
ATTRIB response.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_HALTB

---

```
USHORT CTOS_HALTB(BYTE *baPUPI);
```

**Description** Put the PICC of Type B in the HALT state.

**Parameters** [IN] *baPUPI*  
Pseudo-Unique PICC Identifier from PICC.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_CLTypeBActive

---

```
USHORT CTOS_CLTypeBActive(BYTE* baPUPI);
```

**Description** Activate a Type B PICC. After this command, the PICC directly enter ISO14443-4 T=CL domain, no need to call TypeRATS.

**Parameters** [OUT] *baPUPI*  
Pseudo-Unique PICC Identifier from PICC.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

## **CTOS\_CLTypeBActiveEx**

```
USHORT CTOS_CLTypeBActiveEx(BYTE* baPUPI, UCHAR bAFI, BOOL fWUPB, UCHAR bFSDI, UCHAR bNumAntiColSlot, USHORT usMaxRate);
```

<b>Description</b>	Activate a Type B PICC. After this command, the PICC directly enter ISO14443-4 T=CL domain, no need to call TypeRATS. All related input parameter please refer to descripton as below.
<b>Parameters</b>	<p>[OUT] <u><i>baPUPI</i></u> Pseudo-Unique PICC Identifier from PICC.</p> <p>[ IN ] <u><i>bAFI</i></u> Select AFI (Application Family Identifier) to pre-select PICC. Only the PICC's matching the AFI will answer the REQB/WUPB command. Normally we use 0 for selecting All families and sub-families.</p> <p>[ IN ] <u><i>fWUPB</i></u> Select whether using WUPB or REQB to poll for PICC. Using REQB can only wake up PICC in IDLE state, but using WUPB can wake up PICC IN IDLE state or HALT state.</p> <ul style="list-style-type: none"> <li>• <u><i>d_USE_REQB</i></u></li> <li>• <u><i>d_USE_WUPB</i></u></li> </ul> <p>[ IN ] <u><i>bFSDI</i></u> Select the maximum number of data can be transmitted to the PICC.</p> <ul style="list-style-type: none"> <li>• <u><i>d_FSDI_16</i></u></li> <li>• <u><i>d_FSDI_24</i></u></li> <li>• <u><i>d_FSDI_32</i></u></li> <li>• <u><i>d_FSDI_40</i></u></li> <li>• <u><i>d_FSDI_48</i></u></li> <li>• <u><i>d_FSDI_64</i></u></li> </ul>

- *d\_FSDI\_96*
- *d\_FSDI\_128*
- *d\_FSDI\_256*

[ IN ] *bNumAntiColSlot*

Select the number of slot when performing the Slot-Marker anti-collision method.

- *d\_NUM\_SLOT\_x1*
- *d\_NUM\_SLOT\_x2*
- *d\_NUM\_SLOT\_x4*
- *d\_NUM\_SLOT\_x8*
- *d\_NUM\_SLOT\_x16*

[ IN ] *usMaxRate*

Select the maximum baudrate can be applied when communicating to the PICC.

- *d\_MAXRATE\_106*
- *d\_MAXRATE\_212*
- *d\_MAXRATE\_424*

#### Return Value

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_CLRATS

---

USHORT CTOS\_CLRATS (BYTE bAutoBR, BYTE\* baATS, USHORT\* bATSLen) ;

**Description** Perform the Request Answer To Select (RATS) defined in ISO14443-4. This command will send the RATS request, receive ATS from the PICC, parse the ATS, and setup the communication parameters of PICC.

**Parameters**

[ IN ]	<u>bAutoBR</u>
	RFU (set to 0).
	= 1 Perform PPS automatically after receiving correct ATS.
	= 0 Do not perform PPS, use the default baud rate (106k).

[OUT]	<u>baATS</u>
	Buffer to store ATS received from PICC.

[OUT]	<u>bATSLen</u>
	ATS length received from PICC.

Return Value	Constants	Value
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Example** Please refer to example in appendix [B.25. T=CL](#)

## CTOS\_CLRATSEx

---

```
USHORT CTOS_CLRATSEx(BYTE bAutoBR, BYTE* baATS, USHORT* bATSLen,
USHORT usMaxRate, BYTE bFSDI);
```

**Description** Perform the Request Answer To Select (RATS) defined in ISO14443-4. This command will send the RATS request, receive ATS from the PICC, parse the ATS, and setup the communication parameters of PICC.

**Parameters**

[ IN ]	<u><i>bAutoBR</i></u>
	RFU (set to 0).
	= 1 Perform PPS automatically after receiving correct ATS.
	= 0 Do not perform PPS, use the default baud rate (106k).

[ OUT ]	<u><i>baATS</i></u>
	Buffer to store ATS received from PICC.

[ OUT ]	<u><i>bATSLen</i></u>
	ATS length received from PICC.

[ IN ]	<u><i>usMaxRate</i></u>
	Select the maximum baudrate can be applied when communicating to the PICC.
	<ul style="list-style-type: none"> <li>• <b><i>d_MAXRATE_106</i></b></li> <li>• <b><i>d_MAXRATE_212</i></b></li> <li>• <b><i>d_MAXRATE_424</i></b></li> </ul>

[ IN ]	<u><i>bFSDI</i></u>
	Select the maximum number of data can be transmitted to the PICC.
	<ul style="list-style-type: none"> <li>• <b><i>d_FSDI_16</i></b></li> <li>• <b><i>d_FSDI_24</i></b></li> <li>• <b><i>d_FSDI_32</i></b></li> </ul>

- *d\_FSDI\_40*
- *d\_FSDI\_48*
- *d\_FSDI\_64*
- *d\_FSDI\_96*
- *d\_FSDI\_128*
- *d\_FSDI\_256*

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#"><u>Contactless reader error codes</u></a>	83xxh

## CTOS\_CLAPDU

---

```
USHORT CTOS_CLAPDU(BYTE* baSBuf, USHORT usSLen, BYTE* baRBuf, USHORT*
usRLen);
```

**Description** Send the APDU to the PICC and receive the response from PICC according to the T=CL protocol defined in ISO14443-4. This command will send the APDU with the communication parameters retrieved from RATS.

**Parameters** [ IN ] *baSBuf*  
Buffer storing the APDU to transmit to PICC.

[ IN ] *usSLen*  
Length of APDU to transmit to PICC.

[ OUT ] *baRBuf*  
Buffer to store the response received from PICC.

[ IN ] *usRLen*  
The length wants to receive.

[ OUT ] *usRLen*  
The length wants to receive.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Note** CTOS\_CLAPDU() will deal with T=CL protocol as described in the ISO14443-4 and EMV Contactless Level1 specification. For example, while some error happened during the transmission, the retry will be made inside the CTOS\_CLAPDU(), and the timeout is automatically calculated and managed internally.

**Example**

Please refer to example in appendix [B.25. T=CL](#)

## CTOS\_CLDESELECT

---

```
USHORT CTOS_CLDESELECT(void);
```

**Description** Exit the ISO14443-4 state and enter the HALT state.

**Parameters** None.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

**Example** Please refer to example in appendix [B.25. T=CL](#)

## CTOS\_CLNAKPOLL

---

```
USHORT CTOS_CLNAKPOLL(void);
```

**Description** Issue a T=CL NAK command to the PICC. This function is to prevent some PICC from power saturation.

**Parameters** None.

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_TclTransparent

---

```
USHORT CTOS_TclTransparent(BYTE* baSBuf, USHORT usSLen, BYTE*
baRBuf, USHORT* usRLen);
```

**Description** Send raw data directly to the PICC.

**Parameters**

[ IN ]	<u><i>baSBuf</i></u>
	Data to send.

[ IN ]	<u><i>usSLen</i></u>
	Length of data to send.

[OUT]	<u><i>baRBuf</i></u>
	Buffer to store the response from PICC.

[ IN ]	<u><i>usRLen</i></u>
	Size of baRBuf.

[ OUT ]	<u><i>usRLen</i></u>
	Length of response from PICC.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

**Note** CTOS\_TclTransparent() will just send the content to the card, and receive the response from the card.

## CTOS\_TclTransparentEx

---

```
USHORT CTOS_TclTransparentEx(BYTE* baSBuf, USHORT usSLen, BYTE*
baRBuf, USHORT* usRLen, BOOL fTxCrcEnable, BOOL fRxCrcEnable);
```

**Description** Send raw data directly to the PICC.

**Parameters** [ IN ] *baSBuf*  
Data to send.

[ IN ] *usSLen*  
Length of data to send.

[ OUT ] *baRBuf*  
Buffer to store the response from PICC.

[ IN ] *usRLen*  
Size of baRBuf.

[ OUT ] *usRLen*  
Length of response from PICC.

[ IN ] *fTxCrcEnable*  
Whether to calculate and pad the CRC when sending data to  
PICC.

[ IN ] *fRxCrcEnable*  
Whether to regard the last byte of data read from PICC as  
CRC and calculate its validity.

Return Value	<b>Constants</b>	<b>Value</b>
d_OK		0000h
<a href="#">Contactless reader error codes</a>		83xxh

## CTOS\_TclSetTimeout

---

```
USHORT CTOS_TclSetTimeout(ULONG ultTimeout);
```

**Description** Set the timeout for TclTransparent and TclTransparentEx function.

**Parameters** [ IN ] *ulTimeout*  
Timeout in millisecond.

Return Value	<i>Constants</i>	<i>Value</i>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_FelicaPolling

---

USHORT CTOS\_FelicaPolling(UCHAR\* IDm, UCHAR\* PMm);

**Description** Detect if any Felica card on the landing field.

**Parameters** [OUT] *IDm*  
Manufacture ID.

[OUT] *PMm*  
Manufacture Parameter.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_FelicaReadWrite

---

```
USHORT CTOS_FelicaReadWrite( UCHAR* TxBuf, USHORT TxLen, UCHAR*
RxBuf, USHORT* RxLen, USHORT ultTimeout );
```

**Description** Write data to the Felica and read response from Felica.

**Parameters** [ IN ] TxBuf  
The write data.

[ IN ] TxLen  
Length of the TxBuf.

[ OUT ] RxBuf  
The return data.

[ IN ] RxLen  
The length of RxBuf.

[ OUT ] RxLen  
The length of return data.

[ IN ] ulTimeout  
Set timeout.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## CTOS\_CLLEDSet

---

`USHORT CTOS_CLLEDSet (BYTE bLEDMask,BYTE bOnOffMask) ;`

**Description** Turn on/off one specific LED.

**Parameters** [ IN ] *bLEDMask*

Mask of LED to control.

- *d\_CL\_LED\_RED*
- *d\_CL\_LED\_GREEN*
- *d\_CL\_LED\_YELLOW*
- *d\_CL\_LED\_BLUE*
- *d\_CL\_LED\_EXT\_RED*
- *d\_CL\_LED\_EXT\_GREEN*
- *d\_CL\_LED\_EXT\_BLUE*

[ IN ] *bOnOffMask*

OnOff status to set for corresponding masked LED. For example, while bit0(RED) of bLEDMask is masked to 1, if bit0 of bOnOffMask is set to 1, then turn on the red LED, and if bit0 of bOnOffMask is set to 0, then turn off the red LED.

**Return Value**

<b>Constants</b>	<b>Value</b>
<u><a href="#">d_OK</a></u>	0000h
<u><a href="#">Contactless reader error codes</a></u>	83xxh

## CTOS\_CLSound

---

```
USHORT CTOS_CLSound(USHORT usFreq, USHORT usDuration);
```

**Description** This function generates a simple sound from the speaker in contactless reader. The frequency argument (usFreq) specifies frequency, in hertz of the sound. The duration argument (usDuration) specifies duration of the sound, in ten milliseconds (10 ms).

**Parameters**

[ IN ]	<i>usFreq</i> Frequency=38400/usFreq (Hz).
--------	---

[ IN ]	<i>usDuration</i> Duration=10*((2*usFreq)/38400)*usDuration (ms).
--------	--

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Contactless reader error codes</a>	83xxh

## 2.24.FNT Font Functions

(no need to call these functions when TTF fonts are used)

- CTOS\_LanguageConfig
- CTOS\_LanguageLCDFontSize
- CTOS\_LanguagePrinterFontSize
- CTOS\_LanguageInfo
- CTOS\_LanguageNum
- CTOS\_LanguagePrinterSelectASCII
- CTOS\_LanguageLCDSelectASCII
- CTOS\_LanguagePrinterGetFontInfo
- CTOS\_LanguageLCDGetFontInfo

### FontError Codes

<b>Constants</b>	<b>Value</b>
d_FONT_LANGUAGE_NOT_SUPPORT	1E01h
d_FONT_SIZE_NOT_SUPPORT	1E02h
d_FONT_STYLE_NOT_SUPPORT	1E03h
d_FONT_TYPE_ERROR	1E04h
d_FONT_INVALID_FONT_FILE	1E05h
d_FONT_INDEX_NOT_SUPPORT	1E06h

## CTOS\_LanguageConfig

---

```
USHORT CTOS_LanguageConfig(USHORT usLanguage, USHORT  
usFontSize, USHORT usFontStyle, BOOL boSetDefault);
```

**Description** Select the language/font to be used on LCD/Printer device. ASCII code is from 0x00 to 0x7F, therefore all the other languages are encoded in the range from 0x80 to 0xFF. Calling this function has different meaning for alphabet base language and non-alphabet base language. For alphabet base language, such as Portuguese or Spanish, the alphabet and hex code is 1 to 1 mapping and no extra decoding rule is required, calling this function is similar to select only the font pattern for hex code range from 0x80 to 0xFF. In this case, user is allowed to create their font pattern and to assign a dedicate ID for the font created. Programmer should select the font with dedicate ID through this function. For non-alphabet base language, such as Chinese or Japanese, since these language has its own decoding rule, calling this function will not only select the font pattern but also the the decoding rule for hex code range from 0x80 to 0xFF. In this case, only the font pattern created by Castles can be used. If there is only non-English font exist in the terminal, it will be set as default and no need to be selected through this function.

**Parameters** [ IN ] *usLanguage*  
Language decoding rule to be selected for non-ASCII(code that from 0x80 to 0xFF) characters.

- ***d\_FONT\_CHINESE\_TAIWAN***  
Select Chinese(Taiwan) Language
- ***d\_FONT\_JAPANESE***  
Select Japanese Language
- ***d\_FONT\_KOREAN***  
Select Korean Language

- ***d\_FONT\_PORTUGUESE\_BRAZIL***  
Select Portuguese(Brazil) Language
- ***d\_FONT\_RUSSIAN***  
Select Russian Language
- ***d\_FONT\_THAI***  
Select Thai Language
- ***d\_FONT\_FARSI***  
Select Farsi Language
- ***d\_FONT\_VIETNAMESE***  
Select Vietnamese Language
- ***d\_FONT\_CHINESE\_PRC***  
Select Chinese(PRC) Language
- ***d\_FONT\_PORTUGUESE\_PORTUGAL***  
Select Portuguese(Portugal) Language
- ***d\_FONT\_ARABIC\_QATAR***  
Choose Arabic(Qatar) Language

[ IN ]    *usFontSize*

- ***d\_FONT\_8x8***  
Select 8x8 font size.
- ***d\_FONT\_8x16***  
Select 8x16 font size.
- ***d\_FONT\_16x16***  
Select 16x16 font size.
- ***d\_FONT\_12x24***

Select 12x24 font size.

- ***d\_FONT\_24x24***

Select 24x24 font size.

- ***d\_FONT\_9x9***

Select 9x9 font size

- ***d\_FONT\_9x18***

Select 9x18 font size

- ***d\_FONT\_16x30***

Select 16x30 font size

- ***d\_FONT\_24x24***

Select 24x24 font size

[ IN ]    *usFontSize*

Font Style.

[ IN ]    *boSetFontDefault*

Whether to set this language to default while multiple language is installed in the terminal..

#### Return Value

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Font error codes</a>	1Exxh

#### Example

Please refer to example in appendix [B.9 Language](#)

#### Notes

The font size and font style input will be applied to both LCD and printer, but programmer can select the font and style for LCD and printer individually by calling **CTOS\_LanguageLCDFontSize()** and **CTOS\_LanguagePrinterFontSize()**.

If user's custom FNT fonts are loaded, *usFontSize* should indicate the

proper size of the new font. E.g: for a 10x14 font, *usFontSize* shall be 0xA0E

## CTOS\_LanguageLCDFontSize

---

```
USHORT CTOS_LanguageLCDFontSize(USHORT usFontSize, USHORT  
usFontStyle);
```

**Description** Set the LCD font size and font style for non-ASCII(code that from 0x80 to 0xFF) character.

- Parameters** [ IN ] *usFontSize*
- ***d\_FONT\_8x8***  
Select 8x8 font size.
  - ***d\_FONT\_8x16***  
Select 8x16 font size.
  - ***d\_FONT\_16x16***  
Select 16x16 font size.
  - ***d\_FONT\_12x24***  
Select 12x24 font size.
  - ***d\_FONT\_24x24***  
Select 24x24 font size.
  - ***d\_FONT\_9x9***  
Select 9x9 font size
  - ***d\_FONT\_9x18***  
Select 9x18 font size
  - ***d\_FONT\_16x30***  
Select 16x30 font size
  - ***d\_FONT\_24x24***  
Select 24x24 font size

- [ IN ] *usFontSize*

Font Style.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Font error codes</a>	1Exxh

**Example**

Please refer to example in appendix [B.9 Language](#)

**Notes**

The font size of ASCII for LCD will be set to the same size as input for non-ASCII automatically.

If user's custom FNT fonts are loaded, *usFontSize* should indicate the proper size of the new font. E.g: for a 10x14 font, *usFontSize* shall be 0x0A0E

## CTOS\_LanguagePrinterFontSize

---

```
USHORT CTOS_LanguagePrinterFontSize(USHORT usFontSize, USHORT  
usFontStyle, USHORT usSetType);
```

**Description** Set the printer font size and font style for non-ASCII(code that from 0x80 to 0xFF) character.

- Parameters** [ IN ] *usFontSize*
- ***d\_FONT\_8x8***  
Select 8x8 font size.
  - ***d\_FONT\_8x16***  
Select 8x16 font size.
  - ***d\_FONT\_16x16***  
Select 16x16 font size.
  - ***d\_FONT\_12x24***  
Select 12x24 font size.
  - ***d\_FONT\_24x24***  
Select 24x24 font size.
  - ***d\_FONT\_9x9***  
Select 9x9 font size
  - ***d\_FONT\_9x18***  
Select 9x18 font size
  - ***d\_FONT\_16x16***  
Select 16x16 font size
  - ***d\_FONT\_16x30***

Select 16x30 font size

- **d\_FONT\_24x24**

Select 24x24 font size

[ IN ] *usFontSize*

Font Style.

[ IN ] *usSetType*

Set the font to be a type to be used for the escape code of  
**CTOS\_PrinterPutString()**.

- **d\_FONT\_NO\_SET\_TYPE**

Not set type. Just select font.

- **d\_FONT\_TYPE1**

Set this font to type 1 (escape code \ft1).

- **d\_FONT\_TYPE2**

Set this font to type 2 (escape code \ft2).

- **d\_FONT\_TYPE3**

Set this font to type 3 (escape code \ft3).

#### Return Value

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Font error codes</a>	1Exxh

#### Example

Please refer to example in appendix [B.9 Language](#)

#### Notes

The font size of ASCII for printer will be set to the same size as input for non-ASCII automatically.

Type 1 is the basic font type. After printing each line, the font to be used in next line will be set back to type 1.

If user's custom FNT fonts are loaded, *usFontSize* should indicate the proper size of the new font. E.g: for a 10x14 font, *usFontSize* shall be

0x0A0E

## CTOS\_LanguageInfo

---

```
USHORT CTOS_LanguageInfo(USHORT
usIndex, USHORT*pusLanguage, USHORT*pusFontSize, USHORT*pusFontStyle);
```

**Description** Get the information of the font installed in the terminal.

**Parameters** [ IN ] *usIndex*  
Index of the language installed. (The value of first index is 1).

[ OUT ] *pusLanguage*  
Language ID of the font.

[ OUT ] *pusFontSize*  
Size of the font.

[ OUT ] *pusFontStyle*  
Style of the font.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Font error codes</a>	1Exxh

**Example** Please refer to example in appendix [B.9 Language](#)

## CTOS\_LanguageNum

---

```
USHORT CTOS_LanguageNum(USHORT*pusIndex);
```

**Description**      Return how many fonts are currently installed in the terminal.

**Parameters**      [ OUT ] *pusIndex*  
                        Number of the font installed.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Font error codes</a>	1Exxh

**Example**      Please refer to example in appendix [B.9 Language](#)

## CTOS\_LanguagePrinterSelectASCII

---

```
USHORT CTOS_LanguagePrinterSelectASCII(USHORT usASCIIFontID);
```

**Description** Select the ASCII font for the printer.

**Parameters** [ IN ] [usASCIIFontID](#)  
ID of the ASCII font.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<u><a href="#">Font error codes</a></u>	1Exxh

## CTOS\_LanguageLCDSelectASCII

---

```
USHORT CTOS_LanguageLCDSelectASCII(USHORT usASCIIFontID);
```

**Description** Select the ASCII font for the LCD.

**Parameters** [ IN ] *usASCIIFontID*  
ID of the ASCII font.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Font error codes</a>	1Exxh

## CTOS\_LanguagePrinterGetFontInfo

---

USHORT

```
CTOS_LanguagePrinterGetFontInfo (USHORT*usASCIIFontID, USHORT*usFontSize,  
                                 USHORT*usFontStyle);
```

**Description** Get information of the font currently used for printer.

**Parameters** [ OUT ] *usASCIIFontID*

ID of the font.

[ OUT ] *usFontSize*

Size of the font.

[ OUT ] *usFontStyle*

Style of the font.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Font error codes</a>	1Exxh

## CTOS\_LanguageLCDGetFontInfo

---

USHORT

```
CTOS_LanguageLCDGetFontInfo (USHORT*usASCIIFontID, USHORT*usFontSize, USHORT*usFontStyle);
```

**Description** Get information of the font currently used for LCD.

**Parameters** [ OUT ] *usASCIIFontID*  
ID of the font.

[ OUT ] *usFontSize*  
Size of the font.

[ OUT ] *usFontStyle*  
Style of the font.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Font error codes</a>	1Exxh

## 2.25.GSM Functions

### System Commands

- CTOS\_GSMOpen
- CTOS\_GSMClose
- CTOS\_GSMPowerOff
- CTOS\_GSMPowerOn
- CTOS\_GSMReset
- CTOS\_GSMQueryOperatorName
- CTOS\_GSMHangup
- CTOS\_GSMSendData
- CTOS\_GSMRecvData
- CTOS\_GSMSendATCmd
- CTOS\_GSMSignalQuality
- CTOS\_GSMGetIMEI
- CTOS\_GSMGetNetworkType
- CTOS\_GSMSetBAND
- CTOS\_GSMGetBAND
- CTOS\_GSMDial
- CTOS\_GSMSwitchToCmdMode
- CTOS\_GSMSwitchToCmdMode\_A
- CTOS\_GSMSwitchToDataMode
- CTOS\_GSMGetModuleOpMode
- CTOS\_GSMPowerMode
- CTOS\_MobileSetNetworkType
- CTOS\_MobileGetNetworkTypeCurrent
- CTOS\_MobileGetNetworkTypeSupport

### SIM Commands

- CTOS\_GSMSelectSIM
- CTOS\_GSMGetCurrentSIM
- CTOS\_SIMGetIMSI
- CTOS\_SIMCheckReady

### **PIN**

- CTOS\_PINGetAuthStatus
- CTOS\_PINVerify
- CTOS\_PINCheckLock
- CTOS\_PINLock
- CTOS\_PINUnLock
- CTOS\_PINUpdate

### **GPRS**

- CTOS\_GPRSA.Attach
- CTOS\_GPRSDetach
- CTOS\_GPRSGetRegState
- CTOS\_GPRSPPPConnect

### **Phonebook**

- CTOS\_PBSelect
- CTOS\_PBRead
- CTOS\_PBWrite
- CTOS\_PBDelete

### **SMS**

- CTOS\_SMSGetSCNumber
- CTOS\_SMSSetSCNumber
- CTOS\_SMSGetList
- CTOS\_SMSRead
- CTOS\_SMSDelete
- CTOS\_SMSSend
- CTOS\_SMSSendPDU
- CTOS\_SMSUnicodeSend

### **GSM Error Codes**

<b>Constants</b>	<b>Value</b>
d_GSM_SELECT_FAIL	4601h
d_GSM_OPEN_COMM_FAIL	4602h
d_GSM_SEND_FAIL	4603h
d_GSM_NO_RESP	4604h
d_GSM_RESP_FORMAT_ERR	4605h
d_GSM_INITIAL_FAIL	4606h
d_GSM_CMD_FAIL	4607h
d_GSM_SIM_READY	4608h
d_GSM_SIM_NOT_READY	4609h
d_GSM_NO_AUTH_NEED	460Ah
d_GSM_AUTH_PIN	460Bh
d_GSM_AUTH_PUK	460Ch
d_GSM_AUTH_PIN2	460Dh
d_GSM_AUTH_PUK2	460Eh
d_GSM_AUTH_OTHERS	460Fh
d_GSM_SIM_LOCK	4610h
d_GSM_SIM_NOT_LOCK	4611h
d_GSM_BAD_PARAMETER	4612h
d_GSM_PHONEBOOK_FORMAT_ERR	4613h
d_GSM_BUFFER_OVERFLOW	4614h
d_GSM_RECORD_NOT_FOUND	4615h
d_GSM_SMS_ONE_LINE	461Eh
d_GSM_SMS_TWO_LINE	461Fh
d_GSM_SIM_SELECT_FAIL	4620h
d_GSM_IO_PROCESSING	4621h
d_GSM_NOT_SUPPORT	4622h
d_GSM_UNKNOWN_MODULE	4623h
d_GSM_INIT_CANCEL	4624h
d_GSM_IO_BUSY	4625h
d_GSM_NO_CARRIER	4626h

## CTOS\_GSMOpen

---

```
USHORT CTOS_GSMOpen(ULONG ulBaud, BYTE bInit);
```

**Description** Open the GSM channel and initialize the GSM module.

**Parameters** [ IN ] *ulBaud*  
RFU, default 115200.

[ IN ] *bInit*  
RFU, default 1.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to exampleappendix [B.4 GSM](#)

## **CTOS\_GSMClose**

---

```
USHORT CTOS_GSMClose(void);
```

**Description** Close the GSM channel.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_GSMPowerOff

---

```
USHORT CTOS_GSMPowerOff(void);
```

**Description** Power down the GSM module.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_GSMPowerOn

---

```
USHORT CTOS_GSMPowerOn(void);
```

**Description** Power on the GSM module.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_GSMReset

---

```
void CTOS_GSMReset(void);
```

**Description** Reset the GSM module.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_GSMQueryOperatorName

---

```
USHORT CTOS_GSMQueryOperatorName(BYTE *baName, BYTE *bpLen);
```

**Description** Get name of operator currently selected by the GSM module.

**Parameters**

[OUT] <u><i>baName</i></u>	Name of the operator.
----------------------------	-----------------------

[ OUT ] <u><i>pbLen</i></u>	Length of name.
-----------------------------	-----------------

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

**Note** The GSM module will select the GSM network according to the SIM card selected. If SIM card is not present, the GSM module will select one with most strong signal strength.

## CTOS\_GSMHangup

---

```
USHORT CTOS_GSMHangup (void) ;
```

**Description** Close the current conversation (data, fax or voice).

**Parameters** None

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_GSMSendData

---

```
USHORT CTOS_GSMSendData(BYTE *baData, USHORT usLen);
```

**Description** Send data to GSM module.

**Parameters** [ IN ] *baData*  
Data to send.

[ IN ] *usLen*  
Length of data.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

**Note** Programmer can use this function to directly control the GSM module.

## CTOS\_GSMRecvData

---

```
USHORT CTOS_GSMRecvData ( BYTE *baData, USHORT *pusLen );
```

**Description** Receive data from GSM module.

**Parameters** [OUT] *baData*  
Buffer to store the received data.

[IN] *pusLen*  
Size of buffer. Maximum number of data to get.

[OUT] *pusLen*  
Length of the data actually returned in the buffer.

Return Value	Constants	Value
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

**Note** Programmer can use this function to directly control the GSM module.

## CTOS\_GSMSendATCmd

---

```
USHORT CTOS_GSMSendATCmd(BYTE *baCmd, ULONG ulLen, ULONG ultimeout);
```

**Description** Send the AT command to the GSM and receive the response from.

**Parameters** [ IN ] *baCmd*

The AT command wants to be sent.

[ IN ] *ulLen*

The AT command length.

[ IN ] *ulTimeout*

The command timeout.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example**

Please refer to example appendix [B.4 GSM](#)

## CTOS\_GSMSignalQuality

---

```
USHORT CTOS_GSMSignalQuality(BYTE *bpStrength);
```

**Description** Get signal quality of the GSM network.

**Parameters** [OUT] *bpStrength*  
Signal quality indicator.  
0: -113 dBm or less  
1: -111 dBm  
2~30: -109 dBm ~ -53 dBm, 2 dBm per step  
31: -51 dBm or greater  
99: not known or not detectable

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_GSMGetIMEI

---

```
USHORT CTOS_GSMGetIMEI (BYTE *pbInfo );
```

**Description** Retrieve GSM module IMEI number.

**Parameters** [ OUT ] *pbInfo*  
Buffer to store IMEI. At least 15 bytes is required.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

## CTOS\_GSMGetNetworkType

---

USHORT CTOS\_GSMGetNetworkType (BYTE \*pbType) ;

**Description** Get packet service network type.

**Parameters** [ OUT ] *pbType*

- **d\_GSM\_NETWORK\_GPRS**  
GPRS network
- **d\_GSM\_NETWORK\_EGPRS**  
EDGE GPRS network
- **d\_GSM\_NETWORK\_WCDMA**  
W-CDMA (3G) network
- **d\_GSM\_NETWORK\_HSDPA**  
HSDPA network
- **d\_GSM\_NETWORK\_UNKNOWN**  
Unknown network

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

## CTOS\_GSMSetBAND

---

USHORT CTOS\_GSMSetBAND (BYTE bID) ;

**Description** Select the band to use. The default band is d\_GSM\_900\_1800 (GSM 900MHz + DCS 1800MHz).

<b>Parameters</b>	[ IN ] <i>bID</i>
	Band to select.
	<ul style="list-style-type: none"> <li>• <b>d_GSM_900_1800</b> GSM 900MHz + DCS 1800MHz</li> </ul>
	<ul style="list-style-type: none"> <li>• <b>d_GSM_900_1900</b> GSM 900MHz + PCS 1900MHz</li> </ul>
	<ul style="list-style-type: none"> <li>• <b>d_GSM_850_1800</b> GMS 850MHz + PCS 1800MHz</li> </ul>
	<ul style="list-style-type: none"> <li>• <b>d_GSM_850_1900</b> GMS 850MHz + PCS 1900MHz</li> </ul>

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_GSMGetBAND

---

USHORT CTOS\_GSMGetBAND (BYTE \*bID) ;

**Description** Get the band currently selected.

**Parameters** [OUT] bID

Band currently selected.

- **d\_GSM\_900\_1800**

GSM 900MHz + DCS 1800MHz

- **d\_GSM\_900\_1900**

GSM 900MHz + PCS 1900MHz

- **d\_GSM\_850\_1800**

GMS 850MHz + PCS 1800MHz

- **d\_GSM\_850\_1900**

GMS 850MHz + PCS 1900MHz

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example**

Please refer to example appendix [B.4 GSM](#)

**Remark**

The Telit's module UE910 (3G) does not support this function and it will return d\_GSM\_NOT\_SUPPORT

## CTOS\_GSMDial

---

```
USHORT CTOS_GSMDial(BYTE *baNumber, BYTE bLen);
```

**Description** Start a data, fax, or voice call to the given number.

**Parameters** [ IN ] *baNumber*  
Number to dial.

[ IN ] *bLen*  
Length of number.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

**Note** A ";" must be appended to the end of number for starting the voice call. For example, if the number to dial is "123456", the content of input baNumber should be "123456;".

## CTOS\_GSMSwitchToCmdMode

---

```
USHORT CTOS_GSMSwitchToCmdMode (void);
```

**Description**      Switch the GSM module to command mode for sending AT command.

**Parameters**      None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example**      Please refer to example appendix [B.4 GSM](#)

**Note**      This function is normally used when programmer wish to send AT command to GSM module after GPRS connection is established. The GSM module is in data mode after connecting to GPRS.

## **CTOS\_GSMSwitchToCmdMode\_A**

---

```
USHORT CTOS_GSMSwitchToCmdMode_A(void);
```

**Description**      Switch the GSM module to command mode in asynchronized way for sending AT command.

**Parameters**      None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Note**      This function is normally used when programmer wish to send AT command to GSM module after GPRS connection is established. The GSM module is in data mode after connecting to GPRS. The function will return immediately and programmer has to call **CTOS\_TCP\_GPRSStatus()** for polling whether the operation is finish or not.

## CTOS\_GSMSwitchToDataMode

---

```
USHORT CTOS_GSMSwitchToDataMode (void);
```

**Description**      Switch the GSM module back to data mode.

**Parameters**      None

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example**      Please refer to example appendix [B.4 GSM](#)

**Note**      After GPRS connection is established, programmer can switch to command mode for sending AT command, but programmer must switch back to data mode before performing subsequent GPRS communication.

## CTOS\_GSMGetModuleOpMode

---

```
USHORT CTOS_GSMGetModuleOpMode (BYTE *pbOpMode) ;
```

**Description** Get whether the GSM module is working in data mode or command mode.

**Parameters** [ OUT] *pbOpMode*  
Pointer to BYTE for storing the current operation mode of mobile device.

- **d\_GSM\_MODULE\_CMD\_MODE**

GSM module is working in command mode.

- **d\_GSM\_MODULE\_DATA\_MODE**

GSM module is working in data mode.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

## CTOS\_GSMPowerMode

---

USHORT CTOS\_GSMPowerMode (BYTE bMode) ;

**Description** Control the power state of the GSM module.

**Parameters** [ IN ] bMode

Power mode to set.

- **d\_GSM\_PWR\_WORK**

Wake GSM module back to normal working mode.

- **d\_GSM\_PWR\_STANDBY**

- **d\_GSM\_PWR\_SLEEP**

Put GSM module into power saving mode. Currently stanyby and sleep are the same.

When GSM module is in power saving mode, the connection of GSM/GPRS will be remained.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Note**

When calling **CTOS\_PowerMode()** with d\_PWR\_STANDBY\_MODE or d\_PWR\_SLEEP\_MODE, it will automatically put GSM module into power saving, and wake GSM module back to normal working mode when system wakeup from sleep or standby mode.

The function can be used when programmer want to put GSM module to power saving mode independently.

## CTOS\_MobileSetNetworkType

---

`USHORT CTOS_MobileSetNetworkType (USHORT usType) ;`

**Description** Set the network type supported by the Mobile Communication module.

**Parameters** [ OUT] *usType*  
 Pointer to USHORT for storing the network type supported.  
 Value can be the bitwise-OR of following definition.

- ***d\_MOBILE\_NETWORK\_GPRS***  
 GPRS is supported.

- ***d\_MOBILE\_NETWORK\_UMTS***  
 UMTS is supported.

- ***d\_MOBILE\_NETWORK\_CDMA***  
 CDMA is supported.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

## CTOS\_MobileGetNetworkTypeCurrent

---

USHORT CTOS\_MobileGetNetworkTypeCurrent (USHORT\**pusType*) ;

**Description** Get the network type currently working with the Mobile Communication module.

**Parameters** [ OUT] *pusType*  
Pointer to USHORT for storing the network type currently working.

Value can be one of the following definition.

- ***d\_MOBILE\_NETWORK\_GPRS***  
GPRS is working.

- ***d\_MOBILE\_NETWORK\_UMTS***  
UMTS is working.

- ***d\_MOBILE\_NETWORK\_CDMA***  
CDMA is working.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

## CTOS\_MobileGetNetworkTypeSupport

---

USHORT CTOS\_MobileGetNetworkTypeSupport (USHORT\**pusType*) ;

**Description** Get the supported network types.

**Parameters** [OUT] *pusType*  
Supported network types.  
Value can be one of the following definition.

- ***d\_MOBILE\_NETWORK\_GPRS***  
Select GPRS.
- ***d\_MOBILE\_NETWORK\_UMTS***  
Select UMTS.
- ***d\_MOBILE\_NETWORK\_CDMA***  
Select CDMA.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

## CTOS\_GSMSelectSIM

---

USHORT CTOS\_GSMSelectSIM(BYTE bID) ;

**Description** Select the SIM card to use. The default SIM is SIM1.

**Parameters** [ IN ] bID  
SIM ID to select.

- d\_GPRS\_SIM1
- d\_GPRS\_SIM2

Return Value	<i>Constants</i>	<i>Value</i>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_GSMGetCurrentSIM

---

USHORT CTOS\_GSMGetCurrentSIM(BYTE\*bID) ;

**Description** Get the ID of SIM currently used.

**Parameters** [OUT] *bID*  
SIM ID currently used.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_SIMGetIMSI

---

```
USHORT CTOS_SIMGetIMSI(BYTE *pbInfo);
```

**Description** Get the value of the Internal Mobile Subscriber Identity stored in the SIM.

**Parameters** [ OUT ] *pbInfo*  
Buffer to store IMSI. At least 15 bytes is required.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_SIMCheckReady

---

```
USHORT CTOS_SIMCheckReady(void);
```

**Description** Check if SIM card selected ready to use.

**Parameters** None

- Return Value**
- **d\_GSM\_SIM\_READY**  
SIM is ready
  - **d\_GSM\_SIM\_NOT\_READY**  
SIM is not ready.

**Example** Please refer to example appendix [B.4 GSM](#)

**Note** The SIM card might be not ready due to SIM card is not present, SIM card can not be reset or SIM card is not compatible.

## CTOS\_PINGetAuthStatus

---

```
USHORT CTOS_PINGetAuthStatus(void);
```

**Description** Get the authenticate status of SIM card.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_GSM_NO_AUTH_NEED	460Ah
d_GSM_AUTH_PIN	460Bh
d_GSM_AUTH_PUK	460Ch
d_GSM_AUTH_PIN2	460Dh
d_GSM_AUTH_PUK2	460Eh
d_GSM_AUTH_OTHERS	460Fh
<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

**Note**

Programmer can use this function to check whether a PIN is required for using the SIM card, and what type of PIN should be used if PIN is required.

## CTOS\_PINVerify

---

USHORT CTOS\_PINVerify(BYTE bPinType, BYTE \*baPin, BYTE bPinLen);

**Description** Verify the PIN for SIM card.

**Parameters** [ IN ] bPinType

Type of PIN to verify.

- d\_GSM\_PIN\_PUK\_1

- d\_GSM\_PIN\_PUK\_2

RFU

[ IN ] baPin

PIN in ASCII code.

Example: "1234".

[ IN ] bPinLen

Length of PIN.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example**

Please refer to example appendix [B.4 GSM](#)

## CTOS\_PINCheckLock

---

```
USHORT CTOS_PINCheckLock(BYTE locktype);
```

**Description** Get facility lock state.

**Parameters** [ IN ] *bLockType*

Lock type.

- *d\_GSM\_AUTH\_SC*

SIM lock, device asks SIM password at power-up.

- *d\_GSM\_AUTH\_PS*

Link the terminal to the current SIM card. The device will ask for a password when other than current SIM card inserted.

- *d\_GSM\_AUTH\_FD*

SIM fixed dialling memory feature.

**Return Value**

- *d\_GSM\_SIM\_LOCK*

Specific facility is locked.

- *d\_GSM\_SIM\_NOT\_LOCK*

Specific facility is not locked.

- Others

Please refer to [GSM error codes](#)

**Example**

Please refer to example appendix [B.4 GSM](#)

## CTOS\_PINLock

---

USHORT CTOS\_PINLock(BYTE locktype, BYTE \*pin, BYTE pinlen);

**Description** Lock the facility.

**Parameters** [ IN ] *bLockType*

Lock type.

- *d\_GSM\_AUTH\_SC*

SIM lock, device asks SIM password at power-up.

- *d\_GSM\_AUTH\_PS*

Link the terminal to the current SIM card. The device will ask for a password when other than current SIM card inserted.

- *d\_GSM\_AUTH\_FD*

SIM fixed dialling memory feature.

[ IN ] *baPin*

Current PIN.

[ IN ] *bPinLen*

Pin Length.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example**

Please refer to example appendix [B.4 GSM](#)

## CTOS\_PINUnLock

---

USHORT CTOS\_PINUnLock(BYTE locktype, BYTE \*pin, BYTE pinlen);

**Description**      Unlock the facility.

**Parameters**      [ IN ]      *bLockType*

Lock type.

- *d\_GSM\_AUTH\_SC*

SIM lock, device asks SIM password at power-up.

- *d\_GSM\_AUTH\_PS*

Link the terminal to the current SIM card. The device will ask for a password when other than current SIM card inserted.

- *d\_GSM\_AUTH\_FD*

SIM fixed dialling memory feature.

[ IN ]      *baPin*

Current PIN.

[ IN ]      *bPinLen*

Pin Length.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example**

Please refer to example appendix [B.4 GSM](#)

## CTOS\_PINUpdate

---

```
USHORT CTOS_PINUpdate(BYTE pintype, BYTE *oldpin, BYTE oldpinlen,
BYTE *newpin, BYTE newpinlen);
```

**Description** Change the PIN of the facility.

**Parameters**

[ IN ]	<i>bPinType</i>
	PIN type.
	• <i>d_GSM_AUTH_SC</i> PIN1.
	• <i>d_GSM_AUTH_PS</i> TerminalPIN.
	• <i>d_GSM_AUTH_FD</i> PIN2.
[ IN ]	<i>baOldPin</i>
	OriginalPIN.
[ IN ]	<i>bOldPinLen</i>
	Length of original PIN.
[ IN ]	<i>baNewPin</i>
	New PIN to change.
[ IN ]	<i>bNewPinLen</i>
	Length of New PIN.

**Return Value**

<b>Constants</b>	<b>Value</b>
<i>d_OK</i>	0000h
<a href="#">GSM error codes</a>	46xxh

**Example**

Please refer to example appendix [B.4 GSM](#)

## CTOS\_GPRSAttach

---

USHORT CTOS\_GPRSAttach (BYTE \*baAPN, BYTE bAPNLen) ;

**Description** Attach to the GPRS service.

**Parameters**

[ IN ]	<u><i>baAPN</i></u>
	Access Point Name.
[ IN ]	<u><i>bAPNLen</i></u>
	Length of APN.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_GPRSDetach

---

```
USHORT CTOS_GPRSDetach(void);
```

**Description**      Detach from the GPRS service.

**Parameters**      None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example**      Please refer to example appendix [B.4 GSM](#)

## CTOS\_GPRSGetRegState

---

USHORT CTOS\_GPRSGetRegState (BYTE \*pbState) ;

**Description** Get the GPRS network registration status.

**Parameters** [OUT] *pbState*

Registration status.

- ***d\_GSM\_GPRS\_STATE\_NOT\_REG***

Terminal not registered and is not currently searching a new operator to register to.

- ***d\_GSM\_GPRS\_STATE\_REG***

Terminal is registered to home network.

- ***d\_GSM\_GPRS\_STATE\_TRYING***

Terminal is not registered but is currently searching a new operator to register to.

- ***d\_GSM\_GPRS\_STATE\_DENY***

Terminal attempt to register but is denied.

- ***d\_GSM\_GPRS\_STATE\_UNKNOW***

Unknown status.

- ***d\_GSM\_GPRS\_STATE\_ROAM***

Terminal is registered in roaming condition.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

**Example**

Please refer to example appendix [B.4 GSM](#)

## CTOS\_GPRSPPPConnect

---

```
USHORT CTOS_GPRSPPPConnect (void) ;
```

**Description** Establish communication between the terminal and the network using layer 2 protocol PPP types.

**Parameters** [ IN ] *baBuf*  
xxxxx.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to exampleappendix [B.4 GSM](#)

## CTOS\_PBSelect

---

```
USHORT CTOS_PBSelect(BYTE bPhoneBookType, BYTE *pbUsed, BYTE  
*pbCapacity);
```

**Description** Select phone book.

**Parameters** [ IN ] *bPhoneBookType*

- ***d\_GSM\_PB\_FD***

Fixed Dialing phone book.

- ***d\_GSM\_PB\_SM***

SIM phone book.

- ***d\_GSM\_PB\_ON***

MSISDN list phone book.

- ***d\_GSM\_PB\_ME***

Mobile Equipment phone book.

- ***d\_GSM\_PB\_LD***

Last Dialed number phone book.

- ***d\_GSM\_PB\_MC***

Missed calls list phone book.

- ***d\_GSM\_PB\_RC***

Received Calls list phone book.

[ OUT ] *pbUsed*

Number of record already used in the phone book.

[ OUT ] *pbCapacity*

Maximum number of record can be stored in the phone book.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_PBRead

---

```
USHORT CTOS_PBRead(BYTE bIndex, stPhoneBook *pstPhoneBook);
```

**Description**      Read one record from the phone book by the index.

**Parameters**      [ IN ]      *bIndex*  
                          Index of record to read.

[ OUT ]      *baBuf*  
                          Record read from phone book.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example**      Please refer to example appendix [B.4 GSM](#)

## CTOS\_PBWrite

---

```
USHORT CTOS_PBWrite(stPhoneBook *pstPhoneBook);
```

**Description** Add a new record to the phone book.

**Parameters** [ IN ] *pstPhoneBook*  
Record to add.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_PBDelete

---

```
USHORT CTOS_PBDelete(BYTE bIndex);
```

**Description** Delete a record from the phone book by index.

**Parameters** [ IN ] *bIndex*  
Index of record to remove.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_SMSGetSCNumber

---

```
USHORT CTOS_SMSGetSCNumber(BYTE *baNumber, BYTE *pbLen);
```

**Description** Get the SMS Service Center number.

**Parameters** [OUT] *baNumber*  
Buffer to store Service Center number.

[ IN ] *pbLen*  
Size of buffer.

[ OUT ] *pbLen*  
Length of Service Center number.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_SMSSetSCNumber

---

```
USHORT CTOS_SMSSetSCNumber(BYTE *baNumber, BYTE bLen);
```

**Description** Set the SMS Service Center number.

**Parameters** [ IN ] *baNumber*  
Service Center number to set.

[ IN ] *bLen*  
Length of Service Center number.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_SMSGetList

---

USHORT CTOS\_SMSGetList (BYTE bStorage, stSMS \*pstSMS, BYTE \*pbNum) ;

**Description** Read all SMS records in the specific SMS storage.

**Parameters**

- [ IN ] bStorage
  - **d\_GSM\_SMS\_REC\_UNREAD**  
Record not read.
  - **d\_GSM\_SMS\_REC\_READ**  
Record already read.
  - **d\_GSM\_SMS\_STO\_UNSEND**  
Record not sent.
  - **d\_GSM\_SMS\_STO\_SEND**  
Record already sent.
  - **d\_GSM\_SMS\_ALL**  
All record.

[ OUT ] pstSMS  
Array of the stSMS structure to store the SMS record.

[ IN ] pbNum  
Size of array. Maximum number of record to read.

[ OUT ] pbNum  
Number of record returned in array.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_SMSRead

---

```
USHORT CTOS_SMSRead(BYTE bIndex, stSMS *pstSMS);
```

**Description**      Read one SMS record by the record index.

**Parameters**      [ IN ]      *bIndex*  
                          Index of SMS record to read.

                          [ OUT]    *pstSMS*  
                          SMS record returned.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example**      Please refer to example appendix [B.4 GSM](#)

## CTOS\_SMSDelete

---

```
USHORT CTOS_SMSDelete(BYTE bIndex);
```

**Description** Delete one SMS record by the record index.

**Parameters** [ IN ] *bIndex*  
Index of SMS record to delete.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_SMSSend

---

```
USHORT CTOS_SMSSend(stSMS_Submit *pstSMS, BYTE *pbIndex);
```

**Description** Send SMS.

**Parameters** [ IN ] *pstSMS*  
SMS to send.

[ OUT ] *pbIndex*  
Index of the SMS stored in terminal.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_SMSSendPDU

---

```
USHORT CTOS_SMSSendPDU(BYTE *baPDU, ULONG ulLen, BYTE *pbIndex);
```

**Description** Send SMS using Packet Data Unit mode.

**Parameters** [ IN ] *baPDU*  
PDU to send.

[ IN ] *ulLen*  
Length of PDU.

[ OUT ] *pbIndex*  
Index of the SMS stored in terminal.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GSM error codes</a>	46xxh

**Example** Please refer to example appendix [B.4 GSM](#)

## CTOS\_SMSUnicodeSend

---

```
USHORT CTOS_SMSUnicodeSend(BYTE bMode, stSMS_Unicode_Submit *pstSMS,
BYTE *pbIndex);
```

**Description** Send unicode SMS.

**Parameters**

[ IN ]	<i>bMode</i>
Unicode format to select.	
• <i>d_GSM_SMS_UTF8</i>	
Select UTF8 format.	

• <i>d_GSM_SMS_UCS2</i>
Select UCS2 format.

[ IN ]	<i>pstSMS</i>
SMS to send.	

[ OUT ]	<i>pbIndex</i>
Index of the SMS stored in terminal.	

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GSM error codes</a>	46xxh

## 2.26.GPRS Functions

- CTOS\_TCP\_GPRSInit
- CTOS\_TCP\_GPRSOpen
- CTOS\_TCP\_GPRSOpenEx
- CTOS\_TCP\_GPRSClose
- CTOS\_TCP\_GPRSClose\_A
- CTOS\_TCP\_GPRSGetIP
- CTOS\_TCP\_GPRSConnectEx
- CTOS\_TCP\_GPRSDisconnect
- CTOS\_TCP\_GPRSTx
- CTOS\_TCP\_GPRSRx
- CTOS\_TCP\_GPRSStatus
- CTOS\_TCP\_GPRSSocketStatus
- CTOS\_TCP\_GPRSSwitchAPN
- CTOS\_TCP\_GPRSSwitchAPN\_A
- CTOS\_TCP\_GPRSCancelTask
- CTOS\_TCP\_GPRSGetDNSServer
- CTOS\_TCP\_GPRSSetDNSServer
- CTOS\_TCP\_GPRSConnectURL
- CTOS\_TCP\_GPRSPing
- CTOS\_TCP\_GPRSURL2IP

### GPRS Error Codes

<b>Constants</b>	<b>Value</b>
d_TCP_IO_PROCESSING	2321h
d_TCP_IO_BUSY	2322h
d_TCP_URL_NOT_FOUND	2370h
d_TCP_PING_FAILED	2371h
d_TCP PPP CONNECTION TERMINATED	2372h
d_TCP_SOCKET_FULL	2380h
d_TCP_CONNECTION_NOT_ESTABLISHED	2381h
d_TCP_BAD_FCS	2382h
d_TCP PPP TIMEOUT	2383h

d_TCP_PROTOCOL_ERROR	2384h
d_TCP_LENGTH_ERROR	2385h
d_TCP_PPP_SEND_TIMEOUT	2386h
d_TCP_PPP_SEND_ERROR	2387h
d_TCP_SOCKET_FAILED	2388h
d_TCP_SOCKET_IO_MODE_FAILED	2389h
d_TCP_RX_RECEIVE_TIMEOUT	238Ah
d_TCP_INVALID_PARA	238Bh
d_TCP_CONNECT_TIMEOUT	2390h
d_TCP_DISCONNECT_FAIL	2391h
d_TCP_RESET	2392h
d_TCP_SEQNENCE_INCORRECT	2393h
d_TCP_NO_SERVER	2394h
d_TCP_RETRANSMISSION	2395h
d_TCP_PROTOCOL	2396h
d_TCP_IP_FORMAT_WRONG	2397h
d_TCP_FORMAT_WRONG	2398h
d_TCP_SEND_TIMEOUT	2399h
d_TCP_NO_ACK	239Ah
d_TCP_BUF_FULL	239Bh
d_TCP_RECEIVE_NON_UDP_PACKAGE	239Ch
d_TCP_IP_ADDRESS_NOT_EXIST	239Dh
d_TCP_RECEIVE_DATA_FAILED	239Eh
d_TCP_SEND_DATA_FAILED	239Fh
d_TCP_LCP_TIMEOUT	23A0h
d_TCP_LCP_ACK	23A1h
d_TCP_LCP_NAK	23A2h
d_TCP_LCP_REJECT	23A3h
d_TCP_LCP_CODE_REJECT	23A4h
d_TCP_LCP_PROTO_REJECT	23A5h
d_TCP_LCP_TERM_REJECT	23A6h
d_TCP_PEER_LCP_ACK	23A7h
d_TCP_PEER_LCP_NAK	23A8h
d_TCP_PEER_LCP_REJ	23A9h
d_TCP_PEER_LCP_CODE_REJECT	23AAh

d_TCP_PEER_LCP_PROTO_REJECT	23ABh
d_TCP_PEER_LCP_TERM_REQUEST	23ACh
d_TCP_PEER_LCP_TERM_ACK	23ADh
d_TCP_SEND_EPIPE_FAILED	23AEh
d_TCP_RECEIVE_EPIPE_FAILED	23AFh
d_TCP_CHAP_TIMEOUT	23B0h
d_TCP_CHAP_RESPONSE	23B1h
d_TCP_CHAP_CODE_REJECT	23B2h
d_TCP_PEER_CHAP_SUCCESS	23B3h
d_TCP_PEER_CHAP_AUTH_FAIL	23B4h
d_TCP_IPCP_TIMEOUT	23C0h
d_TCP_IPCP_ACK	23C1h
d_TCP_IPCP_REJECT	23C2h
d_TCP_IPCP_CODE_REJECT	23C3h
d_TCP_IPCP_PROTO_REJECT	23C4h
d_TCP_PEER_IPCP_ACK	23C5h
d_TCP_PEER_IPCP_NAK	23C6h
d_TCP_IPCP_TOSS	23C7h
d_TCP_PEER_IPCP_REJ	23C8h
d_TCP_CCP_REJECT	23CAh
d_TCP_BIND_FAILED	23CBh
d_TCP_CHANNEL_TYPE_FAILED	23E0h
d_TCP_CHANNEL_OFFLINE	23E1h
d_TCP_GPRS_AUTH_SETTING_FAILED	23E2h
d_TCP_GSM_OPEN_FAILED	23E3h
d_TCP_GPRS_ATTACH_FAILED	23E4h
d_TCP_GPRS PPP_CONNECT_FAILED	23E5h
d_TCP_GPRS_IS_CONNECTED	23E6h
d_TCP_USER_INTERRUPT	23FFh

## **CTOS\_TCP\_GPRSInit**

---

```
void CTOS_TCP_GPRSInit(void);
```

**Description** Initialize the GPRS Library, please call this function once in your main() function.

**Parameters** None

**Return Value** None

**Example** Please refer to example in appendix [B.21 GPRS](#)

**Note** There is no need to call this function anymore. It has been kept for backwards compatibility with old firmwares.

## CTOS\_TCP\_GPRSOpen

---

```
USHORT CTOS_TCP_GPRSOpen(BYTE *baIP, STR *strAPN, STR *baID, STR
*baPW);
```

**Description** Open the network connection over TCP/IP through GPRS. Please call **CTOS\_TCP\_GPRSStatus()** to check whether this action is finished. If GPRS open failed, GSM module will be reset.

**Parameters**

[ IN ]	<u><i>baIP</i></u>
The local IP address (4 bytes).	
When baIP is "\x00\x00\x00\x00", the local IP will be assigned by GPRS Server.	

[ IN ]	<u><i>strAPN</i></u>
The APN name.	

[ IN ]	<u><i>baID</i></u>
The ID string used for the CHAP authentication.	

[ IN ]	<u><i>baPW</i></u>
The password string used for the CHAP authentication.	

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GPRS error codes</a>	23xxh

**Example** Please refer to example in appendix [B.21 GPRS](#)

## **CTOS\_TCP\_GPRSOpenEx**

---

```
USHORT CTOS_TCP_GPRSOpenEx (BYTE *baIP, STR *strAPN, STR *baID, STR  
*baPW);
```

**Description** Open the network connection over TCP/IP through GPRS. Please call **CTOS\_TCP\_GPRSSStatus()** to check whether this action is finished. If GPRS open failed, GSM module will **NOT** be reset.

**Parameters** [ IN ] *baIP*  
The local IP address (4 bytes).  
When *baIP* is "\x00\x00\x00\x00", the local IP will be assigned by GPRS Server.

[ IN ] *strAPN*  
The APN name.

[ IN ] *baID*  
The ID string used for the CHAP authentication.

[ IN ] *baPW*  
The password string used for the CHAP authentication.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GPRS error codes</a>	23xxh

## CTOS\_TCP\_GPRSClose

---

```
USHORT CTOS_TCP_GPRSClose(void);
```

**Description** Close the network connection with GPRS in a synchronous way.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GPRS error codes</a>	23xxh

**Example** Please refer to example in appendix [B.21 GPRS](#)

## CTOS\_TCP\_GPRSClose\_A

---

```
USHORT CTOS_TCP_GPRSClose_A(void);
```

**Description** Close the network connection with GPRS in an asynchronous way.  
Please call **CTOS\_TCP\_GPRSStatus()** to check whether this action is finished.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GPRS error codes</a>	23xxh

**Example** Please refer to example in appendix [B.21 GPRS](#)

## CTOS\_TCP\_GPRSGetIP

---

```
USHORT CTOS_TCP_GPRSGetIP(BYTE *baIP);
```

**Description** Get the IP address assigned manually or by the network.

**Parameters** [ OUT ] *baIP*  
The local IP address of the terminal(4 bytes).

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GPRS error codes</a>	23xxh

**Example** Please refer to example in appendix [B.21 GPRS](#)

## CTOS\_TCP\_GPRSConnectEx

---

```
USHORT CTOS_TCP_GPRSConnectEx(BYTE *bSocket, BYTE *baIP, USHORT usPort);
```

**Description** Connect to the remote with remote IP address and port number.  
Please call **CTOS\_TCP\_GPRSSStatus()** to check whether this action is finished.

**Parameters** [ OUT ] *bSocket*  
The socket descriptor of this connection.

[ IN ] *baIP*  
The pointer of the remote IP address (4 bytes).

[ IN ] *usPort*  
The pointer of the remote port number.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
d_OK		0000h
<a href="#">GPRS error codes</a>		23xxh

**Example** Please refer to example in appendix [B.21 GPRS](#)

## CTOS\_TCP\_GPRSDisconnect

---

```
USHORT CTOS_TCP_GPRSDisconnect (BYTE bSocket) ;
```

**Description** Disconnect to the remote.

Please call **CTOS\_TCP\_GPRSStatus()** to check whether this action is finished.

**Parameters** [ IN ] *bSocket*

The socket descriptor to be closed.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GPRS error codes</a>	23xxh

**Example**

Please refer to example in appendix [B.21 GPRS](#)

## CTOS\_TCP\_GPRSTx

---

```
USHORT CTOS_TCP_GPRSTx(BYTE bSocket, BYTE *baBuffer, USHORT usLen);
```

**Description** Send data out.

Please call **CTOS\_TCP\_GPRSStatus()** to check whether this action is finished.

**Parameters** [ IN ] *bSocket*

The socket descriptor of the connection.

[ IN ] *baBuffer*

The pointer address of sending data.

[ IN ] *usLen*

The length of the sending data. The maximum is 1460.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GPRS error codes</a>	23xxh

**Example**

Please refer to example in appendix [B.21 GPRS](#)

## CTOS\_TCP\_GPRSRx

---

USHORT CTOS\_TCP\_GPRSRx(BYTE bSocket, BYTE \*baBuffer, USHORT \*usLen);

**Description** Receive data.

Please call **CTOS\_TCP\_GPRSStatus()** to check whether this action is finished.

**Parameters** [ IN ] *bSocket*

The socket descriptor of the connection.

[ OUT ] *baBuffer*

The pointer address of received buffer.

[ IN ] *usLen*

Buffer size of baBuffer.

[ OUT ] *usLen*

The length of the received data. The maximum is 1460.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GPRS error codes</a>	23xxh

**Example**

Please refer to example in appendix [B.21 GPRS](#)

**Note**

When get d\_OK with insufficient data, you need to call this function again to get left data.

## CTOS\_TCP\_GPRSStatus

---

```
USHORT CTOS_TCP_GPRSStatus (DWORD*pdwState);
```

**Description** Get the current processing status when calling the GPRS.

**Parameters** [OUT] *pdwState*

The current processing status.

- **TCP\_GPRS\_STATE\_ESTABLISHED**
- **TCP\_GPRS\_STATE\_ESTABLISHING**
- **TCP\_GPRS\_STATE\_CONNECTING**
- **TCP\_GPRS\_STATE\_SENDING**
- **TCP\_GPRS\_STATE RECEIVING**
- **TCP\_GPRS\_STATE\_DISCONNECTING**
- **TCP\_GPRS\_STATE\_ONHOOKING**

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GPRS error codes</a>	23xxh

**Example**

Please refer to example in appendix [B.21 GPRS](#)

**Note**

While no GPRS function is running currently, the pdwState value is 0x00000000.

## CTOS\_TCP\_GPRSSocketStatus

---

USHORT CTOS\_TCP\_GPRSSocketStatus (BYTE bSocket, DWORD\*pdwState);

**Description** Get the current GPRS processing status of the specific socket.

**Parameters**

[ IN ]	<u>bSocket</u>
	The socket descriptor return from GPRSConnectEx().

[ OUT ] pdwState

The current processing status.

- **TCP\_GPRS\_STATE\_ESTABLISHED**
- **TCP\_GPRS\_STATE\_ESTABLISHING**
- **TCP\_GPRS\_STATE\_CONNECTING**
- **TCP\_GPRS\_STATE\_SENDING**
- **TCP\_GPRS\_STATE\_RECEIVING**
- **TCP\_GPRS\_STATE\_DISCONNECTING**
- **TCP\_GPRS\_STATE\_ONHOOKING**

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GPRS error codes</a>	23xxh

**Note**

The pdwState value is 0x00000000 while no GPRS function is running currently on specific socket.

## **CTOS\_TCP\_GPRSSwitchAPN**

---

```
USHORT CTOS_TCP_GPRSSwitchAPN (BYTE *baIP, STR *strAPN, STR *baID,
STR *baPW);
```

**Description** SwitchGPRS network connectionin synchronous way.Please call **CTOS\_TCP\_GPRSStatus()** to check whether this action is finished.

- Parameters**
- [ IN ] *baIP*  
The local IP address (4 bytes).  
When baIP is "\x00\x00\x00\x00", the local IP will be assigned by GPRS Server.
  - [ IN ] *strAPN*  
The APN name.
  - [ IN ] *baID*  
The ID string used for the CHAP authentication.
  - [ IN ] *baPW*  
The password string used for the CHAP authentication.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GPRS error codes</a>	23xxh

## **CTOS\_TCP\_GPRSSwitchAPN\_A**

---

```
USHORT CTOS_TCP_GPRSSwitchAPN_A(BYTE *baIP, STR *strAPN, STR *baID,
STR *baPW);
```

**Description** SwitchGPRS network connection in asynchronous way. Please call **CTOS\_TCP\_GPRSStatus()** to check whether this action is finished.

- Parameters**
- [ IN ] *baIP*  
The local IP address (4 bytes).  
When *baIP* is "\x00\x00\x00\x00", the local IP will be assigned by GPRS Server.
  - [ IN ] *strAPN*  
The APN name.
  - [ IN ] *baID*  
The ID string used for the CHAP authentication.
  - [ IN ] *baPW*  
The password string used for the CHAP authentication.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GPRS error codes</a>	23xxh

## CTOS\_TCP\_GPRSCancelTask

---

```
USHORT CTOS_TCP_GPRSCancelTask (void *RFU);
```

**Description** Cancel current in progress GPRS asynchronous function.

**Parameters** [ IN ] *RFU*  
Reserved for future use. Please call this function using  
CTOS\_TCP\_GPRSCancelTask((void \*)0);

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GPRS error codes</a>	23xxh

## CTOS\_TCP\_GPRSGetDNSServer

---

```
USHORT CTOS_TCP_GPRSGetDNSServer (BYTE *baIP);
```

**Description** Get the DNS IP address of the current connection.

**Parameters** [ OUT ] *baIP*  
DNS IP address (4 bytes).

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GPRS error codes</a>	23xxh

## CTOS\_TCP\_GPRSSetDNSServer

---

```
USHORT CTOS_TCP_GPRSSetDNSServer (BYTE *baIP);
```

**Description** Set the DNS IP address of the current connection.

**Parameters** [ IN ] *baIP*  
DNS IP address, 4 bytes.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GPRS error codes</a>	23xxh

## CTOS\_TCP\_GPRSConnectURL

---

```
USHORT CTOS_TCP_GPRSConnectURL (BYTE *pbSocket, BYTE *baIPURL,
USHORT usPort);
```

<b>Description</b>	Connect to the remote server using URL.  Please call <b>CTOS_TCP_GPRSSStatus()</b> to check whether this action is finished.						
<b>Parameters</b>	<p>[ OUT ] <u><i>pbSocket</i></u> The socket descriptor of this connection.</p> <p>[ IN ] <u><i>baIPURL</i></u> URL or IP address in ASCII character. Example: "www.google.com", "61.219.144.201"</p> <p>[ IN ] <u><i>usPort</i></u> The remote port number.</p>						
<b>Return Value</b>	<table border="1"> <thead> <tr> <th><b>Constants</b></th><th><b>Value</b></th></tr> </thead> <tbody> <tr> <td>d_OK</td><td>0000h</td></tr> <tr> <td><a href="#">GPRS error codes</a></td><td>23xxh</td></tr> </tbody> </table>	<b>Constants</b>	<b>Value</b>	d_OK	0000h	<a href="#">GPRS error codes</a>	23xxh
<b>Constants</b>	<b>Value</b>						
d_OK	0000h						
<a href="#">GPRS error codes</a>	23xxh						

## CTOS\_TCP\_GPRSPing

---

```
USHORT CTOS_TCP_GPRSPing (STR *strIPURL, BYTE bTimeout);
```

**Description** Test the reachability of a destination IP address on the network.

**Parameters** [ IN ] *strIPURL*

URL or IP address in ASCII character.

Example: "www.google.com", "61.219.144.201"

[ IN ] *bTimeout*

Timeout in second.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">GPRS error codes</a>	23xxh

## CTOS\_TCP\_GPRSURL2IP

---

```
USHORT CTOS_TCP_GPRSURL2IP (STR* strURL, BYTE* baIP);
```

**Description**      Query IP address for specific URL from DNS server.

**Parameters**      [ IN ]    *strURL*  
                          URL to query in ASCII character.  
                          Example: "www.google.com".

                          [ OUT ] *baIP*  
                          Response IP address, 4 bytes.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">GPRS error codes</a>	23xxh

## 2.27. File System Functions

([C/C++ file input/output](#) functions can be used instead)

- CTOS\_FileOpen
- CTOS\_FileClose
- CTOS\_FileDelete
- CTOS\_FileGetSize
- CTOS\_FileSeek
- CTOS\_FileRead
- CTOS\_FileWrite
- CTOS\_FileDir
- CTOS\_FileCut
- CTOS\_FileRename
- CTOS\_FileGetPosition
- CTOS\_FileReopen
- CTOS\_FileSetAttrib
- CTOS\_FileDirAttrib
- CTOS\_FileGetAttrib
- CTOS\_FileFormat
- CTOS\_FileDirA
- CTOS\_FileGetFreeSpace
- CTOS\_FileOpenAttrib

### File System Error Codes

<b>Constants</b>	<b>Value</b>
d_FS_INVALID_PARAMETER	2001h
d_FS_FHT_FULL	2002h
d_FS_FILE_ALREADY_OPENED	2003h
d_FS_FILE_NOT_OPENED	2004h
d_FS_FILE_NOT_FOUND	2005h
d_FS_FILE_INVALID_HANDLE	2006h
d_FS_DATA_FULL	2007h
d_FS_NOT_INITIALED	2008h
d_FS_CHECKSUM_ERROR	2009h
d_FS_FILE_ALREADY_EXISTED	200Ah
d_FS_NOT_OWNER	200Bh
d_FS_OPEN_FAILED	200Ch
d_FS_FUNCTION_NOT_SUPPORTED	200Fh
d_FS_STORAGE_TYPE_NOT_SUPPORTED	2010h

d_FS_OPERATION_ERROR	20FFh
----------------------	-------

## CTOS\_FileOpen

---

```
USHORT CTOS_FileOpen(BYTE*caFileName, BYTE bStorageType,
ULONG*pulFileHandle);
```

**Description** Open a file and return the file handle. The handle should be used in subsequent calling to file I/O functions. If the file with the input file name doesn't exist, a new file will be created.

**Parameters** [ IN ] *caFileName*  
Name of file.

[ IN ] *bStorageType*  
Specify where the file stored.  
• ***d\_STORAGE\_FLASH***  
File stored in flash

[ OUT ] *pulFileHandle*  
Pointer to ULONG to store the file handle.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">File System error codes</a>	20xxh

**Example** Please refer to example in appendix[B.3 File System](#)

## CTOS\_FileClose

---

```
USHORT CTOS_FileClose(ULONG ulFileHandle);
```

**Description** Close the previous opened file.

**Parameters** [ IN ] *ulFileHandle*  
Handle of the file.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">File System error codes</a>	20xxh

**Example** Please refer to example in appendix [B.3 File System](#)

## CTOS\_FileDelete

---

USHORT CTOS\_FileDelete (BYTE\*caFileName) ;

**Description** Delete a file with the specific file name.

**Parameters** [ IN ] *caFileName*  
Name of the file.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">File System error codes</a>	20xxh

**Example** Please refer to example in appendix [B.3 File System](#)

## CTOS\_FileGetSize

---

```
USHORT CTOS_FileGetSize(BYTE*caFileName, ULONG*pulFileSize);
```

**Description** Get the size of file with the specific file name.

**Parameters** [ IN ] *caFileName*

Name of file.

[ OUT ] *pulFileSize*

Pointer to ULONG to store the size of the file.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">File System error codes</a>	20xxh

**Example**

Please refer to example in appendix [B.3 File System](#)

## CTOS\_FileSeek

---

```
USHORT CTOS_FileSeek(ULONG ulFileHandle, ULONG ulOffset, BYTE
bOrigin);
```

**Description** Move file pointer to a specific position in the file.

**Parameters** [ IN ] *ulFileHandle*  
Handle of file.

[ IN ] *ulOffset*  
Number of bytes to move.

[ IN ] *bOrigin*  
Start position to move.

- *d\_SEEK\_FROM\_BEGINNING*

Move forward from the beginning of the file.

- *d\_SEEK\_FROM\_CURRENT*

Move forward from the current position.

- *d\_SEEK\_FROM\_EOF*

Move backward from the end of the file.

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">File System error codes</a>	20xxh

**Example**

Please refer to example in appendix [B.3 File System](#)

## CTOS\_FileRead

---

```
USHORT CTOS_FileRead(ULONG ulFileHandle, BYTE*baBuffer,  
ULONG*pulActualLength);
```

**Description** Read data from the file.

**Parameters** [ IN ] *ulFileHandle*  
Handle of file.

[ OUT ] *baBuffer*  
Buffer to store the data read from the file.

[ IN ] *pulActualLength*  
Size of buffer. Maximum number of data to read.

[ OUT ] *pulActualLength*  
Length of data actually read from the file.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">File System error codes</a>	20xxh

**Example** Please refer to example in appendix [B.3 File System](#)

## CTOS\_FileWrite

---

```
USHORT CTOS_FileWrite(ULONG ulFileHandle, BYTE*baBuffer, ULONG  
ulBufferLength);
```

**Description** Write data to the file.

**Parameters** [ IN ] *ulFileHandle*  
Handle of file.

[ IN ] *baBuffer*  
Buffer storing the data to write to the file.

[ IN ] *ulBufferLength*  
Length of data.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">File System error codes</a>	20xxh

**Example** Please refer to example in appendix [B.3 File System](#)

## CTOS\_FileDir

---

```
USHORT CTOS_FileDir(BYTE *baFileBuf, ULONG *pulFileSize, USHORT
*pusLen);
```

**Description** List the name and length of files created by the application.

**Parameters**

[OUT] <i>baFileBuf</i>	Buffer to store the file names for each file, 15 bytes per file.
------------------------	--

[ OUT ] <i>pulFileSize</i>	Array of ULONG to store the file size for each file.
----------------------------	--

[ IN ] <i>pusLen</i>	Maximum number of file to list.
----------------------	---------------------------------

[ OUT ] <i>pusLen</i>	Number of file actually listed.
-----------------------	---------------------------------

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">File System error codes</a>	20xxh

**Example** Please refer to example in appendix [B.3 File System](#)

**Note** Programmer should specify the maximum number of file to list in pusLen, and prepare a buffer of size 15\*(maximum\_number) to store the name of file, and an ULONG array of size (maximum\_number) to store the length of file.

## CTOS\_FileCut

---

```
USHORT CTOS_FileCut (BYTE*caFileName, ULONG ulFileLen);
```

**Description** Cut the file to a specific file length. The content of file might be truncated to meet the input length.

**Parameters** [ IN ] *caFileName*  
Name of file.

[ IN ] *ulFileLen*  
Target file length.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">File System error codes</a>	20xxh

**Example** Please refer to example in appendix [B.3 File System](#)

## CTOS\_FileRename

---

USHORT CTOS\_FileRename (BYTE\*caFileName, BYTE\*caFileNewName);

**Description** Change name of a file.

**Parameters** [ IN ] *caFileName*  
Original name of file.

[ IN ] *caFileNewName*  
New name of file.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">File System error codes</a>	20xxh

**Example** Please refer to example in appendix [B.3 File System](#)

## CTOS\_FileGetPosition

---

```
USHORT CTOS_FileGetPosition(ULONG ulFileHandle, ULONG *pulPosition);
```

**Description** Get current position of file pointer in the file.

**Parameters** [ IN ] *ulFileHandle*  
Handle of file.

[ OUT ] *pulPosition*  
Pointer to ULONG to store the position of file pointer.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">File System error codes</a>	20xxh

**Example** Please refer to example in appendix [B.3 File System](#)

## CTOS\_FileReopen

---

```
USHORT CTOS_FileReopen (BYTE*caFileName, BYTE bStorageType,
ULONG*pulFileHandle);
```

**Description** Reopen a existed file with specific file name.

**Parameters** [ IN ] *caFileName*  
Name of file.

[ IN ] *bStorageType*  
Specify where the file stored.  
• *d\_STORAGE\_FLASH*  
File stored in flash

[ OUT ] *pulFileHandle*  
Pointer to ULONG to store the handle of file.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">File System error codes</a>	20xxh

**Example** Please refer to example in appendix [B.3 File System](#)

**Note** The file can be used to test the existence of a file.

## CTOS\_FileSetAttrib

---

```
USHORT CTOS_FileSetAttrib(BYTE*caFileName, BYTE bAttrib);
```

**Description** Set attribute of a file with the specific file name.

**Parameters** [ IN ] *caFileName*

Name of file.

[ IN ] *bAttrib*

Attribute of file.

- *d\_FA\_PRIVATE*

Allow only the application itself can access  
(read/write) the file.

- *d\_FA\_PUBLIC*

Allow other application to access (read/write) the file.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">File System error codes</a>	20xxh

## CTOS\_FileDirAttrib

---

```
USHORT CTOS_FileDirAttrib(CTOS_FILE_ATTRIB *pstFA, USHORT  
*pusFileName);
```

**Description** List the attributes of files created by the application.

**Parameters**

[ OUT]	<i>pstFA</i>	Pointer to a <b>CTOS_FILE_ATTRIB</b> array to store the attributes of file.
[ IN ]	<i>pusFileName</i>	Maximum number of file attribute to list.
[ OUT ]	<i>pusFileName</i>	Number of file attribute actually listed.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">File System error codes</a>	20xxh

## CTOS\_FileGetAttrib

---

```
USHORT CTOS_FileGetAttrib(BYTE*caFileName, CTOS_FILE_ATTRIB *pstFA);
```

**Description** Get the attribute of a file with the specific file name.

**Parameters** [ IN ] *caFileName*

Name of file.

[ OUT ] *pstFA*

Point to **CTOS\_FILE\_ATTRIB** to store the attribute of file.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">File System error codes</a>	20xxh

## CTOS\_FileFormat

---

```
USHORT CTOS_FileFormat (BYTE bType) ;
```

**Description** Delete all files created by the application and release the memory space.

**Parameters** [ IN ] *bType*  
RFU.

Return Value	<i>Constants</i>	<i>Value</i>
	d_OK	0000h
	<a href="#">File System error codes</a>	20xxh

**Example** Please refer to example in appendix [B.3 File System](#)

## CTOS\_FileDirA

---

```
USHORT CTOS_FileDirA(CTOS_FILE_INFO *pstInfo , USHORT *pusLen);
```

**Description** List the information of files created by the application.

**Parameters**

[ OUT] <u><i>pstInfo</i></u>	Point to a <b>CTOS_FILE_INFO</b> array to store the information of file.
[ IN ] <u><i>usLen</i></u>	Maximum number of file information to list.
[ OUT ] <u><i>usLen</i></u>	Number of file information actually listed.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">File System error codes</a>	20xxh

## CTOS\_FileGetFreeSpace

---

```
USHORT CTOS_FileGetFreeSpace(BYTE bStorageType, ULONG*pulFreeSize);
```

**Description** Get the size of free space.

**Parameters** [ IN ] *bStorageType*

The storage type.

- **d\_STORAGE\_FLASH**

File stored in flash

[ OUT ] *pulFreeSize*

Pointer to ULONG to store the size of free space.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">File System error codes</a>	20xxh

**Example**

Please refer to example in appendix [B.3 File System](#)

## **CTOS\_FileOpenAttrib**

---

```
USHORT CTOS_FileOpenAttrib(BYTE*caFileName, BYTE bStorageType,
ULONG*pulFileHandle, BYTE bAttrib);
```

<b>Description</b>	Open a file with specific name and attribute and return the file handle. The handle should be used in subsequent calling to file I/O functions. If the file with the input file name doesn't exist, a new file will be created. The function is a combination of <b>CTOS_FileOpen()</b> and <b>CTOS_FileSetAttrib()</b> .						
<b>Parameters</b>	<p>[ IN ]    <u><i>caFileName</i></u> Name of file.</p> <p>[ IN ]    <u><i>bStorageType</i></u> Specify where the file stored. <ul style="list-style-type: none"> <li>• <b><i>d_STORAGE_FLASH</i></b> File stored in flash.</li> </ul> </p> <p>[ OUT ] <u><i>pulFileHandle</i></u> Pointer to ULONG to store the file handle.</p> <p>[ IN ]    <u><i>bAttrib</i></u> Attribute of file. <ul style="list-style-type: none"> <li>• <b><i>d_FA_PRIVATE</i></b> Allow only the application itself can access (read/write) the file.</li> <li>• <b><i>d_FA_PUBLIC</i></b> Allow other application to access (read/write) the file.</li> </ul> </p>						
<b>Return Value</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;"><b>Constants</b></th> <th style="text-align: right; padding: 2px;"><b>Value</b></th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">d_OK</td> <td style="text-align: right; padding: 2px;">0000h</td> </tr> <tr> <td style="padding: 2px;"><a href="#">File System error codes</a></td> <td style="text-align: right; padding: 2px;">20xxh</td> </tr> </tbody> </table>	<b>Constants</b>	<b>Value</b>	d_OK	0000h	<a href="#">File System error codes</a>	20xxh
<b>Constants</b>	<b>Value</b>						
d_OK	0000h						
<a href="#">File System error codes</a>	20xxh						

## 2.28. User Interface Functions

- CTOS\_LCDGMenu
- CTOS\_LCDGMenuEx
- CTOS\_LCDTMenu
- CTOS\_UIKeypad

### User Interface Error Codes

<b>Constants</b>	<b>Value</b>
d_MENU_LCD_PAGE_OVERFLOW	1B01h
d_MENU_LCD_NOT_SUPPORT	1B02h

## CTOS\_LCDGMenu

---

```
USHORT CTOS_LCDGMenu(BYTE bAttribute, STR *pbaHeaderString, STR  
*pcaItemString);
```

**Description** Show menu on the LCD display and return one item in the menu. The function is for "Graphic Mode" of LCD.

<b>Parameters</b>	[ IN ] <b><u>bAttribute</u></b>
	Attribute of the header string. Value can be the bitwise-OR of following definition.
	<ul style="list-style-type: none"><li>• <b><u>_taNormal</u></b> Normal mode, the header string is showed aligning to the left of the LCD display.</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>_taReverse</u></b> Set the reverse attribute of the header string.</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>_taRight</u></b> The header string is showed aligning to the right of the LCD display.</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>_taCenter</u></b> The header string is showed at the center of the LCD display.</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>_taErrCode</u></b> RFU</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>_taDelay</u></b> RFU</li></ul>
	<p><i>_taNormal,_taRight and _taCenter are exclusive to each other.</i></p>

[ IN ] *pbaHeaderString*

Header string.

[ IN ] *pcaItemString*

Items string.

**Return Value**

- **0xFF**

Cancel key is pressed.

- ***Index of item selected.***

Menu item

- ***Others***

Please refer to [User Interface error codes](#)

**Note**

Page 1:

1. New File  
2. Del File  
3. Directory  
4. Read  
5. Write  
6. Rename File  
→ 1/2

Page 2:

1. Cut Content  
2. Set Attrib  
3. Get Attrib  
4. Dir Attrib  
5. Format  
← ... 2/2

[Function Usage]

The attribute of the header string "File System" in the sample is (\_taCenter | \_taReverse).

The items string to input is declared as follow,

```
STR baltemString[] = {  
    "New File\n"  
    "Del File\n"
```

```
"Directory\n"
"Read\n"
"Write\n"
"Rename File\n"
"Cut Content\n"
"Set Attrib\n"
"Get Attrib\n"
"Dir Attrib\n"
"Format\n");
```

Call the function in the below way to display the menu.

```
usItem = CTOS_LCDGMenu(_taCenter|_taReverse, "File System",
balItemString);
```

#### [Menu Usage]

1. Press "Up" or "Down" key to select pages.
2. Press "1" to "6" key to select item. The index of the item selected will be returned.  
For example, if "New File" is selected, 1 will be returned, if "Rename File" is selected, 6 will be returned, and if "Cut Content" is selected, 7 will be returned.
3. Press "Cancel", and 0xFF will be returned.

Note. The max number of page for displaying menu is 9 pages.

## CTOS\_LCDGMenuEx

---

```
USHORT CTOS_LCDGMenuEx(BYTE bAttribute, STR *pbaHeaderString, STR  
*pcaItemString, USHORT usShowItem);
```

**Description** Show menu on the LCD display and return one item in the menu. The function is for "Graphic Mode" of LCD.

<b>Parameters</b>	[ IN ] <b><u>bAttribute</u></b>
	Attribute of the header string. Value can be the bitwise-OR of following definition.
	<ul style="list-style-type: none"><li>• <b><u>_taNormal</u></b> Normal mode, the header string is showed aligning to the left of the LCD display.</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>_taReverse</u></b> Set the reverse attribute of the header string.</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>_taRight</u></b> The header string is showed aligning to the right of the LCD display.</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>_taCenter</u></b> The header string is showed at the center of the LCD display.</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>_taErrCode</u></b> RFU</li></ul>
	<ul style="list-style-type: none"><li>• <b><u>_taDelay</u></b> RFU</li></ul>
	<p><i>_taNormal,_taRight and _taCenter are exclusive to each other.</i></p>

[ IN ] *pbaHeaderString*

Header string.

[ IN ] *pcaItemString*

Items string.

[ IN ] *usShowItem*

Number of item to show on the Menu. User can input the same item string, but decide how many item to be showed with this parameter.

**Return Value**

- ***0xFF***

Cancel key is pressed.

- ***Index of item selected.***

Menu item

- ***Others***

Please refer to [User Interface error codes](#)

**Note**

Page 1:

FILE SYSTEM  
1. New File  
2. Del File  
3. Directory  
4. Read  
5. Write  
6. Rename File  
->1/2

Page 2:

FILE SYSTEM  
1. Cut Content  
2. Set Attrib  
3. Get Attrib  
4. Dir Attrib  
5. Format  
->2/2

[Function Usage]

The attribute of the header string "File System" in the sample is

(\_taCenter | \_taReverse).

The items string to input is declared as follow,

```
STR baltemString[] = {  
    "New File\n"  
    "Del File\n"  
    "Directory\n"  
    "Read\n"  
    "Write\n"  
    "Rename File\n"  
    "Cut Content\n"  
    "Set Attrib\n"  
    "Get Attrib\n"  
    "Dir Attrib\n"  
    "Format\n"};
```

Call the function in the below way to display whole the menu.

```
usItem = CTOS_LCDCMenuEX(_taCenter|_taReverse, "File System",  
baltemString,11);
```

#### [Menu Usage]

1. Press "Up" or "Down" key to select pages.
2. Press "1" to "6" key to select item. The index of the item selected will be returned.

For example, if "New File" is selected, 1 will be returned, if "Rename File" is selected, 6 will be returned, and if "Cut Content" is selected, 7 will be returned.

3. Press "Cancel", and 0xFF will be returned.

Note. The max number of page for displaying menu is 9 pages.

## CTOS\_LCDTMenu

---

```
USHORT CTOS_LCDTMenu(BYTE bAttribute, STR *pbaHeaderString, STR  
*pcaItemString);
```

**Description** Show menu on the LCD display and return one item in the menu. The function is for "Text Mode" of LCD.

<b>Parameters</b>	[ IN ] <u><i>bAttribute</i></u>
	Attribute of the header string. Value can be the bitwise-OR of following definition.
	<ul style="list-style-type: none"><li>• <u><i>_taNormal</i></u> Normal mode, the header string is showed aligning to the left of the LCD display.</li></ul>
	<ul style="list-style-type: none"><li>• <u><i>_taReverse</i></u> Set the reverse attribute of the header string.</li></ul>
	<ul style="list-style-type: none"><li>• <u><i>_taRight</i></u> The header string is showed aligning to the right of the LCD display.</li></ul>
	<ul style="list-style-type: none"><li>• <u><i>_taCenter</i></u> The header string is showed at the center of the LCD display.</li></ul>
	<ul style="list-style-type: none"><li>• <u><i>_taErrCode</i></u> RFU</li></ul>
	<ul style="list-style-type: none"><li>• <u><i>_taDelay</i></u> RFU</li></ul>
	<p><i>_taNormal,_taRight and _taCenter are exclusive to each other.</i></p>

[ IN ] *pbaHeaderString*

Header string.

[ IN ] *pcaItemString*

Items string.

**Return Value**

- **0xFF**

Cancel key is pressed.

- ***Index of item selected.***

Menu item

- ***Others***

Please refer to [User Interface error codes](#)

**Note**

Page 1:

1. New File  
2. Del File  
3. Directory  
4. Read  
5. Write  
6. Rename File  
→ 1/2

Page 2:

1. Cut Content  
2. Set Attrib  
3. Get Attrib  
4. Dir Attrib  
5. Format  
← ... 2/2

[Function Usage]

The attribute of the header string "File System" in the sample is (\_taCenter | \_taReverse).

The items string to input is declared as follow,

```
STR baltemString[] = {  
    "New File\n"  
    "Del File\n"
```

```
"Directory\n"
"Read\n"
"Write\n"
"Rename File\n"
"Cut Content\n"
"Set Attrib\n"
"Get Attrib\n"
"Dir Attrib\n"
"Format\n");
```

Call the function in the below way to display the menu.

```
usItem = CTOS_LCDTMenu(_taCenter|_taReverse, "File System",
balItemString);
```

#### [Menu Usage]

1. Press "Up" or "Down" key to select pages.
2. Press "1" to "6" key to select item. The index of the item selected will be returned.  
For example, if "New File" is selected, 1 will be returned, if "Rename File" is selected, 6 will be returned, and if "Cut Content" is selected, 7 will be returned.
3. Press "Cancel", and 0xFF will be returned.

Note. The max number of page for displaying menu is 9 pages.

## **CTOS\_UIKeypad**

---

```
USHORT CTOS_UIKeypad(USHORT usX, USHORT usY, STR
*pcaKeyboardLayout[], UCHAR ucCursorBlinkInterval, UCHAR
ucDelayToNextChar, BOOL boEnableCursorMove, BOOL boOneRadixPointOnly,
UCHAR ucDigitAfterRadixPoint, BYTE bPasswordMask, STR caData[],
UCHAR ucDataLen);
```

**Description** Allow user to enter alphabets and numbers in keypad similar to the way of typing SMS on mobile phone. Developer may define custom letter mapping to each key and may mask the data shown on the screen for privacy purpose. The function is the "Text Mode".

**Parameters** [ IN ] *usX*  
Horizontal cursor position for the first character.

[ IN ] *usY*  
Vertical cursor position for the first character.

[ IN ] *pcaKeyboardLayout*  
Assign letter mapping to each key. The array is stored in array of character string (\*pcaKeyboardLayout[]) and the array must have exactly 12 elements. Each element of the array is reflect to the following keys: d\_KBD\_0, d\_KBD\_1, d\_KBD\_2, d\_KBD\_3, d\_KBD\_4, d\_KBD\_5, d\_KBD\_6, d\_KBD\_7, d\_KBD\_8, d\_KBD\_9, d\_KBD\_00, d\_KBD\_DOT. If only one selection is assigned to the key, cursor will move to next character directly. If no character selection is assigned to the key, no action will be preformed and cursor will not move.

Example of English keyboard layout : STR  
\*keyboardLayoutEnglish[]={" 0", "qzQZ1", "abcABC2",
"defDEF3", "ghiGHI4", "jklJKL5", "mnoMNO6", "prsPRS7",
"tuvTUV8", "wxyWXY9", ":\|\?.,<>", ".!@#\$%^&\*()"};

d_KBD_1 [1]:"qzQZ1"	d_KBD_2 [2]:"abcABC2"	d_KBD_3 [3]:"defDEF3"
d_KBD_4 [4]:"ghiGHI4"	d_KBD_5 [5]:"jklJKL5"	d_KBD_6 [6]:"mnoMNO6"
d_KBD_7 [7]:"prsPRS7"	d_KBD_8 [8]:"tuvTUV8"	d_KBD_9 [9]:"wxyWXY9"
d_KBD_00 [10]:";/\?,<>"	d_KBD_0 [0]:" 0"	d_KBD_DOT [11]:"!@#\$%^&*()"

Example of numeric keyboard layout with radix point : STR  
 \*keyboardLayoutNumberWithRadixPoint[]={"0", "1", "2", "3",  
 "4", "5", "6", "7", "8", "9", "", ".};

d_KBD_1 [1]:"1"	d_KBD_2 [2]:"2"	d_KBD_3 [3]:"3"
d_KBD_4 [4]:"4"	d_KBD_5 [5]:"5"	d_KBD_6 [6]:"6"
d_KBD_7 [7]:"7"	d_KBD_8 [8]:"8"	d_KBD_9 [9]:"9"
d_KBD_00 [10]:""	d_KBD_0 [0]:"0"	d_KBD_DOT [11]:"."

Example of numeric keyboard layout without radix point :  
 STR \*keyboardLayoutNumber[]={"0", "1", "2", "3", "4", "5",  
 "6", "7", "8", "9", "", ""};

d_KBD_1 [1]:"1"	d_KBD_2 [2]:"2"	d_KBD_3 [3]:"3"
d_KBD_4 [4]:"4"	d_KBD_5 [5]:"5"	d_KBD_6 [6]:"6"
d_KBD_7 [7]:"7"	d_KBD_8 [8]:"8"	d_KBD_9 [9]:"9"
d_KBD_00 [10]:""	d_KBD_0 [0]:"0"	d_KBD_DOT [11]:""

[ IN ] [ucCursorBlinkInterval](#)

Cursor blink interval in 10ms.

[ IN ] [ucDelayToNextChar](#)

Time in 10ms for cursor to move to next character if no button is pressed.

[ IN ] [boEnableCursorMove](#)

Whether to enable the F3/F4 cursor move capability.

- **d\_TRUE**

Enable.

- **d\_FALSE**

Disable.

[ IN ] **boOneRadixPointOnly**

Whether to allow more than one radix point.

- **d\_TRUE**

Only one radix point is allowed, normally set the option for inputting a string of number. For example, "123.456".

- **d\_FALSE**

More than one radix point is allowed.

[ IN ] **ucDigitAfterRadixPoint**

Number of digit allowed to input after the radix point.

[ IN ] **bPasswordMask**

Mask user data with a character(eg: \*, X). "\0"(null) for plain text.

[ OUT ] **caData**

Pointer to store the user data retrieved. The char string is end with "\0".

[ IN ] **ucDataLen**

Length of baData. Must be greater then 2. The actual length user could type usDataLen-1. The last character is for "\0".

#### Return Value

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_MENU_LCD_PAGE_OVERFLOW	1B01h

**Note**

User may move the cursor right and left by F3 and F4 key and may use backspace key to delete character before cursor.

Example 1:

```
STR *keyboardLayoutEnglish[]={" 0", "qzQZ1", "abcABC2",
"defDEF3", "ghiGHI4", "jklJKL5", "mnoMNO6", "prsPRS7", "tuvTUV8",
"wxyWXY9", ":\|\?.,<>", ".!@#$%^&*()"};
CTOS_Keypad(4, 2, keyboardLayoutEnglish, 40, 80, d_TRUE,
d_FALSE, 0, "\0", baBuff, 9);
```



Example 2:

```
STR *keyboardLayoutEnglish[]={" 0", "qzQZ1", "abcABC2",
"defDEF3", "ghiGHI4", "jklJKL5", "mnoMNO6", "prsPRS7", "tuvTUV8",
"wxyWXY9", ":\|\?.,<>", ".!@#$%^&*()"};
CTOS_Keypad(4, 2, keyboardLayoutEnglish, 40, 80, d_TRUE,
d_FALSE, 0, "*", baBuff, 9);
```



Example 3:

```
STR *keyboardLayoutNumberWithRadixPoint[]={"0", "1", "2", "3", "4",
```

```
"5", "6", "7", "8", "9", "", ".");  
CTOS_LCDTPrintXY(1, 2, "USD$");  
CTOS_Keypad(5, 2, keyboardLayoutNumberWithRadixPoint, 40, 80,  
d_TRUE, d_TRUE, 2,"\\0", baBuff, 9);
```



## 2.29.TCP Setting Functions

- CTOS\_TCP\_SetConnectTO
- CTOS\_TCP\_GetConnectTO
- CTOS\_TCP\_SetRxAckTO
- CTOS\_TCP\_GetRxAckTO
- CTOS\_TCP\_SetRxTO
- CTOS\_TCP\_GetRxTO
- CTOS\_TCP\_SetRetryCounter
- CTOS\_TCP\_GetRetryCounter
- CTOS\_TCP\_BindToDevice
- CTOS\_TCP\_SelectDefaultDevice
- CTOS\_PPP\_SetTO
- CTOS\_PPP\_GetTO
- CTOS\_PPP\_SetRetryCounter
- CTOS\_PPP\_GetRetryCounter

### TCPSetting Error Codes

<b>Constants</b>	<b>Value</b>
d_TCP_IO_PROCESSING	2321h
d_TCP_IO_BUSY	2322h
d_TCP_URL_NOT_FOUND	2370h
d_TCP_PING_FAILED	2371h
d_TCP_PPP_CONNECTION_TERMINATED	2372h
d_TCP_SOCKET_FULL	2380h
d_TCP_CONNECTION_NOT_ESTABLISHED	2381h
d_TCP_BAD_FCS	2382h
d_TCP_PPP_TIMEOUT	2383h
d_TCP_PROTOCOL_ERROR	2384h
d_TCP_LENGTH_ERROR	2385h
d_TCP_PPP_SEND_TIMEOUT	2386h
d_TCP_PPP_SEND_ERROR	2387h
d_TCP_SOCKET_FAILED	2388h
d_TCP_SOCKET_IO_MODE_FAILED	2389h

d_TCP_RX_RECEIVE_TIMEOUT	238Ah
d_TCP_INVALID_PARA	238Bh
d_TCP_CONNECT_TIMEOUT	2390h
d_TCP_DISCONNECT_FAIL	2391h
d_TCP_RESET	2392h
d_TCP_SEQNENCE_INCORRECT	2393h
d_TCP_NO_SERVER	2394h
d_TCP_RETRANSMISSION	2395h
d_TCP_PROTOCOL	2396h
d_TCP_IP_FORMAT_WRONG	2397h
d_TCP_FORMAT_WRONG	2398h
d_TCP_SEND_TIMEOUT	2399h
d_TCP_NO_ACK	239Ah
d_TCP_BUF_FULL	239Bh
d_TCP_RECEIVE_NON_UDP_PACKAGE	239Ch
d_TCP_IP_ADDRESS_NOT_EXIST	239Dh
d_TCP_RECEIVE_DATA_FAILED	239Eh
d_TCP_SEND_DATA_FAILED	239Fh
d_TCP_LCP_TIMEOUT	23A0h
d_TCP_LCP_ACK	23A1h
d_TCP_LCP_NAK	23A2h
d_TCP_LCP_REJECT	23A3h
d_TCP_LCP_CODE_REJECT	23A4h
d_TCP_LCP_PROTO_REJECT	23A5h
d_TCP_LCP_TERM_REJECT	23A6h
d_TCP_PEER_LCP_ACK	23A7h
d_TCP_PEER_LCP_NAK	23A8h
d_TCP_PEER_LCP_REJ	23A9h
d_TCP_PEER_LCP_CODE_REJECT	23AAh
d_TCP_PEER_LCP_PROTO_REJECT	23ABh
d_TCP_PEER_LCP_TERM_REQUEST	23ACh
d_TCP_PEER_LCP_TERM_ACK	23ADh
d_TCP_SEND_EPIPE_FAILED	23AEh
d_TCP_RECEIVE_EPIPE_FAILED	23AFh
d_TCP_CHAP_TIMEOUT	23B0h

d_TCP_CHAP_RESPONSE	23B1h
d_TCP_CHAP_CODE_REJECT	23B2h
d_TCP_PEER_CHAP_SUCCESS	23B3h
d_TCP_PEER_CHAP_AUTH_FAIL	23B4h
d_TCP_IPCP_TIMEOUT	23C0h
d_TCP_IPCP_ACK	23C1h
d_TCP_IPCP_REJECT	23C2h
d_TCP_IPCP_CODE_REJECT	23C3h
d_TCP_IPCP_PROTO_REJECT	23C4h
d_TCP_PEER_IPCP_ACK	23C5h
d_TCP_PEER_IPCP_NAK	23C6h
d_TCP_IPCP_TOSS	23C7h
d_TCP_PEER_IPCP_REJ	23C8h
d_TCP_CCP_REJECT	23CAh
d_TCP_BIND_FAILED	23CBh
d_TCP_CHANNEL_TYPE_FAILED	23E0h
d_TCP_CHANNEL_OFFLINE	23E1h
d_TCP_USER_INTERRUPT	23FFh

## CTOS\_TCP\_SetConnectTO

---

```
USHORT CTOS_TCP_SetConnectTO(ULONG ulTime) ;
```

**Description** Set the timeout for the connect and disconnect functions of GPRS and Modem.

**Parameters** [ IN ] *ulTime*  
The timeout value in millisecond.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Example** Please refer to example in appendix[B.20 TCP Setting](#)

**Note** The default value is 22 seconds

## CTOS\_TCP\_GetConnectTO

---

```
USHORT CTOS_TCP_GetConnectTO(ULONG *ulTime);
```

**Description** Get the timeout for the connect and disconnect functions of GPRS and Modem.

**Parameters** [OUT] *ulTime*  
The timeout value in millisecond.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Example** Please refer to example in appendix [B.20 TCP Setting](#)

**Note** The default value is 22 seconds

## CTOS\_TCP\_SetRxAckTO

---

```
USHORT CTOS_TCP_SetRxAckTO(ULONG ulTime);
```

**Description** Set the timeout value for GPRS and Modem to receive the Ack package.

**Parameters** [ IN ] *ulTime*  
The timeout value in millisecond.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Note** Not supported in VEGA5000.

## CTOS\_TCP\_GetRxAckTO

---

```
USHORT CTOS_TCP_GetRxAckTO(ULONG *ulTime);
```

**Description** Get the timeout value for GPRS and Modem to receive the Ack package.

**Parameters** [OUT] *ulTime*  
The timeout value in millisecond.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Note** Not supported in VEGA5000.

## CTOS\_TCP\_SetRxTO

---

```
USHORT CTOS_TCP_SetRxTO(ULONG ultime);
```

**Description** Set the timeout value for GPRS and Modem to receive the data.

**Parameters** [ IN ] *ultime*  
The timeout value in millisecond.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Example** Please refer to example in appendix [B.20 TCP Setting](#)

**Note** The default value is 68 seconds

## CTOS\_TCP\_GetRxTO

---

```
USHORT CTOS_TCP_GetRxTO(ULONG *ulTime);
```

**Description** Get the timeout value for GPRS and Modem to receive the data.

**Parameters** [OUT] *ulTime*  
The timeout value in millisecond.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Example** Please refer to example in appendix [B.20 TCP Setting](#)

**Note** The default value is 68 seconds

## CTOS\_TCP\_SetRetryCounter

---

```
USHORT CTOS_TCP_SetRetryCounter(USHORT usTime);
```

**Description** Set the retry counter for connect, send, receive, and disconnect functions of GPRS and Modem.

**Parameters** [ IN ] *usTime*  
The number of times in retry counter.  
Please set this value to be larger than 2.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Example** Please refer to example in appendix [B.20 TCP Setting](#)

**Note** The default value is 2 (2 retries after the first time failed)

## CTOS\_TCP\_GetRetryCounter

---

```
USHORT CTOS_TCP_GetRetryCounter(USHORT *usTime);
```

**Description** Get the retry counter for connect, send, receive, and disconnect functions of GPRS and Modem.

**Parameters** [OUT] *usTime*  
The number of times in retry counter.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Example** Please refer to example in appendix [B.20 TCP Setting](#)

**Note** The default value is 2 (2 retries after the first time failed)

## CTOS\_TCP\_BindToDevice

---

USHORT CTOS\_TCP\_BindToDevice(INT iSocket, BYTE bDevice);

**Description** Bind the socket to the desired communication device. This function must be called before connect() function.

**Parameters**

[ IN ]	<i>iSocket</i>
	The value return from socket() function.

[ IN ]	<i>bDevice</i>
	Communication device id.
	<ul style="list-style-type: none"> <li>• <i>d_TCP_DEVICE_ETHERNET</i></li> <li>• <i>d_TCP_DEVICE_GPRS</i></li> <li>• <i>d_TCP_DEVICE_MODEM</i></li> </ul>

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">TCP Setting error codes</a>	23xxh

**Note** If user prefer to use linux socket() and connect() function, then user must use this function to select the communication channel.

## CTOS\_TCP\_SelectDefaultDevice

---

```
USHORT CTOS_TCP_SelectDefaultDevice(BYTE bDevice);
```

**Description** Set the default communication device for connect() function.

**Parameters** [ IN ] bDevice

Communication device id.

- **d\_TCP\_DEVICE\_ETHERNET**
- **d\_TCP\_DEVICE\_GPRS**
- **d\_TCP\_DEVICE\_MODEM**
- **d\_TCP\_DEVICE\_WIFI**

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">TCP Setting error codes</a>	23xxh

**Note**

If user prefer to use linux socket() and connect() function, then user must use this function to select the communication channel. User need to open device( ex: CTOS\_TCP\_GPRSOpen() ) before calling CTOS\_TCP\_SelectDefaultDevice() .

## CTOS\_PPP\_SetTO

---

```
USHORT CTOS_PPP_SetTO(ULONG ulTime);
```

**Description** Set the timeout for the PPP transactions in the Open functions of GPRS and Modem.

**Parameters** [ IN ] *ulTime*  
The timeout value in millisecond.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Example** Please refer to example in appendix [B.20 TCP Setting](#)

**Note** The default value is 60 seconds.

## CTOS\_PPP\_GetTO

---

```
USHORT CTOS_PPP_GetTO(ULONG *ulTime);
```

**Description** Get the timeout for the PPP transactions in the Open functions of GPRS and Modem.

**Parameters** [ OUT] *ulTime*  
The timeout value in millisecond.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Example** Please refer to example in appendix [B.20 TCP Setting](#)

**Note** The default value is 60 seconds.

## CTOS\_PPP\_SetRetryCounter

---

```
USHORT CTOS_PPP_SetRetryCounter(USHORT usTime);
```

**Description** Set the retry counter for the PPP transactions in the Open functions of GPRS and Modem.

**Parameters** [ IN ] *usTime*  
The number of times in retry counter.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Example** Please refer to example in appendix [B.20 TCP Setting](#)

**Note** The default value is 0 (no retries)

## CTOS\_PPP\_GetRetryCounter

---

```
USHORT CTOS_PPP_GetRetryCounter(USHORT *usTime);
```

**Description** Get the retry counter for the PPP transactions in the Open functions of GPRS and Modem.

**Parameters** [ OUT] *usTime*  
The number of times in retry counter.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">TCP Setting error codes</a>	23xxh

**Example** Please refer to example in appendix [B.20 TCP Setting](#)

**Note** The default value is 0 (no retries)

## 2.30.Wifi Functions

- CTOS\_WifiOpen
- CTOS\_WifiClose
- CTOS\_WifiScan
- CTOS\_WifiInfoGet
- CTOS\_WifiConnectAP
- CTOS\_WifiDisconnectAP
- CTOS\_WifiStatus
- CTOS\_WifiConfigSet
- CTOS\_WifiConfigGet
- CTOS\_WifiConnectAPEx
- CTOS\_WifiPing

### Wifi Error Codes

<b>Constants</b>	<b>Value</b>
d_WIFI_INVALID_PARA	4301h
d_WIFI_CONFIG_ERROR	4302h
d_WIFI_NOT_OPEN	4303h
d_WIFI_INSUFFICIENT_CMD_BUF	4305h
d_WIFI_CMD_NOT_SUCCESS	4307h
d_WIFI_IO_PROCESSING	6001h
d_WIFI_IO_TIMEOUT	6002h
d_WIFI_IO_CANCELED	6003h
d_WIFI_IO_SCANFAILED	6004h
d_WIFI_IO_APCONNECTFAILED	6005h

## CTOS\_WifiOpen

---

```
USHORT CTOS_WifiOpen(void);
```

**Description** Open the connection of Wifi.

Please call **CTOS\_WifiStatus()** to check whether this action is finished.

**Parameters** None

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">Wifi error codes</a>	43xxh 06xxh

## CTOS\_WifiClose

---

```
USHORT CTOS_WifiClose(void);
```

**Description** Close the connection of Wifi.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Wifi error codes</a>	43xxh 06xxh

## CTOS\_WifiScan

---

```
USHORT CTOS_WifiScan(void);
```

**Description** Scan the Wifi devices.

**Parameters** None.

Return Value	Constants	Value
	d_OK	0000h
	<a href="#">Wifi error codes</a>	43xxh 06xxh

## CTOS\_WifiInfoGet

---

```
USHORT CTOS_WifiInfoGet(BYTE* bNumInfo, CTOS_stWifiInfo** stInfo);
```

**Description** Get the information of Wifi devices.

**Parameters**

[ OUT ]	<u><a href="#">bNumInfo</a></u>
The number of detected Wifi devices.	

[ OUT ]	<u><a href="#">stInfo</a></u>
The stInfo structure stores the information of the Device.	

```
typedef struct
{
    BYTE Address[32];
    BYTE ESSID[36];
    BYTE Mode[64];
    BYTE Freq[64];
    BYTE Quality[16];
    BYTE SignalLv[16];
    BYTE NoiseLv[16];
    BYTE EncryptionKey[8];
    //-----
    BYTE Type_1[8];
    BYTE GroupCipher_1[8];
    BYTE PairwiseCiphers_1[8];
    BYTE Authentication_1[8];
    //-----
    BYTE Type_2[8];
    BYTE GroupCipher_2[8];
    BYTE PairwiseCiphers_2[8];
    BYTE Authentication_2[8];
}CTOS_stWifiInfo;
```

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<u><a href="#">Wifi error codes</a></u>	43xxh 06xxh

## CTOS\_WifiConnectAP

---

```
USHORT CTOS_WifiConnectAP(CTOS_stWifiInfo* stInfo, BYTE *baPassword,
USHORT usPasswordLen);
```

**Description** Connect to the specific Wifi device.

**Parameters** [ IN ] *stInfo*

The stInfo structure stores the information of the Device.

[ IN ] *baPassword*

The password of the device.

[ IN ] *usPasswordLen*

The password length of the device.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Wifi error codes</a>	43xxh 06xxh

## CTOS\_WifiDisconnectAP

---

```
USHORT CTOS_WifiDisconnectAP(void);
```

<b>Description</b>	Disconnect to the remote.  Please call <b>CTOS_WifiStatus()</b> to check whether this action is finished.
<b>Parameters</b>	None.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Wifi error codes</a>	43xxh 06xxh

## CTOS\_WifiStatus

---

USHORT CTOS\_WifiStatus (DWORD\* pdwState);

**Description** Get the current processing status when calling the Wifi.

**Parameters** [OUT] *pdwState*

The current processing status.

- ***d\_WIFI\_STATE\_AP\_CONNECTED***
- ***d\_WIFI\_STATE\_SCANNING***
- ***d\_WIFI\_STATE\_AP\_CONNECTING***
- ***d\_WIFI\_STATE\_CONNECTING***
- ***d\_WIFI\_STATE\_SENDING***
- ***d\_WIFI\_STATE RECEIVING***
- ***d\_WIFI\_STATE\_DISCONNECTING***
- ***d\_WIFI\_STATE\_AP\_DISCONNECTING***

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Wifi error codes</a>	43xxh 06xxh

## CTOS\_WifiConfigSet

---

USHORT CTOS\_WifiConfigSet(BYTE bTag, BYTE\* baValue, BYTE bLen);

**Description** Set the configuration of the Wifi.

**Parameters** [ IN ] bTag

- **d\_WIFI\_CONFIG\_DHCP**

Set connection mode.

= 0 : No DHCP.

= 1 : DHCP.

- **d\_WIFI\_CONFIG\_AUTOCON\_AP**

Set the auto-conection to the Access Point.

= 0 : No auto-conection to the AP.

= 1 : Auto-conection to the AP.

- **d\_WIFI\_CONFIG\_IP**

Set the Wifi IP Adress.

- **d\_WIFI\_CONFIG\_MASK**

Set the Network MASK.

- **d\_WIFI\_CONFIG\_GATEWAY**

Set the GATEWAY Adress.

- **d\_WIFI\_CONFIG\_DNSIP**

Set the DNS Server IP Address.

- **d\_WIFI\_CONFIG\_MAC**

Set MAC Adress.

[ IN ] baValue

Configuration data to set. Data should be in the form of ASCII

characters.

[ IN ] *bLen*

Length of configuration data.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Wifi error codes</a>	43xxh 06xxh

**Remark**

The Wifi module can memorize the last network it connected to. If the parameter d\_WIFI\_CONFIG\_AUTOCON\_AP is set to 1 with CTOS\_WifiConfigSet() then the Wifi module will connect to that network automatically just after calling CTOS\_WifiOpen().

## CTOS\_WifiConfigGet

---

```
USHORT CTOS_WifiConfigGet(BYTE bTag, BYTE* baValue, BYTE *pbLen);
```

**Description** Get the configuration of the Wifi.

**Parameters** [ IN ] bTag

- **d\_WIFI\_CONFIG\_DHCP**

Set Connection Mode.

= 0 : Close DHCP.

= 1 : Open DHCP.

- **d\_WIFI\_CONFIG\_AUTOCON\_AP**

Set the auto-conection to AP.

= 0 : Close auto-conection to AP.

= 1 : Open auto-conection to AP.

- **d\_WIFI\_CONFIG\_IP**

Set the Wifi IP Adress.

- **d\_WIFI\_CONFIG\_MASK**

Set the Network MASK.

- **d\_WIFI\_CONFIG\_GATEWAY**

Set the GATEWAY Adress.

- **d\_WIFI\_CONFIG\_DNSIP**

Set the DNS Server IP Address.

- **d\_WIFI\_CONFIG\_MAC**

Set MAC Adress.

[ OUT ] baValue

Buffer to store the configuration data.

[ IN ]     *pusLen*

Size of buffer.

**Return Value**

<b><i>Constants</i></b>	<b><i>Value</i></b>
d_OK	0000h
<a href="#"><u>Wifi error codes</u></a>	43xxh 06xxh

## CTOS\_WifiConnectAPEx

---

```
USHORT CTOS_WifiConnectAPEx(BYTE *baSSid, USHORT usSSIDLen, BYTE
bProtocol, BYTE bPairwise, BYTE bGroup, BYTE *baPassword, USHORT
usPasswordLen);
```

**Description** Connect to the hidden Wifi device.

**Parameters** [ IN ] *baSSid*

The Service set identification (SSID) of the Wifi.

[ IN ] *usSSIDLen*

The length of the SSID.

[ IN ] *bProtocol*

The Protocol of Wifi.

- *d\_WIFI\_PROTOCOL\_WEP*
- *d\_WIFI\_PROTOCOL\_WPA*
- *d\_WIFI\_PROTOCOL\_WPA2*

[ IN ] *bPairwise*

The Pairwise of Wifi.

- *d\_WIFI\_PAIRWISE\_TKIP*
- *d\_WIFI\_PAIRWISE\_CCMP*
- *d\_WIFI\_PAIRWISE\_TKIPCCMP*

[ IN ] *bGroup*

The Group of Wifi.

- *d\_WIFI\_GROUP\_TKIP*
- *d\_WIFI\_GROUP\_CCMP*
- *d\_WIFI\_GROUP\_TKIPCCMP*

[ IN ] *baPassword*

The password of the device.

[ IN ] *usPasswordLen*

The password length of the device.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#"><u>Wifi error codes</u></a>	43xxh 06xxh

## CTOS\_WifiPing

---

```
USHORT CTOS_WifiPing(STR *strIPURL, BYTE bTimeout);
```

**Description** Test the reachability of a destination IP address on the network.

**Parameters**

[ IN ]	<u><i>strIPURL</i></u>
	Destination IP address in the form of ASCII character.
	Example: "192.120.100.168".
[ IN ]	<u><i>bTimeout</i></u>
	Ping Timeout.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Wifi error codes</a>	43xxh 06xxh

## 2.31.Touch Signature

- CTOS\_TouchSignatureStart
- CTOS\_GetSignatureStatus
- CTOS\_ToucSignatureConfigSet
- CTOS\_TouchSignatureTerminate
- CTOS\_BMPConverter

### Touch Signature Error Codes

<b>Constants</b>	<b>Value</b>
d_TOUCH_SIGNATURE_NOT_SUPPORT	3201h
d_TOUCH_SIGNATURE_INVALID_PARA	3202h
d_TOUCH_SIGNATURE_PROCESSING	3203h
d_TOUCH_SIGNATURE_NOT_PROCESSING	3204h
d_TOUCH_SIGNATURE_INSUFFICIENT_RESOURCE	3205h
d_BMP_INVALID_PARA	3301h
d_BMP_IFILE_NOT_FOUND	3302h
d_BMP_OPERATION_ERROR	3303h

## CTOS\_TouchSignatureStart

---

```
USHORT CTOS_TouchSignatureStart(ULONG ulXStart , ULONG ulYStart,
ULONG ulWidth, ULONG ulHeight, STR *strBMPFile,ULONG ulTimeout);
```

**Description** Start the Signature Capturing Process by preparing an area on touch panel and wait for user to sign in the area. The signature will be recorded to a BMP file.

**Parameters** [ IN ] [ulXStart](#)  
Starting X coordinate of touch panel for signature.

[ IN ] [ulYStart](#)  
Starting Y coordinate of touch panel for signature.

[ IN ] [ulWidth](#).  
Width of touch panel for signature.

[ IN ] [ulHeight](#)  
Height of touch panel for signature.

[ IN ] [strBMPFile](#)  
Name of BMP file to store the signature.

[ IN ] [ulTimeout](#)  
Th Timeout in 100 ms for terminating the Signature Capturing Process. 0x00000000 means the process will last infinitely.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
d_OK		0000h
<u><a href="#">Touch Signature error codes</a></u>		32xxh

**Note** This function is non-blocked, and will return immediate after the Signature Capturing Process is started. Programmer can call CTOS\_TouchSignatureTerminate() to terminate the signature

capturing process or the process will be terminated automatically after specified timeout is expired. User need to install Signature plugin before using this series API.

## CTOS\_GetSignatureStatus

---

```
USHORT CTOS_GetSignatureStatus(ULONG* pulStatus , ULONG
*pulDuration);
```

**Description** Get the status and duration of the status of Signature Capturing Process.

**Parameters** [ OUT ] *pulStatus*

Status of Signature Capturing Process.

- **d\_SC\_P\_STATUS\_NO\_DATA**
- **d\_SC\_P\_STATUS\_IDLE**
- **d\_SC\_P\_STATUS\_SIGNING**

[ OUT ] *pulDuration*

Duration of specific status lasted in 100ms.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Touch Signature error codes</a>	32xxh

**Note**

User could decide the scenario to terminate the Signature Capturing. Process with the status return by `CTOS_GetSignatureStatus()`. NO\_DATA status is not having any data of signing. IDLE status is already have one or more signing data but now is not signing. SIGNING status is mean signing now. User need to install Signature plugin before using this series API.

## CTOS\_ToucSignatureConfigSet

---

```
USHORT CTOS_ToucSignatureConfigSet (BYTE bTag, STR *strValue, BYTE bLen);
```

**Description** Set configurations for signature.

**Parameters** [ IN ] bTag

Status of Signature Capturing Process.

- **d\_TOUCH\_SIGNATURE\_CONFIG\_COLOR\_RED**  
1: Color code of Red, default is 0.
- **d\_TOUCH\_SIGNATURE\_CONFIG\_COLOR\_GREEN**  
2: Color code of Green, default is 0.
- **d\_TOUCH\_SIGNATURE\_CONFIG\_COLOR\_BLUE**  
3: Color code of Blue, default is 0.
- **d\_TOUCH\_SIGNATURE\_CONFIG\_FONT\_WIDTH**  
4: Font width

[ IN ] strValue

Configuration data to set. Data should be in the form of ASCII characters, i.e. "255".

[ IN ] bLen

Length of configuration data.

Signature line width value in pixel.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Touch Signature error codes</a>	32xxh

**Note**

When changing signature color or width parameter, the same effect of LCD and BMP file. User need to install Signature plugin before using this series API.

## CTOS\_TouchSignatureTerminate

---

```
USHORT CTOS_TouchSignatureTerminate(void);
```

**Description** Terminate the Signature Capturing Process immediately.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Touch Signature error codes</a>	32xxh

**Note** Terminate for Signature Capturing Process created by CTOS\_TouchSignatureStart(). User need to install Signature plugin before using this series API.

## CTOS\_BMPConverter

---

```
USHORT CTOS_BMPConvert (STR *strInputBMPFile, STR *strOutputBMPFile,
BYTE bTag);
```

**Description** Convert BMP file. This API have two convert type , one is rotation and the other one is converting to mono color BMP format.

**Parameters** [ IN ] [strInputBMPFile](#)  
Original name of file.

[ OUT ] [strOutputBMPFile](#)  
New name of file.

[ IN ] [bTag](#)  
Status of Signature Capturing Process.  

- **[d\\_BMP\\_CONVERT\\_ONE\\_BIT\\_COLOR](#)**
  - 1: To 1 bit color.***
  - 2: Right rotate 90 degrees.***
  - 3: Left rotate 90 degrees.***

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<u><a href="#">Touch Signature error codes</a></u>	32xxh

**Note** The BMP file retrieved from Signature Capturing Process is in full color, while printer accepts only mono (1-bit) color BMP. This function is provided as a tool to convert the full color BMP to mono color. User need to install Signature plugin before using this series API.

## 2.32. Bluetooth Functions

- CTOS\_BluetoothConfigSet
- CTOS\_BluetoothConfigGet
- CTOS\_BluetoothTxReady
- CTOS\_BluetoothTxData
- CTOS\_BluetoothRxReady
- CTOS\_BluetoothRxData
- CTOS\_BluetoothDisconnect
- CTOS\_BluetoothStatus
- CTOS\_BluetoothListen
- CTOS\_BluetoothOpen
- CTOS\_BluetoothClose

### Bluetooth Error Codes

<b>Constants</b>	<b>Value</b>
d_BLUETOOTH_INVALID_PARA	4501h
d_BLUETOOTH_CONFIG_ERROR	4502h
d_BLUETOOTH_TIMEOUT	4503h
d_BLUETOOTH_SOCKET_ALREADYOPENED	4504h
d_BLUETOOTH_INSUFFICIENT_CMD_BUF	4505h
d_BLUETOOTH_CMD_NOT_SUCCESS	4507h

## CTOS\_BluetoothConfigSet

---

```
USHORT CTOS_BluetoothConfigSet(BYTE bTag, BYTE* baValue, USHORT usLen);
```

**Description** Set the configuration of the Bluetooth.

**Parameters** [ IN ] bTag

- **d\_BLUETOOTH\_CONFIG\_AUTOCON**  
Set connection mode.  
= 0 : Close auto-connect.  
= 1 : Open auto-connect.
- **d\_BLUETOOTH\_CONFIG\_DEVICE\_NAME**  
Set the name of the device.
- **d\_BLUETOOTH\_CONFIG\_PASSKEY**  
Set the pass key of the device.
- **d\_BLUETOOTH\_CONFIG\_SECURE**  
Set Secure Simple Pairing (SSP).  
= 0 : Close SSP.  
= 1 : Open SSP.

[ IN ] baValue

Configuration data to set. Data should be in the form of ASCII characters.

[ IN ] usLen

Length of configuration data.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Bluetooth error codes</a>	45xxh

## CTOS\_BluetoothConfigGet

---

```
USHORT CTOS_BluetoothConfigGet (BYTE bTag, BYTE* baValue, USHORT  
*pusLen);
```

**Description** Get the configuration of the Bluetooth.

**Parameters** [ IN ] bTag

- **d\_BLUETOOTH\_CONFIG\_AUTOCON**

Get connection mode.

= 0 : Close auto-connect.

= 1 : Open auto-connect.

- **d\_BLUETOOTH\_CONFIG\_DEVICE\_NAME**

Get the name of the device.

- **d\_BLUETOOTH\_CONFIG\_PASSKEY**

Get the pass key of the device.

- **d\_BLUETOOTH\_CONFIG\_SECURE**

Get Secure Simple Pairing (SSP).

= 0 : Close SSP.

= 1 : Open SSP.

- **d\_BLUETOOTH\_CONFIG\_MAC**

Get the MAC of the device.

[ OUT ] baValue

Buffer to store the configuration data.

[ IN ] pusLen

Size of buffer.

[ OUT ] *pusLen*

Length of configuration data returned.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#"><u>Bluetooth error codes</u></a>	45xxh

## CTOS\_BluetoothTxReady

---

```
USHORT CTOS_BluetoothTxReady(void);
```

**Description** Check if the Bluetooth is ready for sending next data.

**Parameters** None.

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">Bluetooth error codes</a>	45xxh

## CTOS\_BluetoothTxData

---

```
USHORT CTOS_BluetoothTxData(BYTE* baData, USHORT usLen);
```

**Description** Send data through the Bluetooth.

**Parameters** [ IN ] *baData*  
Data to send.

[ IN ] *usLen*  
Length of data. The maximum value is 2048.

<b>Return Value</b>	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Bluetooth error codes</a>	45xxh

**Note** Before sending any data, programmer should call **CTOS\_BluetoothTxReady ()** to check whether the port is ready.

## CTOS\_BluetoothRxReady

---

```
USHORT CTOS_BluetoothRxReady(USHORT* pusLen);
```

**Description** Get the number of data currently received from the Bluetooth.

**Parameters** [ OUT ] *pusLen*  
Length of data received.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Bluetooth error codes</a>	45xxh

## CTOS\_BluetoothRxData

---

USHORT CTOS\_BluetoothRxData (BYTE\* baData, USHORT\* pusLen);

**Description** Get the data received from the Bluetooth.

**Parameters** [ OUT] *baData*  
Buffer to store the received data.

[ IN ] *pusLen*  
Size of buffer. Maximum number of data to get.

[ OUT ] *pusLen*  
Length of the data actually returned in the buffer.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	<a href="#">Bluetooth error codes</a>	45xxh

**Note** Programmer can call **CTOS\_BluetoothRxReady()** to check the number of data currently received, and then call this function with corresponding size of buffer get the data.  
If the size input is smaller than the actual number of data received, then only the size number of data will be returned.  
If the size input is bigger than the actual number of data, then the actual number of data returned will be stored in pusLen.

## CTOS\_BluetoothDisconnect

---

```
USHORT CTOS_BluetoothDisconnect(void);
```

**Description** Disconnect to the remote.

Please call **CTOS\_BluetoothStatus()** to check whether this action is finished.

**Parameters** None.

**Return Value**

<i>Constants</i>	<i>Value</i>
d_OK	0000h
<a href="#">Bluetooth error codes</a>	45xxh

## CTOS\_BluetoothStatus

---

```
USHORT CTOS_BluetoothStatus (DWORD* pdwState);
```

**Description** Get the current processing status when calling the Bluetooth.

**Parameters** [OUT] *pdwState*

The current processing status.

- ***BLUETOOTH\_STATE\_CONNECTED***
- ***BLUETOOTH\_STATE\_LISTENING***
- ***BLUETOOTH\_STATE\_SENDING***

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Bluetooth error codes</a>	45xxh

## CTOS\_BluetoothListen

---

```
USHORT CTOS_BluetoothListen(BYTE* baServiceName, BYTE* baUUID);
```

**Description** Enable the listening mode of Bluetooth.

**Parameters** [ IN ] *baServiceName*  
Name of the service.

[ IN ] *baUUID*  
Universally Unique Identifier (UUID)

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Bluetooth error codes</a>	45xxh

## CTOS\_BluetoothUnlisten

---

```
USHORT CTOS_BluetoothUnlisten(void);
```

**Description** Disable the listening mode of Bluetooth.

**Parameters** None.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Bluetooth error codes</a>	45xxh

## CTOS\_BluetoothOpen

---

```
USHORT CTOS_BluetoothOpen(void);
```

**Description** Open the connection of Bluetooth.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Bluetooth error codes</a>	45xxh

## CTOS\_BluetoothClose

---

```
USHORT CTOS_BluetoothClose(void);
```

**Description** Close the connection of Bluetooth.

**Parameters** None

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
<a href="#">Bluetooth error codes</a>	45xxh

## CTOS\_BluetoothPairedListGet

---

```
USHORT CTOS_BluetoothPairedListGet(BYTE *pbNum, stDeviceInfo
**sppDeviceInfo);
```

**Description** Retrieve the paired list of Bluetooth.

**Parameters** [ OUT ] *pbNum*

The number of paired devices in the paired list.

[ OUT ] *sppDeviceInfo*

The double pointer that point to the memory pointer of the paired device structure.

**Return Value**

<b>Constants</b>	<b>Value</b>
d_OK	0000h
d_BLUETOOTH_CMD_NOT_SUCCESS	4507h

**Note**

Usage :

BYTE bNum;

stDeviceInfo \*spDeviceInfo;

**CTOS\_BluetoothPairedListGet(&bNum, &spDeviceInfo);**

1<sup>st</sup> : spDeviceInfo->pbDeviceName;

2<sup>nd</sup>: (spDeviceInfo + 1)->pbDeviceName;

## CTOS\_BluetoothUnpair

---

```
USHORT CTOS_BluetoothUnpair(BYTE bIndex);
```

**Description** Unpair the device which is in the paired list.

**Parameters** [ IN ] bIndex  
The index of the devices in the paired list.

Return Value	<b>Constants</b>	<b>Value</b>
	d_OK	0000h
	d_BLUETOOTH_CMD_NOT_SUCCESS	4507h

**Note** The list could be gotten by **CTOS\_BluetoothPairedListGet**.

## Appendix A: Language Id

<b>Constants</b>	<b>Unicode Offset</b>
d_TTF_C0_CONTROLS	0000h
d_TTF_BASIC_LATIN	0020h
d_TTF_C1_CONTROLS	0080h
d_TTF_LATIN_1_SUPPLEMENT	00A0h
d_TTF_LATIN_EXTENDED_A	0100h
d_TTF_LATIN_EXTENDED_B	0180h
d_TTF_IPA_EXTENSIONS	0250h
d_TTF_SPACING_MODIFIERS	02B0h
d_TTF_COMBINING_DIACRITICS_MARKS	0300h
d_TTF_GREEK_AND_COPTIC	0370h
d_TTF_CYRILLIC	0400h
d_TTF_CYRILLIC_SUPPLEMENT	0500h
d_TTF_ARMENIAN	0530h
d_TTF_HEBREW	0590h
d_TTF_ARABIC	0600h
d_TTF_SYRIAC	0700h
d_TTF_ARABIC_SUPPLEMENT	0750h
d_TTF_THAANA	0780h
d_TTF_N_KO	07C0h
d_TTF_SAMARITAN	0800h
d_TTF_MANDAIC	0840h
d_TTF_DEVANAGARI	0900h
d_TTF_BENGALI	0980h
d_TTF_GURMUKHI	0A00h
d_TTF_GUJARATI	0A80h
d_TTF_ORIYA	0B00h
d_TTF_TAMIL	0B80h
d_TTF_TELUGU	0C00h
d_TTF_KANNADA	0C80h

d_TTF_MALAYALAM	0D00h
d_TTF_SINHALA	0D80h
d_TTF_THAI	0E00h
d_TTF_LAOS	0E80h
d_TTF_TIBETAN	0F00h
d_TTF_MYANMAR	1000h
d_TTF_GEORGIAN	10A0h
d_TTF_HANGUL_JAMO	1100h
d_TTF_ETHIOPIC	1200h
d_TTF_ETHIOPIC_SUPPLEMENT	1380h
d_TTF_CHEROKEE	13A0h
d_TTF_UNIFIED_CANADIAN_ABORIGINAL_SYLLABICS	1400h
d_TTF_OGHAM	1680h
d_TTF_RUNIC	16A0h
d_TTF_TAGALOG	1700h
d_TTF_HANUNOO	1720h
d_TTF_BUHID	1740h
d_TTF_TAGBANWA	1760h
d_TTF_KHMER	1780h
d_TTF_MONGOLIAN	1800h
d_TTF_UNIFIED_CANADIAN_ABORIGINAL_SYLLABICS_EXTENDED	18B0h
d_TTF_LIMBU	1900h
d_TTF_TAI_LE	1950h
d_TTF_NEW_TAI_LUE	1980h
d_TTF_KHMER_SYMBOLS	19E0h
d_TTF_BUGINESE	1A00h
d_TTF_TAI_THAM	1A20h
d_TTF_BALINESE	1B00h
d_TTF_SUNDANESE	1B80h
d_TTF_BATAK	1BC0h
d_TTF_LEPCHA	1C00h
d_TTF_OL_CHIKI	1C50h
d_TTF_SUDANESE_SUPPLEMENT	1CC0h
d_TTF_VEDIC_EXTENSIONS	1CD0h
d_TTF_PHONETIC_EXTENSIONS	1D00h

d_TTF_PHONETIC_EXTENSIONS_SUPPLEMENT	1D80h
d_TTF_COMBINING_DIACRITICS_MARKS_SUPPLEMENT	1DC0h
d_TTF_LATIN_EXTENDED_ADDITIONAL	1E00h
d_TTF_GREEK_EXTENDED	1F00h
d_TTF_GENERAL_PUNCTUATION	2000h
d_TTF_SUPERSCRIPTS_AND_SUBSCRIPTS	2070h
d_TTF_CURRENCY_SYMBOLS	20A0h
d_TTF_COMBINING_DIACRITICS_MARKS_FOR_SYMBOLS	20D0h
d_TTF_LETTERLIKE_SYMBOLS	2100h
d_TTF_NUMBER_FORM	2150h
d_TTF_ARROWS	2190h
d_TTF_MATHEMATICAL_OPERATOR	2200h
d_TTF_MISCLENEOUS_TECHNICAL	2300h
d_TTF_CONTROL_PICTURES	2400h
d_TTF_OPTICAL_CHARACTER_RECOGNITION	2440h
d_TTF_ENCLOSURE_ALPHANUMERICS	2460h
d_TTF_BOX_DRAWING	2500h
d_TTF_BLOCK_ELEMENT	2580h
d_TTF_GEOMETRIC_SHAPES	25A0h
d_TTF_MISCLENEOUS_SYMBOLS	2600h
d_TTF_DINGBATS	2700h
d_TTF_MISCLENEOUS_MATHEMATICAL_SYMBOLS_A	27C0h
d_TTF_SUPPLEMENTAL_ARROWS_A	27F0h
d_TTF_BRAILLE_PATTERNS	2800h
d_TTF_SUPPLEMENTAL_ARROWS_B	2900h
d_TTF_MISCLENEOUS_MATHEMATICAL_SYMBOLS_B	2980h
d_TTF_SUPPLEMENTAL_MATHEMATICAL_OPERATOR	2A00h
d_TTF_MISCLENEOUS_SYMBOLS_AND_ARROWS	2B00h
d_TTF_GLAGOLITIC	2C00h
d_TTF_LATIN_EXTENDED_C	2C60h
d_TTF_COPTIC	2C80h
d_TTF_GEORGIAN_SUPPLEMENT	2D00h
d_TTF_TIFINAGH	2D30h
d_TTF_ETHIOPIC_EXTENDED	2D80h
d_TTF_SUPPLEMENTAL_PUNCTUATION	2E00h

d_TTF_CJK_RADICALS_SUPPLEMENT	2E80h
d_TTF_KANGXI_RADICALS	2F00h
d_TTFIDEOGRAPHIC_DESCRIPTION_CHARACTERS	2FF0h
d_TTF_CJK_SYMBOLS_AND_PUNCTUATION	3000h
d_TTF_HIRAGANA	3040h
d_TTF_KATAKANA	30A0h
d_TTF_BOPOMOFO	3100h
d_TTF_HANGUL_COMPATIBILITY_JAMO	3130h
d_TTF_KANBUN	3190h
d_TTF_BOPOMOFO_EXTENDED	31A0h
d_TTF_CJK_STROKES	31C0h
d_TTF_KATAKANA_PHONETIC_EXTENSIONS	31F0h
d_TTF_ENCLOSURE_CJK LETTERS_AND_MONTHS	3200h
d_TTF_CJK_COMPATIBILITY	3300h
d_TTF_CJK_UNIFIED_IDEOGRAPHS_EXTENSION_A	3400h
d_TTF_YIJING_HEXAGRAMS_SYMBOLS	4DC0h
d_TTF_CJK_UNIFIED_IDEOGRAPHS	4E00h
d_TTF_YI_SYLLABLES	A000h
d_TTF_YI_RADICALS	A490h
d_TTF_LISU	A4D0h
d_TTF_VAI	A500h
d_TTF_CYRILLIC_EXTENDED_B	A640h
d_TTF_BAMUM	A6A0h
d_TTF_MODIFIER_TONE LETTERS	A700h
d_TTF_LATIN_EXTENDED_D	A720h
d_TTF_SYLOTI_NAGRI	A800h
d_TTF_IND_NO	A830h
d_TTF_PHAGS_PA	A840h
d_TTF_SAURASHTRA	A880h
d_TTF_DEVA_EXT	A8E0h
d_TTF_KAYAH_LI	A900h
d_TTF_REJANG	A930h
d_TTF_JAVANESE	A980h
d_TTF_CHAM	AA00h
d_TTF_MYANMAR_EXTA	AA60h

d_TTF_TAI_VIET	AA80h
d_TTF_MEETEI_EXT	AAE0h
d_TTF_ETHIOPIC_EXT_A	AB00h
d_TTF_MEETEI_MAYEK	ABC0h
d_TTF_HANGUL_SYLLABLES	AC00h
d_TTF_HANGUL_JAMO_EXTENDED_B	D7B0h
d_TTF_HIGH_HALF_ZONE_OF_UTF_16	D800h
d_TTF_LOW_HALF_ZONE_OF_UTF_16	DC00h
d_TTF_PRIVATE_USE_ZONE	E000h
d_TTF_CJK_COMPATIBILITY_IDEOGRAPHS	F900h
d_TTF_ALPHABETIC_PRESENTATION_FORMS	FB00h
d_TTF_ARABIC_PRESENTATION_FORMS_A	FB50h
d_TTF_VARIATION_SELECTOR	FE00h
d_TTF_VERTICAL_FORMS	FE10h
d_TTF_COMBINING_HALF_MARKS	FE20h
d_TTF_CJK_COMPATIBILITY_FORMS	FE30h
d_TTF_SMALL_FORM_VARIANTS	FE50h
d_TTF_ARABIC_PRESENTATION_FORMS_B	FE70h
d_TTF_HALFWIDTH_AND_FULLWIDTH_FORMS	FF00h
d_TTF_SPECIALS	FFF0h

# Appendix B: Examples

## B.1. Encryption and Decryption

```
/*=====
* FILE NAME: Encryption / Decryption
* MODULE NAME: Encryption / Decryption
* PROGRAMMER: Peggy Chang
* DESCRIPTION: Test encryption & decryption functions
* REVISION: 01.00
=====*/
/*=====
*           I N C L U D E S           *
=====*/
#include <string.h>
#include <stdio.h>
#include <ctosapi.h>
#include <stdlib.h>
#include <stdarg.h>

// Private functions
unsigned int wub_find_str_end_pt(unsigned char *str)
{
    unsigned int i;
    i = 0;
    while (str[i] != 0x00)
        i++;
    return i;
}

unsigned char wub_hex_2_ascii(unsigned char hex)
{
    if (hex <= 9)
        return hex + '0';
    else
        return hex - 10 + 'A';
}

void wub_str_append_byte_hex(unsigned char *str, unsigned char dat)
{
    unsigned int i;
    i = wub_find_str_end_pt(str);
    str[i ++] = wub_hex_2_ascii(dat / 16);
    str[i ++] = wub_hex_2_ascii(dat % 16);
    str[i] = 0x00;
}

unsigned int wub_hex_2_str(unsigned char *hex, unsigned char *str, unsigned int len)
{
    unsigned int i;
    str[0] = 0;
    for (i = 0; i < len; i++)
        wub_str_append_byte_hex(str, hex[i]);
    return len * 2;
}

/*=====
*           D E F I N E S           *
=====*/
#define d_BUFF_SIZE 128 //Define buffer size

//Define Encryption Key
#define d_ISSUER_MMK "\x01\x02\x03\x04\x05\x06\x07\x08"
//The Encryption Key in DES or MAC
```

```

#define d_AES_KEY
"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x10\x11\x12\x13\x14\x15\x16"
//The Encryption Key in AES

#define d_ICV_KEY      "\x08\x07\x06\x05\x04\x03\x02\x01"
//The ICV Key in MAC

#define d_ENCRYPTION_DATA    "\x31\x32\x33\x34\x35\x36\x37\x38"
//Define the Encryption Data

#define d_LINE_DOT 12 //A line is 12 dots in Printer

//The Modulus parameter
const BYTE baModulus[] = {0xE4,0x3C,0x11,0x36,0xFB,0x66,0x3A,0x75,0xD9,0x13,
                         0x19,0x4C,0x2A,0x6F,0x06,0x5C,0xC9,0x69,0x44,0x16,
                         0x24,0x89,0x14,0x23,0x94,0x90,0x0E,0x0D,0xE4,0x98,
                         0xF8,0xCF,0x02,0xBC,0xB5,0x57,0x8E,0x57,0x4D,0xB8,
                         0x67,0x68,0x28,0x90,0x02,0x4E,0x98,0xA3,0xD4,0x68,
                         0xBC,0x87,0xBC,0xF5,0x50,0x19,0xDD,0x99,0xF1,0xE6,
                         0x62,0xE8,0xD4,0xD3
};

//The Deponent Parameter
const BYTE baDeponent[] = {0xCE,0xFB,0x12,0x79,0x94,0x60,0x4F,0x60,0x14,0xAE,
                           0xD7,0x60,0x55,0x93,0x3E,0x67,0xE3,0x58,0x8D,0xA0,
                           0xAF,0x1D,0x89,0xA0,0x02,0xD8,0xE9,0x85,0xFB,0xEC,
                           0x26,0xA6,0x8E,0xB0,0x44,0x55,0xDC,0x3B,0x99,0xFB,
                           0x4C,0x3C,0x1B,0x07,0x0A,0x6F,0xBF,0x71,0x52,0x5B,
                           0xB4,0x10,0xD5,0x02,0x78,0x01,0xBD,0xC6,0x6F,0x02,
                           0x3D,0xEB,0xAD,0x21
};

//The Exponent Parameter
const BYTE baExponent[] = {0x01,0x00,0x01};

/* =====
 * FUNCTION NAME: main
 * DESCRIPTION: function main is responsible for all the tasks listed in the
introduction to this demo.
 * RETURN:      none.
 * NOTES:       none.
 *===== */
int main(int argc,char *argv[])
{
    //Declare Local Variable //
    BYTE key;
    BYTE babuff[d_BUFF_SIZE];
    BYTE baRSA[d_BUFF_SIZE],baRNG[d_BUFF_SIZE],baDES[d_BUFF_SIZE];
    BYTE baAES[d_BUFF_SIZE],baMAC[d_BUFF_SIZE],baSHA[d_BUFF_SIZE];
    SHA_CTX SHA_CTX;
    int i;

    CTOS_LCDTClearDisplay ();

    CTOS_LCDTSetReverse(TRUE);
    CTOS_LCDTPrintXY(1,1,"  Encrpytion   ");
    CTOS_LCDTSetReverse(FALSE);

    /******
     * The RSA Encryption
     *****/
    CTOS_LCDTPrintXY(1,2,"RSA Encryption");
    memset(babuff,0x00,sizeof(babuff));
    memcpy(&babuff[sizeof(baModulus)-
strlen(d_ENCRYPTION_DATA)],d_ENCRYPTION_DATA,strlen(d_ENCRYPTION_DATA));

    //Perform the RSA exponent operation for encryption //
    CTOS_RSA((BYTE *)baModulus,sizeof(baModulus),(BYTE
*)baExponent,sizeof(baExponent),babuff,baRSA);

    //Print ant display the RSA result for encryption //
    CTOS_PrinterPutString("RSA Encryption");
    wub_hex_2_str(&baRSA[0],babuff,15); //Hex value change to string //
    CTOS_PrinterPutString(babuff);
    wub_hex_2_str(&baRSA[15],babuff,15);
    CTOS_PrinterPutString(babuff);
}

```

```

wub_hex_2_str(&baRSA[30],babuff,15);
CTOS_PrinterPutString(babuff);
wub_hex_2_str(&baRSA[45],babuff,15);
CTOS_PrinterPutString(babuff);
wub_hex_2_str(&baRSA[60],babuff,4);
CTOS_PrinterPutString(babuff);

*****  

*   The RSA Decryption           *  

*****  

CTOS_LCDTPrintXY(1,3,"RSA Decryption");

//Perform the RSA exponent operation for decryption //
CTOS_RSA((BYTE *)baModulus,sizeof(baModulus),(BYTE
*)baDeponent,sizeof(baDeponent),baRSA,babuff);

//Print and display the RSA result for decryption //
CTOS_PrinterPutString("\nRSA Decryption");
wub_hex_2_str(&babuff[0],baRSA,15);
CTOS_PrinterPutString(baRSA);
wub_hex_2_str(&babuff[15],baRSA,15);
CTOS_PrinterPutString(baRSA);
wub_hex_2_str(&babuff[30],baRSA,15);
CTOS_PrinterPutString(baRSA);
wub_hex_2_str(&babuff[45],baRSA,15);
CTOS_PrinterPutString(baRSA);
wub_hex_2_str(&babuff[60],baRSA,4);
CTOS_PrinterPutString(baRSA);

*****  

*   Generate the RNG           *  

*****  

CTOS_LCDTPrintXY(1,4,"RNG:");
//Generate 8 bytes random number
CTOS_RNG(baRNG);

//Print and display RNG result
wub_hex_2_str(baRNG,babuff,8);
CTOS_LCDTPrintXY(5,4,babuff);
CTOS_PrinterPutString("\nRNG");
CTOS_PrinterPutString(babuff);

*****  

*   The DES Encryption/Decryption      *  

*****  

CTOS_LCDTPrintXY(1,5,"DES:");

//This API perform the DES encryption/decryption calculation //
CTOS_DES(d_ENCRYPTION, d_ISSUER_MMK, 8, d_ENCRYPTION_DATA, 8, baDES);

//Print and display the DES result //
wub_hex_2_str(baDES,babuff,8);
CTOS_LCDTPrintXY(5,5,babuff);
CTOS_PrinterPutString("\nDES Encryption");
CTOS_PrinterPutString(babuff);

*****  

*   The AES Encryption/Decryption      *  

*****  

CTOS_LCDTPrintXY(1,6,"AES");
memset(babuff,0x00,sizeof(babuff));
memcpy(babuff,d_ENCRYPTION_DATA,8);
memcpy(&babuff[8],d_ENCRYPTION_DATA,8);

//Perform the AES encryption/decryption calculation //
CTOS_AES(d_ENCRYPTION, d_AES_KEY, babuff, 16, baAES);

//Print and display the AES result //
wub_hex_2_str(baAES,babuff,8);
CTOS_LCDTPrintXY(5,6,babuff);
CTOS_PrinterPutString("\nAES Encryption");
CTOS_PrinterPutString(babuff);
wub_hex_2_str(&baAES[8],babuff,8);
CTOS_PrinterPutString(babuff);

*****
```

```

*   The MAC Calculation
*****
CTOS_LCDTPrintXY(1,7,"MAC:");

//Perform MAC calculation //
CTOS_MAC(d_ISSUER_MMK, 8, d_ICV_KEY, d_ENCRYPTION_DATA, 8, baMAC);

//Print and display the MAC result //
wub_hex_2_str(baMAC,babuff,8);
CTOS_LCDTPrintXY(5,7,babuff);
CTOS_PrinterPutString("\nMAC Calculation");
CTOS_PrinterPutString(babuff);

*****
*   Generater the SHA1
*****
CTOS_LCDTPrintXY(1,8,"SHA1");

//Initialize the SHA_CTX structure and perpart for the SHA1 operation //
CTOS_SHA1Init(&SHA_CTX);

//Perform the SHA1 algorithm with the input data //
CTOS_SHA1Update(&SHA_CTX,d_ENCRYPTION_DATA,8);

//Finalize the SHA1 operation and retrun the result //
CTOS_SHA1Final(baSHA,&SHA_CTX);

//Print and display the SHA1 result //
wub_hex_2_str(baSHA,babuff,15);
CTOS_PrinterPutString("\nSHA1");
CTOS_PrinterPutString(babuff);
wub_hex_2_str(&baSHA[15],babuff,5);
CTOS_PrinterPutString(babuff);

CTOS_PrinterLine(d_LINE_DOT*10); //Print a space 1 line

CTOS_KBDGet ( &key );

return 0;
}

// eof

```

## B.2. Ethernet

```
/*
 * FILE NAME: Ethenent
 * MODULE NAME: Ethernet
 * PROGRAMMER: Peggy Chang
 * DESCRIPTION: Test Ethernet Functions
 * REVISION: 01.00
 */

=====
 *      I N C L U D E S      *
=====

#include <string.h>
#include <stdio.h>
#include <ctosapi.h>
#include <stdlib.h>
#include <stdarg.h>

=====
 *      D E F I N E S      *
=====

#define d_BUFF_SIZE 128           // Buff Size

//Define const Ethernet Data //
BYTE ip_addr[] = "192.120.100.220"; //IP Address
BYTE mask[] = "255.255.255.0"; //Mask
BYTE gate_ip[] = "192.120.100.200"; //Gateway IP
BYTE host_ip[] = "192.120.100.22"; //Default Host IP
BYTE host_port[] = "6666"; //Default Host Port
BYTE auto_connect[] = "0";
BYTE manual_connect[] = "2";
BYTE blank_str[] = " ";

//Declare Global Variable //
USHORT ret; //return Value
BYTE babuff[d_BUFF_SIZE]; //Global Buffer
BYTE key;

enum emKeyMode // Key Mode
{
    eMain,
    eConfig,
    eConnected,
    eTxData,
    eRxData,
};

enum emKeyMode emKM;

/*
 * FUNCTION NAME: show_screen
 * DESCRIPTION: Show on the LCD Display
 * RETURN: none.
 * NOTES: none.
 */
void show_screen(int tag)
{
    //Clear LCD Display //
    CTOS_LCDTClearDisplay();

    switch(tag)
    {
        case 0:
            CTOS_LCDTSetReverse(TRUE); //The reverse enable //
            CTOS_LCDTPrintXY(1, 1, " ETHERNET TEST "); //Display a string in the 1 line //
            CTOS_LCDTSetReverse(FALSE); //The reverse disable //
            CTOS_LCDTPrintXY(1, 2, "1.Config"); //Display a string in the 2 line //
            CTOS_LCDTPrintXY(1, 3, "2.Connect"); //Display a string in the 3 line //
            CTOS_LCDTPrintXY(1, 4, "3.Tx Data"); //Display a string in the 4 line //
            CTOS_LCDTPrintXY(1, 5, "4.Rx Data"); //Display a string in the 5 line //
    }
}
```

```

        CTOS_LCDTPrintXY(1, 6, "5.Get Status"); //Display a string in the 6 line //
        CTOS_LCDTPrintXY(1, 7, "6.Ping"); //Display a string in the 7 line //
        CTOS_LCDTPrintXY(1, 8, "7.Exit"); //Display a string in the 8 line //
        break;
    case 1:
        CTOS_LCDTSetReverse(TRUE);
        CTOS_LCDTPrintXY(1, 1, " Config Setting ");
        CTOS_LCDTSetReverse(FALSE);
        CTOS_LCDTPrintXY(1, 2, "1. Set");
        CTOS_LCDTPrintXY(1, 3, "2. Get");
        break;
    case 2:
        CTOS_LCDTSetReverse(TRUE);
        CTOS_LCDTPrintXY(1, 1, " Connected ");
        CTOS_LCDTSetReverse(FALSE);
        CTOS_LCDTPrintXY(1, 2, "1. Connect");
        CTOS_LCDTPrintXY(1, 3, "2. DisConnect");
        break;
    }
}

/* =====
 * FUNCTION NAME: EthernetSetConfig
 * DESCRIPTION:
 * RETURN:      none.
 * NOTES:       none.
 * =====*/
void EthernetSetConfig(void)
{
    BYTE buff[16+1];
    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, "--Ethernet Set--");

    /*Set up the configuration of Ethernet */

    // Set IP Address //
    memset(buff, 0, sizeof(buff));
    memcpy(buff, (BYTE*)ip_addr,strlen(ip_addr));
    ret = CTOS_EthernetConfigSet(d_ETHERNET_CONFIG_IP, buff, strlen(buff));
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY(1, 8, "Set IP Not OK");
        return ;
    }
    //CTOS_LCDTPrintXY(1, 2, "Set IP OK");
    CTOS_LCDTPrintXY(1, 2, buff);

    // Set Mask //
    memset(buff, 0, sizeof(buff));
    memcpy(buff, (BYTE*)mask,strlen(mask));
    ret = CTOS_EthernetConfigSet(d_ETHERNET_CONFIG_MASK, buff, strlen(buff));
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY(1, 8, "Set Mask Not OK");
        CTOS_Delay(1000);
        return ;
    }
    CTOS_LCDTPrintXY(1, 3, buff);

    // Set Geteway IP //
    memset(buff, 0, sizeof(buff));
    memcpy(buff, (BYTE*)gate_ip,strlen(gate_ip));
    ret = CTOS_EthernetConfigSet(d_ETHERNET_CONFIG_GATEWAY, buff, strlen(buff));
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY(1, 8, "Set GATE Not OK");
        return ;
    }
    CTOS_LCDTPrintXY(1, 4, buff);

    // Set Host IP //
    memset(buff, 0, sizeof(buff));
    memcpy(buff, (BYTE*)host_ip,strlen(host_ip));
    ret = CTOS_EthernetConfigSet(d_ETHERNET_CONFIG_HOSTIP, buff, strlen(buff));
    if(ret != d_OK)

```

```

{
    CTOS_LCDTPrintXY(1, 8, "Set H_IP Not OK");
    return ;
}
CTOS_LCDTPrintXY(1, 5, buff);

// Set Host Port //
memset(buff, 0, sizeof(buff));
memcpy(buff, (BYTE*)host_port, strlen(host_port));
ret = CTOS_EthernetConfigSet(d_ETHERNET_CONFIG_HOSTPORT, buff, strlen(buff));
if(ret != d_OK)
{
    CTOS_LCDTPrintXY(1, 8, "Set H_IP Not OK");
    return ;
}
CTOS_LCDTPrintXY(1, 6, buff);

memset(buff, 0, sizeof(buff));
buff[0] = 0x32;
ret = CTOS_EthernetConfigSet(d_ETHERNET_CONFIG_AUTOCON, buff, 1);
if(ret != d_OK)
{
    CTOS_LCDTPrintXY(1, 8, "Set M_Mod Not OK");
    return ;
}
CTOS_LCDTPrintXY(1, 7, buff);

buff[0] = 0x30;
ret = CTOS_EthernetConfigSet(d_ETHERNET_CONFIG_DHCP, buff, 1);
if(ret != d_OK)
{
    CTOS_LCDTPrintXY(1, 8, "Set M_Mod Not OK");
    return ;
}
CTOS_LCDTPrintXY(1, 7, buff);

ret = CTOS_EthernetConfigSet(d_ETHERNET_CONFIG_UPDATE_EXIT, babuff, 0);
if(ret != d_OK)
{
    CTOS_LCDTPrintXY(1, 8, "Save Exit Not OK");
    return ;
}
CTOS_LCDTPrintXY(1, 8, "Save Exit OK");
//CTOS_Delay(5000);
}

/* =====
* FUNCTION NAME: EthernetGetConfig
* DESCRIPTION: Get the Ethernet Configuration.
* RETURN: none.
* NOTES: none.
* =====*/
void EthernetGetConfig(void)
{
    //Declare Local Variable //
    BYTE bLength;

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, "--Ethernet Get--");

    // Get the configuration value of Ethernet //
    memset(babuff, 0x00, sizeof(babuff));
    bLength = sizeof(babuff);
    //Get the IP Address //
    ret = CTOS_EthernetConfigGet(d_ETHERNET_CONFIG_IP, babuff, &bLength);
    if (ret == d_OK)
    {
        //CTOS_LCDTPrintXY(1, 2, "IP Add:");
        CTOS_LCDTPrintXY(1, 2, babuff);
    }else{
        CTOS_LCDTPrintXY(1, 2, "IP Error");
    }

    memset(babuff, 0x00, sizeof(babuff));
    bLength = sizeof(babuff);
    //Get the mask //
}

```

```

ret = CTOS_EthernetConfigGet(d_ETHERNET_CONFIG_MASK, babuff, &bLength);
if (ret == d_OK)
{
    //CTOS_LCDTPrintXY(1, 3, "Mask:");
    CTOS_LCDTPrintXY(1, 3, babuff);
}else{
    CTOS_LCDTPrintXY(1, 3, "Mask Error");
}

memset(babuff,0x00,sizeof(babuff));
bLength = sizeof(babuff);
//Get the Gateway IP//
ret = CTOS_EthernetConfigGet(d_ETHERNET_CONFIG_GATEWAY, babuff, &bLength);
if (ret == d_OK)
{
    //CTOS_LCDTPrintXY(1, 4, "GateIP:");
    CTOS_LCDTPrintXY(1, 4, babuff);
}else{
    CTOS_LCDTPrintXY(1, 4, "GateIP Error");
}

memset(babuff,0x00,sizeof(babuff));
bLength = sizeof(babuff);
//Get the Host IP //
ret = CTOS_EthernetConfigGet(d_ETHERNET_CONFIG_HOSTIP, babuff, &bLength);
if (ret == d_OK)
{
    //CTOS_LCDTPrintXY(1, 5, "HostIP:");
    CTOS_LCDTPrintXY(1, 5, babuff);
}else{
    CTOS_LCDTPrintXY(1, 5, "HostIP Error");
}

memset(babuff,0x00,sizeof(babuff));
bLength = sizeof(babuff);
//Get the Host Port //
ret = CTOS_EthernetConfigGet(d_ETHERNET_CONFIG_HOSTPORT, babuff, &bLength);
if (ret == d_OK)
{
    //CTOS_LCDTPrintXY(1, 6, "HostPt:");
    CTOS_LCDTPrintXY(1, 6, babuff);
}else{
    CTOS_LCDTPrintXY(1, 6, "HostPort Error");
}

/*
=====
* FUNCTION NAME: EthernetRxData
* DESCRIPTION:   Receive data from the host.
* RETURN:
* NOTES:         none.
* =====*/
void EthernetRxData(void)
{
    //Declare Local Variable //
USHORT usr_Len = 0, usR_Len = 0;

CTOS_LCDTClearDisplay();
CTOS_LCDTPrintXY(1, 1, "----Rx Data----");
CTOS_LCDTPrintXY(1, 8, "Exit->'X'");
memset(babuff,0x00,sizeof(babuff));
usr_Len = sizeof(babuff);
CTOS_LCDTPrintXY(1, 2, "Receive Data....");
CTOS_LCDTPrintXY(1, 3, "Length=");

while(1){
    //Check if the data is currently availabl in Ethernet //
    ret = CTOS_EthernetRxReady(&usr_Len);
    if (ret == d_OK){
        //Receive data via Ethernet channel //
        CTOS_EthernetRxData(&babuff[usR_Len],&usr_Len);
        usR_Len += usr_Len;
        sprintf(&babuff[100],"%dbytes",usR_Len);
        CTOS_LCDTPrintXY(8, 3, &babuff[100]);

        CTOS_Delay(1000);
    }else{
}
}

```

```

        CTOS_LCDTPrintXY(1, 8, "RxReady Failed");
    }

    CTOS_KBDHit(&key);
    //retrun main() //
    if (key == d_KBD_CANCEL){
        //Clear the receive buffer of Ethernet //
        CTOS_EthernetFlushRxData();
        show_screen(0);
        emKM = eMain;
        return;
    }
}

/* =====
 * FUNCTION NAME: Connected
 * DESCRIPTION: Connect or Disconnect the host with IP Address
 *               and Port number which are stored in the Ethernet module.
 * RETURN:      1-->Connect or Disconnect Success,
 *               0-->Connect or Disconnect Fail.
 * NOTES:       none.
 * =====*/
USHORT EthernetConnect(BOOL IsConnected) {
    //Declare Local Variable //
    DWORD dwStatus;
    USHORT usr_Len = 0;

    if (IsConnected){
        // Use default Host IP address & Port number //

        ret = CTOS_EthernetConnect();
        if (ret != d_OK)
        {
            memset(babuff, 0,sizeof(babuff));
            sprintf(babuff, "ret=%04X", ret);
            CTOS_LCDTPrintXY(1, 7, babuff);
            CTOS_KBDGet(&key);
            return 0;
        }else
            ret = CTOS_EthernetStatus(&dwStatus);
            if (dwStatus & d_STATUS_ETHERNET_RX_READY){
                ret = CTOS_EthernetRxReady(&usr_Len);
                CTOS_EthernetRxData(&babuff[0],&usr_Len);
            }
            sprintf(babuff,"0x%08X",dwStatus);
            CTOS_LCDTPrintXY(1, 7, babuff);
            return 1;
    }else{
        //Disconnect from the host //
        ret = CTOS_EthernetDisconnect();
        if (ret != d_OK)
            return 0;
        else
            return 1;
    }
}

//Use the specific IP adrress & Port number //
USHORT EthernetConnectEx(BOOL IsConnected, BYTE* baDestIP, BYTE bIPLen, BYTE* baPort,
BYTE bPortLen)
{
    //Declare Local Variable //
    DWORD dwStatus;
    USHORT usr_Len = 0;

    CTOS_LCDTPrintXY(1, 7, baDestIP);
    CTOS_LCDTPrintXY(1, 8, baPort);
    if (IsConnected){
        ret = CTOS_EthernetConnectEx(baDestIP, bIPLen, baPort, bPortLen);
        if (ret != d_OK)
        {
            memset(babuff, 0,sizeof(babuff));
            sprintf(babuff, "ret=%04X", ret);
            CTOS_LCDTPrintXY(1, 7, babuff);
            CTOS_KBDGet(&key);
        }
    }
}

```

```

        return 0;
    }else
        ret = CTOS_EthernetStatus(&dwStatus);
        if (dwStatus & d_STATUS_ETHERNET_RX_READY) {
            ret = CTOS_EthernetRxReady(&usr_Len);
            CTOS_EthernetRxData(&babuff[0],&usr_Len);
        }
        return 1;
    }else{
        //Disconnect from the host //
        ret = CTOS_EthernetDisconnect();
        if (ret != d_OK)
            return 0;
        else
            return 1;
    }
}

/* =====
 * FUNCTION NAME: EthernetTxData
 * DESCRIPTION: Send data to the destination HOST using the
 *                  TCP/IP protocols.
 * RETURN:
 * NOTES:      none.
 * =====*/
void EthernetTxData(void)
{
    //Declare Local Variable //
    DWORD dwStatus;
    int i,iLength = 0;

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, "----Tx Data----");

    //Get the status of Ethernet //
    ret = CTOS_EthernetStatus(&dwStatus);
    if (ret != d_OK){
        CTOS_LCDTPrintXY(1, 7, "Status Fail");
    }else{
        sprintf(babuff,"0x%08X",dwStatus);
        CTOS_LCDTPrintXY(1, 7, babuff);
    }
    if (dwStatus & d_STATUS_ETHERNET_CONNECTED) {
        sprintf(babuff,"0123456789ABCDEF");
        CTOS_LCDTPrintXY(1, 2, "Send Data....");
        CTOS_LCDTPrintXY(1, 3, "Length=");
        for (i=0; i<=10; i++){
            //Check Ethernet hether ready to transmit data.
            if (CTOS_EthernetTxReady()== d_OK){
                //Transmit data via Ethernet channel //
                CTOS_EthernetTxData(babuff,strlen(babuff));
                iLength += strlen(babuff);
                sprintf(&babuff[100],"%dbyte",iLength);
                CTOS_LCDTPrintXY(8, 3, &babuff[100]);
                CTOS_Delay(1000);
            }else{
                CTOS_LCDTPrintXY(1, 8, "Tx Data Fail");
            }
        }
        CTOS_LCDTPrintXY(1, 8, "Exit->'X'");
    }
    //Get a key from keyboard //
    CTOS_KBDGet(&key);
    //retrun main() //
    if (key == d_KBD_CANCEL){
        show_screen(0);
        emKM = eMain;
    }
}

/* =====
 * FUNCTION NAME: Ethernet Get Status
 * DESCRIPTION: Get the status of Ethernet.
 * RETURN:
 * NOTES:      none.
 * =====*/

```

```

void EthernetGetStatus(void)
{
    //Declare Local Variable //
    DWORD dwStatus;
    int i = 1;

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, i, "---Get Status---");
    i++;

    //Get the status //
    CTOS_EthernetStatus(&dwStatus);
    if (dwStatus & d_STATUS_ETHERNET_CONNECTED){
        CTOS_LCDTPrintXY(1, i, "Connected"); //if Ethernet is connected //
        i++;
    }
    if (dwStatus & d_STATUS_ETHERNET_COMMAND_MODE){
        CTOS_LCDTPrintXY(1, i, "Command Mode"); //if Ethernet is command mode //
        i++;
    }
    if (dwStatus & d_STATUS_ETHERNET_PHYICAL_ONLINE){
        CTOS_LCDTPrintXY(1, i, "Phyical Online"); //if Ethernet is phyical online //
        i++;
    }
    if (dwStatus & d_STATUS_ETHERNET_RX_READY){
        CTOS_LCDTPrintXY(1, i, "Rx Ready"); //if Ethernet is Rx ready //
        i++;
    }
    if (dwStatus & d_STATUS_ETHERNET_TX_BUSY){
        CTOS_LCDTPrintXY(1, i, "Tx Busy"); //if Ethernet is Tx busy //
        i++;
    }
    CTOS_KBDGet(&key);
    if(key == d_KBD_CANCEL){
        emKM = eMain;
        show_screen(0);
    }
}

/* =====
 * FUNCTION NAME: Ethernet Ping
 * DESCRIPTION: Used to test whether a particular host is
 *               reachable across an IP network.
 * RETURN:
 * NOTES:      none.
 * =====*/
void EthernetPing(void)
{
    BYTE PingBuff[15+1];
    sprintf(PingBuff,"192.120.100.22");
    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, "----Ping----");
    CTOS_LCDTPrintXY(1, 2, PingBuff);
    sprintf(babuff, "%d", strlen(PingBuff));
    CTOS_LCDTPrintXY(1, 3, babuff);

    //Ping the other IP Address //
    ret = CTOS_EthernetPing(PingBuff, strlen(PingBuff));
    if (ret == d_OK){
        CTOS_LCDTPrintXY(1, 8, "Ping Success");
    }else{
        memset(babuff, 0, sizeof(babuff));
        sprintf(babuff, "ret=%04X", ret);
        CTOS_LCDTPrintXY(1, 8, babuff);
    }
    CTOS_KBDGet(&key);
    if(key == d_KBD_CANCEL){
        emKM = eMain;
        show_screen(0);
    }
}

int main(int argc,char *argv[])
{
    // TODO: Add your program here //
    //Open the Back Light in the LCD Display //
}

```

```

CTOS_BackLightSet(d_BKLIT_LCD, d_ON);

CTOS_LCDTPrintXY(1, 1, "----Ethernet----");

//Open the Ethernet cannel //
ret = CTOS_EthernetOpen();
if(ret != d_OK)
{
    CTOS_LCDTPrintXY(1, 7, "Open Not OK  ");
    CTOS_KBDGet(&key);
    return 0;
}
CTOS_LCDTPrintXY(1, 2, "Open OK      ");

// Show Main screen //
show_screen(0);
emKM = eMain;

while(1)
{
    CTOS_KBDGet(&key);
    switch(emKM)
    {
        case eMain:
            switch(key)
            {
                case d_KBD_1:
                    show_screen(1); //Show sub screen //
                    emKM = eConfig; //key mode change //
                    break;
                case d_KBD_2:
                    show_screen(2);
                    emKM = eConnected;
                    break;
                case d_KBD_3:
                    emKM = eTxData;
                    break;
                case d_KBD_4:
                    emKM = eRxData;
                    break;
                case d_KBD_5:
                    EthernetGetStatus();
                    break;
                case d_KBD_6:
                    EthernetPing();
                    break;
                case d_KBD_7:
                case d_KBD_CANCEL:
                    //Disconnect from the host //
                    CTOS_EthernetDisconnect();
                    //Close an opened Ethernet channel and release its working buffer //
                    CTOS_EthernetClose();
                    return 1;
                case d_KBD_8:
                    EthernetConnectEx(TRUE, (BYTE *)host_ip, strlen(host_ip), (BYTE *)host_port,
strlen(host_port));
                    break;
            }
            break;
        case eConfig:
            switch(key)
            {
                case d_KBD_1:
                    EthernetSetConfig();
                    break;
                case d_KBD_2:
                    EthernetGetConfig();
                    break;
                case d_KBD_CANCEL:
                    show_screen(0);
                    emKM = eMain;
                    break;
            }
            break;
        case eConnected:
            switch(key)

```

```

{
    case d_KBD_1:
        if (EthernetConnect(TRUE) == 1){
            CTOS_LCDTPrintXY(1, 8, "Connect OK");
        }else{
            CTOS_LCDTPrintXY(1, 8, "Connect Failed");
        }
        CTOS_Delay(1000);
        show_screen(0);
        emKM = eMain;
        break;
    case d_KBD_2:
        if (EthernetConnect(FALSE) == 1){
            CTOS_LCDTPrintXY(1, 8, "DisConnect OK");
        }else{
            CTOS_LCDTPrintXY(1, 8, "DisConnect Failed");
        }
        CTOS_Delay(1000);
        show_screen(0);
        emKM = eMain;
        break;
    case d_KBD_7:
    case d_KBD_CANCEL:
        show_screen(0);
        emKM = eMain;
        break;
    }
    break;
case eTxData:
    EthernetTxData();
    break;
case eRxData:
    EthernetRxData();
    break;
}
}

//Close the Back Light in the LCD Display //
CTOS_BackLightSet(d_BKLIT_LCD, d_OFF);
return 0;
}

// eof

```

## B.3. FileSystem

```
/*
 * FILE NAME: FileSystem.c
 * PROGRAMMER: Vance Ke
 * DESCRIPTION: Test File System Functions
 * REVISION:01.00
 */

=====
 * I N C L U D E S
 =====

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctosapi.h>

=====
 * D E F I N E S
 =====

#define d_BUFF_SIZE 128 //Buffer Size

/* Define Error Code*/
#define d_FILE_EXIST      0x01
#define d_FILE_NO_FILES   0x02
#define d_FILE_DIR_OK     0x03
#define d_FILE_LENGTH_SHORT 0x04

=====
 * GLOBAL VARIABLES
 =====

BYTE key;
USHORT ret;
USHORT usTotal;
BYTE babuff[d_BUFF_SIZE];
BYTE baFileBuff[d_BUFF_SIZE];

/* Define Key Mode Option */
typedef enum
{
    eMain , eNewFile , eDelFile , eDirFile , eWrite, eRead , eRename , eMainPage2 ,
eSetAttrib
} emKeyMode;
emKeyMode emKM;

=====
 * FUNCTION NAME: DirectoryFile
 * DESCRIPTION: List all created files by the same application
 * RETURN:      d_FILE_DIR_OK --> Search files success
 *              d_FILE_NO_FILES --> No any files found
 * NOTES:      none.
 * =====
USHORT DirectoryFile(USHORT *pusTotal, BYTE *pbaFileBuff)
{
    int i;
    ULONG ulFileSize[256]; //The maximum value is 256
    BYTE baTemp[d_BUFF_SIZE];

    memset(baTemp,0x00,sizeof(baTemp));

    /* List all created files by the same application.
     * pbaFileBuff --> the file names of the files. Each file name occupies 15 bytes
     * ulFileSize --> the file sizes of the files. it is array of ULONG
     * pusTotal --> the number of the files created by the same AP */
    CTOS_FileDir(pbaFileBuff , ulFileSize , pusTotal);

    if (*pusTotal == 0)
        return d_FILE_NO_FILES;

    /* Show the files on the LCD screen */
}
```

```

        for (i=1 ; i<=*pusTotal ; i++) {
            sprintf(baTemp,"%d.%s %d", i ,&pbaFileBuff[15*(i-1)],(int )ulFileSize[i-1]);
            CTOS_LCDTPrintXY(1, i+1, baTemp);
        }
        return d_FILE_DIR_OK;
    }

/*=====
 * FUNCTION NAME: show_screen
 * DESCRIPTION: Show on the LCD screen
 * RETURN: none.
 * NOTES: none.
 * ===== */
void show_screen(int tag)
{
    CTOS_LCDTClearDisplay ();
    switch(tag)
    {
        case 0:
            emKM = eMain;
            CTOS_LCDTPrintXY(1, 1, "\fr FILE SYSTEM \fn");
            CTOS_LCDTPrintXY(1, 2, "1.New File");
            CTOS_LCDTPrintXY(1, 3, "2.Del File");
            CTOS_LCDTPrintXY(1, 4, "3.Directory");
            CTOS_LCDTPrintXY(1, 5, "4.Read");
            CTOS_LCDTPrintXY(1, 6, "5.Write");
            CTOS_LCDTPrintXY(1, 7, "6.Rename File");
            CTOS_LCDTPrintXY(1, 8, "'DOWN'->NextPage");
            break;
        case 1:
            emKM = eNewFile;
            CTOS_LCDTPrintXY(1, 1, "\fr STORAGE TYPE \fn");
            CTOS_LCDTPrintXY(1, 2, "1.Flash");
            CTOS_LCDTPrintXY(1, 3, "2.SRAM");
            break;
        case 2:
            CTOS_LCDTPrintXY(1, 1, "\fr DELETE FILE \fn");
            break;
        case 3:
            CTOS_LCDTPrintXY(1, 1, "\fr DIRETCORY \fn");
            break;
        case 4:
            CTOS_LCDTPrintXY(1, 1, "\fr WRITE FILE \fn");
            break;
        case 5:
            CTOS_LCDTPrintXY(1, 1, "\fr READ FILE \fn");
            break;
        case 6:
            CTOS_LCDTPrintXY(1, 1, "\fr RENAME FILE \fn");
            break;
        case 7:
            emKM = eMainPage2;
            CTOS_LCDTPrintXY(1, 1, "\fr FILE SYSTEM \fn");
            CTOS_LCDTPrintXY(1, 2, "1.Cut Content");
            CTOS_LCDTPrintXY(1, 3, "2.Set Attrib");
            CTOS_LCDTPrintXY(1, 4, "3.Get Attrib");
            CTOS_LCDTPrintXY(1, 5, "4.Dir Attrib");
            CTOS_LCDTPrintXY(1, 6, "5.Format");
            CTOS_LCDTPrintXY(1, 7, "6.Disk Free Size");
            CTOS_LCDTPrintXY(1, 8, "'UP'->PrevPage");
            break;
        case 8:
            CTOS_LCDTPrintXY(1, 1, "\fr SET ATTRIB \fn");
            CTOS_LCDTPrintXY(1, 2, "1.Private");
            CTOS_LCDTPrintXY(1, 3, "2.Public");
            break;
        case 9:
            CTOS_LCDTPrintXY(1, 1, "\fr SET ATTRIB \fn");
            break;
        case 10:
            CTOS_LCDTPrintXY(1, 1, "\fr GET ATTRIB \fn");
            break;
        case 11:
            CTOS_LCDTPrintXY(1, 1, "\fr FORMAT \fn");
            CTOS_LCDTPrintXY(1, 3, "Note!!!");
    }
}

```

```

        CTOS_LCDTPrintXY(1, 4, "Files will Erase!!");
        CTOS_LCDTPrintXY(1, 8, "Format->'OK' ?");
        break;
    case 12:
        CTOS_LCDTPrintXY(1, 1, "\fr Cut Content \fn");
        break;
    case 13:
        CTOS_LCDTPrintXY(1, 1, "\fr Disk Free Size \fn");
        break;
    }
}

/* =====
 * FUNCTION NAME: show_errormsg
 * DESCRIPTION: Show error message on the LCD screen
 * RETURN: none.
 * NOTES: none.
 * ===== */
void show_errormsg(USHORT ret)
{
    switch(ret)
    {
        case d_OK:
        case d_USER_CANCEL:
        case d_FILE_DIR_OK:
            break;
        case d_FILE_EXIST:
            CTOS_LCDTPrintXY(1,8,"File Exist      ");
            break;
        case d_FILE_LENGTH_SHORT:
            CTOS_LCDTPrintXY(1,8,"Length<20bytes");
            break;
        case d_FILE_NO_FILES:
            CTOS_LCDTPrintXY(1,8,"No Any Files   ");
            break;
        case d_FS_INVALID_PARAMETER:
            CTOS_LCDTPrintXY(1,8,"Invalid Parameter");
            break;
        case d_FS_FHT_FULL:
            CTOS_LCDTPrintXY(1,8,"FHT Full      ");
            break;
        case d_FS_FILE_ALREADY_OPENED:
            CTOS_LCDTPrintXY(1,8,"Already Opened  ");
            break;
        case d_FS_FILE_NOT_OPENED:
            CTOS_LCDTPrintXY(1,8,"Not Opened     ");
            break;
        case d_FS_FILE_NOT_FOUND:
            CTOS_LCDTPrintXY(1,8,"File Not Found  ");
            break;
        case d_FS_FILE_INVALID_HANDLE:
            CTOS_LCDTPrintXY(1,8,"Invalid handle  ");
            break;
        case d_FS_DATA_FULL:
            CTOS_LCDTPrintXY(1,8,"Data Full      ");
            break;
        case d_FS_NOT_INITIALED:
            CTOS_LCDTPrintXY(1,8,"Not Initialed  ");
            break;
        case d_FS_CHECKSUM_ERROR:
            CTOS_LCDTPrintXY(1,8,"Checksum Error  ");
            break;
        case d_FS_FILE_ALREADY_EXISTED:
            CTOS_LCDTPrintXY(1,8,"Already Existed ");
            break;
        case d_FS_NOT_OWNER:
            CTOS_LCDTPrintXY(1,8,"Not Owner      ");
            break;
        default:
            CTOS_LCDTPrintXY(1,8,"Other Error !!  ");
    }
}

/* =====
 * FUNCTION NAME: NewFile
 * DESCRIPTION: Create a new file in the Flash. (*NOT* support SRAM type)

```

```

* RETURN:      d_FILE_EXIST --> File already is existed
*             ret --> Return other messages
* NOTES:      none.
* =====
USHORT NewFile(BYTE *pbaFileName, BYTE bStorageType)
{
    ULONG ulFileSize = 0, ulHandle;

    /* Get the file size with specific file name.
     * If Get file size > 0, the file is already existed      */
    ret = CTOS_FileGetSize(pbaFileName, &ulFileSize);

    if (ulFileSize > 0)
        return d_FILE_EXIST;

    if ((ret != d_OK) && (ret != d_FS_FILE_NOT_FOUND))
        return ret;

    /* Open a file and return a number called a file handle.
     * If the specified file name does not exist , it will be created first. */
    ret = CTOS_FileOpen(pbaFileName, bStorageType, &ulHandle);

    if (ret == d_OK)
        CTOS_FileClose(ulHandle);

    return ret;
}

/* =====
* FUNCTION NAME: DeleteFile
* DESCRIPTION: Delete a file with specific file name
* RETURN:      d_USER_CANCEL --> Cancel to delete the file
*             d_FILE_NO_FILES --> No any files by the same application
*             d_FS_FILE_NOT_FOUND --> User select wrong file item
*             ret --> Return other messages
* NOTES:      none.
* =====
USHORT DeleteFile(void)
{
    /* Search & list the files by the same application */
    if (DirectoryFile(&usTotal, baFileBuff) != d_FILE_DIR_OK)
        return d_FILE_NO_FILES;

    /* To select one file */
    CTOS_KBDGet(&key);
    if (key == d_KBD_CANCEL)
        return d_USER_CANCEL;
    if ((key - 0x30) > usTotal)
        return d_FS_FILE_NOT_FOUND;

    /* Delete selected file */
    return CTOS_FileDelete(&baFileBuff[15*(key-0x31)]);
}

/* =====
* FUNCTION NAME: FileRead
* DESCRIPTION: Read data from the opened file
* RETURN:      d_USER_CANCEL --> Cancel to Read the file
*             d_FILE_NO_FILES --> No any files by the same application
*             d_FS_FILE_NOT_FOUND --> User select wrong file item
*             ret --> Return other messages
* NOTES:      none.
* =====
USHORT FileRead(BYTE *pcaReadString, ULONG *pulLength)
{
    ULONG ulHandle, ulbuffLen;

    memset(pcaReadString, 0x00, *pulLength);

    /* Search & list the files by the same application */
    if (DirectoryFile(&usTotal, baFileBuff) != d_FILE_DIR_OK)
        return d_FILE_NO_FILES;

    /* To select one file */
    CTOS_KBDGet(&key);
    if (key == d_KBD_CANCEL)

```

```

        return d_USER_CANCEL;
    if ((key - 0x30) > usTotal)
        return d_FS_FILE_NOT_FOUND;

    /* Open the selected file */
    if (strncpy(&baFileBuff[15*(key-0x31)], "FL", 2) == 0)
    {
        /* Reopen a file with specific file name. the file should be created first if
        not exist */
        ret = CTOS_FileReopen(&baFileBuff[15*(key-0x31)], d_STORAGE_FLASH, &ulHandle);
        if (ret != d_OK)
            return ret;
    } else {
        return d_FS_FILE_NOT_FOUND;
    }

    /* Read data from this opened file */
    ret = CTOS_FileRead(ulHandle, pcaReadString, pulLength);
    if (ret != d_OK)
        return ret;

    /* Close the opened file */
    CTOS_FileClose(ulHandle);
    return ret;
}

/* =====
 * FUNCTION NAME: FileWrite
 * DESCRIPTION: Write data into the opened file
 * RETURN: d_USER_CANCEL --> Cancel to wirte the file
 *          d_FILE_NO_FILES --> No any files by the same application
 *          d_FS_FILE_NOT_FOUND --> User select wrong file item
 *          ret --> Return other messages
 * NOTES: none.
 * ===== */
USHORT FileWrite(BYTE *pcaString)
{
    ULONG ulHandle, ulPosition;

    /* Search & list the files by the same application */
    if (DirectoryFile(&usTotal, baFileBuff) != d_FILE_DIR_OK)
        return d_FILE_NO_FILES;

    /* To select one file */
    CTOS_KBDGet(&key);
    if (key == d_KBD_CANCEL)
        return d_USER_CANCEL;
    if ((key - 0x30) > usTotal)
        return d_FS_FILE_NOT_FOUND;

    /* Open the selected file */
    if (strncpy(&baFileBuff[15*(key-0x31)], "FL", 2) == 0)
    {
        /*Reopen a file with specific file name. the file should be created first if not
        exist */
        ret = CTOS_FileReopen(&baFileBuff[15*(key-0x31)], d_STORAGE_FLASH, &ulHandle);
        if (ret != d_OK)
            return ret;
    } else {
        return d_FS_FILE_NOT_FOUND;
    }

    /* Move the file pointer to a specific position.
     * Move backward from the end of the file. */
    ret = CTOS_FileSeek(ulHandle, 0, d_SEEK_FROM_EOF);
    if (ret != d_OK)
        return ret;

    CTOS_FileGetPosition(ulHandle, &ulPosition);
    sprintf(babuff, "BPos=%d", (int)ulPosition);
    CTOS_LCDTPrintXY(1, 7, babuff);

    /* Write data into this opened file */
    ret = CTOS_FileWrite(ulHandle, pcaString, strlen(pcaString));
    if (ret != d_OK)
        return ret;
}

```

```

CTOS_FileGetPosition(ulHandle ,&ulPosition);
sprintf(babuff,"FPos=%d", (int )ulPosition);
CTOS_LCDTPrintXY(9,7,babuff);

/* Close the opened file */
CTOS_FileClose(ulHandle);
return ret;
}

/* =====
* FUNCTION NAME: FileRename
* DESCRIPTION: Change the name of the file
* RETURN: d_USER_CANCEL --> Cancel to rename the file
*          d_FILE_NO_FILES --> No any files by the same application
*          d_FS_FILE_NOT_FOUND --> User select wrong file item
*          ret --> Return other messages
* NOTES: none.
* ===== */
USHORT FileRename(void)
{
    BYTE tmpKey;
    BYTE baNewName[d_BUFF_SIZE];

    memset(baNewName,0x00,sizeof(baNewName));
    memset(baFileBuff,0x00,sizeof(baFileBuff));

    /* Search & list the files by the same application */
    if (DirectoryFile(&usTotal, baFileBuff) != d_FILE_DIR_OK)
        return d_FILE_NO_FILES;

    /* To select one file */
    CTOS_KBDGet(&key);
    if (key == d_KBD_CANCEL)
        return d_USER_CANCEL;
    if ((key - 0x30) > usTotal)
        return d_FS_FILE_NOT_FOUND;

    if (strncmp(&baFileBuff[15*(key-0x31)],"FL",2) == 0)
    {
        sprintf(baNewName,"%.*s_1.txt",4,&baFileBuff[15*(key-0x31)]);
        CTOS_LCDTPrintXY(1,8,baNewName);
        CTOS_KBDGet(&tmpKey);
    } else {
        return d_FS_FILE_NOT_FOUND;
    }

    /* Change the name of selected file */
    ret = CTOS_FileRename(&baFileBuff[15*(key-0x31)] , baNewName);
    return ret;
}

/* =====
* FUNCTION NAME: FileCut
* DESCRIPTION: Cut off the file content such that the file length
*               is the specific value
* RETURN: d_FILE_LENGTH_SHORT --> The file data length is short
*          d_USER_CANCEL --> Cancel to cut off the file
*          d_FILE_NO_FILES --> No any files by the same application
*          d_FS_FILE_NOT_FOUND --> User select wrong file item
*          ret --> Return other messages
* NOTES: none.
* ===== */
USHORT FileCut(void)
{
    USHORT usret;
    ULONG ulFileSize;

    /* Search & list the files by the same application */
    if (DirectoryFile(&usTotal, baFileBuff) != d_FILE_DIR_OK)
        return d_FILE_NO_FILES;

    /* To select one file */
    CTOS_KBDGet(&key);
    if (key == d_KBD_CANCEL)
        return d_USER_CANCEL;

```

```

if ((key - 0x30) > usTotal)
    return d_FS_FILE_NOT_FOUND;

/* Get the file size with specific file name */
CTOS_FileGetSize(&baFileBuff[15*(key-0x31)],&ulFileSize);

/* Check whether this file size is greater than 20 bytes */
if (ulFileSize <= 20)
    return d_FILE_LENGTH_SHORT;

/* Cut off the file size to be specific value.
 * The length 20 bytes is the final size of the file. */
usret = CTOS_FileCut(&baFileBuff[15*(key-0x31)], 20);

if (DirectoryFile(&usTotal, baFileBuff) != d_FILE_DIR_OK) //Re-get file lists
    return d_FILE_NO_FILES;
return usret;
}

/*
 * =====
 * FUNCTION NAME: main
 * DESCRIPTION: Main entry for the FileSystem testing sample code.
 * RETURN: none.
 * NOTES: none.
 * ===== */
int main (void)
{
    int i;
    USHORT usFileCount = 0;
    ULONG ulReadLen;
    ULONG ulFreeSize ;
    BYTE baTemp[d_BUFF_SIZE];

    memset(babuff,0x00,sizeof(babuff));
    memset(baTemp,0x00,sizeof(baTemp));

    CTOS_BackLightSet(d_BKLIT_LCD,d_ON);
    show_screen(0);
    while(1)
    {
        CTOS_KBDGet ( &key );
        switch(emKM)
        {
            /*** The Main Page ***/
            case eMain:
                switch(key)
                {
                    case d_KBD_1: //Run 'NewFile' Function
                        show_screen(1);
                        CTOS_KBDGet ( &key );
                        switch(key)
                        {
                            case d_KBD_1: //Flash
                                DirectoryFile(&usTotal ,baFileBuff);
                                usFileCount = ( usTotal !=7 ? usTotal+1 : 1 );
                                CTOS_LCDTPrintXY(1,7,"New:");
                                sprintf(babuff,"FL%02d.txt",usFileCount);
                                CTOS_LCDTPrintXY(5,7 ,babuff);
                                ret = NewFile(babuff , 0);
                                if (ret == d_OK)
                                    CTOS_LCDTPrintXY(1 ,8 , "New File Success");
                                else
                                    show_errormsg(ret);
                                CTOS_KBDGet (&key);
                                show_screen(0);
                                break;
                            case d_KBD_2: //SRAM -- Not support in Vega 5000
                                CTOS_LCDTPrintXY(1,8,"Not supported !!");
                                CTOS_KBDGet (&key);
                                show_screen(1);
                                break;
                            case d_KBD_CANCEL: //Go back main layer
                                show_screen(0);
                                break;
                        }
                }
        }
    }
}

```

```

        break;

    case d_KBD_2:      //Run 'DeleteFile' Function
        show_screen(2);
        ret = DeleteFile();
        if(ret == d_OK)
            CTOS_LCDTPrintXY(1,8,"Del File Success");
        else if (ret != d_USER_CANCEL)
            show_errormsg(ret);

        if (ret != d_USER_CANCEL)
            CTOS_KBDGet ( &key );
        show_screen(0);
        break;

    case d_KBD_3:      //Run 'DirectoryFile' Function
        memset(baFileBuff,0x00,sizeof(baFileBuff));
        show_screen(3);
        ret = DirectoryFile(&usTotal,baFileBuff);
        if (ret!= d_OK)
            show_errormsg(ret);
        CTOS_KBDGet ( &key );
        show_screen(0);
        break;

    case d_KBD_4:      //Run 'FileRead' Function
        show_screen(5);
        ulReadLen = sizeof(babuff); //Set the maximum reading size
        memset(babuff ,0 ,sizeof(babuff));
        ret = FileRead(babuff ,&ulReadLen);
        if (ret == d_OK)
        {
            sprintf(baTemp,"Read OK (%lu)",ulReadLen);
            CTOS_LCDTPrintXY(1 ,8 ,baTemp);
            if (ulReadLen > 0)
            {
                CTOS_KBDGet(&key);
                CTOS_LCDTClearDisplay();
                CTOS_LCDTPrint("The Content: \n");
                for (i=0; i<112; i+=16) {
                    if (i > ulReadLen)
                        break;
                    strncpy(baTemp ,&babuff[i] ,16);
                    baTemp[16]='\n';
                    if (i >= (112-16))
                        baTemp[16]='\0';
                    CTOS_LCDTPrint(baTemp);
                }
            }
            else if (ret != d_USER_CANCEL) {
                show_errormsg(ret);
                CTOS_KBDGet ( &key );
            }
        }

        if (ret != d_USER_CANCEL)
            CTOS_KBDGet ( &key );

        show_screen(0);
        break;

    case d_KBD_5:      //Run 'FileWrite' Function
        show_screen(4);
        ret = FileWrite("12345ABCDEFG6789");
        if (ret == d_OK)
            CTOS_LCDTPrintXY(1,8,"Write Success      ");
        else if (ret != d_USER_CANCEL)
            show_errormsg(ret);

        if (ret != d_USER_CANCEL)
            CTOS_KBDGet ( &key );
        show_screen(0);
        break;

    case d_KBD_6:      //Run 'FileRename' Function
        show_screen(6);

```

```

    ret = FileRename();

    if (ret == d_OK)
        CTOS_LCDTPrintXY(1,8,"Rename Success    ");
    else if (ret != d_USER_CANCEL)
        show_errormsg(ret);

    if (ret != d_USER_CANCEL)
        CTOS_KBDGet (&key);
    show_screen(0);
    break;

    case d_KBD_7:
        break;

    case d_KBD_CANCEL: //Exit Application
        CTOS_BackLightSet(d_BKLIT_LCD,d_OFF);
        exit(0);

    case d_KBD_DOWN: //Go to next page
        show_screen(7);
        break;
    }
    break;

/** The Second Page ***/
case eMainPage2:
    switch(key)
    {
        case d_KBD_1: //Run 'FileCut' Function
            show_screen(12);
            ret = FileCut();
            if (ret == d_OK)
                CTOS_LCDTPrintXY(1,8,"Cut Success    ");
            else if (ret != d_USER_CANCEL)
                show_errormsg(ret);
            if (ret != d_USER_CANCEL)
                CTOS_KBDGet (&key);
            show_screen(7);
            break;

        case d_KBD_2: //Run 'FileSetAttrib' Function
            CTOS_LCDTPrintXY(1,8,"Not supported !!");
            CTOS_KBDGet (&key);
            show_screen(7);
            break;

        case d_KBD_3: //Run 'FileGetAttrib' Function
            CTOS_LCDTPrintXY(1,8,"Not supported !!");
            CTOS_KBDGet (&key);
            show_screen(7);
            break;

        case d_KBD_4: //Run 'FileDirAttrib' Function
            CTOS_LCDTPrintXY(1,8,"Not supported !!");
            CTOS_KBDGet (&key);
            show_screen(7);
            break;

        case d_KBD_5: //Run 'FileFormat' Function
            show_screen(11);
            CTOS_KBDGet (&key);
            if (key == d_KBD_ENTER)
            {
                ret = CTOS_FileFormat(0); //Delete whole files created by the AP
                if (ret == d_OK)
                    CTOS_LCDTPrintXY(1,8,"Format Success    ");
                else if (ret != d_USER_CANCEL)
                    show_errormsg(ret);
                CTOS_Delay(2000);
            }
            show_screen(7);
            break;

        case d_KBD_6: //Run 'Get Disk Free Size' Function
            show_screen(13);
    }
}

```

```

ret = CTOS_FileGetFreeSpace(0 , &ulFreeSize);
if (ret == d_OK) {
    memset(baTemp , 0 , sizeof(baTemp));
    sprintf(baTemp,"%ld Bytes",ulFreeSize);
    CTOS_LCDTPrintXY(1,5,baTemp);
    CTOS_LCDTPrintXY(1,8,"Get Success   ");
}
else if (ret != d_USER_CANCEL) {
    show_errormsg(ret);
    CTOS_KBDGet (&key);
}

if (ret != d_USER_CANCEL)
    CTOS_KBDGet (&key);

show_screen(7);
break;

case d_KBD_UP: //Go back Main Page
    show_screen(0);
    break;

case d_KBD_CANCEL: //Exit Application
    CTOS_BackLightSet(d_BKLIT_LCD , d_OFF);
    exit(0);
    break;
}
break;
}
}
exit(0);
}

// eof

```

## B.4. GSM

```
/******  
* FILE NAME:      GSM  
* MODULE NAME:    GSM  
* PROGRAMMER:    Andy Chang  
* DESCRIPTION:   Test Functions of GSM  
* REVISION:01.00  
******/  
  
/*=====*  
*           I N C L U D E S           *  
*=====*/  
#include <string.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <stdarg.h>  
/** These two files are necessary for calling CTOS API **/  
#include <ctosapi.h>  
/** Uncomment this line if you want to show debug information in the simulator  
#include <debugprint.h>  
**/  
  
#define TITLE_Y          1  
#define RTN_MSG_Y        8  
  
typedef struct  
{  
    BOOL    boIsEnabled;  
    BOOL    boIsShowTitle;  
    STR     strTitle[32];  
    USHORT  usRtn;  
}stReturn;  
  
stReturn stRtn;  
  
BOOL GetNumber(IN BYTE bX, IN BYTE bY, IN BYTE bNumSize, IN BOOL boIsNeedEnter, OUT  
ULONG *ulValue)  
{  
    BYTE    bKey, baBuf[10], i, bCurrentDigit;  
  
    memset(baBuf, 0x00, sizeof(baBuf));  
    bCurrentDigit = 0;  
    *ulValue = 0;  
  
    CTOS_LCDTgotoXY(bX, bY);  
    for (i = bX; i < bNumSize; i++)  
        CTOS_LCDTPutCh(' ');  
  
    while (1)  
    {  
        CTOS_KBDGet(&bKey);  
  
        if ( (bKey >= '0') && (bKey <= '9') )  
        {  
            if (bCurrentDigit < bNumSize)  
            {  
                baBuf[bCurrentDigit++] = bKey - '0';  
                CTOS_LCDTPutChXY(bX++, bY, bKey);  
            }  
        }  
        else if (bKey == d_KBD_CLEAR)  
        {  
            if (bCurrentDigit > 0)  
            {  
                bCurrentDigit--;  
                CTOS_LCDTPutChXY(--bX, bY, ' ');  
            }  
        }  
        else if (bKey == d_KBD_ENTER)  
            break;  
    }  
}
```

```

        else if (bKey == d_KBD_CANCEL)
            return FALSE;

        if (boIsNeedEnter == FALSE && bCurrentDigit == bNumSize)
            break;
    }

    for (i = 0; i < bCurrentDigit ; i++)
    {
        *ulValue *= 10;
        *ulValue += baBuf[i];
    }

    return TRUE;
}

BYTE GetString(IN BYTE bX, IN BYTE bY, OUT BYTE *baBuf, IN BYTE bBufSize)
{
    BYTE      bKey;
    BYTE      bStrLen;

    memset(baBuf, 0x00, bBufSize);
    bStrLen = 0;

    while(1)
    {
        CTOS_KBDGet(&bKey);

        switch(bKey)
        {
            case d_KBD_DOT:
                if (bStrLen < bBufSize - 1)
                {
                    baBuf[bStrLen++] = '.';
                    CTOS_LCDTPutchXY(bX++, bY, '.');
                }
                break;
            case d_KBD_CLEAR:
                if (bStrLen > 1)
                {
                    bStrLen--;
                    CTOS_LCDTPutchXY(--bX, bY, ' ');
                }
                break;
            case d_KBD_CANCEL:
                return 0;
            case '1':
            case '2':
            case '3':
            case '4':
            case '5':
            case '6':
            case '7':
            case '8':
            case '9':
            case '0':
                if (bStrLen < bBufSize - 1)
                {
                    baBuf[bStrLen++] = bKey;
                    CTOS_LCDTPutchXY(bX++, bY, bKey);
                }
                break;
            case d_KBD_ENTER:
                baBuf[bStrLen] = '\0';
                return bStrLen;
        }
    }

    return 0;
}

void ShowResult(void)
{
    BYTE      bKey, baMsg[32];
    if(stRtn.boIsEnabled == FALSE)

```

```

    return;

    if (stRtn.boIsShowTitle == TRUE)
        CTOS_LCDTPrintXY(1, TITLE_Y, stRtn.strTitle);

    if (stRtn.usRtn != d_OK)
    {
        sprintf(baMsg, "Error: 0x%04X", stRtn.usRtn);
        CTOS_LCDTPrintXY(1, RTN_MSG_Y, baMsg);
    }
    else
    {
        CTOS_LCDTPrintXY(1, RTN_MSG_Y, "OK");
    }

    CTOS_KBDGet(&bKey);
}

int GSM_Functions(void)
{
    BOOL    boRtn;
    BYTE    bRtn, bKey, baStr[32];
    USHORT   usRtn;

    BYTE    baTmp[32], baTmp2[32], bTmp, bTmp2, bTmpLen;
    USHORT   usTmpLen;
    ULONG    ulTmp, ulCase, ulNextCase;
    stPhoneBook stPB;
    stSMS    stSMS_t[50];
    stSMS_Submit    stSMS_Submit_t;

    ulCase = 0;

    while (1)
    {
        CTOS_LCDTClearDisplay();

        stRtn.boIsEnabled = TRUE;
        stRtn.boIsShowTitle = TRUE;
        stRtn.usRtn = d_OK;

        memset(baStr, 0x00, sizeof(baStr));
        memset(baTmp, 0x00, sizeof(baTmp));

        switch(ulCase)
        {
        case 0:
            CTOS_LCDTPrintXY(1, 1, "10. System 1");
            CTOS_LCDTPrintXY(1, 2, "20. System 2");
            CTOS_LCDTPrintXY(1, 3, "30. SIM");
            CTOS_LCDTPrintXY(1, 4, "40. PIN");
            CTOS_LCDTPrintXY(1, 5, "50. GSM GPRS");
            CTOS_LCDTPrintXY(1, 6, "60. Phone Book");
            CTOS_LCDTPrintXY(1, 7, "70. SMS");
            break;
        ///////////////////////////////////////////////////////////////////
        ///////////////////////////////////////////////////////////////////
        case 10:
            CTOS_LCDTPrintXY(1, 1, "11. GSM Open");
            CTOS_LCDTPrintXY(1, 2, "12. GSM Close");
            CTOS_LCDTPrintXY(1, 3, "13. Power On");
            CTOS_LCDTPrintXY(1, 4, "14. Power Off");
            CTOS_LCDTPrintXY(1, 5, "15. Reset");
            CTOS_LCDTPrintXY(1, 6, "16. Op Name");
            CTOS_LCDTPrintXY(1, 7, "17. Hang Up");
            break;
        case 11:
            strcpy(stRtn.strTitle, "CTOS_GSMOpen");
            // Open GSM channel and initial to start to use the GSM functions.
            stRtn.usRtn = CTOS_GSMOpen(115200, TRUE);
            break;
        case 12:
            strcpy(stRtn.strTitle, "CTOS_GSMClose");
            // Close the GSM channel.
            stRtn.usRtn = CTOS_GSMClose();
            break;
        }
    }
}

```

```

case 13:
    strcpy(stRtn.strTitle, "CTOS_GSMPowerOn");
    // Power on the GSM.
    stRtn.usRtn = CTOS_GSMPowerOn();
    break;
case 14:
    strcpy(stRtn.strTitle, "CTOS_GSMPowerOff");
    // Power off the GSM.
    stRtn.usRtn = CTOS_GSMPowerOff();
    break;
case 15:
    strcpy(stRtn.strTitle, "CTOS_GSMReset");
    // Reset the GSM module.
    CTOS_GSMReset();
    break;
case 16:
    strcpy(stRtn.strTitle, "CTOS_GSMQueryOperatorName");
    // This command returns the currently selected operator.
    stRtn.usRtn = CTOS_GSMQueryOperatorName(baTmp, &bTmpLen);
    if (stRtn.usRtn == d_OK)
    {
        baTmp[bTmpLen] = '\0';
        CTOS_LCDTPrintXY(1, 2, "Operator Name:");
        CTOS_LCDTPrintXY(1, 3, baTmp);
    }
    break;
case 17:
    strcpy(stRtn.strTitle, "CTOS_GSMHangup");
    // This command may be used to deactivate all active connection.
    stRtn.usRtn = CTOS_GSMHangup();
    break;
///////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////
case 20:
    CTOS_LCDTPrintXY(1, 1, "21. Send AT Command");
    CTOS_LCDTPrintXY(1, 2, "22. Send Data");
    CTOS_LCDTPrintXY(1, 3, "23. Receive Data");
    CTOS_LCDTPrintXY(1, 4, "24. Signal Quality");
    CTOS_LCDTPrintXY(1, 5, "25. Set Band");
    CTOS_LCDTPrintXY(1, 6, "26. Get Band");
    break;
case 21:
    strcpy(stRtn.strTitle, "CTOS_GSMSendATCmd");
    // Send the AT command to the GSM.
    stRtn.usRtn = CTOS_GSMSendATCmd("AT", 2, 200);
    break;
case 22:
    strcpy(stRtn.strTitle, "CTOS_GSMSendData");
    // Send data to the GSM. Use this function to transmit data via the GSM or
GPRS after a connection already built.
    stRtn.usRtn = CTOS_GSMSendData("AT\x0D", 3);
    break;
case 23:
    strcpy(stRtn.strTitle, "CTOS_GSMRecvData");
    // Receive data from the GSM. Use this function to receive data via the GSM
or GPRS after a connection already built.
    stRtn.usRtn = CTOS_GSMRecvData(baTmp, &usTmpLen);
    if (stRtn.usRtn == d_OK)
    {
        baTmp[bTmpLen] = '\0';
        CTOS_LCDTPrintXY(1, 2, "Recv Data:");
        CTOS_LCDTPrintXY(1, 3, baTmp);
    }
    break;
case 24:
    strcpy(stRtn.strTitle, "CTOS_GSMSignalQuality");
    // Get signal quality of the GSM network.
    stRtn.usRtn = CTOS_GSMSignalQuality(&bTmp);
    if (stRtn.usRtn == d_OK)
    {
        sprintf(baStr, "Strength: %d", bTmp);
        CTOS_LCDTPrintXY(1, 2, baStr);
    }
    break;
case 25:
    strcpy(stRtn.strTitle, "CTOS_GSMSetBAND");

```

```

CTOS_LCDTPrintXY(1, 2, "1. 900 1800");
CTOS_LCDTPrintXY(1, 3, "2. 900 1900");
CTOS_LCDTPrintXY(1, 4, "3. 850 1800");
CTOS_LCDTPrintXY(1, 5, "4. 850 1900");
boRtn = GetNumber(8, 6, 1, FALSE, &ulTmp);
if (boRtn == FALSE)
{
    stRtn.boIsEnabled = FALSE;
    break;
}

if (ulTmp > 4)
{
    stRtn.boIsEnabled = FALSE;
    CTOS_LCDTPrintXY(1, 8, "Invalid Band Type");
    CTOS_KBDGet(&bTmp);
    break;
}

// Select the band to use. The default band is d_GSM_900_1800 (GSM 900MHz +
DCS 1800MHz).
if (ulTmp == 1)
    stRtn.usRtn = CTOS_GSMSetBAND(d_GSM_900_1800);
else if (ulTmp == 2)
    stRtn.usRtn = CTOS_GSMSetBAND(d_GSM_900_1900);
else if (ulTmp == 3)
    stRtn.usRtn = CTOS_GSMSetBAND(d_GSM_850_1800);
else
    stRtn.usRtn = CTOS_GSMSetBAND(d_GSM_850_1900);
break;
case 26:
strcpy(stRtn.strTitle, "CTOS_GSMGetBAND");
// Get the current band settings.
stRtn.usRtn = CTOS_GSMGetBAND(&bTmp);
if (stRtn.usRtn == d_OK)
{
    CTOS_LCDTPrintXY(1, 2, "Band:");
    if (bTmp == d_GSM_900_1800)
        CTOS_LCDTPrintXY(1, 2, "Band: 900 1800");
    else if (bTmp == d_GSM_900_1900)
        CTOS_LCDTPrintXY(1, 2, "Band: 900 1900");
    else if (bTmp == d_GSM_850_1800)
        CTOS_LCDTPrintXY(1, 2, "Band: 850 1800");
    else
        CTOS_LCDTPrintXY(1, 2, "Band: 850 1900");
}
break;
///////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////
case 30:
CTOS_LCDTPrintXY(1, 1, "31. Select SIM");
CTOS_LCDTPrintXY(1, 2, "32. Get SIM");
CTOS_LCDTPrintXY(1, 3, "33. SIM Ready?");
CTOS_LCDTPrintXY(1, 4, "34. Get IMSI");
break;
case 31:
strcpy(stRtn.strTitle, "GSM Select SIM 1");
// Select the SIM card to use. The default SIM is SIM1.
stRtn.usRtn = CTOS_GSMSelctSIM(d_GPRS_SIM1);
break;
case 32:
strcpy(stRtn.strTitle, "CTOS_GSMGetCurrentSIM");
// Get the current SIM ID.
stRtn.usRtn = CTOS_GSMGetCurrentSIM(&bTmp);
if (stRtn.usRtn == d_OK)
{
    if (bTmp == d_GPRS_SIM1)
        CTOS_LCDTPrintXY(1, 2, "Current SIM1");
    else
        CTOS_LCDTPrintXY(1, 2, "Current SIM2");
}
break;
case 33:
strcpy(stRtn.strTitle, "CTOS_SIMCheckReady");
CTOS_LCDTPrintXY(1, 1, stRtn.strTitle);
stRtn.boIsEnabled = FALSE;

```

```

// Check is the GSM ready to use the SMS or Phone Book with the SIM card
inserted.
stRtn.usRtn = CTOS_SIMCheckReady();
if (stRtn.usRtn == d_GSM_SIM_READY)
{
    CTOS_LCDTPrintXY(1, 2, "SIM Ready");
    CTOS_KBDGet(&bTmp);
}
else if (stRtn.usRtn == d_GSM_SIM_READY)
{
    CTOS_LCDTPrintXY(1, 2, "SIM Not Ready");
    CTOS_KBDGet(&bTmp);
}
else
    stRtn.boIsEnabled = TRUE;
break;
case 34:
strcpy(stRtn.strTitle, "CTOS_SIMGetIMSI");
// Get the SIM card IMSI.
stRtn.usRtn = CTOS_SIMGetIMSI(baTmp);
if (stRtn.usRtn == d_OK)
{
    CTOS_LCDTPrintXY(1, 2, "IMSI:");
    CTOS_LCDTPrintXY(1, 3, baTmp);
}
break;
///////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////
case 40:
CTOS_LCDTPrintXY(1, 1, "41. Get Auth Status");
CTOS_LCDTPrintXY(1, 2, "42. PIN Verify");
CTOS_LCDTPrintXY(1, 3, "43. PIN Check Lock");
CTOS_LCDTPrintXY(1, 4, "44. PIN Lock");
CTOS_LCDTPrintXY(1, 5, "45. PIN UnLock");
CTOS_LCDTPrintXY(1, 6, "46. PIN Update");
break;
case 41:
strcpy(stRtn.strTitle, "CTOS_PINGetAuthStatus");
CTOS_LCDTPrintXY(1, 1, stRtn.strTitle);
stRtn.boIsEnabled = FALSE;
// This command indicates whether or not a password is required to use the
SIM.
stRtn.usRtn = CTOS_PINGetAuthStatus();
if (stRtn.usRtn == d_GSM_NO_AUTH_NEED)
{
    CTOS_LCDTPrintXY(1, 2, "No Auth Need");
    CTOS_KBDGet(&bTmp);
}
else if (stRtn.usRtn == d_GSM_AUTH_PIN)
{
    CTOS_LCDTPrintXY(1, 2, "Auth PIN");
    CTOS_KBDGet(&bTmp);
}
else if (stRtn.usRtn == d_GSM_AUTH_PUK)
{
    CTOS_LCDTPrintXY(1, 2, "Auth PUK");
    CTOS_KBDGet(&bTmp);
}
else if (stRtn.usRtn == d_GSM_AUTH_PIN2)
{
    CTOS_LCDTPrintXY(1, 2, "Auth PIN2");
    CTOS_KBDGet(&bTmp);
}
else if (stRtn.usRtn == d_GSM_AUTH_PUK2)
{
    CTOS_LCDTPrintXY(1, 2, "Auth PUK2");
    CTOS_KBDGet(&bTmp);
}
else if (stRtn.usRtn == d_GSM_AUTH_OTHERS)
{
    CTOS_LCDTPrintXY(1, 2, "Auth others");
    CTOS_KBDGet(&bTmp);
}
else
    stRtn.boIsEnabled = TRUE;
break;

```

```

case 42:
    strcpy(stRtn.strTitle, "CTOS_PINVerify");
    CTOS_LCDTPrintXY(1, 2, "1: PUK1 2: PUK2");
    boRtn = GetNumber(8, 3, 1, FALSE, &ulTmp);
    if (boRtn == FALSE)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }

    if (ulTmp > 2)
    {
        stRtn.boIsEnabled = FALSE;
        CTOS_LCDTPrintXY(1, 8, "Invalid Pin Type");
        CTOS_KBDGet(&bTmp);
        break;
    }

    CTOS_LCDTPrintXY(1, 4, "PIN:");
    bTmpLen = GetString(8, 4, baTmp, 5);
    if (bTmpLen == 0)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }

    // Verify the PIN if required.
    if (ulTmp == 1)
        stRtn.usRtn = CTOS_PINVerify(d_GSM_PIN_PUK_1, baTmp, bTmpLen);
    else
        stRtn.usRtn = CTOS_PINVerify(d_GSM_PIN_PUK_2, baTmp, bTmpLen);
    break;
case 43:
    strcpy(stRtn.strTitle, "CTOS_PINCheckLock");
    CTOS_LCDTPrintXY(1, 2, "1: SC 2: PS");
    CTOS_LCDTPrintXY(1, 3, "3: FD");
    boRtn = GetNumber(8, 4, 1, FALSE, &ulTmp);
    if (boRtn == FALSE)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }

    if (ulTmp > 3)
    {
        stRtn.boIsEnabled = FALSE;
        CTOS_LCDTPrintXY(1, 8, "Invalid Pin Type");
        CTOS_KBDGet(&bTmp);
        break;
    }

    // Get SC, PS, FD locking state.
    if (ulTmp == 1)
        stRtn.usRtn = CTOS_PINCheckLock(d_GSM_AUTH_SC);
    else if (ulTmp == 2)
        stRtn.usRtn = CTOS_PINCheckLock(d_GSM_AUTH_PS);
    else
        stRtn.usRtn = CTOS_PINCheckLock(d_GSM_AUTH_FD);

    if (stRtn.usRtn == d_GSM_SIM_LOCK)
    {
        CTOS_LCDTPrintXY(1, 5, "SIM Lock");
        CTOS_KBDGet(&bTmp);
        stRtn.boIsEnabled = FALSE;
    }
    else if (stRtn.usRtn == d_GSM_SIM_NOT_LOCK)
    {
        CTOS_LCDTPrintXY(1, 5, "SIM Not Lock");
        CTOS_KBDGet(&bTmp);
        stRtn.boIsEnabled = FALSE;
    }
    else
    {
        stRtn.boIsEnabled = TRUE;
    }
}

```

```

        break;
case 44:
    strcpy(stRtn.strTitle, "CTOS_PINLock");

    CTOS_LCDTPrintXY(1, 2, "1: SC 2: PS");
    CTOS_LCDTPrintXY(1, 3, "3: FD");
    boRtn = GetNumber(8, 4, 1, FALSE, &ulTmp);
    if (boRtn == FALSE)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }

    if (ulTmp > 3)
    {
        stRtn.boIsEnabled = FALSE;
        CTOS_LCDTPrintXY(1, 8, "Invalid Pin Type");
        CTOS_KBDGet(&bTmp);
        break;
    }

    CTOS_LCDTPrintXY(1, 5, "PIN:");
    bTmpLen = GetString(8, 5, baTmp, 5);
    if (bTmpLen == 0)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }

    // Lock the SC, PS or FD using current PIN.
    if (ulTmp == 1)
        stRtn.usRtn = CTOS_PINLock(d_GSM_AUTH_SC, baTmp, bTmpLen);
    else if (ulTmp == 2)
        stRtn.usRtn = CTOS_PINLock(d_GSM_AUTH_PS, baTmp, bTmpLen);
    else
        stRtn.usRtn = CTOS_PINLock(d_GSM_AUTH_FD, baTmp, bTmpLen);
    break;
case 45:
    strcpy(stRtn.strTitle, "CTOS_PINUnLock");

    CTOS_LCDTPrintXY(1, 2, "1: SC 2: PS");
    CTOS_LCDTPrintXY(1, 3, "3: FD");
    boRtn = GetNumber(8, 4, 1, FALSE, &ulTmp);
    if (boRtn == FALSE)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }

    if (ulTmp > 3)
    {
        stRtn.boIsEnabled = FALSE;
        CTOS_LCDTPrintXY(1, 8, "Invalid Pin Type");
        CTOS_KBDGet(&bTmp);
        break;
    }

    CTOS_LCDTPrintXY(1, 5, "PIN:");
    bTmpLen = GetString(8, 5, baTmp, 5);
    if (bTmpLen == 0)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }

    // Unlock the SC, PS or FD.
    if (ulTmp == 1)
        stRtn.usRtn = CTOS_PINUnLock(d_GSM_AUTH_SC, baTmp, bTmpLen);
    else if (ulTmp == 2)
        stRtn.usRtn = CTOS_PINUnLock(d_GSM_AUTH_PS, baTmp, bTmpLen);
    else
        stRtn.usRtn = CTOS_PINUnLock(d_GSM_AUTH_FD, baTmp, bTmpLen);
    break;
case 46:
    strcpy(stRtn.strTitle, "CTOS_PINUpdate");

```

```

CTOS_LCDTPrintXY(1, 2, "1: SC 2: PS");
CTOS_LCDTPrintXY(1, 3, "3: FD");
boRtn = GetNumber(8, 4, 1, FALSE, &ulTmp);
if (boRtn == FALSE)
{
    stRtn.boIsEnabled = FALSE;
    break;
}

if (ulTmp > 3)
{
    stRtn.boIsEnabled = FALSE;
    CTOS_LCDTPrintXY(1, 8, "Invalid Pin Type");
    CTOS_KBDGet(&bTmp);
    break;
}

CTOS_LCDTPrintXY(1, 5, "Old PIN:");
bTmpLen = GetString(10, 5, baTmp, 5);
if (bTmpLen == 0)
{
    stRtn.boIsEnabled = FALSE;
    break;
}

CTOS_LCDTPrintXY(1, 6, "New PIN:");
bTmpLen = GetString(10, 6, baTmp2, 5);
if (bTmpLen == 0)
{
    stRtn.boIsEnabled = FALSE;
    break;
}

// Change the PIN.
if (ulTmp == 1)
    stRtn.usRtn = CTOS_PINUpdate(d_GSM_AUTH_SC, baTmp, bTmpLen, baTmp2,
bTmpLen);
else if (ulTmp == 2)
    stRtn.usRtn = CTOS_PINUpdate(d_GSM_AUTH_PS, baTmp, bTmpLen, baTmp2,
bTmpLen);
else
    stRtn.usRtn = CTOS_PINUpdate(d_GSM_AUTH_FD, baTmp, bTmpLen, baTmp2,
bTmpLen);
break;
/////////////////////////////////////////////////////////////////
case 50:
    CTOS_LCDTPrintXY(1, 1, "51. Dial");
    CTOS_LCDTPrintXY(1, 2, "52. To Cmd Mode");
    CTOS_LCDTPrintXY(1, 3, "53. To Data Mode");
    CTOS_LCDTPrintXY(1, 4, "54. GPRS Attach");
    CTOS_LCDTPrintXY(1, 5, "55. GPRS Detach");
    CTOS_LCDTPrintXY(1, 6, "56. Reg State?");
    CTOS_LCDTPrintXY(1, 7, "57. PPP Connect");
    break;
case 51:
    strcpy(stRtn.strTitle, "CTOS_GSMDial");
    CTOS_LCDTPrintXY(1, 2, "Phone Number:");
    bTmpLen = GetString(1, 3, baTmp, 11);
    if (bTmpLen == 0)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }
    baTmp[bTmpLen++] = ';';
    // This command can be used to set up outgoing data calls.
    stRtn.usRtn = CTOS_GSMDial(baTmp, bTmpLen);
    break;
case 52:
    strcpy(stRtn.strTitle, "To Cmd Mode");
    // Return to the command mode after you have built a connection and entered
the data mode.
    stRtn.usRtn = CTOS_GSMSwitchToCmdMode();
    break;
case 53:
    strcpy(stRtn.strTitle, "To Data Mode");

```

```

        // Return to the data mode after you have built a connection and entered the
        command mode.
        stRtn.usRtn = CTOS_GSMSwitchToDataMode();
        break;
    case 54:
        strcpy(stRtn.strTitle, "CTOS_GPRSAttach");
        // This command is used to attach the MT from the GPRS service.
        stRtn.usRtn = CTOS_GPRSAttach("internet", strlen("internet"));
        break;
    case 55:
        strcpy(stRtn.strTitle, "CTOS_GPRSDetach");
        // This command is used to detach the MT from the GPRS service.
        stRtn.usRtn = CTOS_GPRSDetach();
        break;
    case 56:
        strcpy(stRtn.strTitle, "CTOS_GPRSGetRegState");
        // Get the GPRS network registration status.
        stRtn.usRtn = CTOS_GPRSGetRegState(&bTmp);
        if (stRtn.usRtn == d_OK)
        {
            sprintf(baStr, "Register: %d", bTmp);
            CTOS_LCDTPrintXY(1, 2, baStr);
        }
        break;
    case 57:
        strcpy(stRtn.strTitle, "CTOS_GPRSPPPConnect");
        // This command causes the MT to perform all actions which are necessary to
        establish communication between the TE
        // and the network using layer 2 protocol PPP types.
        stRtn.usRtn = CTOS_GPRSPPPConnect();
        break;
///////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////
case 60:
    CTOS_LCDTPrintXY(1, 1, "61. PB Select");
    CTOS_LCDTPrintXY(1, 2, "62. PB Read");
    CTOS_LCDTPrintXY(1, 3, "63. PB Write");
    CTOS_LCDTPrintXY(1, 4, "64. PB Delete");
    break;
case 61:
    strcpy(stRtn.strTitle, "CTOS_PBSelect");

    CTOS_LCDTPrintXY(1, 2, "1: FD 2: SM");
    CTOS_LCDTPrintXY(1, 3, "3: ON 4: ME");
    CTOS_LCDTPrintXY(1, 4, "5: LD 6: MC");
    CTOS_LCDTPrintXY(1, 5, "7: RC");
    boRtn = GetNumber(8, 5, 1, FALSE, &ulTmp);
    if (boRtn == FALSE)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }

    if (ulTmp > 7)
    {
        stRtn.boIsEnabled = FALSE;
        CTOS_LCDTPrintXY(1, 8, "Invalid PB Type");
        CTOS_KBDGet(&bTmp);
        break;
    }

    // Select Phone Book.
    if (ulTmp == 1)
        stRtn.usRtn = CTOS_PBSelect(d_GSM_PB_FD, &bTmp, &bTmp2);
    else if (ulTmp == 2)
        stRtn.usRtn = CTOS_PBSelect(d_GSM_PB_SM, &bTmp, &bTmp2);
    else if (ulTmp == 3)
        stRtn.usRtn = CTOS_PBSelect(d_GSM_PB_ON, &bTmp, &bTmp2);
    else if (ulTmp == 4)
        stRtn.usRtn = CTOS_PBSelect(d_GSM_PB_ME, &bTmp, &bTmp2);
    else if (ulTmp == 5)
        stRtn.usRtn = CTOS_PBSelect(d_GSM_PB_LD, &bTmp, &bTmp2);
    else if (ulTmp == 6)
        stRtn.usRtn = CTOS_PBSelect(d_GSM_PB_MC, &bTmp, &bTmp2);
    else
        stRtn.usRtn = CTOS_PBSelect(d_GSM_PB_RC, &bTmp, &bTmp2);

```

```

if (stRtn.usRtn == d_OK)
{
    sprintf(baStr, "Used: %d", bTmp);
    CTOS_LCDTPrintXY(1, 6, baStr);

    sprintf(baStr, "Total: %d", bTmp2);
    CTOS_LCDTPrintXY(1, 7, baStr);
}
break;
case 62:
strcpy(stRtn.strTitle, "CTOS_PBRead");

CTOS_LCDTPrintXY(1, 2, "Index:");
boRtn = GetNumber(8, 2, 3, TRUE, &ulTmp);
if (boRtn == FALSE)
{
    stRtn.boIsEnabled = FALSE;
    break;
}

// Read one record from the phone book by the index.
stRtn.usRtn = CTOS_PBRead((BYTE)ulTmp, &stPB);
if (stRtn.usRtn == d_OK)
{
    stPB.baName[stPB.bNameLen] = '\0';
    sprintf(baStr, "Name: %s", stPB.baName);
    CTOS_LCDTPrintXY(1, 3, baStr);

    stPB.baNumber[stPB.bNumberLen] = '\0';
    sprintf(baStr, "Num: %s", stPB.baNumber);
    CTOS_LCDTPrintXY(1, 4, baStr);

    sprintf(baStr, "Type: %d", stPB.bAddrType);
    CTOS_LCDTPrintXY(1, 5, baStr);
}
break;
case 63:
strcpy(stRtn.strTitle, "CTOS_PBWrite");

CTOS_LCDTPrintXY(1, 2, "Index:");
boRtn = GetNumber(8, 2, 3, TRUE, &ulTmp);
if (boRtn == FALSE)
{
    stRtn.boIsEnabled = FALSE;
    break;
}

CTOS_LCDTPrintXY(1, 3, "Name:");
bTmpLen = GetString(7, 3, stPB.baName, 9);
if (bTmpLen == 0)
{
    stRtn.boIsEnabled = FALSE;
    break;
}
else
    stPB.bNameLen = bTmpLen;

CTOS_LCDTPrintXY(1, 4, "Number:");
bTmpLen = GetString(7, 4, stPB.baNumber, 11);
if (bTmpLen == 0)
{
    stRtn.boIsEnabled = FALSE;
    break;
}
else
    stPB.bNumberLen = bTmpLen;

stPB.bAddrType = 129;

// Add a new record to the phone book.
stRtn.usRtn = CTOS_PBWrite(&stPB);
break;
case 64:
strcpy(stRtn.strTitle, "CTOS_PBDDelete");

```

```

CTOS_LCDTPrintXY(1, 2, "Index:");
boRtn = GetNumber(8, 2, 3, TRUE, &ulTmp);
if (boRtn == FALSE)
{
    stRtn.boIsEnabled = FALSE;
    break;
}

// Remove a record from the phone book.
stRtn.usRtn = CTOS_PBDelete((BYTE)ulTmp);
break;
///////////////////////////////////////////////////////////////////
///////////////////////////////////////////////////////////////////
case 70:
    CTOS_LCDTPrintXY(1, 1, "71. Get SC Num");
    CTOS_LCDTPrintXY(1, 2, "72. Set SC Num");
    CTOS_LCDTPrintXY(1, 3, "73. Get List");
    CTOS_LCDTPrintXY(1, 4, "74. SMS Read");
    CTOS_LCDTPrintXY(1, 5, "75. SMS Delete");
    CTOS_LCDTPrintXY(1, 6, "76. SMS Send");
    CTOS_LCDTPrintXY(1, 7, "77. SMS Send PDU");
    break;
case 71:
    strcpy(stRtn.strTitle, "CTOS_SMSGetSCNumber");
    // Get the SMS service center address.
    stRtn.usRtn = CTOS_SMSGetSCNumber(baTmp, &bTmpLen);
    if (stRtn.usRtn == d_OK)
    {
        baTmp[bTmpLen] = '\0';
        sprintf(baStr, "Num: %s", baTmp);
        CTOS_LCDTPrintXY(1, 2, baStr);
    }
    break;
case 72:
    strcpy(stRtn.strTitle, "CTOS_SMSSetSCNumber");

    CTOS_LCDTPrintXY(1, 2, "Num:");
    bTmpLen = GetString(6, 2, baTmp, 11);
    if (bTmpLen == 0)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }
    // Set the SMS service center address.
    stRtn.usRtn = CTOS_SMSSetSCNumber(baTmp, bTmpLen);
    break;
case 73:
    strcpy(stRtn.strTitle, "CTOS_SMSGetList");

    CTOS_LCDTPrintXY(1, 2, "1: Rec Unread");
    CTOS_LCDTPrintXY(1, 3, "2: Rec Read");
    CTOS_LCDTPrintXY(1, 4, "3: Sto Unsend");
    CTOS_LCDTPrintXY(1, 5, "4: Sto send");
    CTOS_LCDTPrintXY(1, 6, "5: All");
    boRtn = GetNumber(8, 7, 1, FALSE, &ulTmp);
    if (boRtn == FALSE)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }

    if (ulTmp > 5)
    {
        stRtn.boIsEnabled = FALSE;
        CTOS_LCDTPrintXY(1, 8, "Invalid SMS Type");
        CTOS_KBDGet(&bTmp);
        break;
    }

    // Read the all SMS records in the specify SMS storage.
    if (ulTmp == 1)
        stRtn.usRtn = CTOS_SMSGetList(d_GSM_SMS_REC_UNREAD, stSMS_t, &bTmp);
    else if (ulTmp == 2)
        stRtn.usRtn = CTOS_SMSGetList(d_GSM_SMS_REC_READ, stSMS_t, &bTmp);
    else if (ulTmp == 3)
        stRtn.usRtn = CTOS_SMSGetList(d_GSM_SMS_STO_UNSEND, stSMS_t, &bTmp);

```

```

    else if (ulTmp == 4)
        stRtn.usRtn = CTOS_SMSGetList(d_GSM_SMS_STO_SEND, stSMS_t, &bTmp);
    else
        stRtn.usRtn = CTOS_SMSGetList(d_GSM_SMS_ALL, stSMS_t, &bTmp);
    break;
case 74:
    strcpy(stRtn.strTitle, "CTOS_SMSRead");

    CTOS_LCDTPrintXY(1, 2, "Index:");
    boRtn = GetNumber(8, 2, 3, TRUE, &ulTmp);
    if (boRtn == FALSE)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }

    // Read one SMS record by the record index.
    stRtn.usRtn = CTOS_SMSRead((BYTE)ulTmp, stSMS_t);
    if (stRtn.usRtn == d_OK)
    {
        stSMS_t[0].baPhoneNumber[stSMS_t[0].bPhoneNumberLen] = '\0';
        sprintf(baStr, "Num: %s", stSMS_t[0].baPhoneNumber);
        CTOS_LCDTPrintXY(1, 3, baStr);

        stSMS_t[0].baMessage[stSMS_t[0].usMessageLen] = '\0';
        sprintf(baStr, "%s", stSMS_t[0].baMessage);
        CTOS_LCDTPrintXY(1, 4, baStr);
    }
    break;
case 75:
    strcpy(stRtn.strTitle, "CTOS_SMSDelete");

    CTOS_LCDTPrintXY(1, 2, "Index:");
    boRtn = GetNumber(8, 2, 3, TRUE, &ulTmp);
    if (boRtn == FALSE)
    {
        stRtn.boIsEnabled = FALSE;
        break;
    }

    // Delete SMS by the record index.
    stRtn.usRtn = CTOS_SMSDelete((BYTE)ulTmp);
    break;
case 76:
    strcpy(stRtn.strTitle, "CTOS_SMSSend");

    CTOS_LCDTPrintXY(1, 2, "Num:");
    bTmpLen = GetString(6, 2, stSMS_Submit_t.baDa, 11);
    if(bTmpLen == 0)
    {
        stRtn.boIsEnabled == FALSE;
        break;
    }

    stSMS_Submit_t.baDa[bTmpLen++] = ';';
    stSMS_Submit_t.bDaLen = bTmpLen;
    stSMS_Submit_t.bVp = 12;
    stSMS_Submit_t.bDcs = 0;
    strcpy(stSMS_Submit_t.baMessage, "Hello World!");
    stSMS_Submit_t.usMessageLen = 12;

    CTOS_LCDTPrintXY(1, 3, "Sending...");
    // Send SMS.
    stRtn.usRtn = CTOS_SMSSend(&stSMS_Submit_t, &bTmp);
    if (stRtn.usRtn == d_OK)
    {
        sprintf(baStr, "SMS Index: %d", bTmp);
        CTOS_LCDTPrintXY(1, 4, baStr);
    }
    break;
case 77:
    strcpy(baTmp, "11000B916407281553F80000AA0AE8329BFD4697D9EC37");
    // Send SMS using PDU mode.
    stRtn.usRtn = CTOS_SMSSendPDU(baTmp, strlen(baTmp), &bTmp);
    if (stRtn.usRtn == d_OK)
    {

```

```

        sprintf(baStr, "Index: %d", bTmp);
        CTOS_LCDTPrintXY(1, 2, baStr);
    }
    break;
}

if (ulCase % 10)
{
    ShowResult();

    ulCase /= 10;
    ulCase *= 10;
    continue;
}

CTOS_LCDTPrintXY(1, 8, "Case:");
boRtn = GetNumber(8, 8, 2, FALSE, &ulNextCase);

if (boRtn == FALSE)
{
    if (ulCase == 0)
        return 0;
    else
        ulCase = 0;
}
else
{
    ulCase = ulNextCase;
}
}

/*=====
 * FUNCTION NAME: main
 * DESCRIPTION: function main is responsible for all the tasks listed in the
introduction to this demo.
 * RETURN: none.
 * NOTES: none.
===== */
int main(int argc,char *argv[])
{
    BYTE   bKey, baStr[32];
    USHORT usRtn;

    CTOS_LCDTClearDisplay();

    // Open GSM channel and initial to start to use the GSM functions.
    usRtn = CTOS_GSMOpen(115200, TRUE);
    if (usRtn != d_OK)
    {
        sprintf(baStr, "GSM Open: 0x%04X", usRtn);
        CTOS_LCDTPrintXY(1, 8, baStr);
        CTOS_KBDGet(&bKey);
        return 1;
    }

    GSM_Functions();

    // Close the GSM channel.
    CTOS_GSMClose();

    return 0;
}

// eof

```

## B.5. Hardware

```
/*
 * FILE NAME: hwtest
 * MODULE NAME: Hardware Test
 * PROGRAMMER: Peggy Chang
 * DESCRIPTION:
 * REVISION: 01.00
 */

=====
 *           I N C L U D E S
 =====

#include <string.h>
#include <ctosapi.h>

=====
 *           D E F I N E S
 =====
#define d_BUFF_SIZE 128      //Buffer Size

=====
 * FUNCTION NAME: ith
 * DESCRIPTION:
 * RETURN:    1 Byte
 * NOTES:    none.
 * =====
BYTE ith(BYTE hex_digit)
{
    hex_digit &= 0x0F;
    if(hex_digit > 9)
    {
        return hex_digit - 10 + 'A';
    }

    return hex_digit + '0';
}
=====
 * FUNCTION NAME: IntToStr
 * DESCRIPTION: Integer transform string
 * RETURN:    1 Byte
 * NOTES:    none.
 * =====
BYTE IntToStr(BYTE* buf, DWORD v)
{
    DWORD upp;
    DWORD i;

    upp = 0;
    i = v;
    while(i)
    {
        i /= 10;
        upp++;
    }

    if(!upp)
    {
        upp = 1;
    }

    for(i = 0; i < upp; i++)
    {
        buf[upp-1-i] = (v % 10) + '0';
        v /= 10;
    }
    return upp;
}
=====
```

```

* FUNCTION NAME: unpack
* DESCRIPTION:
* RETURN:      none.
* NOTES:      none.
* ===== */
void unpack(OUT char* pStr, IN BYTE* pData, IN USHORT Len, IN char* pSpilt)
{
    USHORT i;
    USHORT offset;
    BYTE s_len;

    s_len = strlen(pSpilt);
    for(i = 0; i < Len; ++i)
    {
        offset = (s_len + 2) * i;
        pStr[offset] = ith(pData[i] >> 4);
        pStr[offset+1] = ith(pData[i]);
        if(i < Len - 1)
        {
            memcpy(&pStr[offset+2], pSpilt, s_len);
        }
    }
    pStr[(s_len + 2) * i] = 0;
}
/* ===== */
* FUNCTION NAME: CryptoTest
* DESCRIPTION:  Test Crypto
* RETURN:      TRUE -> Crypto Test Success
*              FALSE-> Crypto Test Fail
* NOTES:      none.
* ===== */
BOOL CryptoTest(void)
{
    //Declare Local Variable //
    USHORT ret;
    BYTE key;
    BYTE rng[8];
    BYTE temp[d_BUFF_SIZE];

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, "--CRYPTO TEST---");

    memset(temp, 0x00, 128);
    ret = CTOS_RNG(rng);
    if(ret != d_OK)
    {
        goto crypto_fail;
    }
    unpack(temp, rng, 8, "");
    CTOS_LCDTPrintXY(1, 3, temp);

    CTOS_LCDTPrintXY(1, 8, "CRYPTO TEST OK ");
    CTOS_KBDGet(&key);

    return TRUE;

crypto_fail:
    CTOS_LCDTPrintXY(1, 8, "CRYPTO TEST FAIL");
    CTOS_KBDGet(&key);
    return FALSE;
}
/* ===== */
* FUNCTION NAME: InfoTest
* DESCRIPTION:  System Information
* RETURN:      none.
* NOTES:      none.
* ===== */
BOOL InfoTest(void)
{
    //Declare Local Variable //
    BYTE* const info_str[] = {"BootSULD      ",  

                             "Basic :      ",  

                             "CryptoKMS :   ",  

                             "SCMSR :      ",  

                             "ULK :        ",  

                             "FSTMS :      ",  

                             "          "
}

```

```

    "Peripheral :      ",
    "ULDPM :          ",
    "EMVL2UEP :       ",
    "CTOS :           ",
    "Peripheral2 :     "};

USHORT ret;
BYTE key;
BYTE sn[d_BUFF_SIZE];
BYTE str[d_BUFF_SIZE];
int i;

CTOS_LCDTClearDisplay();
CTOS_LCDTPrintXY(1, 1, "---INFO TEST----");

CTOS_LCDTPrintXY(1, 3, "Get Serial      ");

//Retrieves the serial number of the device //
ret = CTOS_GetSerialNumber(sn);
if(ret != d_OK)
{
    goto info_fail;
}
unpack(str, sn, 8, "");
CTOS_LCDTPrintXY(1, 4, str);
unpack(str, &sn[8], 8, "");
CTOS_LCDTPrintXY(1, 5, str);
CTOS_KBDGet(&key);

CTOS_LCDTPrintXY(1, 5, "      ");

memset(str, 0x00, 30);
for(i = 0; i <= ID_MAXIMUM; ++i) // d_KERNEL_PERIPHERAL2
{
    CTOS_LCDTPrintXY(1, 3, info_str[i]);

    //Retrieves the specified system component information fo the current system //
    ret = CTOS_GetSystemInfo(i, str);
    if(ret != d_OK)
    {
        goto info_fail;
    }
    CTOS_LCDTPrintXY(1, 4, str);
    CTOS_KBDGet(&key);
}

CTOS_LCDTPrintXY(1, 8, "INFO TEST OK      ");
CTOS_KBDGet(&key);

return TRUE;

info_fail:
    CTOS_LCDTPrintXY(1, 8, "INFO TEST FAIL  ");
    CTOS_KBDGet(&key);
    return FALSE;
}
/* =====
 * FUNCTION NAME: TimerTest
 * DESCRIPTION:
 * RETURN:      none.
 * NOTES:      none.
 * ===== */
BOOL TimerTest(void)
{
    //Declare Local Variable //
    USHORT ret;
    DWORD start, end, distance;
    BYTE key;
    BYTE upp;
    BYTE const TimerID[] = {TIMER_ID_1, TIMER_ID_2, TIMER_ID_3, TIMER_ID_4};
    BYTE str[d_BUFF_SIZE];
    int i;

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, "---TIMER TEST---");

    CTOS_LCDTPrintXY(1, 3, "Tick      ");

```

```

//The system timer tick is only incremented at set periods //
start = CTOS_TickGet();
CTOS_Delay(1000);
end = CTOS_TickGet();
distance = end - start;
if(distance < 99 || distance > 101)
{
    goto timer_fail;
}
CTOS_LCDTPrintXY(6, 3, "OK");

memcpy(str, "Timer           ", 16);
for(i = 0; i < 4; ++i)
{
    str[5] = '1' + TimerID[i];
    CTOS_LCDTPrintXY(1, 3, str);

    //Set an alarm flag that will be set to d_YES after the time configured by the
    specified time //
    ret = CTOS_TimeOutSet(TimerID[i], 500);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY(1, 4, "TimeoutSet Fail");
        goto timer_fail;
    }
    start = CTOS_TickGet();

    //Check the specified timer is timeout or not //
    while(CTOS_TimeOutCheck(TimerID[i]) != d_YES);
    end = CTOS_TickGet();
    distance = end - start;
    if(distance < 499 || distance > 501)
    {
        memset(str, ' ', 16);
        str[16] = 0;
        upp = 0;
        upp += IntToStr(&str[upp], distance);
        CTOS_LCDTPrintXY(1, 4, str);
        goto timer_fail;
    }
    CTOS_LCDTPrintXY(8, 3, "OK");
    CTOS_Delay(500);
}

CTOS_LCDTPrintXY(1, 8, "TIMER TEST OK    ");
CTOS_KBDGet(&key);

return TRUE;

timer_fail:
    CTOS_LCDTPrintXY(1, 8, "TIMER TEST FAIL ");
    CTOS_KBDGet(&key);
    return FALSE;
}
/* =====
 * FUNCTION NAME: RTC Test
 * DESCRIPTION: Use this function to read the date and the time settings of the Real
Time Clock
 * RETURN:      none.
 * NOTES:       none.
 * ===== */
BOOL RTCTest(void)
{
    //Declare Local Variable //
    USHORT ret;
    CTOS_RTC stRTC;
    BYTE upp;
    BYTE key;
    BYTE str[17];

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, "----RTC TEST----");

    //Get date & time //
    ret = CTOS_RTCGet(&stRTC);

```

```

if(ret != d_OK)
{
    goto rtc_fail;
}
CTOS_LCDTPrintXY(1, 3, "RTC Get      ");
memset(str, ' ', 16);
str[16] = 0;
upp = 0;
upp += IntToStr(&str[upp], stRTC.bYear) + 1;
upp += IntToStr(&str[upp], stRTC.bMonth) + 1;
upp += IntToStr(&str[upp], stRTC.bDay) + 1;
upp += IntToStr(&str[upp], stRTC.bHour) + 1;
upp += IntToStr(&str[upp], stRTC.bMinute) + 1;
upp += IntToStr(&str[upp], stRTC.bSecond) + 1;
CTOS_LCDTPrintXY(1, 4, str);
upp = 0;
upp += IntToStr(&str[upp], stRTC.bDoW) + 1;
str[upp] = 0;
CTOS_LCDTPrintXY(1, 5, str);

CTOS_LCDTPrintXY(1, 8, "RTC TEST OK      ");
CTOS_KBDGet(&key);

return TRUE;

rtc_fail:
CTOS_LCDTPrintXY(1, 8, "RTC TEST FAIL      ");
CTOS_KBDGet(&key);
return FALSE;
}
/* =====
* FUNCTION NAME: Buzzer Test
* DESCRIPTION:
* RETURN:      none.
* NOTES:      none.
* ===== */
BOOL BuzzerTest(void)
{
//Declare Local Variable //
USHORT ret;
BYTE key;

CTOS_LCDTClearDisplay();
CTOS_LCDTPrintXY(1, 1, "--BUZZER TEST---");
CTOS_LCDTPrintXY(1, 3, "Beep      ");

//Generates a simple tone on the system speaker //
ret = CTOS_Beep();
if(ret != d_OK)
{
    goto buuzer_fail;
}
CTOS_LCDTPrintXY(6, 3, "OK");
CTOS_Delay(500);

CTOS_LCDTPrintXY(1, 3, "Sound      ");

//Generates a simple sound from the system speaker //
ret = CTOS_Sound(1500, 100);
if(ret != d_OK)
{
    goto buuzer_fail;
}
CTOS_LCDTPrintXY(7, 3, "OK");
CTOS_Delay(500);

CTOS_LCDTPrintXY(1, 8, "BUZZER TEST OK      ");
CTOS_KBDGet(&key);

return TRUE;

buuzer_fail:
CTOS_LCDTPrintXY(1, 8, "BUZZER TEST FAIL");
CTOS_KBDGet(&key);
return FALSE;
}

```

```

}

/* =====
 * FUNCTION NAME: LED Test
 * DESCRIPTION:
 * RETURN:      none.
 * NOTES:      none.
 * ===== */
BOOL LEDTest(void)
{
    //Declare Local Variable //
    USHORT ret;
    BYTE key;
    BYTE const Led[] = {d_LED1, d_LED1, d_LED2, d_LED2, d_LED3, d_LED3};
    BYTE const OnOff[] = {d_ON, d_OFF, d_ON, d_OFF, d_ON, d_OFF};
    BYTE str[d_BUFF_SIZE];
    int i;

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, "----LED TEST----");
    str[16] = 0;
    memcpy(str, "LED1", 16);
    for(i = 0; i < 6; ++i)
    {
        //Turn on/off specific LED //
        ret = CTOS_LEDSet(Led[i], OnOff[i]);
        str[3] = '1' + Led[i];
        if(OnOff[i]) memcpy(&str[5], "ON ", 3);
        else memcpy(&str[5], "OFF", 3);

        CTOS_LCDTPrintXY(1, 3, str);

        if(ret != d_OK) goto led_fail;

        CTOS_LCDTPrintXY(10, 3, "OK");
        CTOS_Delay(500);
    }

    CTOS_LCDTPrintXY(1, 8, "LED TEST OK      ");
    CTOS_KBDGet(&key);

    return TRUE;

led_fail:
    CTOS_LCDTPrintXY(1, 8, "LED TEST FAIL   ");
    CTOS_KBDGet(&key);
    return FALSE;
}
/* =====
 * FUNCTION NAME: main
 * DESCRIPTION:      function main is responsible for all the tasks listed in the
introduction to this demo.
 * RETURN:      none.
 * NOTES:      none.
 * ===== */
int main(int argc,char *argv[])
{
    //Declare Local Variable //
    BYTE key;

    while(1)
    {
        CTOS_LCDTClearDisplay();
        CTOS_LCDTSetReverse(TRUE);
        CTOS_LCDTPrintXY(1, 1, "      HW TEST      ");
        CTOS_LCDTSetReverse(FALSE);
        CTOS_LCDTPrintXY(1, 2, "1. LED Test");
        CTOS_LCDTPrintXY(1, 3, "2. RTC Test");
        CTOS_LCDTPrintXY(1, 4, "3. Timer Test");
        CTOS_LCDTPrintXY(1, 5, "4. Buzzer Test");
        CTOS_LCDTPrintXY(1, 6, "5. Crypto Test");
        CTOS_LCDTPrintXY(1, 7, "6. Info Test");
        CTOS_LCDTPrintXY(1, 8, "7. Exit");

        CTOS_KBDGet(&key);
    }
}

```

```
switch(key)
{
case d_KBD_1:
    LEDTest();
    break;
case d_KBD_2:
    RTCTest();
    break;
case d_KBD_3:
    TimerTest();
    break;
case d_KBD_4:
    BuzzerTest();
    break;
case d_KBD_5:
    CryptoTest();
    break;
case d_KBD_6:
    InfoTest();
    break;
case d_KBD_7:
case d_KBD_CANCEL:
    return 1;
}
}

return 1;
}

// eof
```

## B.6. Hello

```
*****  
* FILE NAME: hello  
* MODULE NAME:  
* PROGRAMMER:  
* DESCRIPTION: Show "hello" on the LCD Display  
* REVISION: 01.00  
*****  
  
#include <string.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <stdarg.h>  
/** These two files are necessary for calling CTOS API **/  
#include <ctosapi.h>  
/** Uncomment this line if you want to show debug information in the simulator  
#include <debugprint.h>  
**/  
  
/**  
* The main entry of the terminal application  
**/  
int main(int argc,char *argv[])  
{  
    BYTE key;  
  
    // TODO: Add your program here //  
    CTOS_LCDTClearDisplay();  
  
    CTOS_LCDTPrintXY(1, 1, "Hello");  
  
    CTOS_KBDGet(&key);  
  
    return 0;  
}  
  
// eof
```

## B.7. Keyboard

```
/*
 * FILE NAME: KBDTest
 * MODULE NAME: Keyboard & Time
 * PROGRAMMER: Peggy Chang
 * DESCRIPTION: Test Keyboard and Modify Date & Time
 * REVISION:01.00
 */

=====
 *           I N C L U D E S           *
=====

#include <string.h>
#include <stdio.h>
#include <ctosapi.h>
#include <stdlib.h>
#include <stdarg.h>

unsigned long wub_str_2_long(unsigned char *str)
{
    unsigned char i;
    unsigned long k;
    i = 0;
    k = 0;
    while (str[i] != 0x00)
    {
        k += str[i ++] - '0';
        if (str[i] != 0x00)
            k *= 10;
    }
    return k;
}

=====
 *           D E F I N E S           *
=====

#define d_BUFF_SIZE 32      //Buffer Size

//Declare Global Variable //
BYTE key;
BYTE babuff[d_BUFF_SIZE];
BYTE isSoundTurnON;

=====
 * FUNCTION NAME: show_screen
 * DESCRIPTION: Show on the LCD Display
 * RETURN: none.
 * NOTES: none.
=====
void show_screen(int tag)
{
    //Clear LCD Display //
    CTOS_LCDTClearDisplay();

    switch(tag)
    {
        case 0:
            CTOS_LCDTSetReverse(TRUE);
            CTOS_LCDTPrintXY(1, 1, "      Key Board      ");
            CTOS_LCDTSetReverse(FALSE);
            CTOS_LCDTPrintXY(1, 2, "1.Get One Key");
            if (isSoundTurnON) CTOS_LCDTPrintXY(1, 3, "2.Sound Turn OFF");
            else CTOS_LCDTPrintXY(1, 3, "2.Sound Turn ON ");
            CTOS_LCDTPrintXY(1, 4, "3.Set Frequence");
            CTOS_LCDTPrintXY(1, 5, "4.Set RTC");
            break;
        case 1:
            CTOS_LCDTSetReverse(TRUE);
            CTOS_LCDTPrintXY(1, 1, "      InputString      ");
    }
}
```

```

        CTOS_LCDTSetReverse(FALSE);
        break;
    case 2:
        CTOS_LCDTSetReverse(TRUE);
        CTOS_LCDTPrintXY(1, 1, " Set Frequence ");
        CTOS_LCDTSetReverse(FALSE);
        break;
    }
}

/*=====
* FUNCTION NAME: GetOneKey
* DESCRIPTION:   Get one key from the keyboard and show key value on the LCD Display.
* RETURN: none.
* NOTES:  none.
*=====
*/
void GetOneKey(void)
{
    //Declare Local Variable //
    BOOL isKey;

    CTOS_LCDTClearDisplay();
    CTOS_LCDTSetReverse(TRUE);
    CTOS_LCDTPrintXY(1, 1, " GetOneKey ");
    CTOS_LCDTSetReverse(FALSE);
    CTOS_LCDTPrintXY(1, 8, "Key -> ");
    while (1)
    {
        //Detect if any key is pressed //
        CTOS_KBDInKey(&isKey);
        if (isKey)
        { //If isKey is TRUE, represent key be pressed //

            //Get a key from keyboard //
            CTOS_KBDGet(&key);

            if (key == d_KBD_CANCEL){
                CTOS_LCDTPrintXY(8, 8, "X      ");
                CTOS_Delay(1000);
                show_screen(0);
                return;
            }else if (key == d_KBD_CLEAR){
                CTOS_LCDTPrintXY(8, 8, "<-      ");
            }else if (key == d_KBD_ENTER){
                CTOS_LCDTPrintXY(8, 8, "OK      ");
            }else if (key == d_KBD_DOT){
                CTOS_LCDTPrintXY(8, 8, ".      ");
            }else if (key == d_KBD_F1){
                CTOS_LCDTPrintXY(8, 8, "F1      ");
            }else if (key == d_KBD_F2){
                CTOS_LCDTPrintXY(8, 8, "F2      ");
            }else if (key == d_KBD_F3){
                CTOS_LCDTPrintXY(8, 8, "F3      ");
            }else if (key == d_KBD_F4){
                CTOS_LCDTPrintXY(8, 8, "F4      ");
            }else if (key == d_KBD_UP){
                CTOS_LCDTPrintXY(8, 8, "UP      ");
            }else if (key == d_KBD_DOWN){
                CTOS_LCDTPrintXY(8, 8, "DOWN    ");
            }else if (((key >=0x30)&&(key <=0x39))){
                CTOS_LCDTPutChXY(8, 8,key);
                CTOS_LCDTPrintXY(9, 8, "      ");
            }
        }
    }
}

/*=====
* FUNCTION NAME: InputString
* DESCRIPTION:   Continue to input two or more characters at a time, build a string
* RETURN: none.
* NOTES:  none.
*=====
*/
int InputString(USHORT usX, USHORT usY, BYTE isMask, BYTE *pbaStr, USHORT *usStrLen)
{
    //Declare Local Variable //

```

```

int i = 0;

memset(pbaStr, 0x00, *usStrLen);
CTOS_LCDTPutchXY(usX, usY, '_');

while(1)
{
    //Scan the keyboard to detect whether a key is pressed //
    CTOS_KBDHit(&key);

    //Cancel InputString() //
    if (key == d_KBD_CANCEL){
        CTOS_LCDTPutchXY(usX+i, usY, ' ');
        memset(pbaStr, 0x00, *usStrLen);
        usStrLen = 0;
        return 0;
    }

    //Done input and output a string //
    if (key == d_KBD_ENTER){
        CTOS_LCDTPutchXY(usX+i, usY, ' ');
        *usStrLen = strlen(pbaStr);
        if (*usStrLen > 0) return 1;
        else return 0;
    }else if (key == d_KBD_CLEAR){ //Backspace //
        CTOS_LCDTPutchXY(usX+i, usY, ' ');
        i--;
        pbaStr[i] = 0x00;
        CTOS_LCDTPutchXY(usX+i, usY, '_');
    }else if(key == d_KBD_00){ //Clear all input data //
        memset(pbaStr, 0x00, *usStrLen);
        i = 0;
        CTOS_LCDTGotoXY(usX+i, usY);
        CTOS_LCDTClear2EOL();
        CTOS_LCDTPutchXY(usX, usY, '_');
    }
    if (i+1 <= *usStrLen){
        if ((key >= 0x30) && (key <= 0x39)){
            pbaStr[i] = key;

            if (isMask) CTOS_LCDTPutchXY(usX+i, usY, '*');
            else CTOS_LCDTPutchXY(usX+i, usY, pbaStr[i]);
            i++;
            CTOS_LCDTPutchXY(usX+i, usY, '_');
        }
    }
}

/*=====
 * FUNCTION NAME: SetRTC
 * DESCRIPTION: Use this function to set the real-time clock's data and time.
 * RETURN: none.
 * NOTES: none.
 *=====
 */
void SetRTC(void)
{
    //Declare Local Variable //
    CTOS_RTC SetRTC;
    USHORT i;
    BYTE isSet = FALSE;
    BYTE baYear[4+1], baMonth[2+1], baDay[2+1], baHour[2+1], baMinute[2+1], baSecond[2+1];

    CTOS_LCDTClearDisplay();
    CTOS_LCDTSetReverse(TRUE);
    CTOS_LCDTPrintXY(1, 1, "      Set RTC      ");
    CTOS_LCDTSetReverse(FALSE);

    //Read the date and the time //
    CTOS_RTCGet(&SetRTC);

    //Show on the LCD Dispaly //
    CTOS_LCDTPrintXY(1, 2, "      Get      Set");
    sprintf(babuff, "YY:%04d", SetRTC.bYear + 2000);
    CTOS_LCDTPrintXY(1, 3, babuff);
}

```

```

sprintf(babuff,"MM:%02d",SetRTC.bMonth);
CTOS_LCDTPrintXY(1, 4, babuff);
sprintf(babuff,"DD:%02d",SetRTC.bDay);
CTOS_LCDTPrintXY(1, 5, babuff);
sprintf(babuff,"hh:%02d",SetRTC.bHour);
CTOS_LCDTPrintXY(1, 6, babuff);
sprintf(babuff,"mm:%02d",SetRTC.bMinute);
CTOS_LCDTPrintXY(1, 7, babuff);
sprintf(babuff,"ss:%02d",SetRTC.bSecond);
CTOS_LCDTPrintXY(1, 8, babuff);

sprintf(babuff,"%02d",SetRTC.bDoW);
CTOS_LCDTPrintXY(15, 8, babuff);

//Input data for the setting //
i = sizeof(baYear);
if (InputString(10,3,FALSE, baYear, &i) == 1){ //Input Year //
    SetRTC.bYear = (wub_str_2_long(baYear) - 2000);
    isSet = TRUE;
}
i = sizeof(baMonth);
if (InputString(10,4,FALSE, baMonth, &i) == 1){ //Input Month //
    SetRTC.bMonth = wub_str_2_long(baMonth);
    isSet = TRUE;
}
i = sizeof(baDay);
if (InputString(10,5,FALSE, baDay, &i) == 1){ //Input Day //
    SetRTC.bDay = wub_str_2_long(baDay);
    isSet = TRUE;
}
i = sizeof(baHour);
if (InputString(10,6,FALSE, baHour, &i) == 1){ //Input Hour //
    SetRTC.bHour = wub_str_2_long(baHour);
    isSet = TRUE;
}
i = sizeof(baMinute);
if (InputString(10,7,FALSE, baMinute, &i) == 1){ //Input Minute //
    SetRTC.bMinute = wub_str_2_long(baMinute);
    isSet = TRUE;
}
i = sizeof(baSecond);
if (InputString(10,8,FALSE, baSecond, &i) == 1){ //Input Second //
    SetRTC.bSecond = wub_str_2_long(baSecond);
    isSet = TRUE;
}
///////////////////////////////
if (isSet){
    //Set the date and time //
    if (CTOS_RTCSet(&SetRTC) == d_OK) CTOS_LCDTPrintXY(15, 2, "OK");
    else CTOS_LCDTPrintXY(13, 2, "Fail");
    isSet = FALSE;
}

CTOS_KBDGet ( &key );
show_screen(0);
return;
}

/*=====
 * FUNCTION NAME: main
 * DESCRIPTION: function main is responsible for all the tasks listed in the
introduction to this demo.
 * RETURN: none.
 * NOTES: none.
 *=====
 */
int main (int argc, char *argv[])
{
    //Declare Local Variable //
    USHORT i, usFreq=0, usDur=0;
    BYTE isFreq = FALSE, isDur = FALSE;

    // TODO: Add your program here //
    //Initial //
    isSoundTurnON = TRUE;
}

```

```

//Turn on the keyboard sound //
CTOS_KBDSetSound(d_ON); //Default //

show_screen(0);

while(1){
    CTOS_KBDGet ( &key );
    switch(key){
        case d_KBD_1:
            GetOneKey();
            break;
        case d_KBD_2:
            if (isSoundTurnON){
                //Turn off the keyboard sound //
                CTOS_KBDSetSound(d_OFF);
                isSoundTurnON = FALSE;
                CTOS_LCDTPrintXY(1, 3, "2.Sound Turn ON ");
            }else{
                //Turn on the keyboard sound //
                CTOS_KBDSetSound(d_ON);
                isSoundTurnON = TRUE;
                CTOS_LCDTPrintXY(1, 3, "2.Sound Turn OFF");
            }
            break;
        case d_KBD_3:
            //Set the sound frequency and duration for keyboard //
            show_screen(2);
            i = sizeof(babuff);
            CTOS_LCDTPrintXY(1, 2, "Sound Frequency");
            //Input values for the settings //
            if (InputString(1,3,FALSE,babuff,&i) == 1){
                usFreq = wub_str_2_long(babuff);
                sprintf(babuff,"%d",usFreq);
                CTOS_LCDTPrintXY(10, 3, babuff);
                isFreq = TRUE;
            }
            CTOS_LCDTPrintXY(1, 4, "Duration");
            i = sizeof(babuff);
            if (InputString(1,5,FALSE,babuff,&i) == 1){
                usDur = wub_str_2_long(babuff);
                sprintf(babuff,"%d",usDur);
                CTOS_LCDTPrintXY(10, 5, babuff);
                isDur = TRUE;
            }
            if ((isFreq) && (isDur)){
                //Set to the keyboard //
                if (CTOS_KBDSetFrequence(usFreq, usDur) == d_OK){
                    isFreq = FALSE;
                    isDur = FALSE;
                    CTOS_LCDTPrintXY(1, 8, "Set....OK");
                }else{
                    CTOS_LCDTPrintXY(1, 8, "Set....Fail");
                }
            }else{
                CTOS_LCDTPrintXY(1, 8, "Set....Cancel");
            }
            CTOS_KBDGet ( &key );
            show_screen(0);
            break;
        case d_KBD_4:
            SetRTC();
            break;
        case d_KBD_CANCEL:
            return 0;
            break;
    }
}
return 0;
}

// eof

```

## B.8. Keypad

```
/*
 * FILE NAME: keypadtest
 * MODULE NAME: CTOS_UIKeypad
 * PROGRAMMER: Karl Chao
 * DESCRIPTION:
 * REVISION: 01.00
 ****

=====
 *           I N C L U D E S
 =====*/
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <ctosapi.h>

=====
 * FUNCTION NAME: main
 * DESCRIPTION: function main is responsible for all the tasks listed in the
introduction to this demo.
 * RETURN: none.
 * NOTES: none.
 ====
int main(int argc,char *argv[])
{
    BYTE key;
    BYTE baBuff[256];

    //Define letter mapping to each key
    //normal english keyboard
    STR *keyboardLayoutEnglish[]={" 0", "qzQZ1", "abcABC2", "defDEF3", "ghiGHI4",
"jklJKL5", "mnoMNO6", "prsPRS7", "tuvTUV8", "wxyWXY9", ":;/\\|?,.<>", ".!@#$%^&*()"};
    //numeric keyboard (0123456789) and radix point '.'
    STR *keyboardLayoutNumberWithRadixPoint[]={"0", "1", "2", "3", "4", "5", "6", "7",
"8", "9", "", "."};
    //numeric keyboard (0123456789) without radix point
    STR *keyboardLayoutNumber[]={"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "",
""};

    //initial LCD screen and set font to Chinese_Taiwan 16x16
    CTOS_BackLightSet(d_BKLIT_LCD, d_ON); //turn on backlight
    CTOS_LanguageConfig(d_FONT_CHINESE_TAIWAN, d_FONT_16x16, 0, FALSE);
    CTOS_LanguageLCDFontSize(d_FONT_16x16, 0);
    CTOS_LCDTSelectFontSize(d_LCD_FONT_8x16);
    CTOS_LCDTClearDisplay();

    //Output example:
    //  Username:
    //  Casauto!_-      <-User type in this row
    //  Casauto!
    //  strlen(): 8
    //Allow user to enter a 8 bytes(9 including escape'\0') char string
    CTOS_LCDTPrintXY(4, 1, "Username:");
    //Parameter:
    //1. usX: 4 => Cursor horizontal position at 4
    //2. usY: 2 => Cursor vertical position at 2
    //3. *pcaKeyboardLayout[]: keyboardLayoutEnglish => use english keyboard layout
defined at the beginning of the program
    //4. ucCursorBlinkInterval: 40 => Cursor blink at 400ms interval
    //5. ucDelayToNextChar: 80 => Move Cursor to next character if user did not press
any key within 800ms
    //6. boEnableCursorMove: d_TRUE => Allow user to move cursor by F3 and F4 key
    //7. boOneRadixPointOnly: d_FALSE => Do not check radix point
    //8. ucDigitAfterRadixPoint: 0 => Do not limit digits after radix point
    //9. bPasswordMask: 0 => Do not mask user data.
    //10. caData[]: baBuff => Pointer to store the user data retrieved
    //11. ucDataLen: 9 => baBuff has 9 bytes. User may input 8 bytes max.
```

```

    CTOS_UIKeypad(4, 2, keyboardLayoutEnglish, 40, 80, d_TRUE, d_FALSE, 0, 0, baBuff,
9);
    CTOS_LCDTPrintXY(4, 3, baBuff);
    sprintf(baBuff, "strlen(): %d", strlen(baBuff));
    CTOS_LCDTPrintXY(3, 4, baBuff);
    CTOS_KBDGet( &key ); //wait until a key is pressed

    CTOS_LCDTClearDisplay(); //Clear Display

    //Output example:
    //  Password:
    //  *****_- <-User type in this row
    //  Casauto!
    //  strlen(): 8
    //Allow user to enter a 8 bytes(9 including escape'\0') char string) and mask data
with '**'
    CTOS_LCDTPrintXY(4, 1, "Password:");
    //Parameter:
    //9. bPasswordMask: '*' => Mask user data with '*'.
    CTOS_UIKeypad(4, 2, keyboardLayoutEnglish, 40, 80, d_TRUE, d_FALSE, 0, '*',
baBuff, 9);
    CTOS_LCDTPrintXY(4, 3, baBuff);
    sprintf(baBuff, "strlen(): %d", strlen(baBuff));
    CTOS_LCDTPrintXY(3, 4, baBuff);
    CTOS_KBDGet( &key ); //wait until a key is pressed

    CTOS_LCDTClearDisplay(); //Clear Display

    //Output Example:
    //Credit Card Nr.:
    //4567567878901234
    //4567567878901234
    //strlen(): 16
    //Allow user to enter a 17 bytes(16 including escape'\0') number and disable F3 and
F4 key
    CTOS_LCDTPrintXY(1, 1, "Credit Card Nr.:");
    //Parameter:
    //3. *pcaKeyboardLayout[]: keyboardLayoutNumber => use numeric keyboard
layout(0123456789) without radix point
    //6. boEnableCursorMove: d_FALSE => Disable moving cursor by F3 and F4 key
    //11. ucDataLen: 17 => baBuff has 17 bytes. User may input 16 bytes max.
    CTOS_UIKeypad(1, 2, keyboardLayoutNumber, 40, 80, d_FALSE, d_FALSE, 0, baBuff,
17);
    CTOS_LCDTPrintXY(1, 3, baBuff);
    sprintf(baBuff, "strlen(): %d", strlen(baBuff));
    CTOS_LCDTPrintXY(1, 4, baBuff);
    CTOS_KBDGet( &key ); //wait until a key is pressed

    CTOS_LCDTClearDisplay(); //Clear Display

    //Output Example:
    //CVV2:
    //***-
    //123
    //strlen(): 3
    //Allow user to enter a 3 bytes(4 including escape'\0') number and and mask data
with '**'
    CTOS_LCDTPrintXY(1, 1, "CVV2:");
    //Parameter:
    //3. *pcaKeyboardLayout[]: keyboardLayoutNumber => use numeric keyboard
layout(0123456789) without radix point
    //6. boEnableCursorMove: d_FALSE => Allow user to move cursor by F3 and F4 key
    //11. ucDataLen: 17 => baBuff has 17 bytes. User may input 16 bytes max.
    CTOS_UIKeypad(1, 2, keyboardLayoutNumber, 40, 80, d_TRUE, d_FALSE, 0, '*', baBuff,
4);
    CTOS_LCDTPrintXY(1, 3, baBuff);
    sprintf(baBuff, "strlen(): %d", strlen(baBuff));
    CTOS_LCDTPrintXY(1, 4, baBuff);
    CTOS_KBDGet( &key ); //wait until a key is pressed

    CTOS_LCDTClearDisplay(); //Clear Display

    //Output Example:
    //Amount:
    //USD$123.45_
    //123.45

```

```

//strlen(): 6
//Allow user to enter a 8 bytes(9 including escape '\0') number with max 1 radix
point and max 2 digits after radix point
    CTOS_LCDTPrintXY(1, 1, "Amount:");
    CTOS_LCDTPrintXY(1, 2, "USD$");
    //3. *pcaKeyboardLayout[]: keyboardLayoutNumberWithRadixPoint => use numeric
keyboard layout(0123456789) with radix point
    //7. boOneRadixPointOnly: d_TRUE => Allow only one radix point in user data
    //8. ucDigitAfterRadixPoint: 2 => Max 2 digits after radix point
    CTOS_UIKeypad(5, 2, keyboardLayoutNumberWithRadixPoint, 40, 80, d_TRUE, d_TRUE, 2,
0, baBuff, 9);
    CTOS_LCDTPrintXY(1, 3, baBuff);
    sprintf(baBuff, "strlen(): %d", strlen(baBuff));
    CTOS_LCDTPrintXY(1, 4, baBuff);
    CTOS_KBDGet( &key ); //wait until a key is pressed

    CTOS_LCDTSelectFontSize(d_LCD_FONT_8x8); //restore default font size

    return 0;
}

// eof

```

## B.9. Language

```
/*
 * FILE NAME: Chinese Language Test
 * MODULE NAME: Language
 * PROGRAMMER: Peggy Chang
 * DESCRIPTION: Display or print non-English font pattern. This sets the Chinese Language
 * REVISION:01.00
 */

=====
 *           I N C L U D E S           *
=====

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
/** These two files are necessary for calling CTOS API **/
#include <ctosapi.h>
/** Uncomment this line if you want to show debug information in the simulator
#include <debugprint.h>
**/

=====
 *           D E F I N E S           *
=====

#define d_BUFF_SIZE    256      //Buffer Size
#define d_LINE_DOT 12        //A line is 12 dots in Printer

=====
 * FUNCTION NAME: main
 * DESCRIPTION: function main is responsible for all the tasks listed in the
introduction to this demo.
 * RETURN: none.
 * NOTES: none.
 * =====

int main(int argc,char *argv[])
{
    BYTE key;
    BYTE babuff[d_BUFF_SIZE];
    USHORT usIndex = 0,usLanguage=0,usFontSize=0,usFontStyle=0;
    int i;

    memset(babuff,0x00,sizeof(babuff));

    // Set Language Config //
    CTOS_LanguageConfig(d_FONT_CHINESE_TAIWAN,d_FONT_16x16,0,FALSE);

    // Set LCD of Langguage (Chinese & English) Font Size //
    CTOS_LanguageLCDFontSize(d_FONT_16x16,0);
    CTOS_LCDTSelectFontSize(d_LCD_FONT_8x16);

    CTOS_LCDTClearDisplay ();
    CTOS_LCDTSetReverse(TRUE);
    sprintf(babuff,"%c%c%c%c%c%c%c",0xAD,0x69,0xB3,0xF9,0xAC,0xEC,0xA7,0xDE);
    CTOS_LCDTPrintXY(1,1,babuff); //Show Chinese Font in the LCD Display //
    CTOS_LCDTSetReverse(FALSE);

    sprintf(babuff,"VEGA7000%c%c%c%c%c%",0xA8,0xEA,0xA5,0x64,0xBE,0xF7);
    CTOS_LCDTPrintXY(1,2,babuff); //Show Chinese Font in the LCD Display //

    //How many language font installed on system //
    CTOS_LanguageNum(&usIndex);
    sprintf(babuff,"%c%c%c%c%c:c%d",0xA6,0x72,0xAB,0xAC,0xBA,0xD8,0xC3,0xFE,usIndex);
;

    CTOS_LCDTPrintXY(1,3,babuff); //Show Chinese Font in the LCD Display //

    sprintf(babuff,"%c%c%c%c%c:c%c:c...",0xBD,0xD0,0xAB,0xF6,0xA5,0xF4,0xA6,0xF3,0xC1
,0xE4);
    CTOS_LCDTPrintXY(1,4,babuff); //Show Chinese Font in the LCD Display //
```

```

//Set Printer Font Size //
CTOS_LanguagePrinterFontSize(d_FONT_16x16,0,d_FONT_TYPE1);

sprintf(babuff,"\\ft1%c%c%c%c%c%c",0xAD,0x69,0xB3,0xF9,0xAC,0xEC,0xA7,0xDE);
CTOS_PrinterPutString(babuff); //Print Chinese Font //
sprintf(babuff,"\\ft1VEGA7000%c%c%c%c%c",0xA8,0xEA,0xA5,0x64,0xBE,0xF7);
CTOS_PrinterPutString(babuff); //Print Chinese Font //

if (usIndex > 0){
    for (i=1; i<=usIndex; i++){
        //The Language font information installed //
        CTOS_LanguageInfo(usIndex,&usLanguage,&usFontSize,&usFontStyle);
        sprintf(babuff,"Index: %d",i);
        CTOS_PrinterPutString(babuff);
        sprintf(babuff,"Language: %d",usLanguage);
        CTOS_PrinterPutString(babuff);
        sprintf(babuff,"FontSize: %d",usFontSize);
        CTOS_PrinterPutString(babuff);
        sprintf(babuff,"FontStyle:%d",usFontStyle);
        CTOS_PrinterPutString(babuff);
    }
}

sprintf(babuff,"\\ft1%c%c%c%c%c%c%c...",0xBD,0xD0,0xAB,0xF6,0xA5,0xF4,0xA6,0xF3,
0xC1,0xE4);
CTOS_PrinterPutString(babuff); //Print Chinese Font //

CTOS_PrinterFline(d_LINE_DOT*10); //a space 1 line

CTOS_LCDTSelectFontSize(d_LCD_FONT_8x8); //Set default font size //
CTOS_KBDGet ( &key );

return 0;
}

// eof

```

## B.10. LCD Display

```

0x00,0x00,0x01,0x83,0xC1,0x71,0x39,0x1F,0xFF,0x0C,0x18,0x18,0x04,0x04,0x04,
0x04,0x04,0x06,0x06,0xFF,0xFF,0x07,0x02,0x03,0x02,0x02,0x00,0x00,0x00,0x00,0x00,
0x40,0xC0,0xE0,0x60,0x30,0x30,0xF8,0xFF,0x06,0x02,0x00,0x00,0x04,0x1C,0x34,0x66,
0xC7,0x83,0xF2,0x7E,0x1F,0x06,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x02,0x06,0x07,0x03,0x01,0x01,0x00,0x00,0x00,0x02,0x02,0x02,
0x02,0x03,0x03,0x03,0x03,0x03,0x03,0x03,0x03,0x03,0x03,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x10,0x30,0x30,0x30,0x31,0x11,0x11,0x11,0x1B,0x1F,0x1F,
0x19,0x19,0x19,0x18,0x18,0x18,0x18,0x18,0x18,0x18,0x10,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x01,0x01,0x00,0x00,0x00,0x1E,0x3F,0x1F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x3F,0x1F,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x0C,0x1C,0x1F,0x1F,0x00,0x00,0x10,0x10,0x18,0x08,0x0C,0x06,
0x07,0x07,0x0F,0x1E,0x1C,0x3C,0x38,0x38,0x10,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
};

/*=====
 * FUNCTION NAME: main
 * DESCRIPTION: function main is responsible for all the tasks listed in the
introduction to this demo.
 * RETURN: none.
 * NOTES: none.
 * ===== */
int main(int argc, char *argv[])
{
    //Declare Variable //
    BYTE key;
    USHORT ret;
    int i;
    BYTE babuff[d_BUFF_SIZE];

    // TODO: Add your program here //

    //Open the Back Light in the LCD Display //
    CTOS_BackLightSet(d_BKLIT_LCD,d_ON);

    //Set the LCD Display is Graphic Mode //
    CTOS_LCDSelectMode(d_LCD_GRAPHIC_MODE);

    //Clear Canvas //
    CTOS_LCDClearCanvas();

    //Draw a pixel on the canvas //
    for (i=0; i<=127; i++){
        CTOS_LCDGPixel(i,10,TRUE);      //Draw a horizontal line
        CTOS_LCDGPixel(20,i,TRUE);      //Draw a vertical line
    }

    //Draw a pattern on the canvas //
    CTOS_LCDGShowPic(80,0,(BYTE *)baLCDLogo,sizeof(baLCDLogo),46); //Draw the "CASTLES"
Graphic Logo //

    CTOS_LCDGSetBox(40,17,20,7,1); //Draw a rectangle //
    CTOS_LCDGSetBox(45,43,60,7,1); //Draw a rectangle //

    //Draw a string on the canvas //
    CTOS_LCDGTextOut(0,20,"CASTLES",d_LCD_FONT_8x16,FALSE); //Draw the "CASTLES"
string, Font size is 8x16, and no reverse //
    CTOS_LCDGTextOut(50,40,"Technology",d_LCD_FONT_8x8,FALSE); //Draw the "Technology"
string, Font size is 8x8 and no reverse//

    CTOS_Delay(3000); //Delay 3 second

    //Move the display window to display desire position of canvas //
    for (i=0; i<=20;i++){
        CTOS_LCDGMoveWindow(i); //Move 20 dot //
        CTOS_Delay(50); //Delay 50ms every dot //
    }
}

```

```

    CTOS_LCDGShowPic(0,30,(BYTE *)baLCDLogo_1,sizeof(baLCDLogo_1),128); //Draw the
    "CASTLES" Chinese Pattern Logo //
    CTOS_LCDGTextOut(0,20,"CASTLES",d_LCD_FONT_8x16,FALSE); //Draw the "CASTLES" string,
Font size is 8x16, and no reverse //

    //Get the current position of window //
    ret = CTOS_LCDGGetWindowOffset(); //ret = retrun value of offset of the current
window //

    CTOS_Delay(3000);
    for (i=ret; i>=0;i--) {
        CTOS_LCDGMoveWindow(i);
        CTOS_Delay(50);
    }

    CTOS_Delay(3000);
    CTOS_LCDGSetBox(0,0,128,64,1);

    for (i=0; i<=127; i++) {
        CTOS_LCDGPixel(i,10,FALSE);
        CTOS_LCDGPixel(20,i,FALSE);
    }
    CTOS_LCDGTextOut(0,20,"CASTLES",d_LCD_FONT_8x16,TRUE);
    CTOS_LCDGTextOut(50,40,"Technology",d_LCD_FONT_8x8,TRUE);

    for (i=0; i<=20;i++) {
        CTOS_LCDGMoveWindow(i);
        CTOS_Delay(50);
    }

    CTOS_Delay(3000);

    CTOS_LCDGClearWindow();

    ret = CTOS_LCDGGetWindowOffset();

    CTOS_LCDGShowPic(80,0,(BYTE *)baLCDLogo,sizeof(baLCDLogo),46);

    CTOS_Delay(3000);
    for (i=ret; i>=0;i--) {
        CTOS_LCDGMoveWindow(i);
        CTOS_Delay(50);
    }

    CTOS_Delay(3000);

    //Set the LCD Display is Text Mode //
    CTOS_LCDSelectMode(d_LCD_TEXT_MODE);

    //Display a character at current cursor position //
    CTOS_LCDTPutCh('C');
    CTOS_Delay(300);

    //Display a character at specific cursor position //
    CTOS_LCDTPutChXY(3,1,'A');
    CTOS_Delay(300);
    CTOS_LCDTPutChXY(5,1,'S');
    CTOS_Delay(300);
    CTOS_LCDTPutChXY(7,1,'T');
    CTOS_Delay(300);
    CTOS_LCDTPutChXY(9,1,'L');
    CTOS_Delay(300);
    CTOS_LCDTPutChXY(11,1,'E');
    CTOS_Delay(300);
    CTOS_LCDTPutChXY(13,1,'S');

    //Locate the cursor to specific coordinate //
    CTOS_LCDTGotoXY(1,2);

    CTOS_Delay(300);
    CTOS_LCDTPutCh('T');
    CTOS_Delay(300);
    CTOS_LCDTPutChXY(2,2,'e');
    CTOS_Delay(300);
    CTOS_LCDTPutChXY(3,2,'c');

```

```

CTOS_Delay(300);
CTOS_LCDTPutchXY(4,2,'h');
CTOS_Delay(300);
CTOS_LCDTPutchXY(5,2,'n');
CTOS_Delay(300);
CTOS_LCDTPutchXY(6,2,'o');
CTOS_Delay(300);
CTOS_LCDTPutchXY(7,2,'l');
CTOS_Delay(300);
CTOS_LCDTPutchXY(8,2,'o');
CTOS_Delay(300);
CTOS_LCDTPutchXY(9,2,'g');
CTOS_Delay(300);
CTOS_LCDTPutchXY(10,2,'y');

CTOS_Delay(500);

//Set the reverse attribute of the character //
CTOS_LCDTSetReverse(TRUE); //the reverse enable //

//Display a string at specific cursor position //
CTOS_LCDTPrintXY(1,4,"Castles"); //Display the "Castles" string in the 4 line//
CTOS_Delay(500);
CTOS_LCDTPrintXY(5,5,"Technology"); //Display the "Technology" string in the 5 line//
CTOS_LCDTSetReverse(FALSE); //the reverse disable //

memset(babuff,0x00,sizeof(babuff));
CTOS_Delay(500);

//Return the current X cursor position //
ret = CTOS_LCDTWhereX();
sprintf(babuff,"X => %d",ret);
CTOS_LCDTPrintXY(1,7,babuff); //Display value of return the current X crusor
position in the 7 line //

//Return the current Y cursor position //
ret = CTOS_LCDTWhereY();
sprintf(babuff,"Y => %d",ret);
CTOS_LCDTPrintXY(10,7,babuff); //Display value of return the current Y crusor
position in the 7 line //

CTOS_Delay(2000);
for (i=ret; i>=1; i--) {
    CTOS_LCDTGotoXY(1,i);

    //Clear all characters displayed from the cursor position to the end of line.
    CTOS_LCDTClear2EOL();
    CTOS_Delay(500);
}

//Set the English (ASCII code) font size attribute of the character //
CTOS_LCDTSelectFontSize(d_LCD_FONT_8x16);

CTOS_LCDTSetReverse(TRUE);

//Display a character at current cursor position //
CTOS_LCDTPrint("CASTLES"); //Dispaly the "CASTLES" string in currnet cursor position
// CTOS_Delay(500);
CTOS_LCDTPrintXY(1,2,"Technology");
CTOS_LCDTSetReverse(FALSE);

CTOS_Delay(500);
CTOS_LCDTPrintXY(1,3,"Castles");
CTOS_Delay(500);
CTOS_LCDTPrintXY(5,4,"Technology");

CTOS_Delay(2000);
//Clear LCD display and reset the cursor //
CTOS_LCDTClearDisplay();

CTOS_LCDTSelectFontSize(d_LCD_FONT_12x24);
CTOS_LCDTPrintXY(1,1,"CASTLES");
CTOS_Delay(500);
CTOS_LCDTSetReverse(TRUE);
CTOS_LCDTPrintXY(1,2,"Technology");

```

```
CTOS_LCDTSetReverse(FALSE);
CTOS_Delay(3000);

CTOS_LCDTSelectFontSize(d_LCD_FONT_8x16);
CTOS_LCDTPrintXY(1,4,"Press Any Key..");

//Set default font size //
CTOS_LCDTSelectFontSize(d_LCD_FONT_8x8);

//Close the Back Light in the LCD Display //
CTOS_BackLightSet(d_BKLIT_LCD,d_OFF);

CTOS_KBDGet ( &key );

return 0;
}

// eof
```

## B.11. Memory Card

```
#include <string.h>
#include <stdio.h>
#include <ctosapi.h>
#include <stdlib.h>
#include <stdarg.h>

BYTE Key;
//-----
void ShowRsp(char *str, BYTE *baB, USHORT usL)
{
char caMe[16];
ULONG i;
    CTOS_LCDTClearDisplay();
    sprintf(caMe,"%s",str);
    CTOS_LCDTPrintXY(1,1,caMe);
    sprintf(caMe,"[i=%d]",usL);
    CTOS_LCDTPrintXY(1,2,caMe);
    for(i=0;i<usL;i++)
    {
        sprintf(caMe,"%02X ",baB[i]);
        CTOS_LCDTPrintXY((i%8)*2+1,i/8+3,caMe);
    }
    CTOS_KBDGet(&Key);
    CTOS_LCDTClearDisplay();
}
//-----
void ShowRtn(USHORT usRtn,BYTE *baT)
{
STR sBuf[80];

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1,1,baT);
    sprintf(sBuf,"Rtn:0x%04X",usRtn);
    CTOS_LCDTPrintXY(1,2,sBuf);
    CTOS_KBDGet(&Key);
}
//-----
void SLESample(int iCase)
{
int i,j,iloop;
BYTE c,bLen,bID;
BYTE CardType;
USHORT usRtn,usLen;
STR sBuf[80];
BYTE baBuf[256],baPBuf[256],baATR[128];

    CTOS_LCDTClearDisplay();
    bID=d_SC_USER;
    switch(iCase)
    {
        case 1://SEL44x2
            CTOS_LCDTPrintXY(1,2,"Insert SLE44x2");
            CTOS_KBDGet(&Key);
            bLen = sizeof(baATR);
            usRtn=CTOS_SyncICCReset(baATR,&bLen);
            ShowRtn(usRtn,"Reset");
            if (usRtn==d_OK)
            {
                ShowRsp("ATR",&baATR[0],(USHORT)bLen);
                usRtn=CTOS_44x2VerifyPSC("\xFF\xFF\xFF");
                ShowRtn(usRtn,"Cmp VerifyData");
                if(usRtn==d_OK)
                {
                    usRtn=CTOS_44x2WriteMainMemory(0xA0,"xA0\xA1\xA2\xA3\xA4\xA5\xA6\xA7",0x08);
                    ShowRtn(usRtn,"Write MM");
                    if(usRtn==d_OK)
                    {
                        usLen=8;
                    }
                }
            }
        }
    }
}
```

```

        usRtn=CTOS_44x2ReadMainMemory(0xA0,baBuf,&usLen);
        ShowRtn(usRtn,"Reaed MM");
        if(usRtn==d_OK)
        {
            ShowRsp("MainMemory",&baBuf[0],usLen);
        }

usRtn=CTOS_44x2WriteMainMemory(0xA0,"\x11\x22\x33\x44\x55\x66\x77\x88",0x08);
ShowRtn(usRtn,"Write MM");
if(usRtn==d_OK)
{
    usLen=8;
    usRtn=CTOS_44x2ReadMainMemory(0xA0,baBuf,&usLen);
    ShowRtn(usRtn,"Reaed MM");
    if(usRtn==d_OK)
    {
        ShowRsp("MainMemory",&baBuf[0],usLen);
    }
}
CTOS_KBDGet(&Key);
break;
case 2://SEL44x2
CTOS_LCDTPrintXY(1,2,"Insert SLE44x2");
CTOS_KBDGet(&Key);
bLen = sizeof(baATR);
usRtn=CTOS_SyncICCReset(baATR,&bLen);
ShowRtn(usRtn,"Reset");
if (usRtn==d_OK)
{
    usRtn=CTOS_44x2VerifyPSC("\xFF\xFF\xFF");
    ShowRtn(usRtn,"Cmp VerifyData");
    if(usRtn==d_OK)
    {
        usLen=4;
        usRtn=CTOS_44x2ReadProtectionMemory(baBuf);
        ShowRtn(usRtn,"Read ProtM");
        if(usRtn==d_OK)
        {
            ShowRsp("ReadProM",&baBuf[0],usLen);
            usLen=3;
            usRtn=CTOS_44x2SetProtectionBit(0x01);
            ShowRtn(usRtn,"Write ProtM");
        }
    }
}
CTOS_KBDGet(&Key);
break;
case 3://SEL44x2
CTOS_LCDTPrintXY(1,2,"Insert SLE44x2");
CTOS_KBDGet(&Key);
bLen = sizeof(baATR);
usRtn=CTOS_SyncICCReset(baATR,&bLen);
ShowRtn(usRtn,"Reset");
if (usRtn==d_OK)
{
    usLen=4;
    usRtn=CTOS_44x2ReadSecurityMemory(baBuf);
    ShowRtn(usRtn,"Read SecM");
    if(usRtn==d_OK)
    {
        ShowRsp("ReadSecM",&baBuf[0],usLen);
        usRtn=CTOS_44x2VerifyPSC("\xFF\xFF\xFF");
        ShowRtn(usRtn,"Cmp VerifyData");
        if(usRtn==d_OK)
        {
            usLen=4;
            usRtn=CTOS_44x2ReadSecurityMemory(baBuf);
            ShowRtn(usRtn,"Read SecM");
            if(usRtn==d_OK)
            {
                ShowRsp("ReadSecM",&baBuf[0],usLen);
                usLen=3;
            }
        }
    }
}

```



```

        }

usRtn=CTOS_44x8WritePBitWithDataCompare(0x214,"\x05\x36\x47\x48",0x04);
    ShowRtn(usRtn,"WritePBW/DataCmp");
    if(usRtn==d_OK)
    {
        usLen=8;
        usRtn=CTOS_44x8ReadMemoryWithPBit(0x210,baBuf,baPBuf,&usLen);
        ShowRtn(usRtn,"ReaedDataW/PBit");
        if(usRtn==d_OK)
        {
            ShowRsp ("ReadData",&baBuf[0],usLen);
            ShowRsp ("P_Bit",&baPBuf[0],usLen);
        }
    }
}

CTOS_KBDGet (&Key);
break;
case 6://SLE44x8
CTOS_LCDTPrintXY(1,2,"Insert SLE44x8");
CTOS_KBDGet (&Key);
bLen = sizeof(baATR);
usRtn=CTOS_SyncICCReset(baATR,&bLen);
ShowRtn(usRtn,"Reset");
if (usRtn==d_OK)
{
    ShowRsp ("ATR",&baATR[0],bLen);
    usRtn=CTOS_44x8VerifyPSC ("\xFF\xFF");
    ShowRtn(usRtn,"PSC Verify");
    if(usRtn==d_OK)
    {
        usLen=8;
        usRtn=CTOS_44x8ReadMemoryWithPBit(0x210,baBuf,baPBuf,&usLen);
        ShowRtn(usRtn,"ReaedDataW/PBit");
        if(usRtn==d_OK)
        {
            ShowRsp ("ReadData",&baBuf[0],usLen);
            ShowRsp ("P_Bit",&baPBuf[0],usLen);
        }
        usRtn=CTOS_44x8WriteMemoryWithPBit(0x210,"\xAA\xBB\xCC\xDD",0x04);
        ShowRtn(usRtn,"WriteEraseW/PB");
        if(usRtn==d_OK)
        {
            usLen=8;
            usRtn=CTOS_44x8ReadMemoryWithPBit(0x210,baBuf,baPBuf,&usLen);
            ShowRtn(usRtn,"ReaedDataW/PBit");
            if(usRtn==d_OK)
            {
                ShowRsp ("ReadData",&baBuf[0],usLen);
                ShowRsp ("P_Bit",&baPBuf[0],usLen);
            }
        }
    }
}
CTOS_KBDGet (&Key);
break;
case 7://I2C
CTOS_LCDTPrintXY(1,2,"Insert I2C");
CTOS_KBDGet (&Key);
bLen = sizeof(baATR);
usRtn=CTOS_I2CCReset(baATR,&bLen);
ShowRtn(usRtn,"Reset");
if (usRtn==d_OK)
{
    ShowRsp ("ATR",&baATR[0],bLen);
    bLen=0;
    for(i=0;i<256;i++)
    {
        baBuf[bLen++]=i;
    }
    usLen=256;
    usRtn=CTOS_I2CWriteMemory(0xA0,0x02,0x00,baBuf,usLen);
    ShowRtn(usRtn,"I2CWrite");

    usRtn=CTOS_I2CReadMemory(0xA1,0x02,0x00,1,baBuf,&usLen);
}

```

```

        ShowRtn(usRtn,"I2CRead");
        if(usRtn==d_OK)
        {
            for(i=0;i<256;)
            {
                if((i+48)>256)
                    j=256-i;
                else
                    j=48;
                ShowRsp("ReadData",&baBuf[i],j);
                i+=48;
            }
        }
        CTOS_KBDGet(&Key);
        break;
    case 8://SLE44x6 Reset
        CTOS_LCDTPrintXY(1,2,"Insert SLE44x6");
        CTOS_KBDGet(&Key);
        bLen = sizeof(baATR);
        usRtn=CTOS_SyncICCReset(baATR,&bLen);
        ShowRtn(usRtn,"Reset");
        if (usRtn==d_OK)
        {
            ShowRsp("ATR",&baATR[0],(USHORT)bLen);
        }
        break;
    case 9://SLE44x6 VerifyTSC
        usRtn=CTOS_44x6VerifyTSC("\xF3\xFF\xFF");
        ShowRtn(usRtn,"SLE44X6VerifyTSC");
        break;
    case 10://SLE44x6 Read
        usLen=48;
        usRtn=CTOS_44x6ReadData(0x00,baBuf,&usLen);
        ShowRtn(usRtn,"SLE44X6Read");
        if(usRtn==d_OK)
        {
            ShowRsp("ReadData",&baBuf[0],usLen);
        }
        break;
    case 11://SLE44x6 Write
        //usLen=5;
        usRtn=CTOS_44x6WriteData(0x04,"\x01",1);
        ShowRtn(usRtn,"SLE44X6Write");
        usLen=10;
        usRtn=CTOS_44x6ReadData(0x00,baBuf,&usLen);
        ShowRtn(usRtn,"SLE44X6Read");
        if(usRtn==d_OK)
        {
            ShowRsp("ReadData",&baBuf[0],usLen);
        }
        break;
    case 12://SLE44x6 Reload
        usRtn=CTOS_44x6Reload(1,0x40);
        ShowRtn(usRtn,"SLE44X6Reload");
        break;
    case 13://SLE44x6 Authen
        usRtn=CTOS_44x6Auth(0,0x6E,0x03,"\x01\x02\x03\x04\x05\x06",baBuf);
        ShowRtn(usRtn,"SLE44X6Authen");
        if(usRtn==d_OK)
        {
            ShowRsp("RepCode",&baBuf[0],2);
        }
        break;
    case 14://SLE44x6 Authen
        usRtn=CTOS_44x6Auth(1,0x6F,0x03,"\x01\x02\x03\x04\x05\x06",baBuf);
        ShowRtn(usRtn,"SLE44X6ExtAuthen");
        if(usRtn==d_OK)
        {
            ShowRsp("RepCode",&baBuf[0],2);
        }
        break;
    case 15://SLE44x6 Write
        usRtn=CTOS_44x6WriteData(0x0F,"\x31\x32",2);
        ShowRtn(usRtn,"SLE44X6Write");
        usLen=48;

```

```

        usRtn=CTOS_44x6ReadData(0x00,baBuf,&usLen);
        ShowRtn(usRtn,"SLE44X6Read");
        if(usRtn==d_OK)
        {
            ShowRsp("ReadData",&baBuf[0],usLen);
        }
        break;
    default:
        break;
    }
}
//-----
void ShowMenu(void)
{
    CTOS_LCDTClearDisplay();
    CTOS_LCDTSelectFontSize(d_LCD_FONT_8x8);
    CTOS_LCDTSetReverse(d_TRUE);
    CTOS_LCDTPrintXY(1,1,"Memory Card TEST");
    CTOS_LCDTSetReverse(d_FALSE);
    CTOS_LCDTPrintXY(1,8,"          SLE v001");
}
//=====
int main(int argc, char *argv[])
{
    BYTE      c;
    int       iCase;
    BYTE      strShow[18];
    ULONG     ulAPRtn;
    BYTE      Key;

    CTOS_LCDTClearDisplay();
    CTOS_BackLightSet(d_BKLIT_LCD,d_ON);
    ShowMenu();
    do{
        CTOS_LCDTClearDisplay();
        CTOS_LCDTSetReverse(d_TRUE);
        CTOS_LCDTPrintXY(1,1,"Memory Card TEST");
        CTOS_LCDTSetReverse(d_FALSE);
        CTOS_LCDTPrintXY(1, 3, "Case# 00...99");
        CTOS_LCDTPrintXY(1, 8, "00 Exit");
        CTOS_LCDTPrintXY(1, 5, "Input #: ");
        CTOS_KBDGet(&c);
        CTOS_LCDTPutchXY(10,5,c);
        iCase = (c - '0')*10;
        CTOS_KBDGet(&c);
        CTOS_LCDTPutchXY(11,5,c);
        iCase += (c - '0');

        sprintf(strShow, "Case: %ld      ", iCase);
        CTOS_LCDTPrintXY(1, 5, strShow);

        if (iCase != 0)
            SLESample(iCase);
    }while (iCase != 0);

    return 0;
}
// eof

```

## B.12. Modem

```
/******  
*  
* FILE NAME: Modem  
* MODULE NAME: Modem  
* PROGRAMMER: Peggy Chang  
* DESCRIPTION: Test Modem Functions  
* REVISION: 01.00  
  
*****  
/  
  
/*=====*  
* I N C L U D E S *  
*=====*/  
#include <string.h>  
#include <stdio.h>  
#include <ctosapi.h>  
#include <stdlib.h>  
#include <stdarg.h>  
  
/*=====*  
* D E F I N E S *  
*=====*/  
#define d_BUFF_SIZE 128 //Buffer Size  
#define d_LINE_DOT 12 //A line is 12 dots  
  
//const BYTE baPhoneNO[] = "803"; //Define phone number of call out  
BYTE baPhoneNO[16];  
const BYTE TestData[] = //Define send data  
{  
    0x60,0x09,0x50,0x00,0x00,0x02,0x00,0x30,0x24,0x05,0x80,0x20,0xC0,0x00,0x04,0x00,  
    0x30,0x00,0x00,0x00,0x01,0x68,0x00,0x00,0x59,0x05,0x12,0x59,0x02,0x09,  
    0x50,0x00,0x37,0x49,0x99,0x37,0x00,0x00,0x01,0x31,0x04,0xD0,0x51,0x21,0x01,0x18,  
    0x99,0x15,0x89,0x45,0x67,0x80,0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,  
    0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,0x31,  
    0x30,0x30,0x30,0x35,0x39  
};  
  
//Declare Global Variable //  
USHORT ret;  
BYTE key;  
BYTE babuff[d_BUFF_SIZE];  
BYTE isDialUp, isListen, isSetOnfi;  
DWORD dwStatus;  
  
/* ======  
* FUNCTION NAME: show_screen  
* DESCRIPTION: Show on the LCD Display  
* RETURN: none.  
* NOTES: none.  
* ====== */  
void show_screen(int tag){  
    CTOS_LCDTClearDisplay ();  
    switch(tag){  
        case 0:  
            CTOS_LCDTSetReverse(TRUE);  
            CTOS_LCDTPrintXY(1, 1, " MODEM TEST ");  
            CTOS_LCDTSetReverse(FALSE);  
            if (!isDialUp)  
                CTOS_LCDTPrintXY(1, 2, "1.Dial Up ");  
            else  
                CTOS_LCDTPrintXY(1, 2, "1.Hang Up ");  
  
            if (!isListen)  
                CTOS_LCDTPrintXY(1, 3, "2.Listen (OFF)");  
            else  
                CTOS_LCDTPrintXY(1, 3, "2.Listen (ON) ");  
  
            CTOS_LCDTPrintXY(1, 4, "3.TxData");  
    }  
}
```

```

        CTOS_LCDTPrintXY(1, 5, "4.RxData");
        CTOS_LCDTPrintXY(1, 6, "5.Set PhoneNO");
        CTOS_LCDTPrintXY(1, 7, "6.Set Config");
        break;
    case 1:
        CTOS_LCDTPrintXY(1, 1, "CONFIG      ");
        CTOS_LCDTPrintXY(1, 2, "1.Set");
        CTOS_LCDTPrintXY(1, 3, "2.Get");
        CTOS_LCDTPrintXY(1, 4, "3.SetDialPrecheck");
        CTOS_LCDTPrintXY(1, 5, "4.Hook OFF");
        CTOS_LCDTPrintXY(1, 6, "5.ATCommand");
        break;
    }
}

/* =====
 * FUNCTION NAME: InputString
 * DESCRIPTION: Continue to input two or more characters at a time, build a string
 * RETURN: none.
 * NOTES: none.
 * ===== */
int InputString(USHORT usX, USHORT usY, BYTE isMask, BYTE *pbaStr, USHORT *usStrLen) {
    //Declare Local Variable //
    int i = 0;

    memset(pbaStr, 0x00, *usStrLen);
    CTOS_LCDTPutchXY(usX, usY, '_');

    while(1) {

        //Scan the keyboard to detect whether a key is pressed //
        CTOS_KBDHit(&key);

        //Cancel InputString() //
        if (key == d_KBD_CANCEL){
            CTOS_LCDTPutchXY(usX+i, usY, ' ');
            memset(pbaStr, 0x00, *usStrLen);
            usStrLen = 0;
            return 0;
        }

        //Done input and output a string //
        if (key == d_KBD_ENTER){
            CTOS_LCDTPutchXY(usX+i, usY, ' ');
            *usStrLen = strlen(pbaStr);
            if (*usStrLen > 0) return 1;
            else return 0;
        }else if (key == d_KBD_CLEAR){ //Backspace //
            CTOS_LCDTPutchXY(usX+i, usY, ' ');
            i--;
            pbaStr[i] = 0x00;
            CTOS_LCDTPutchXY(usX+i, usY, '_');
        }else if(key == d_KBD_00){ //Clear all input data //
            memset(pbaStr, 0x00, *usStrLen);
            i = 0;
            CTOS_LCDTGotoXY(usX+i, usY);
            CTOS_LCDTClear2EOL();
            CTOS_LCDTPutchXY(usX, usY, '_');
        }
        if (i+1 <= *usStrLen){
            if ((key >= 0x30)&& (key <=0x39)){
                pbaStr[i] = key;

                if (isMask) CTOS_LCDTPutchXY(usX+i, usY, '*');
                else CTOS_LCDTPutchXY(usX+i, usY, pbaStr[i]);
                i++;
                CTOS_LCDTPutchXY(usX+i, usY, '_');
            }
        }
    }

    /* =====
     * FUNCTION NAME: Set Phone Number
     * DESCRIPTION:
     * RETURN: none.
     * NOTES: none.
     */
}

```

```

/* ===== */
void SetPhoneNumber(void) {
    USHORT len;
    CTOS_LCDTClearDisplay ();
    CTOS_LCDTPrintXY(1, 1, "\fr SET PHONE NO. \fn");
    CTOS_LCDTPrintXY(1, 2, "Input Phone No.:");

    len = sizeof(baPhoneNO);

    if (InputString(1, 3, FALSE, baPhoneNO,&len) == 1){
        CTOS_LCDTPrintXY(1, 8, "Set OK");
    }else{
        CTOS_LCDTPrintXY(1, 8, "Set Fail");
    }

    CTOS_KBDGet(&key);

    show_screen(0);
}

/* ===== */
/* FUNCTION NAME: ModemDialUp
 * DESCRIPTION: Dial up to connect the host or hang up
 * RETURN: none.
 * NOTES: none.
 * ===== */
void ModemDialUp(BYTE isUP) {
    if (isUP){

        //Dial up to connect the host //
        ret = CTOS_ModemDialup((BYTE *)baPhoneNO,strlen(baPhoneNO));

        CTOS_LCDTPrintXY(1, 7, (BYTE *)baPhoneNO);
        CTOS_LCDTPrintXY(1, 8, "DialUp...");

        //Check whether connect success //
        while(1){
            //Cancel connect the host
            CTOS_KBDHit(&key);
            if (key == d_KBD_CANCEL){

                //Hang up
                CTOS_ModemHookOn();
                CTOS_LCDTPrintXY(11, 8, "Cancal");
                return;
            }

            //Get the status of modem //
            CTOS_ModemStatus(&dwStatus);
            sprintf(babuff, "%08X", dwStatus);
            CTOS_LCDTPrintXY(1, 8, babuff);

            if (!(dwStatus & d_M_STATUS_DIALING)){
                //Connect success to the host //
                if (dwStatus & d_M_STATUS_MODEM_ONLINE){
                    CTOS_LCDTPrintXY(11, 8, "OK");
                    CTOS_LCDTPrintXY(1, 2, "1.Hang Up      ");
                    isDialUp = TRUE;
                }else{ //Connect fail to the host //
                    CTOS_LCDTPrintXY(11, 8, "Fail");
                    isDialUp = FALSE;
                }
                return;
            }
        }
    }
}
//Hook on the modem (Hang up) //
if (ret == d_OK){
    CTOS_LCDTPrintXY(1, 2, "1.Dial Up      ");
    CTOS_LCDTPrintXY(1, 8, "Hang UP      ");
    isDialUp = FALSE;
}
}

/* ===== */

```

```

* FUNCTION NAME: ModemListen
* DESCRIPTION: Listen & wait to host connecting
* RETURN: none.
* NOTES: none.
* ===== */
void ModemListen(BYTE isON){

    //Declare Local Variable //
    BYTE baATCmd[d_BUFF_SIZE],baRep[d_BUFF_SIZE];
    USHORT usATCmdLen,usRepLen;

    //Listen Start //
    if (isON){
        CTOS_LCDTPrintXY(1, 8, "Listen...");

        //Wait to host connecting //
        ret = CTOS_ModemListen(1);
        while(1){
            CTOS_KBDHit(&key);
            if (key == d_KBD_CANCEL){
                CTOS_LCDTPrintXY(11, 8, "Cancel");
                return;
            }
            CTOS_ModemStatus(&dwStatus);
            if (dwStatus & d_M_STATUS_LISTENING){
                CTOS_LCDTPrintXY(11, 8, "OK");
                CTOS_LCDTPrintXY(1, 3, "2.Listen (ON) ");
                isListen = TRUE;
                return;
            }else{
                CTOS_LCDTPrintXY(11, 8, "Fail");
                isListen = FALSE;
                return;
            }
        }
    }else{ //Listen Stop //

        CTOS_ModemClose();
        CTOS_Delay(500);
        CTOS_ModemOpen(d_M_MODE_AYNC_NORMAL,d_M_HANDSHAKE_V22_ONLY, d_M_COUNTRY_TAIWAN);
        CTOS_LCDTPrintXY(1, 3, "2.Listen (OFF)");
        CTOS_LCDTPrintXY(1, 8, "Listen Stop ");
        isListen = FALSE;
    }
}

/* =====
* FUNCTION NAME: ModemTxData
* DESCRIPTION: Send data to the host via the modem channel
* RETURN: none.
* NOTES: none.
* ===== */
void ModemTxData(void){
    //Declare Local Variable //
    DWORD dwStatus;
    int i,iLength = 0;

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, "----Tx Data----");

    CTOS_LCDTPrintXY(1, 2, "Send Data....");
    CTOS_LCDTPrintXY(1, 3, "Length=");

    //Check modem whether ready to transmit data
    if (CTOS_ModemTxReady()== d_OK){

        //Transmit data via the modem channel //
        CTOS_ModemTxData((BYTE *)TestData,sizeof(TestData));
        iLength = sizeof(TestData);
        sprintf(&babuff[100],"%dbyte",iLength);
        CTOS_LCDTPrintXY(8, 3, &babuff[100]);
        CTOS_Delay(1000);
    }else{
        CTOS_LCDTPrintXY(1, 7, "Tx Data Fail");
    }
}

```

```

//Go back to main() //
CTOS_LCDTPrintXY(1, 8, "Exit->'X'");
CTOS_KBDGet(&key);
if (key == d_KBD_CANCEL) {
    show_screen(0);
}
}

/* =====
* FUNCTION NAME: ModemRxData
* DESCRIPTION:   Receive data from the host via modem channel
* RETURN:        none.
* NOTES:        none.
* ===== */
void ModemRxData(void){
    //Declare Local Variable //
    USHORT usr_Len = 0, usR_Len = 0;

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, "----Rx Data----");
    CTOS_LCDTPrintXY(1, 8, "Exit->'X'");
    memset(babuff, 0x00, sizeof(babuff));
    usr_Len = sizeof(babuff);
    CTOS_LCDTPrintXY(1, 2, "Receive Data....");
    CTOS_LCDTPrintXY(1, 3, "Length=");

    while(1){
        //Check if the data is currently available in Modem //
        ret = CTOS_ModemRxReady(&usr_Len);
        if (ret == d_OK){
            //Receive data via the modem channel //
            CTOS_ModemRxData(&babuff[usR_Len],&usr_Len);
            usR_Len += usr_Len;
            sprintf(&babuff[100],"%dbytes",usR_Len);
            CTOS_LCDTPrintXY(8, 3, &babuff[100]);

            CTOS_Delay(1000);
        }else{
            CTOS_LCDTPrintXY(1, 8, "RxReady Failed");
        }

        CTOS_KBDHit(&key);

        //Go back to main() //
        if (key == d_KBD_CANCEL){
            //Clear the receive buffer of Modem //
            CTOS_ModemFlushRxData();
            show_screen(0);
            return;
        }
    }
}

/* =====
* FUNCTION NAME: GetModemStatus
* DESCRIPTION:   Get Modem current status
* RETURN:        none.
* NOTES:        none.
* ===== */
void GetModemStatus(void){

    //Declare Local Variable //
    DWORD dwStatus;
    int i = 2;

    CTOS_LCDTClearDisplay ();
    CTOS_LCDTPrintXY(1, 1, " Modem Status ");

    //Get the status of the modem //
    CTOS_ModemStatus(&dwStatus);
    if (dwStatus & d_M_STATUS_MODEM_ONLINE){
        CTOS_LCDTPrintXY(1, i, "Modem Online");
        i++;
    }
    if (dwStatus & d_M_STATUS_SDLC_MODE){
        CTOS_LCDTPrintXY(1, i, "SDLC Mode");
    }
}

```

```

        i++;
    }
    if (dwStatus & d_M_STATUS_SDLC_ONLINE){
        CTOS_LCDTPrintXY(1, i, "SDLC Online");
        i++;
    }
    if (dwStatus & d_M_STATUS_DIALING){
        CTOS_LCDTPrintXY(1, i, "Dialing");
        i++;
    }
    if (dwStatus & d_M_STATUS_NO_DIAL_TONE){
        CTOS_LCDTPrintXY(1, i, "No Dial Tone");
        i++;
    }
    if (dwStatus & d_M_STATUS_LINE_BUSY){
        CTOS_LCDTPrintXY(1, i, "Line Busy");
        i++;
    }
    if (dwStatus & d_M_STATUS_RING_BACK){
        CTOS_LCDTPrintXY(1, i, "Ring Back");
        i++;
    }
    if (dwStatus & d_M_STATUS_TX_BUSY){
        CTOS_LCDTPrintXY(1, i, "Tx Busy");
        i++;
    }
    if (dwStatus & d_M_STATUS_REMOTE_NOT_ANSWER){
        CTOS_LCDTPrintXY(1, i, "Remote Not Answer");
        i++;
    }
    if (dwStatus & d_M_STATUS_NO_CARRIER){
        CTOS_LCDTPrintXY(1, i, "NO Carrier");
        i++;
    }
    if (dwStatus & d_M_STATUS_ALL_DATA_SENT){
        CTOS_LCDTPrintXY(1, i, "All Data Sent");
        i++;
    }
    if (dwStatus & d_M_STATUS_RX_DATA_VALID){
        CTOS_LCDTPrintXY(1, i, "Rx Data Valid");
        i++;
    }
    if (dwStatus & d_M_STATUS_LISTENING){
        CTOS_LCDTPrintXY(1, i, "Listening");
        i++;
    }
    if (dwStatus & d_M_STATUS_OTHER_ERROR){
        CTOS_LCDTPrintXY(1, i, "Other Error");
        i++;
    }
    if (dwStatus & d_M_STATUS_DATA_SENT_ERROR){
        CTOS_LCDTPrintXY(1, i, "Data Sent Error");
        i++;
    }

    CTOS_KBDGet(&key);
    show_screen(0);
}

/* =====
 * FUNCTION NAME: ModemConfig
 * DESCRIPTION: Set or get modem configuration
 * RETURN: none.
 * NOTES: none.
 * ===== */
void ModemConfig(BYTE bConfigMode){ //bConfigMode = '1'->Set, '2'->Get

    //Declare Local Variable //
    USHORT usValue;

    if(bConfigMode == '1'){ //Set the modem configuration //

        //After set the config, print the setting status //
        if (CTOS_ModemSetConfig(d_M_CONFIG_DAILING_DURATION,30) == d_OK)
            CTOS_PrinterPutString("\nDailing Diration OK");
    else

```

```

        CTOS_PrinterPutString("\nDailing Diration Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_MIN_ONHOOK_DURATION,3) == d_OK)
            CTOS_PrinterPutString("\nMin OnHook Duration OK");
        else
            CTOS_PrinterPutString("\nMin OnHook Duration Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_PREDIAL_DELAY_TIME,0) == d_OK)
            CTOS_PrinterPutString("\nPreDial Delay Tiem OK");
        else
            CTOS_PrinterPutString("\nPreDial Delay Tiem Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_DIAL_TONE_DETECT_DURATION,7000) == d_OK)
            CTOS_PrinterPutString("\nDial Tone Detect Duration OK");
        else
            CTOS_PrinterPutString("\nDial Tone Detect Duration Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_DIAL_TONE_MIN_ON_TIME,14400) == d_OK)
            CTOS_PrinterPutString("\nDial Tone Min On Time OK");
        else
            CTOS_PrinterPutString("\nDial Tone Min On Time Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_DTMF_ON_TIME,100) == d_OK)
            CTOS_PrinterPutString("\nDTMF On Time OK");
        else
            CTOS_PrinterPutString("\nDTMF On Time Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_DTMF_OFF_TIME,100) == d_OK)
            CTOS_PrinterPutString("\nDTMF Off Time OK");
        else
            CTOS_PrinterPutString("\nDTMF Off Time Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_BUSY_TONE_MIN_TOTAL_TIME,1728) == d_OK)
            CTOS_PrinterPutString("\nBusy Tone Min Total Time OK");
        else
            CTOS_PrinterPutString("\nBusy Tone Min Total Time Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_BUSY_TONE_DELTA_TIME,7632) == d_OK)
            CTOS_PrinterPutString("\nBusy Tone Delta Time OK");
        else
            CTOS_PrinterPutString("\nBusy Tone Delta Time Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_BUSY_TONE_MIN_ON_TIME,864) == d_OK)
            CTOS_PrinterPutString("\nBusy Tone Min On Time OK");
        else
            CTOS_PrinterPutString("\nBusy Tone Min On Time Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_RINGBACK_TONE_MIN_TOTAL_TIME,18000) == d_OK)
            CTOS_PrinterPutString("\nRingBack Tone Min Total OK");
        else
            CTOS_PrinterPutString("\nRingBack Tone Min Total Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_RINGBACK_TONE_DELTA_TIME,61200) == d_OK)
            CTOS_PrinterPutString("\nRingBack Tone Delta Time OK");
        else
            CTOS_PrinterPutString("\nRingBack Tone Delta Time Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_RINGBACK_TONE_MIN_ON_TIME,4608) == d_OK)
            CTOS_PrinterPutString("\nRingBack Tone Delta Time OK");
        else
            CTOS_PrinterPutString("\nRingBack Tone Delta Time Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_ANSWER_TONE_WAIT_DURATION,50) == d_OK)
            CTOS_PrinterPutString("\nAnswer Tone Wait Duration OK");
        else
            CTOS_PrinterPutString("\nAnswer Tone Wait Duration Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_BLIND_DIAL_DELAY_TIME,2) == d_OK)
            CTOS_PrinterPutString("\nBlind Dial Delay Time OK");
        else
            CTOS_PrinterPutString("\nBlind Dial Delay Time Fail");

        if (CTOS_ModemSetConfig(d_M_CONFIG_CARRIER_PRESENT_TIME,6) == d_OK)
            CTOS_PrinterPutString("\nCarrier Present Time OK");
        else
            CTOS_PrinterPutString("\nCarrier Present Time Fail");

```

```

        }else{ //Read the modem configuration //

        //Print the configuration //
        CTOS_ModemReadConfig(d_M_CONFIG_DAILING_DURATION,&usValue);
        sprintf(babuff,"\\nDailing Duration = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_MIN_ONHOOK_DURATION,&usValue);
        sprintf(babuff,"\\nMin OnHook Duration = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_PREDIAL_DELAY_TIME,&usValue);
        sprintf(babuff,"\\nPreDial Delay Tiem = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_DIAL_TONE_DETECT_DURATION,&usValue);
        sprintf(babuff,"\\nDial Tone Detect Duration = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_DIAL_TONE_MIN_ON_TIME,&usValue);
        sprintf(babuff,"\\nDial Tone Min On Time = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_DTMF_ON_TIME,&usValue);
        sprintf(babuff,"\\nDTMF On Time = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_DTMF_OFF_TIME,&usValue);
        sprintf(babuff,"\\nDTMF Off Time = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_BUSY_TONE_MIN_TOTAL_TIME,&usValue);
        sprintf(babuff,"\\nBusy Tone Min Total Time = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_BUSY_TONE_DELTA_TIME,&usValue);
        sprintf(babuff,"\\nBusy Tone Delta Time = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_BUSY_TONE_MIN_ON_TIME,&usValue);
        sprintf(babuff,"\\nBusy Tone Min On Time = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_RINGBACK_TONE_MIN_TOTAL_TIME,&usValue);
        sprintf(babuff,"\\nRingBack Tone Min Total Time = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_RINGBACK_TONE_DELTA_TIME,&usValue);
        sprintf(babuff,"\\nRingBack Tone Delta Time = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_RINGBACK_TONE_MIN_ON_TIME,&usValue);
        sprintf(babuff,"\\nRing Back Tone Min On Time = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_ANSWER_TONE_WAIT_DURATION,&usValue);
        sprintf(babuff,"\\nAnswer Tone Wait Duration = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_BLIND_DIAL_DELAY_TIME,&usValue);
        sprintf(babuff,"\\nBlind Dial Delay Time = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
        CTOS_ModemReadConfig(d_M_CONFIG_CARRIER_PRESENT_TIME,&usValue);
        sprintf(babuff,"\\nCarrier Present Time = \\n%d", usValue);
        CTOS_PrinterPutString(babuff);
    }
    CTOS_PrinterLine(d_LINE_DOT*10);
    show_screen(0);
}

/* =====
 * FUNCTION NAME: SetDialPrecheck
 * DESCRIPTION: Set precheck parameters
 * RETURN: none.
 * NOTES: none.
 * ===== */
void SetDialPrecheck(void){

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, " Dial Precheck ");
    CTOS_LCDTPrintXY(1, 2, "1.No Dial Tone or Busy"); //No detect dial tone or busy //
    CTOS_LCDTPrintXY(1, 3, "2.Only Dial Tone"); //Only detect dial tone //
    CTOS_LCDTPrintXY(1, 4, "3.Only Busy"); //Only detect busy //
    CTOS_LCDTPrintXY(1, 5, "4.Dial Tone and Busy"); //Detect dial tone and busy
    (default) //
    CTOS_LCDTPrintXY(1, 6, "5.ALL"); //Detect dial tone, busy, and ringback //

    while(1){
        CTOS_KBDGet ( &key );
}

```

```

switch(key){
    case d_KBD_1:
    case d_KBD_2:
    case d_KBD_3:
    case d_KBD_4:
    case d_KBD_5:
        //Set precheck parameters for dialing up //
        //The hex value of key is 0x31, the mode of the dial precheck start 0x00 to 0x04,
        so (the hex value - 0x31 = dial precheck)
        ret = CTOS_ModemSetDialPrecheck(key - 0x31);
        if (ret == d_OK)
            CTOS_LCDTPrintXY(1, 8, "Set.....OK");
        else
            CTOS_LCDTPrintXY(1, 8, "Set.....Fail");
        break;
    case d_KBD_CANCEL:
        //Go back to main() //
        show_screen(0);
        return;
        break;
    }
}
}

/* =====
 * FUNCTION NAME: main
 * DESCRIPTION: function main is responsible for all the tasks listed in the
 *               introduction to this demo.
 * RETURN:      none.
 * NOTES:       none.
 * ===== */
int main(int argc,char *argv[])
{
BYTE baAt_cmd[d_BUFF_SIZE];
USHORT usReqLen;
BYTE key;

// TODO: Add your program here //
//Initial //
isDialUp = FALSE;
isListen = FALSE;
memset(baAt_cmd,0x00,sizeof(baAt_cmd));
memset(baPhoneNO,0,sizeof(baPhoneNO));

//Open Modem channel //
CTOS_LCDTPrintXY(1, 1, " OPEN Modem ");
ret = CTOS_ModemOpen(d_M_MODE_AYNC_NORMAL,d_M_HANDSHAKE_V22_ONLY, d_M_COUNTRY_TAIWAN);
if (ret != d_OK){
    CTOS_LCDTPrintXY(1, 2, " OPEN Fail ");
    CTOS_KBDGet(&key);
    return 0;
}
CTOS_LCDTPrintXY(1, 1, " OPEN OK ");
show_screen(0);

while(1){
    CTOS_KBDGet ( &key );
    switch(key)
    {
        case d_KBD_1:
            //Dial up or hang up via modem channel //
            ModemDialUp(!isDialUp);
            break;
        case d_KBD_2:
            //Start or Stop Listen //
            ModemListen(!isListen);
            break;
        case d_KBD_3:
            ModemTxData();
            break;
        case d_KBD_4:
            ModemRxData();
            break;
        case d_KBD_5:
            //GetModemStatus();
            SetPhoneNumber();
    }
}
}

```

```

        break;
    case d_KBD_6:
        show_screen(1);
        CTOS_KBDGet ( &key );
        switch(key)
        {
            case d_KBD_1:
            case d_KBD_2:
                ModemConfig(key);
                break;
            case d_KBD_3:
                SetDialPrecheck();
                break;
            case d_KBD_4:
                ret = CTOS_ModemHookOff();
                sprintf(babuff,"ret = 0x%04X",ret);
                CTOS_LCDTPrintXY(1, 8, babuff);
                break;
            case d_KBD_5:
                // AT Command: Disable V.42 bis and MNP5 data compression //
                // Wait for dial tone delay = 20 seconds //
                memcpy(baAt_cmd,"AT%C0\rATS14=20\r",strlen("AT%C0\rATS14=20\r"));
                CTOS_LCDTPrintXY(1, 7, baAt_cmd);
                CTOS_ModemATCommand(baAt_cmd,strlen(baAt_cmd),babuff,&usReqLen);
                CTOS_LCDTPrintXY(1, 8, babuff);
                break;
            case d_KBD_CANCEL:
                show_screen(0);
                break;
        }
        break;
    case d_KBD_CANCEL:
        CTOS_ModemClose();
        return 0;
        break;
    }
}

// eof

```

## B.13. MSReader

```
/*
 * FILE NAME: MSReader
 * MODULE NAME: MSReader
 * PROGRAMMER: Peggy Chang
 * DESCRIPTION: Read magnetic stripe card data.
 * REVISION: 01.00
 */

=====
 *           I N C L U D E S           *
=====

#include <string.h>
#include <stdio.h>
#include <ctosapi.h>
#include <stdlib.h>
#include <stdarg.h>

=====
 *           D E F I N E S           *
=====

#define d_BUFF_SIZE 128

=====
 * FUNCTION NAME: main
 * DESCRIPTION: function main is responsible for all the tasks listed in the
introduction to this demo.
 * RETURN: none.
 * NOTES: none.
=====
int main(int argc, char *argv[])
{
    //Declare Local Variable //
    USHORT rtn;
    BYTE key;
    USHORT usTk1Len, usTk2Len, usTk3Len; //Track 1,2,3 length //
    //Track 1,2,3 data buffer //
    BYTE baTk1Buf[d_BUFF_SIZE], baTk2Buf[d_BUFF_SIZE], baTk3Buf[d_BUFF_SIZE];
    BYTE baTk1Err, baTk2Err, baTk3Err; //Track 1,2,3 individual status //
    BYTE baBuff[d_BUFF_SIZE];

    //sysheap_setup ( 256 * 1024 ); // initial newlib and create 256K heap

    // TODO: Add your program here //
    CTOS_BackLightSetEx(d_BKLIT_LCD, d_ON, 1000);

    CTOS_LCDTClearDisplay();
    CTOS_LCDTSetReverse(TRUE);
    CTOS_LCDTPrintXY(1, 1, "----MSR Test----");
    CTOS_LCDTSetReverse(FALSE);
    CTOS_LCDTPrintXY(1, 3, "Pls Swipe MSR...");

    usTk1Len = usTk2Len = usTk3Len = d_BUFF_SIZE;

    do{
        //Scan the deyboard to detect whether a key is pressed //
        CTOS_KBDHit(&key);
        if (key == d_KBD_CANCEL) return 0;

        //Read magnetic stripe card data //
        rtn = CTOS_MSRRRead(baTk1Buf, &usTk1Len, baTk2Buf, &usTk2Len, baTk3Buf, &usTk3Len);

    }while(rtn != d_OK || (usTk1Len == 0 && usTk2Len == 0 && usTk3Len == 0));

    memset(baBuff,0x00,sizeof(baBuff));

    sprintf(baBuff,"Track 1 Len = %d", usTk1Len);
    CTOS_LCDTPrintXY(1, 2, baBuff); //Display the track 1 data length in the 2 line //

    sprintf(baBuff,"Track 2 Len = %d", usTk2Len);
```

```

CTOS_LCDTPrintXY(1, 3, baBuff); //Display the track 2 data length in the 3 line //

sprintf(baBuff,"Track 3 Len = %d", usTk3Len);
CTOS_LCDTPrintXY(1, 4, baBuff); //Display the track 3 data length in the 4 line //

//Get the individual status of each track of last swipe of magnetic card //
rtn = CTOS_MSRRGetLastErr(&baTk1Err,&baTk2Err,&baTk3Err);

//Display the track 1 individual status in the 5 line //
sprintf(baBuff,"Track1Err %02X", baTk1Err);
CTOS_LCDTPrintXY(1, 5, baBuff);

//Display the track 2 individual status in the 6 line //
sprintf(baBuff,"Track2Err %02X", baTk2Err);
CTOS_LCDTPrintXY(1, 6, baBuff);

//Display the track 3 individual status in the 7 line //
sprintf(baBuff,"Track3Err %02X", baTk3Err);
CTOS_LCDTPrintXY(1, 7, baBuff);

CTOS_LCDTPrintXY(1, 8, "Press Any Key...");

CTOS_KBDGet(&key);

CTOS_BackLightSet(d_BKLIT_LCD, d_OFF);
return 0;
}

// eof

```

## B.14. Printer





```

for (i = 0; i < 6; i++)           // Height / 8
{
    for (j = 0; j < 60; j++)   // Width
    {
        baTemp[a + j] = baPrinterBufferLogo_Single[b + j];
    }

    a += 384;
    b += 361;
}

// Put a string into the printer buffer
CTOS_PrinterBufferPutString((BYTE *)baTemp, 61, 13, "CASTLES Technology",
&stFont_Attrib);

// The printer will print out an image which inside the Printer Buffer.
//Print the "CASTLES" Graphic & Text Logo
CTOS_PrinterBufferOutput((BYTE *)baTemp, 6);

CTOS_PrinterFline(4);

// Get the current status of printer
usRtn = CTOS_PrinterStatus();
switch(usRtn)
{
case d_OK:
    CTOS_PrinterPutString("d_OK");
    break;
case d_PRINTER_HEAD_OVERHEAT:
    CTOS_PrinterPutString("d_PRINTER_HEAD_OVERHEAT");
    break;
case d_PRINTER_PAPER_OUT:
    CTOS_PrinterPutString("d_PRINTER_PAPER_OUT");
    break;
case d_PRINTER_BARCODE_GENERATE_ERR:
    CTOS_PrinterPutString("d_PRINTER_BARCODE_GENERATE_ERR");
    break;
case d_PRINTER_BARCODE_CONTENT_ERR:
    CTOS_PrinterPutString("d_PRINTER_BARCODE_CONTENT_ERR");
    break;
case d_PRINTER_BARCODE_CONTENT_LEN_ERR:
    CTOS_PrinterPutString("d_PRINTER_BARCODE_CONTENT_LEN_ERR");
    break;
case d_PRINTER_BARCODE_OUTSIDE_PAPER:
    CTOS_PrinterPutString("d_PRINTER_BARCODE_OUTSIDE_PAPER");
    break;
}

return 0;
}

// eof

```

## B.15. RS232

```
/*
 * FILE NAME: RS232
 * MODULE NAME: RS232 Communication
 * PROGRAMMER:
 * DESCRIPTION: Test Rs232 communication functions
 * REVISION: 01.00
 */

=====
 *      I N C L U D E S      *
=====

#include <string.h>
#include <stdio.h>
#include <ctosapi.h>

=====
 *      D E F I N E S      *
=====

#define d_BUFF_SIZE    128      //Buff size
#define d_LCD_Width    16+1     //LCD width

=====
 * =====
 * FUNCTION NAME: main
 * DESCRIPTION: function main is responsible for all the tasks listed in the
introduction to this demo.
 * RETURN: none.
 * NOTES: none.
 * =====
int main()
{
    //Declare Local Variable //
    USHORT ret;
    USHORT len;
    BYTE key;
    BYTE str[d_LCD_Width];
    BYTE buf[d_BUFF_SIZE];

    CTOS_LCDTPrintXY(1, 1, "----RS232 ECHO---");

    // Open COM1 with baudrate 115200 //
    ret = CTOS_RS232Open(d_COM1, 115200, 'N', 8, 1);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY(1, 7, "Open Not OK");
        CTOS_LCDTPrintXY(1, 8, "Press any key");
        CTOS_KBDGet(&key);
        return 1;
    }
    CTOS_LCDTPrintXY(1, 2, "Baudrate: 115200");
    //The RTS control signal from this system which indicates this system is ready to
receive data //
    ret = CTOS_RS232SetRTS(d_COM1, d_ON);
    if (ret != d_OK)
    {
        CTOS_LCDTPrintXY(1, 4, "V7 Rec Err");
        CTOS_KBDGet(&key);
        return 1;
    }
    else
        CTOS_LCDTPrintXY(1, 4, "V7 Rec OK");

    //The CTS status of remote host/device for the specified port //
    ret = CTOS_RS232GetCTS(d_COM1);
    if (ret != d_OK)
    {
        CTOS_LCDTPrintXY(1, 5, "PC Rec Err");
        CTOS_KBDGet(&key);
        return 1;
    }
}
```

```

    }
    else
        CTOS_LCDTPrintXY(1, 5, "PC Rec OK");

    CTOS_Delay(1000);
    CTOS_LCDTClearDisplay();

    while(1)
    {
        // Check if Cancel key is pressed //
        CTOS_KBDHit(&key);
        if(key == d_KBD_CANCEL)
        {
            break;
        }
        CTOS_LCDTPrintXY(1, 3, "Start");

        // Check if data is available in COM1 port //
        CTOS_RS232RxReady(d_COM1, &len);
        if(len)
        {
            // Get Data from COM1 port
            CTOS_RS232RxData(d_COM1, buf, &len);

            // Display
            sprintf(str, "Got = %c%c%c", buf[0], buf[1], buf[2]);
            CTOS_LCDTPrintXY(1, 4, "          ");
            CTOS_LCDTPrintXY(1, 4, str);

            // Check if COM1 is ready to send data
            while(CTOS_RS232TxReady(d_COM1) != d_OK);

            // Send data via COM1 port
            if(CTOS_RS232TxData(d_COM1, buf, len-2) != d_OK)
            {
                CTOS_LCDTPrintXY(1, 7, "TxData Not OK");
                CTOS_LCDTPrintXY(1, 8, "Press any key");
                CTOS_KBDGet(&key);
                //Flushing the RS232 receive buffer //
                CTOS_RS232FlushRxBuffer(d_COM1);
                //Close RS232 comport //
                CTOS_RS232Close(d_COM1);
                return 1;
            }
        }
    }
    //Close RS232 comport //
    CTOS_RS232Close(d_COM1);
    return 0;
}

// eof

```

## B.16. SmartCard

```
/*
 * FILE NAME:     SmartCard
 * MODULE NAME:   Smart Card
 * PROGRAMMER:    Peggy Chang
 * DESCRIPTION:   Test smart card function
 * REVISION:      01.00
 */

=====
 *           I N C L U D E S           *
=====

#include <string.h>
#include <stdio.h>
#include <ctosapi.h>
#include <stdlib.h>
#include <stdarg.h>

=====
 *           D E F I N E S           *
=====

#define d_BUFF_SIZE    128    //Buffer Size

=====
 * FUNCTION NAME: main
 * DESCRIPTION:   function main is responsible for all the tasks listed in the
introduction to this demo.
 * RETURN:        none.
 * NOTES:         none.
=====

int main(int argc, char *argv[])
{
    //Declare Local Variable //
    BYTE key;
    BYTE bStatus;
    BYTE baATR[d_BUFF_SIZE], bATRLen, CardType;
    BYTE baSAPDU[d_BUFF_SIZE],baRAPDU[d_BUFF_SIZE];
    USHORT bSLen,bRLen;

    // TODO: Add your program here //
    CTOS_BackLightSet(d_BKLIT_LCD,d_ON);
    CTOS_LCDTClearDisplay ();

    CTOS_LCDTSetReverse(TRUE);
    CTOS_LCDTPrintXY(1,1,"Smart Card");
    CTOS_LCDTSetReverse(FALSE);

    CTOS_LCDTPrintXY(1,2,"Plz Ins Card");

    do{
        CTOS_KBDHit(&key);
        if (key == d_KBD_CANCEL) return;
        //Check the ICC status //
        CTOS_SCStatus(d_SC_USER,&bStatus);
    }while(!(bStatus & d_MK_SC_PRESENT)); //Break until the ICC Card is inserted //

    bATRLen = sizeof(baATR);
    CTOS_LCDTPrintXY(1,3,"Reset EMV..");
    //Power on the ICC and retrun the ATR contents metting the EMV2000 specification //
    if (CTOS_SCResetEMV(d_SC_USER, d_SC_5V, baATR, &bATRLen, &CardType) == d_OK)
        CTOS_LCDTPrintXY(13,3,"OK");
    else
        CTOS_LCDTPrintXY(13,3,"Fail");

    CTOS_Delay(1000);
    bATRLen = sizeof(baATR);
    CTOS_LCDTPrintXY(1,4,"Reset ISO..");
    //Power on the ICC and retrun the ATR content metting the ISO-7816 specification //
    if (CTOS_SCResetISO(d_SC_USER, d_SC_5V, baATR, &bATRLen, &CardType) == d_OK)
        CTOS_LCDTPrintXY(13,4,"OK");
```

```

else
    CTOS_LCDTPrintXY(13,4,"Fail");

    CTOS_LCDTPrintXY(1,5,"Send APDU..");

//APDU Data
baSAPDU[0]=0x00; //CLA
baSAPDU[1]=0xB2; //INS
baSAPDU[2]=0x01; //P1
baSAPDU[3]=0x0C; //P2
baSAPDU[4]=0x00; //Le
bSLen = 5;
bRLen = sizeof(baRAPDU);
//Send out an APDU command and get the response from ICC //
if (CTOS_SCSendAPDU(d_SC_USER, baSAPDU, bSLen, baRAPDU, &bRLen) == d_OK)
    CTOS_LCDTPrintXY(13,5,"OK");
else
    CTOS_LCDTPrintXY(13,5,"Fail");

//Turn off the power of ICC //
if (CTOS_SCPowerOff(d_SC_USER) == d_OK)
    CTOS_LCDTPrintXY(1,6,"Power off OK");
else
    CTOS_LCDTPrintXY(1,6,"Power off Fail");

CTOS_LCDTPrintXY(1,7,"Pls Take ICC Out");
do{
    CTOS_SCStatus(d_SC_USER,&bStatus);
}while(bStatus & d_MK_SC_PRESENT); //Break until the ICC Card does not exist //

CTOS_BackLightSet(d_BKLIT_LCD,d_OFF);

return 0;
}

// eof

```

## B.17. System Mode

```
/*
 * FILE NAME: SystemMode
 * MODULE NAME: System
 * PROGRAMMER: Peggy Chang
 * DESCRIPTION: Test System Functions
 * REVISION: 01.00
 */

=====
 *      I N C L U D E S      *
=====

#include <string.h>
#include <stdio.h>
#include <ctosapi.h>
#include <stdlib.h>
#include <stdarg.h>

=====
 *      D E F I N E S      *
=====

#define d_BUFF_SIZE 32      //Buffer Size

//Declare Global Variable //
BYTE key;

=====
 *      FUNCTION NAME: show_screen
 *      DESCRIPTION: Show on the LCD Display
 *      RETURN: none.
 *      NOTES: none.
 * =====
void show_screen(int tag){
    CTOS_LCDTClearDisplay ();
    switch(tag){
        case 0:
            CTOS_LCDTPrint("\fr  SYSTEM MODE  \fn");
            CTOS_LCDTPrintXY(1, 2, "1.System Reset");
            CTOS_LCDTPrintXY(1, 3, "2.Download Mode");
            CTOS_LCDTPrintXY(1, 4, "3.PowerOFF");
            CTOS_LCDTPrintXY(1, 5, "4.LCDSetContrast");
            break;
        case 1:
            CTOS_LCDTPrint("\fr  SYSTEM WAIT  \fn");
            CTOS_LCDTPrintXY(1, 3, "Plz Press Key");
            break;
        case 2:
            CTOS_LCDTPrint("\fr LCD CONTRAST \fn");
            CTOS_LCDTPrintXY(1, 3, "AAAAAAAAAAAAA");
            CTOS_LCDTPrintXY(1, 4, "bbbbbbbbbbb");
            CTOS_LCDTPrintXY(1, 5, "EEEEEEEEE");
            CTOS_LCDTPrintXY(1, 8, "Down->+'Up->-'");
            break;
    }
}

=====
 *      FUNCTION NAME: SetLCDContrast
 *      DESCRIPTION: Set LCD Contrast
 *      RETURN: none.
 *      NOTES: none.
 * =====
void SetLCDContrast(void)
{
    //Declare Local Variable //
    BYTE bValue = 0x38;

    show_screen(2);

    while(1) {
```

```

CTOS_KBDHit(&key);
if (key == d_KBD_CANCEL) {
    show_screen(0);
    return;
}
if (key == d_KBD_DOWN) {
    bValue++;
    //To setup the contrast of the LCD //
    CTOS_LCDSetContrast(bValue);
}
if (key == d_KBD_UP){
    bValue--;
    CTOS_LCDSetContrast(bValue);
}
}
/*
* =====
* FUNCTION NAME: main
* DESCRIPTION: function main is responsible for all the tasks listed in the
introduction to this demo.
* RETURN: none.
* NOTES: none.
* ===== */
int main(int argc, char *argv[])
{
    // TODO: Add your program here //

    show_screen(0);

    while(1){
        CTOS_KBDGet (&key );
        switch(key){
            case d_KBD_1:
                //Calling this function will reset all the peripherals, all of the open files will
                be closed,
                //and all the connection session with external device will be close, too. //
                CTOS_SystemReset();
                break;
            case d_KBD_2:
                //To enter the PM(Program Manager) "Download AP" mode //
                CTOS_EnterDownloadMode();
                show_screen(0);
                break;
            case d_KBD_3:
                //To power off the system //
                CTOS_PowerOff();
                break;
            case d_KBD_4:
                SetLCDContrast();
                break;
            case d_KBD_CANCEL:
                return 0;
                break;
        }
    }
}

// eof

```

## B.18. Timer Led

```
/*
 * FILE NAME: timer_led
 * MODULE NAME: Timer & LED
 * PROGRAMMER: Peggy Chang
 * DESCRIPTION:
 * REVISION: 01.00
 */

=====
 *      I N C L U D E S
 =====

#include <string.h>
#include <stdio.h>
#include <ctosapi.h>

/* =====
 * FUNCTION NAME: main
 * DESCRIPTION: function main is responsible for all the tasks listed in the
introduction to this demo.a
 * RETURN: none.
 * NOTES: none.
 * ===== */
int main(int argc, char *argv[])
{
    BYTE key;
    BYTE const Led[] = {d_LED1, d_LED2,d_LED3};
    int i = 0;

    CTOS_LCDTClearDisplay();
    CTOS_LCDTPrintXY(1, 1, " Timer LED Test ");
    CTOS_LCDTPrintXY(1, 8, "Press X for Exit");

    //Turn on/off specific LED //
    CTOS_LEDSet(Led[i], d_ON);
    // Set 300 ms //
    CTOS_TimeOutSet(TIMER_ID_1, 30);

    while(1)
    {
        // Check if Cancel key is pressed
        CTOS_KBDHit(&key);
        if(key == d_KBD_CANCEL)
        {
            CTOS_LEDSet(Led[i], d_OFF);
            break;
        }

        if(CTOS_TimeOutCheck(TIMER_ID_1) == d_YES)
        {
            // Turn off current and Turn on next one
            CTOS_LEDSet(Led[i++], d_OFF);
            if(i >= 3)
            {
                i = 0;
            }
            CTOS_LEDSet(Led[i], d_ON);

            // Set 300 ms
            CTOS_TimeOutSet(TIMER_ID_1, 30);
        }
    }

    return 0;
}

// eof
```

## B.19. USB

```
/*=====
* FILE NAME: USBTTest
* MODULE NAME: USB Function
* PROGRAMMER: Peggy Chang
* DESCRIPTION: Test USB Functions
* REVISION: 01.00
=====*/
/*=====
*      I N C L U D E S
=====
*/
#include <string.h>
#include <stdio.h>
#include <ctosapi.h>
#include <stdlib.h>
#include <stdarg.h>

/*=====
*      D E F I N E S
=====
*/
#define d_BUFF_SIZE 128      //Buffer Size

//Declare Global Variable //
BYTE babuff[d_BUFF_SIZE];
BYTE key;

/* =====
* FUNCTION NAME: USBTxData
* DESCRIPTION: Send data to the host via the device
* RETURN: none.
* NOTES: none.
* =====*/
void USBTxData(BYTE *pbaTxData, USHORT usTxLen)
{
    CTOS_LCDTPrintXY(1,6,pbaTxData);
    //Check whether it is ready to send data from device to host, or check whether the
    previous transmission is done //
    if (CTOS_USBTxReady() == d_OK)
    {
        //Send data from device to host //
        if (CTOS_USBTxData(pbaTxData,usTxLen) == d_OK){
            CTOS_LCDTPrintXY(1, 8, "Tx OK");
        }else{
            CTOS_LCDTPrintXY(1, 8, "Tx Fail");
        }
    }else{
        CTOS_LCDTPrintXY(1, 8, "Tx Ready Fail");
    }
}
/* =====
* FUNCTION NAME: USBRxData
* DESCRIPTION: Receive data from the host.
* RETURN:
* NOTES: none.
* =====*/
void USBRxData(BYTE *baRxData, USHORT *pusRxLen)
{
    USHORT usr_Len = 0;
    memset(baRxData,0x00,*pusRxLen);

    //Check whether is any data received from the host //
    if (CTOS_USBRxReady(pusRxLen) == d_OK){
        if(*pusRxLen> d_BUFF_SIZE) *pusRxLen = d_BUFF_SIZE;
        //Retrieve the data received from the host //
        if (CTOS_USBRxData(baRxData,pusRxLen) != d_OK) *pusRxLen = 0;
    }
}
```

```

int main(int argc,char *argv[])
{
    int i = 1;
    BYTE RxData[256];
    USHORT RxLen = 0;
    USHORT RxTotalLen = 0, TxTotalLen = 0;

    // TODO: Add your program here //

    //Enable USB Function //
    if (CTOS_USBOpen() != d_OK){
        CTOS_LCDTPrintXY(1,8,"USB Open Fail    ");
    }

    //Cancel the currently transmission of data to host //
    CTOS_USBTxFlush();
    //Clear all the data currently received from host //
    CTOS_USBRxFlush();

    memset(RxData,0x00,sizeof(RxData));

    sprintf(RxData,"0123456789ABCDEF");
    RxLen = 16;

    CTOS_LCDTClearDisplay ();
    CTOS_KBDGet(&key);
    //Send the data of received to the host directly //
    USBTxData(RxData,RxLen);

    CTOS_LCDTPrintXY(1, 1, "----Tx Data----");
    CTOS_LCDTPrintXY(1, 2, "Length=");
    CTOS_LCDTPrintXY(1, 3, "----Rx Data----");
    CTOS_LCDTPrintXY(1, 4, "Length=");

    while(1){

        memset(RxData,0x00,sizeof(RxData));
        RxLen = sizeof(RxData);

        CTOS_Delay(1000);

        //Receive the data via USB channel from the host //
        USBRxData(RxData,&RxLen);
        if (RxLen > 0){
            RxTotalLen += RxLen;
            sprintf(&babuff[100],"%dbytes",RxTotalLen);
            CTOS_LCDTPrintXY(8, 4, &babuff[100]);

            CTOS_Delay(1000);

            //Send the data of received to the host directly //
            USBTxData(RxData,RxLen);

            sprintf(&babuff[100],"%dbytes",TxTotalLen);
            CTOS_LCDTPrintXY(8, 2, &babuff[100]);
            TxTotalLen += RxLen;
        }

        sprintf(&babuff[100],"i = %d",i);
        CTOS_LCDTPrintXY(1, 7, &babuff[100]);
        i++;

        CTOS_KBDHit(&key);
        //retrun main() //
        if (key == d_KBD_CANCEL){
            //Clear all the data currently received from host //
            CTOS_USBRxFlush();
            //Disable USB Function //
            CTOS_USBClose();
            return 0;
        }
    }
}

// eof

```



## B.20. TCP Setting

```
/* **** */
* FILE NAME: Setting.c
* PROGRAMMER: Vance Ke
* DESCRIPTION: Test PPP/TCP Setting Functions
* REVISION: 01.00
* **** */

/* ===== */
*           I N C L U D E S
* ===== */
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctosapi.h>

/* ===== */
* FUNCTION NAME: ConnectTO
* DESCRIPTION: Set and get timeout for the connect and disconnect functions of GPRS
and Modem.
* RETURN:      none
* NOTES:       none
* ===== */
void ConnectTO(void)
{
    USHORT ret;
    ULONG ulTime_s = 10000;
    ULONG ulTime_g = 0;
    BYTE str[17];

    ret = CTOS_TCP_SetConnectTO(ulTime_s);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY (1,1, "Set Timeout FAIL");
        sprintf(str, "ret = %X", ret);
        CTOS_LCDTPrintXY (1,2, str);
        return;
    }
    CTOS_LCDTPrintXY (1,1, "Set ConnTO OK");

    ret = CTOS_TCP_GetConnectTO(&ulTime_g);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY (1,3, "Get Timeout FAIL");
        sprintf(str, "ret = %X", ret);
        CTOS_LCDTPrintXY (1,4, str);
        return;
    }
    CTOS_LCDTPrintXY (1,3, "Get ConnTO OK");
    sprintf(str, "Time = %ld", ulTime_g);
    CTOS_LCDTPrintXY (1,6, str);
}

/* ===== */
* FUNCTION NAME: RxTO
* DESCRIPTION: Set and get the timeout value of receiving data.
* RETURN:      none
* NOTES:       none.
* ===== */
void RxTO(void)
{
    USHORT ret;
    ULONG ulTime_s = 12000;
    ULONG ulTime_g = 0;
    BYTE str[17];

    ret = CTOS_TCP_SetRxTO(ulTime_s);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY (1,1, "Set Timeout FAIL");
```

```

        sprintf(str, "ret = %X", ret);
        CTOS_LCDTPrintXY (1,2, str);
        return;
    }
    CTOS_LCDTPrintXY (1,1, "Set RxTO OK");

    ret = CTOS_TCP_GetRxTO(&ulTime_g);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY (1,3, "Get Timeout FAIL");
        sprintf(str, "ret = %X", ret);
        CTOS_LCDTPrintXY (1,4, str);
        return;
    }
    CTOS_LCDTPrintXY (1,3, "Get RxTO OK");
    sprintf(str, "Time = %ld", ulTime_g);
    CTOS_LCDTPrintXY (1,6, str);
}

/*=====
* FUNCTION NAME: RetryCounter
* DESCRIPTION: Set and get retry counter for connect, send, receive, and disconnect
functions of GPRS and Modem.
* RETURN:      none
* NOTES:       none.
*=====
void RetryCounter(void)
{
    USHORT ret;
    USHORT usTime_s = 20;
    USHORT usTime_g = 0;
    BYTE   str[17];

    ret = CTOS_TCP_SetRetryCounter(usTime_s);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY (1,1, "Set Timeout FAIL");
        sprintf(str, "ret = %X", ret);
        CTOS_LCDTPrintXY (1,2, str);
        return;
    }
    CTOS_LCDTPrintXY (1,1, "Set Retry OK");

    ret = CTOS_TCP_GetRetryCounter(&usTime_g);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY (1,3, "Get Timeout FAIL");
        sprintf(str, "ret = %X", ret);
        CTOS_LCDTPrintXY (1,4, str);
        return;
    }
    CTOS_LCDTPrintXY (1,3, "Get Retry OK");
    sprintf(str, "Time = %d", usTime_g);
    CTOS_LCDTPrintXY (1,6, str);
}

/*=====
* FUNCTION NAME: PPP_SetTO
* DESCRIPTION: Set and get the timeout value for PPP
* RETURN:      none
* NOTES:       none.
*=====
void PPP_SetTO(void)
{
    USHORT ret;
    ULONG  ulTime_s = 45000;
    ULONG  ulTime_g = 0;
    BYTE   str[17];

    ret = CTOS_PPP_SetTO(ulTime_s);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY (1,1, "Set Timeout FAIL");
        sprintf(str, "ret = %X", ret);
        CTOS_LCDTPrintXY (1,2, str);
    }
}

```

```

        return;
    }
    CTOS_LCDTPrintXY (1,1, "Set PPPTO OK");

    ret = CTOS_PPP_GetTO(&ulTime_g);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY (1,3, "Get Timeout FAIL");
        sprintf(str, "ret = %X", ret);
        CTOS_LCDTPrintXY (1,4, str);
        return;
    }
    CTOS_LCDTPrintXY (1,3, "Get PPPTO OK");
    sprintf(str, "Time = %ld", ulTime_g);
    CTOS_LCDTPrintXY (1,6, str);
}

/*=====
* FUNCTION NAME: PPP_Retry
* DESCRIPTION: Set and get the retry counter for PPP.
* RETURN:      none
* NOTES:       none.
*=====
void PPP_Retry(void)
{
    USHORT ret;
    ULONG usTime_s = 15;
    USHORT usTime_g = 0;
    BYTE str[17];

    ret = CTOS_PPP_SetRetryCounter(usTime_s);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY (1,1, "Set Retry FAIL");
        sprintf(str, "ret = %X", ret);
        CTOS_LCDTPrintXY (1,1, str);
        return;
    }
    CTOS_LCDTPrintXY (1,1, "Set Retry OK");

    ret = CTOS_PPP_GetRetryCounter(&usTime_g);
    if(ret != d_OK)
    {
        CTOS_LCDTPrintXY (1,3, "Get Retry FAIL");
        sprintf(str, "ret = %X", ret);
        CTOS_LCDTPrintXY (1,4, str);
        return;
    }
    CTOS_LCDTPrintXY (1,3, "Get Retry OK");
    sprintf(str, "Time = %d", usTime_g);
    CTOS_LCDTPrintXY (1,6, str);
}

/*=====
* FUNCTION NAME: main
* DESCRIPTION: Main entry for the PPP/TCP setting sample code
* RETURN:      none.
* NOTES:       none.
*=====
int main (void)
{
    BYTE key;

    CTOS_BackLightSet(d_BKLIT_LCD,d_ON);

    while(1)
    {
START:
        CTOS_LCDTPrintXY(1, 1, "\fr    TCP Setting \fn");
        CTOS_LCDTPrintXY (1,2, "1.ConnectTO");
        CTOS_LCDTPrintXY (1,3, "2.RxAckTO");
        CTOS_LCDTPrintXY (1,4, "3.RxTO");
        CTOS_LCDTPrintXY (1,5, "4.RetryCounter");
        CTOS_LCDTPrintXY (1,6, "5.PPP_SetTO");
        CTOS_LCDTPrintXY (1,7, "6.PPP_Retry");
    }
}

```

```

CTOS_KBDGet ( &key );
CTOS_LCDTClearDisplay ();

switch(key)
{
    case d_KBD_1:
        ConnectTO();
        break;
    case d_KBD_2:
        CTOS_LCDTPrintXY(1,8,"Not supported !!");
        break;
    case d_KBD_3:
        RxTO();
        break;
    case d_KBD_4:
        RetryCounter();
        break;
    case d_KBD_5:
        PPP_SetTO();
        break;
    case d_KBD_6:
        PPP_Retry();
        break;
    case d_KBD_CANCEL: //Exit Application //
        CTOS_BackLightSet(d_BKLIT_LCD,d_OFF);
        exit(0);
    default:
        goto START;
}
CTOS_KBDGet (&key);
CTOS_LCDTClearDisplay ();
}

CTOS_BackLightSet(d_BKLIT_LCD,d_OFF);
exit(0);
}

// eof

```

## B.21. GPRS

```
/*=====
* FILE NAME:    GPRS.c
* PROGRAMMER:   Vance Ke
* DESCRIPTION:  Test GPRS Functions
* REVISION:    01.00
***** */

/*=====
*      I N C L U D E S
***** */

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <ctosapi.h>

/*=====
*      D E F I N E S
***** */

#define PPP_TO_MS    45000 //45000ms
#define CONNECT_TO_MS 5000 // 5000ms
#define CONNECT_RETRY 2

/*=====
* FUNCTION NAME: pollStatusResult
* DESCRIPTION: Show the status and return value of GPRSStatus and return until not
get IO_PROCESSING state.
* RETURN:       The return value of GPRSStatus
* NOTES:        none
***** */

USHORT pollStatusResult(BYTE *func, USHORT theRTN)
{
    BYTE key , str[17];
    USHORT usret;
    DWORD status;

    CTOS_LCDTClearDisplay ();
    CTOS_LCDTPrintXY(1, 1, func);
    memset(str, 0 , sizeof(str));
    CTOS_LCDTPrintXY(1, 2, str);

    status = 0;
    while (1)
    {
        usret = CTOS_TCP_GPRSStatus(&status);
        sprintf(str , "Return:%04x      " , usret);
        CTOS_LCDTPrintXY(1, 3, str);
        sprintf(str , "Status:%x      " , status);
        CTOS_LCDTPrintXY(1, 4, str);

        if (usret == 0x2321) //keep polling status
        {
            CTOS_Delay(2000);
            continue;
        }
        else
            break;
    }
    return usret;
}

/*=====
* FUNCTION NAME: main
* DESCRIPTION: Main entry for the GPRS testing sample code.
* RETURN:       none
* NOTES:        none
***** */
int main(void)
```

```

{
    BYTE iSocket; //TCP socket descriptor
    USHORT usrtn , usrtn2 = d_OK;
    BYTE key;
    BYTE bStr[30];
    BYTE baLocalIP[4] = {0};
    BYTE baRemoteIP[4] = { 218 , 211 , 35 , 213}; //Host IP Address, put yours
    USHORT usPort = 8000;
    char APN[20] = "internet";
    USHORT usSendLen;
    USHORT usRecvLen;
    BYTE baSendBuf[1024]; //Send buffer
    BYTE baRecvBuf[1100]; //Receive buffer

    CTOS_BackLightSet(d_BKLIT_LCD,d_ON);

    strcpy(baSendBuf, "0123456789ABCDEFGH" );//Set sending buffer
    usSendLen = usRecvLen = strlen(baSendBuf);

    while (1)
    {
        START_MENU:
        CTOS_LCDTClearDisplay ();
        CTOS_LCDTPrintXY(1, 1, "\fr TCP GPRS Test \fn");
        CTOS_LCDTPrintXY(1, 3, " OK      -> Start");
        CTOS_LCDTPrintXY(1, 4, " Cancel -> Exit");

        CTOS_KBDGet ( &key );
        if( key == d_KBD_CANCEL )
            break;
        else if( key != d_KBD_ENTER )
            goto START_MENU;

        /* Initialize(Reset) the GSM device */
        CTOS_LCDTClearDisplay ();
        CTOS_LCDTPrint ( "GPRS Init...\n" );
        CTOS_TCP_GPRSInit();
        CTOS_LCDTPrint ( "OK" );

        CTOS_KBDGet ( &key );

        /* Set PPP timeout (ms) */
        usrtn = CTOS_PPP_SetTO(PPP_TO_MS);

        /* Do GPRS open with related config value */
        CTOS_LCDTClearDisplay ();
        usrtn = CTOS_TCP_GPRSOpen(baLocalIP ,APN ,"" ,"" );
        usrtn = pollStatusResult("GPRS Open.....", usrtn);
        if (usrtn != d_OK)
            goto EXIT_ERR;

        /* Get the local IP address of the current connection */
        CTOS_LCDTPrint ( "\n\n" );
        CTOS_LCDTPrint ( "Get IP...\n" );
        usrtn = CTOS_TCP_GPRSGetIP(baLocalIP );
        if(usrtn != d_OK)
            goto EXIT_ERR_CLOSE;
        sprintf(bStr, "%d.%d.%d.%d",
baLocalIP[0],baLocalIP[1],baLocalIP[2],baLocalIP[3]);
        CTOS_LCDTPrint ( bStr );

        CTOS_KBDGet ( &key );

        /* Set TCP connecting timeout (ms) and retry time */
        usrtn = CTOS_TCP_SetConnectTO(CONNECT_TO_MS);
        usrtn = CTOS_TCP_SetRetryCounter(CONNECT_RETRY);

        /* Connect to the remote server */
        usrtn = CTOS_TCP_GPRSConnectEx(&iSocket ,baRemoteIP ,usPort);
        usrtn = pollStatusResult("GPRS Connect ", usrtn);
        if (usrtn != d_OK)
            goto EXIT_ERR_CLOSE;

        CTOS_KBDGet ( &key );

        /* Send TCP packet data out */
}

```

```

CTOS_LCDTPrint ("Transmitting...\n");
usrtn = CTOS_TCP_GPRSTx(iSocket ,baSendBuf ,usSendLen);
usrtn = pollStatusResult("GPRS Tx", usrtn);
CTOS_LCDTPrint ("\n");
sprintf(bStr, "TX Len = %d \n", usSendLen);
CTOS_LCDTPrint (bStr);
if (usrtn != d_OK)
    goto EXIT_ERR_DISCONNECT;

CTOS_KBDGet ( &key );

/* Receive TCP packet data ---
 * When get d_OK with insufficient data , you need to call CTOS_TCP_GPRSRx()
again to get left. */
CTOS_LCDTPrint ("Receiving... \n");
usrtn = CTOS_TCP_GPRSRx(iSocket ,baRecvBuf ,&usRecvLen);
usrtn = pollStatusResult("GPRS Rx", usrtn);
CTOS_LCDTPrint ("\n");
sprintf(bStr, "RX Len = %d \n", usRecvLen);
CTOS_LCDTPrint (bStr);
if (usrtn != d_OK)
    goto EXIT_ERR_DISCONNECT;

CTOS_KBDGet ( &key );

/* Disconnect the remote server */
usrtn = CTOS_TCP_GPRSDisconnect(iSocket);
usrtn = pollStatusResult("GPRS Disconnect", usrtn);
if (usrtn != d_OK)
    goto EXIT_ERR_DISCONNECT;

CTOS_KBDGet ( &key );

#if 1 /* Close the GSM device -- SYNC way */
    CTOS_LCDTClearDisplay ();
    CTOS_LCDTPrintXY(1, 1, "GPRS Close...");
    usrtn = CTOS_TCP_GPRSClose();
    sprintf(bStr, "Return:%04x ", usrtn);
    CTOS_LCDTPrintXY(1, 3, bStr);
#else /* Close the GSM device -- ASYNC way */
    usrtn = CTOS_TCP_GPRSClose_A();
    usrtn = pollStatusResult("GPRS Close A...", usrtn);
#endif

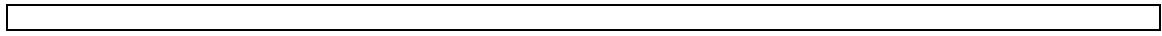
if (usrtn != d_OK)
    goto EXIT_ERR_CLOSE;

CTOS_LCDTPrint ("\nOK");
CTOS_KBDGet ( &key );
CTOS_LCDTClearDisplay ();
}
exit(0);

EXIT_ERR_DISCONNECT:
usrtn2 = CTOS_TCP_GPRSDisconnect(iSocket);
usrtn2 = pollStatusResult("GPRS Disconnect*", usrtn2);
EXIT_ERR_CLOSE:
usrtn2 = CTOS_TCP_GPRSClose();
usrtn2 = pollStatusResult("GPRS Close* ", usrtn2);
EXIT_ERR:
    CTOS_KBDGet ( &key );
    CTOS_LCDTPrint ("\n");
    sprintf(bStr, "Err Ret:%04x ", usrtn);
    CTOS_LCDTPrint(bStr );
    CTOS_LCDTPrint ("\n");
    if (usrtn2 != d_OK)
    {
        sprintf(bStr, "Err Exit:%04x ", usrtn2);
        CTOS_LCDTPrint(bStr );
    }
    CTOS_KBDGet ( &key );
    CTOS_LCDTClearDisplay ();
    exit(1);
}

// eof

```



## B.22. Modem TCP

```
/*=====
 * FILE NAME:    TCPModem.c
 * PROGRAMMER:   Vance Ke
 * DESCRIPTION:  Test Modem Functions
 * REVISION:     01.00
 =====*/

/*=====
 *      I N C L U D E S
 =====*/
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <ctosapi.h>

/*=====
 *      D E F I N E S
 =====*/
#define d_BUFF_SIZE (1024)           // Buffer Size
#define Mode        d_M_MODE_ASYNC_FAST // Mode type
#define HandShake   d_M_HANDSHAKE_V32BIS_AUTO_FB // HandShake type
#define CountryCode d_M_COUNTRY_TAIWAN // CountryCode

/*=====
 *      GLOBAL VARIABLES
 =====*/
char *ptr;
char progress[]=". . . . .";

/*=====
 * FUNCTION NAME: Open_Show
 * DESCRIPTION:  Open the modem device & show the return value
 * RETURN:       TRUE and FALSE
 * NOTES:        none
 =====*/
BOOL Open_Show(void)
{
    USHORT RET;
    BYTE str[30];

    memset(str, 0, sizeof(str));
    RET = CTOS_TCP_ModemOpen (Mode , HandShake , CountryCode);
    CTOS_LCDTPrintXY ( 1, 1, "Open...          " );
    sprintf(str, "ret:%x", RET);
    CTOS_LCDTPrintXY ( 1, 2,str );
    if(RET != d_OK)
        return FALSE;
    return TRUE;
}

/*=====
 * FUNCTION NAME: status_Show
 * DESCRIPTION:  Show the status and return value of ModemStatus
 * RETURN:       The return value of ModemStatus
 * NOTES:        none
 =====*/
USHORT status_Show(DWORD* Status, BYTE page)
{
    USHORT RET;
    BYTE str[30];
    page*=3;
    memset(str, 0, sizeof(str));
    RET = CTOS_TCP_ModemStatus(Status);
    sprintf(str, "Status:%04x ", *Status);
    CTOS_LCDTPrintXY (1,2+page, str );
    sprintf(str, "ret:%04x      ", RET);
    CTOS_LCDTPrintXY (1,3+page, str );
```

```

        if (ptr == progress)
            ptr = progress + strlen(progress) -1;
        else
            ptr--;
        sprintf(str, "%s          ", ptr);
        CTOS_LCDTPrintXY (1,4+page, str );
        return RET;
    }

/*=====
* FUNCTION NAME: ContinueCheckStatus
* DESCRIPTION:   Check the status until not get IO_PROCESSING state.
* RETURN:         The return value of ModemStatus
* NOTES:         none
*=====
USHORT ContinueCheckStatus(USHORT statusRET, DWORD* Status, BYTE page)
{
    while(statusRET == d_TCP_IO_PROCESSING)
    {
        statusRET = status_Show(Status, page);
        if(statusRET == d_OK)
            return statusRET;
    }
    return statusRET;
}

/*=====
* FUNCTION NAME: Dialup_Show
* DESCRIPTION:   Dial up and show the return value of ModemDialup
* RETURN:         TRUE and FALSE
* NOTES:         none
*=====
BOOL Dialup_Show(BYTE* Phone, BYTE* ID, BYTE* PW, ULONG Timeout , DWORD* Status)
{
    USHORT usret;
    BYTE str[30],i;

    memset(str, 0, sizeof(str));
    CTOS_LCDTClearDisplay ( );
    CTOS_LCDTPrintXY ( 1, 1, "Dialup...    " );
    usret = CTOS_TCP_ModemDialup(Phone, ID, PW, Timeout);
    sprintf(str, "ret:%04x ", usret);
    CTOS_LCDTPrintXY (1,3, str );

    ptr = progress;
    while (1)
    {
        usret = status_Show(Status, 0);
        if (usret == d_TCP_IO_PROCESSING) //keep polling dial-up status
        {
            CTOS_Delay(2000);
            continue;
        }

        if(*Status == d_M_STATUS_MODEM_ONLINE)
            return TRUE;
        else
            return FALSE;
    }
}

/*=====
* FUNCTION NAME: setBuffer
* DESCRIPTION:   Set up the Buffer
* RETURN:         none
* NOTES:         none
*=====
void setBuffer(BYTE* buf)
{
    BYTE RNG[10],i;
    for(i=0; i<128; i++)
    {
        CTOS_RNG (RNG);           //Set 8 bytes random values
        memcpy(buf + i*8, RNG, 8);
    }
}

```

```

        memcpy(buf, "\x03\xFE", 2);
    }

/*=====
 * FUNCTION NAME: main
 * DESCRIPTION: Main entry for the TCP Modem testing sample code
 * RETURN:      none
 * NOTES:       none
 *===== */
int main (void)
{
    UINT Times;
    int iSocket , ret;
    DWORD dwStatus = 0;
    USHORT usret , len;
    BYTE key , bstr[30];
    BYTE baPhone[9] = "40508999"; //ISP Setting -- phone number
    BYTE baID [6] = "yahoo"; //ISP Setting -- User Name
    BYTE baPW [6] = "yahoo"; //ISP Setting -- Password
    BYTE baSendBuf[d_BUFF_SIZE];
    BYTE baRecvBuf[d_BUFF_SIZE];
    BYTE baLocoalIP [] = {0x00, 0x00, 0x00, 0x00 };
    BYTE baRemoteIP [] = {218, 211, 35, 213};

    // TCP port Setting
    USHORT usTCPPort = 8000;

    //UDP Setting
    BYTE baUDPSrcDestIP[8] = {218 , 211 , 35 , 213 , 0 , 0 , 0 , 0};
    USHORT usUDPRemotePort = 8010; //remote UDP port
    USHORT usUDPLocalPort = 7000; //Local port for remote server to send data in .

    CTOS_BackLightSet(d_BKLIT_LCD,d_ON);

START_MENU:
    CTOS_LCDTClearDisplay ( );
    CTOS_LCDTPrintXY(1, 1, "\fr TCP/UDP Test \fn");
    CTOS_LCDTPrintXY(1, 3, " OK -> Start");
    CTOS_LCDTPrintXY(1, 4, " Cancel -> Exit");

    CTOS_KBDGet ( &key );
    if( key == d_KBD_CANCEL )
        exit(0);
    else if (key != d_KBD_ENTER)
        goto START_MENU;

START:
    memset(bStr, 0, sizeof(bStr));

    CTOS_LCDTClearDisplay ( );

    /* Initialize the modem device */
    CTOS_TCP_ModemInit();

    /* Open the modem device */
    ret = Open_Show();
    CTOS_KBDGet ( &key );
    if(!ret)
        goto END;

    CTOS_LCDTClearDisplay ( );
    CTOS_LCDTPrint ( "Set Local IP...\n" );

    /* Set the local IP address of the connection */
    usret = CTOS_TCP_ModemSetIP(baLocoalIP);
    sprintf(bStr, "ret:%x", usret);
    CTOS_LCDTPrint (bStr);
    CTOS_KBDGet ( &key );
    if(usret != d_OK)
        goto END_CLOSE;

    /* Dial the number of ISP and get the IP address of this connection */
    ret = Dialup_Show(baPhone, baID, baPW, 5000, &dwStatus); //Timeout : 5000 * 10ms =
50s
    CTOS_KBDGet ( &key );
    if(!ret)

```

```

        goto END;

    CTOS_LCDTClearDisplay ();
    CTOS_LCDTPrint ( "Get IP...\n" );

    /* Get the local IP address of the current connection */
    usret = CTOS_TCP_ModemGetIP(baLocoalP);
    sprintf(bStr, "ret:%x\n", usret);
    CTOS_LCDTPrint (bStr);
    sprintf(bStr, "%d.%d.%d.%d", baLocoalP[0], baLocoalP[1], baLocoalP[2],
baLocoalP[3] );
    CTOS_LCDTPrint (bStr );
    CTOS_KBDGet ( &key );
    if(usret != d_OK)
        goto END_CLOSE;

    /* Connect to the remote server */
    CTOS_LCDTClearDisplay ( );
    CTOS_LCDTPrintXY(1, 1, "Connect...  ");
    CTOS_LCDTPrintXY(1, 2, "ret:");
    usret = CTOS_TCP_ModemConnectEx (&iSocket, baRemoteIP, usTCPPort);
    usret = ContinueCheckStatus(usret, &dwStatus, 0);
    CTOS_KBDGet ( &key );
    if(usret != d_OK)
        goto END_CLOSE;

    /* Send TCP packet data out */
    setBuffer(baSendBuf);
    CTOS_LCDTClearDisplay ( );
    CTOS_LCDTPrint ("Transmitting...\n" );
    usret = CTOS_TCP_ModemTx(iSocket, baSendBuf, d_BUFF_SIZE);
    usret = ContinueCheckStatus(usret, &dwStatus, 0);
    if(usret != d_OK)
        goto END_CLOSE;

    /* Receive TCP packet data ---
       * When get d_OK with insufficient data , you need to call CTOS_TCP_ModemRx() again
       to get left. */
    len = d_BUFF_SIZE;
    memset(baRecvBuf, 0x00, sizeof(baRecvBuf));
    CTOS_LCDTPrint ("Receiving... \n" );
    usret = CTOS_TCP_ModemRx(iSocket, baRecvBuf, &len);
    usret = ContinueCheckStatus(usret, &dwStatus, 1);
    CTOS_LCDTPrint ("\n");
    sprintf(bStr, "len = %d \n", len);
    CTOS_LCDTPrint (bStr);
    CTOS_KBDGet ( &key );
    if(usret != d_OK)
        goto END_CLOSE;

    /* Disconnect the remote server */
    CTOS_LCDTClearDisplay ( );
    CTOS_LCDTPrint( "Disconnect... \n" );
    usret = CTOS_TCP_ModemDisconnect (iSocket);
    usret = ContinueCheckStatus(usret, &dwStatus, 0);
    CTOS_KBDGet ( &key );
    if(usret != d_OK)
        goto END_CLOSE;

    /* Send UDP packets to specified remote IP address and port. */
    CTOS_LCDTClearDisplay ( );
    CTOS_LCDTPrint( "UDP TX Data...\n" );
    usret = CTOS_UDP_ModemTx(baRemoteIP,usUDPRemotePort ,baSendBuf ,d_BUFF_SIZE);
    usret = ContinueCheckStatus(usret, &dwStatus, 0);
    CTOS_KBDGet ( &key );
    if(usret != d_OK)
        goto END_CLOSE;

    /* Receive UDP packets from remote IP address. */
    len = d_BUFF_SIZE;
    memcpy( baUDPSrcDestIP + 4 , baLocoalP , 4);
    CTOS_LCDTClearDisplay ( );
    CTOS_LCDTPrint( "UDP RX Data...\n" );
    usret = CTOS_UDP_ModemRx(baUDPSrcDestIP ,&usUDPLocalPort ,baRecvBuf ,&len);
    usret = ContinueCheckStatus(usret, &dwStatus, 0);
    CTOS_KBDGet ( &key );

```

```

if(usret != d_OK)
    goto END_CLOSE;

/* Hook on this connection */
CTOS_LCDTClearDisplay ( );
CTOS_LCDTPrint ("OnHook... \n" );
usret = CTOS_TCP_ModemOnHook();
usret = ContinueCheckStatus(usret, &dwStatus, 0);
CTOS_KBDGet ( &key );

END_CLOSE:
    CTOS_TCP_ModemClose();      //Close the modem device
END:
    CTOS_LCDTClearDisplay ( );
    CTOS_LCDTPrint ( "Test End \n\n" );
    CTOS_LCDTPrint ( "CANCEL->Close\n" );
    CTOS_LCDTPrint ( "CLEAR ->Restart\n" );
    while(1)
    {
        CTOS_KBDGet(&key);
        if( key == d_KBD_CANCEL )
            break;
        else if( key == d_KBD_CLEAR )
            goto START;
    }
    exit(0);
}

// eof

```

## B.23. Mifare

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <ctosapi.h>

BOOL Authenticae(BYTE Block, BYTE KeyType, BYTE Key[6], BYTE *sn)
{
    BYTE i;
    BYTE KeyBuf=0x00;
    BYTE AuthenKey [12];
    memset(AuthenKey,0x00,sizeof(AuthenKey));
    for(i=0;i<6;i++)
    {
        KeyBuf=Key[i];
        AuthenKey[i*2]=(KeyBuf^0xF0)&0xF0;
        AuthenKey[i*2]|= (KeyBuf>>4);
        AuthenKey[i*2+1]=((KeyBuf^0x0F)<<4);
        AuthenKey[i*2+1]|= (KeyBuf&0x0F);
    }

    if(CTOS_MifareLOADKEY(AuthenKey) !=d_OK)
        return FALSE;
    if(CTOS_MifareAUTH(KeyType, Block, sn) !=d_OK)
        return FALSE;
    return TRUE;
}

BOOL Decrease_Part (UCHAR bBlockNr,UCHAR* bValue)
{
    if(CTOS_MifareDECREASE(bBlockNr,bValue) !=d_OK)
        return FALSE;
    if(CTOS_MifareTRANSFER(bBlockNr) !=d_OK)
        return FALSE;
    return TRUE;
}

BOOL Increase_Part (UCHAR bBlockNr,UCHAR* bValue)
{
    if(CTOS_MifareINCREASE(bBlockNr,bValue) !=d_OK)
        return FALSE;
    if(CTOS_MifareTRANSFER(bBlockNr) !=d_OK)
        return FALSE;
    return TRUE;
}

BOOL Check_Card_state(BYTE *ATQA_f, BYTE *SAK_f, BYTE *SN, BYTE *SN_Len)
{
    UCHAR times=0;
    while(CTOS_CLTypeAActiveFromIdle(0, ATQA_f, SAK_f,SN, SN_Len) !=d_OK)
    {
        if(times<5)
            times++;

    }
    if(times<5)
    {
        times=0;
        return FALSE;
    }
    return TRUE;
}

#define      TypeA      (0x60)
#define      TypeB      (0x61)
#define      BlockNr     (10)
#define      Value       (20)
#define      Valuebyte   (4)

int main ( )
```

```

{
    BYTE Key[6]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};
    BYTE CSN[10];
    BYTE KeyType=TypeB;
    BYTE CSN_Len=0;
    BYTE ATQA[2];
    BYTE SAK[10];

    UCHAR ValueBuf [16];
    UCHAR DataeBuf [16];
    UCHAR ValueBlock=10;
    UCHAR DataBlock=8;
    ULONG Money;
    UCHAR str[5];
    UCHAR Card_state=0;

    syslib_init ();
    Money = 5;

    while(1)
    {
        // TODO: Add your program here //
        CTOS_LCDTClearDisplay ( );
        CTOS_LCDTPrint ( "[Hello World]\n" );
        memset(DataeBuf, 0x34, sizeof(DataeBuf));

        Card_state=Check_Card_state(ATQA, SAK, CSN, &CSN_Len);
        if(Card_state)
        {
            if(!Authenticae(ValueBlock, KeyType, Key, CSN))
                continue;
            CTOS_LCDTPrint("[OK]");
            Increase_Part(ValueBlock, (UCHAR*)&Money);
            if(CTOS_MifareREADBLOCK(ValueBlock, ValueBuf)==d_OK)
            {
                memcpy(str, ValueBuf, 4);
                str[5]='\0';
                CTOS_LCDTPrintXY(1,3,str);
                CTOS_LCDTPrintXY(1,4,ValueBuf);
            }
            else
                CTOS_LCDTPrintXY(1,3,"[Read ERROR]");

            if(CTOS_MifareWRITEBLOCK(DataBlock, DataeBuf)!=d_OK)
                CTOS_LCDTPrintXY(1,3,"[Write ERROR]");
            if(CTOS_MifareREADBLOCK(DataBlock, DataeBuf)==d_OK)
                CTOS_LCDTPrintXY(1,4,DataeBuf);
            else
                CTOS_LCDTPrintXY(1,3,"[Read ERROR]");

            CTOS_LCDTPrintXY(10,2,"[OVER]");
            CTOS_Delay(1000);
        }
    }
    return 0;
}

```

## B.24. Mifare EEPROM

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <ctosapi.h>
#define TypeA      (0x60)
#define TypeB      (0x61)
#define BlockNr    (10)
#define Value      (20)
#define Valuebyte  (4)

void PrintResponse(USHORT response, UCHAR* baBuf)
{
    if(response==d_OK)
    {
        CTOS_LCDTPrintXY(1,3,baBuf);
        CTOS_LCDTPrintXY(1,2,"[OK]");
    }
    else
        CTOS_LCDTPrintXY(1,2,"[ERROR]");
}

int main ( )
{
    BYTE Key[6]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};
    BYTE CSN[10];
    BYTE KeyType=TypeB;
    BYTE CSN_Len=0;
    BYTE ATQA[2];
    BYTE SAK[10];
    UCHAR Block=10;
    UCHAR str[5];
    UCHAR bAddrMSB=0x00, bAddrLSB=0x30, bNumByte=12;
    UCHAR ReadBuf[12];
    UCHAR WriteBuf[12];
    USHORT ret;
    syslib_init ( );
    while(1)
    {
        CTOS_LCDTClearDisplay ( );
        CTOS_LCDTPrintXY (1,1, "[Hello World]" );
        memset(ReadBuf,0,sizeof(ReadBuf));
        memset(WriteBuf,0x0F,sizeof(WriteBuf));
        while(CTOS_CLTypeAActiveFromIdle(0, ATQA, SAK,CSN, &CSN_Len)!=d_OK);

        ret = CTOS_MifareREADE2(bAddrMSB, bAddrLSB, bNumByte, ReadBuf);
        PrintResponse(ret, ReadBuf);
        CTOS_Delay(1000);

        ret = CTOS_MifareWRITEE2(bAddrMSB, bAddrLSB, bNumByte, WriteBuf);
        PrintResponse(ret, WriteBuf);
        CTOS_Delay(1000);

        ret = CTOS_MifareLOADKEYE2(bAddrMSB, bAddrLSB);
        if(ret!=d_OK)
        {
            CTOS_LCDTPrintXY(1,4,"[LOAD ERROR]");
            CTOS_Delay(1000);
            continue;
        }
        ret = CTOS_MifareAUTH(KeyType, Block, CSN);
        if(ret!=d_OK)
        {
            CTOS_LCDTPrintXY(1,4,"[AUTH ERROR]");
            CTOS_Delay(1000);
            continue;
        }
        CTOS_LCDTPrintXY(1,4,"[AUTH OK]");
        CTOS_Delay(1000);
    }
}
```

```

    }
    return 0;
}

```

## B.25. T=CL

```

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <ctosapi.h>

USHORT AntiCollisionSample(void)
{
USHORT r;

BYTE baATQA1[20], bSAK1[20], baTCSN1[20];
BYTE bTCSNLen1;

BYTE baATQA2[20], bSAK2[20], baTCSN2[20];
BYTE bTCSNLen2;

BYTE bAutoBR;
BYTE baATS[30];
USHORT bATSLen;

BYTE RxBuf[1024];
USHORT RxLen;

BYTE ShowTemp[64];

//=====
// Type A check

CTOS_LCDTPrintXY(1,1,"Put Contactless Card");
while(1)
{
    bTCSNLen1 = sizeof(baTCSN1);
    r = CTOS_CLTypeAActiveFromIdle(0, baATQA1, bSAK1, baTCSN1, &bTCSNLen1);
    if(r == d_OK) break;
}
r = CTOS_CLTypeAHalt();
if(r != d_OK)
{
    CTOS_LCDTPrintXY(1,2,"Halt Error");
    return -1;
}

memset(ShowTemp,0x00,sizeof(ShowTemp));
sprintf(ShowTemp,"%02X %02X %02X %02X", baTCSN1[0], baTCSN1[1], baTCSN1[2],
baTCSN1[3]);
CTOS_LCDTPrintXY(1,2,"Card 1 SN:");
CTOS_LCDTPrintXY(1,3,ShowTemp);

bTCSNLen2 = sizeof(baTCSN2);
r = CTOS_CLTypeAActiveFromIdle(0, baATQA2, bSAK2, baTCSN2, &bTCSNLen2);
if(r != 0x83E5)
{
    memset(ShowTemp,0x00,sizeof(ShowTemp));
    sprintf(ShowTemp,"%02X %02X %02X %02X", baTCSN2[0], baTCSN2[1], baTCSN2[2],
baTCSN2[3]);
    CTOS_LCDTPrintXY(1,4,"Card 2 SN:");
    CTOS_LCDTPrintXY(1,5,ShowTemp);
    CTOS_LCDTPrintXY(1,8,"anti-collision");

    return -1;
}

r = CTOS_CLTypeAActiveFromHalt(0, baATQA1, bSAK1, baTCSN1, bTCSNLen1);
if(r != d_OK)
{
    CTOS_LCDTPrintXY(1,7,"Active Halt Error");
    return -1;
}

```

```

r = CTOS_CLRATS(bAutoBR,baATS,&bATSLen);
if(r != d_OK)
{
    CTOS_LCDTPrintXY(1,7,"RATS Error");
    return -1;
}

r = CTOS_CLDESELECT();
if(r != d_OK)
{
    CTOS_LCDTPrintXY(1,7,"Deselect Error");
    return -1;
}

r = CTOS_CLTypeAActiveFromHalt(0, baATQA1, bSAK1, baTCSN1, btCSNLen1);
if(r != d_OK)
{
    CTOS_LCDTPrintXY(1,7,"Active Halt Error");
    return -1;
}

r = CTOS_CLRATS(bAutoBR,baATS,&bATSLen);
if(r != d_OK)
{
    CTOS_LCDTPrintXY(1,7,"RATS Error");
    return -1;
}

r = CTOS_CLAPDU("\x00\x00\x00\x00\x00",5,RxBuf,&RxLen);
if(r != d_OK)
{
    CTOS_LCDTPrintXY(1,7,"APDU Error");
    return -1;
}

}

int main ( )
{
    USHORT r;

    CTOS_LCDTSelectFontSize(d_LCD_FONT_8x8);

    BYTE a[16],b[16];
    while(1)
    {
        AntiCollisionSample();

        CTOS_Delay(2000);
        CTOS_LCDTClearDisplay();
    }
    return 0;
}

```

## B.26. PowerAutoMode

```
/**  
** A Template for developing new terminal application  
**/  
  
#include <string.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <stdarg.h>  
/** These two files are necessary for calling CTOS API **/  
#include <ctosapi.h>  
  
#define PARENT_MSG_LINE1      1  
#define PARENT_MSG_LINE2      2  
#define PARENT_MSG_LINE3      3  
#define PARENT_MSG_LINE4      4  
#define PARENT_MSG_LINE5      5  
#define PARENT_MSG_LINE6      6  
#define PARENT_MSG_LINE7      7  
#define PARENT_MSG_LINE8      8  
  
//The function that will execute when sleep or standby  
BOOL sleep_test(void *arg,BYTE mach_state)  
{  
    BYTE *buf;  
    BYTE Display[20];  
    CTOS_LCDTClearDisplay();  
    buf = (BYTE *)arg;  
    sprintf(Display,"Sendstring = %s",buf);  
    CTOS_LCDTPrintXY(1, PARENT_MSG_LINE1, Display);  
    CTOS_Delay(2000);  
  
    if (mach_state == d_PWR_SLEEP_MODE )  
    {  
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE2, "Doing sleep ....      ");  
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE3, "Sleep.....      ");  
        CTOS_Delay(2000);  
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE3, "Sleep Done ....      ");  
        CTOS_Delay(1000);  
        CTOS_LCDTClearDisplay();  
    }  
    else if ( mach_state == d_PWR_STANDBY_MODE )  
    {  
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE2, "Doing standby ....      ");  
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE3, "Standby.....      ");  
        CTOS_Delay(2000);  
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE3, "Standby Done ....      ");  
        CTOS_Delay(1000);  
        CTOS_LCDTClearDisplay();  
    }  
  
    return d_OK;  
}  
  
//The function that will execute when resume from sleep or standby  
BOOL resume_test(void *arg,BYTE last_state,USHORT *wakesrc)  
{  
    const char *buf;  
    BYTE buff[32];  
    buf = arg;  
    printf("%s: last_state:%d, wakesrc:0x%x", buf, last_state,*wakesrc);  
  
    CTOS_LCDTClearDisplay();
```

```

    if ( last_state == d_PWR_STANDBY_MODE )
    {
        sprintf(buff,"Wakeup ..... 0x%02X ",*wakesrc);
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE4, buff);
        CTOS_Delay(2000);
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE5, "Wakeup Done ....      ");
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE7, "Press any key to test");
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE8, "again or 'X' to leave");
        CTOS_Delay(1000);
    }
    else if (last_state == d_PWR_SLEEP_MODE )
    {
        sprintf(buff,"Resume ..... 0x%02X ",*wakesrc);
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE4, buff);
        CTOS_Delay(2000);
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE5, "Resume Done ....      ");
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE7, "Press any to test... ");
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE8, "again, 'X' to leave");
        CTOS_Delay(1000);
    }

    return d_OK ;
}

//If you do not need to excute extra movment before or after the terminal sleep.
//Please mark or delete line 101 to 106 and 109.
int main(int argc, char **argv)
{
    BYTE *Sendstring="TEST";
    ULONG ulSec = 12;
    BYTE key;
    CTOS_LCDSelectMode(d_LCD_GRAPHIC_320x240_MODE);

    CTOS_PowerAutoModeEnable(); //Open Auto Power Managment

    //---- [Start]initial sleep function.
    CTOS_stPowerModeCallback sleep_ops;
    memset(&sleep_ops,0 , sizeof(CTOS_stPowerModeCallback));
    sleep_ops.OnEnterPowerMode = sleep_test;           //Excete when Enter
PowerMode(Sleep/Standy)
    sleep_ops.pEnterPowerModeData = (void *)Sendstring; //Pass data into function
"resume_test"
    sleep_ops.OnExitPowerMode = resume_test;           //Excete when Exit
PowerMode(Sleep/Standy)
    CTOS_PowerAutoModeRegisterCallback(&sleep_ops);   //RegisterCallBack fuction of
Auto Power Managment
    //---- [End]initial sleep function.

    //---- Set the timeout of Standby Mode or Sleep Mode
    CTOS_PowerAutoModeTimeToSleep(0); // prevent the system get into Sleep Mode
    CTOS_PowerAutoModeTimeToStandby(ulSec); //If you did not enter any key/Inset
SmartCard/Swipe MSR, the AP will get into Stanby Mode
    //CTOS_PowerAutoModeTimeToStandby(0); // prevent the system get into Standby Mode
    //CTOS_PowerAutoModeTimeToSleep(ulSec); //If you did not enter any key/Inset
SmartCard/Swipe MSR, the AP will get into Sleep Mode

    do{
        CTOS_LCDTClearDisplay();
        CTOS_LCDTPrintXY(1, PARENT_MSG_LINE1, "Wiat 12s to Standby.");
        CTOS_KBDGet(&key);
    }while(key != d_KBD_CANCEL);

    CTOS_PowerAutoModeUNRegisterCallback(); //Unegist CallBack fuction of Auto Power
Management
    CTOS_PowerAutoModeDisable(); //Close Auto Power Managment
}

```

```
    exit(0);  
}
```

~ END ~