VT-OS V1.1 API Library

Alarm Manager Functions List (Sys)

1 AlarmAddHit

Purpose Add an event to event hit queue

Prototype BOOLEAN AlarmAddHit(AlarmEvent *alm_evt)

Scope Internal

Input parameters

alm_evt	The event to add

Output parameters None

Return

TRUE	Success
FALSE	Fail

Comment

2 AlarmAddQuery

Purpose Add an entry to the query queue

Prototype BOOLEAN AlarmAddQuery(AppID app, UBYTE type, RecordID rec_id, USHORT app_data)

Scope Internal

Input parameters

app	Application ID
type	Event type to query
rec_id	RecordID of the event
app_data	Data passed to the query application

Output parameters None

Return

TRUE	Success
FALSE	Fail

Comment

3 AlarmDelHit

Purpose Remove an event from the hit queue

Prototype BOOLEAN AlarmDelHit(AlarmEvent alm_evt);

Scope Internal

Input parameters

alm_evt	Event to delete

Output parameters None

Return

TRUE	Success
FALSE	Fail

Comment

4 AlarmMgrInit

Purpose Reset alarm manager

Prototype void AlarmMgrInit(void)

Scope OS

Input parameters None

Output parameters None

Return None

Comment Call when system boot up or reset the system

5 AlarmOnHit

Purpose Call by OS when alarm event hit

Prototype void AlarmOhHit(void)

Scope OS

Input parameters None

Output parameters None

Return None

Comment

6 AlarmSetAlarm

Purpose Find the next alarm and set the hardware timer

Prototype void AlarmSetAlarm(void)

Scope Internal

Input parameters None

Output parameters None

Return None

Comment

Alarm Manager Functions Summary (User)

1 AlarmAddEvent

Purpose Inset an event to the alarm queue, replace the scheduled one if same event existed

Prototype BOOLEAN AlarmAddEvent(AlarmEvent alm_evt)

Scope All

Input parameters

alm_evt	The event to add

Output parameters None

Return

TRUE	Success
FALSE	Fail

Comment The content in alm_evt.alert_msg will be copied by alarm manager in system memory space

2 AlarmDelAppEvent

Purpose Remove all event of a specific application

Prototype void AlarmDelAppEvent(AppID appid)

Scope All

Input parameters

appid	Application ID

Output parameters None

Return None

Comment

3 AlarmDelEvent

Purpose Delete an event

Prototype Err AlarmDelEvent (AlarmEvent alm_evt)

Scope All

Input parameters

alm_evt	Event to be deleted, an event is identifed by AppID and type

Output parameters None

Return

TRUE	Success
FALSE	Event not found

Comment This function will call AlarmDelEventEx() to remove the event

4 AlarmFindEvent

Purpose Find an event in the alarm queue

Prototype BOOLEAN AlarmFindEvent(AlarmEvent *alm_evt, AlarmEventLink **evt_ptr);

Scope Internal

Input parameters

alm_evt	The event to find, an event is identified by the AppID, type and record ID

Output parameters

evt_ptr	Pointer to the found event

Return

TRUE	Found
FALSE	Not found

Comment

5 AlarmGetLastAlarmTime

Purpose Get last scheduled event of an application

Prototype BOOLEAN AlarmGetLastAlarmTime(AppID appid, RTM* last_time)

Scope All

Input parameters appid Application ID

Output parameters last_time Date and time of last scheduled event

Return None

Comment

6 AlarmTotalEvent

Purpose Get total number of event in the alarm queue

Prototype USHORT AlarmTotalEvent(void)

Scope All

Input parameters None

Output parameters None

Return Number of event in the alarm queue

Comment

Auto Power Functions

1 AppDisOnPdaTimerInt

Purpose For Application to disable the Auto Power off (the Pda will not power off if use this function)

Prototype void AppDisOnPdaTimerInt()

Scope Application

Input Parameters None

Output Parameters None
Result None
Comment None

2 CheckPdaPwr

Purpose Reset the Auto Power off counter, let the Pda power off after next minutes. But if the Pda status Off, this function nothing to do.
Prototype void CheckPdaPwr()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

3 DisOnPdaTimerInt

Purpose For System to disable the Auto Power off (the Pda will not power off if use this function)
Prototype void DisOnPdaTimerInt()
Scope System
Input Parameters None
Output Parameters None
Result None
Comment None

4 EnOnPdaTimerInt

Purpose Set / Enable how long the Pda will Power Off
Prototype void EnOnPdaTimerInt()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

5 OffPda

Purpose Turn Off the Pda
Prototype void OffPda()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

6 OnPda

Purpose Start to count the Pda On timer
Prototype void OnPda()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

7 OnPdaInit

Purpose Init the Auto Power function
Prototype void OnPdaInit()
Scope System / Internal
Input Parameters None

Output Parameters None
Result None
Comment None

8 PdaPwrState

Purpose Check the Auto power status
Prototype void PdaPwrState(UWORD *pdapwrstate)
Scope System / Appliction / Internal
Input Parameters UWORD

Output Parameters UWORD
Result TRUE Pda On
FALSE Pda Off
Comment None

9 ResetOnPda

Purpose Reset the Auto Power off counter, let the Pda power off after next minutes.
Prototype void ResetOnPda()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

Battery Warning Functions

1 BatteryWarningInit

Purpose This function is called by the Kernel to initialize the global variables of the battery warning process.

Prototype void BatteryWarningInit()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

2 BatteryWarningPopup

Purpose This function is called by the *SystemHandleEvent* to handle the POWER_EVENT if exists.

Prototype void BatteryWarningPopup(EvtType *Event)

Scope System

Input Parameters Event Pointer to input event

Output Parameters None

Result None

Comment
- if battery door is opened, the *BatteryWarningPopup* sends events to shut down the PDA.
- if battery low is detected, the *BatteryWarningPopup* popup the battery low dialog box.

3 BatteryWarningReset

Purpose This function is called to initialize part of global variables.

Prototype void BatteryWarningReset()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

4 BatteryWarningRestore

Purpose This function is called to redraw the previous screen that is saved before the battery low dialog box popup. It should be called after the x on the top right corner of the dialog box is clicked.

Prototype void BatteryWarningRestore()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment The previous screen (160 pixels x 160 pixels) is saved before popping up the battery low dialog box

5 BatteryWarningRestorePreviousFormStatus

Purpose This function is called to by *BatteryWarningRestore*. Sometimes, menu or popup trigger objects are popup in the previous application screen. Therefore, *BatteryWarningRestorePreviousFormStatus* is to close the popup menu object or popup trigger and the application screen can be returned to normal situation.

Prototype void BatteryWarningRestorePreviousFormStatus()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

6 BatteryWarningSetEnableStatus

Purpose This function is called to set whether allowing the popping up of the battery low dialog box when battery low is detected.

Prototype void BatteryWarningSetEnableStatus(BOOLEAN enable)

Scope Application

Input Parameters enable TRUE if allowing the popping up of

battery low dialog box
FALSE if not allowing the popping up of battery low dialog box

Output Parameters None

Result None

Comment If poping up of battery low dialog box is not allowed and battery low is detected, the battery low event is saved. Once the dialog box is enabled again, the battery low dialog box is popup.

7 BatteryWarningShutDown

Purpose This function is called to shut down the PDA.

Prototype void BatteryWarningShutDown()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment Two events are sent out to active application and the system. One is EVT_APP_STOP to notify the active application that the application is requested to stop. The other event is EVT_POWER_DOWN to notify the system that power down sequence is requested.

Chkmem Functions

1 Dummy1

Purpose Determine the reset type.

Prototype void Dummy1()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment This function is called when the reset button has been pressed. There are two types of reset, hard-reset and soft reset. The boot-loader will determine the reset type and indicate the result to the system by placing an value to memory location 0x80000020.
content of 0x80000020 = 0x55555555 indicate soft-reset
content of 0x80000020 = 0xaaaaaaaa indicate hard-reset

2 Dummy2

Purpose Check memory integrity when system reset.

Prototype void Dummy2()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment If memory is corrupted, it will call the hard-reset routine to reset the PDA.

Dbinit Functions

1 AllApplicationDatabaseInit

Purpose To initialize all the database of the applications that are pre-programmed in the FLASH

Prototype AllApplicationDatabaseInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

2 AppAnniversariesDatabaseInit

Purpose To initialize database of Anniversaries application

Prototype AppAnniversariesDatabaseInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

3 AppEmailDatabaseInit

Purpose To initialize database of Email application

Prototype AppEmailDatabaseInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

4 AppExpenseDatabaseInit

Purpose To initialize database of Expense application

Prototype AppExpenseDatabaseInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

5 AppMemoDatabaseInit

Purpose To initialize database of Memo application

Prototype AppMemoDatabaseInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

6 AppPhonebookDatabaseInit

Purpose To initialize database of Phonebook application

Prototype AppPhonebookDatabaseInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

7 AppSchedulerDatabaseInit

Purpose To initialize database of Scheduler application

Prototype AppSchedulerDatabaseInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

8 AppSketchDatabaseInit

Purpose To initialize database of Sketch application

Prototype AppSketchDatabaseInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

9 AppToDoDatabaseInit

Purpose To initialize database of To Do List application

Prototype AppToDoDatabaseInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

10 AppVoxMemoDatabaseInit

Purpose To initialize database of Voice Memo application

Prototype AppVoxMemoDatabaseInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

11 SchedulerDBAppmtToBuffer

Purpose To pack the appointments and strings of appointments to two different packed buffers for storage in database

Prototype
```
SchedulerDBAppmtToBuffer(AppointmentDB *appmt_rec,
                         BYTE **string, WORD num_appmt,
                         BYTE **appmt_rec_buf,
                         BYTE  **string_buf,
                         WORD *num_bytes_appmt,
                         WORD *num_bytes_string)
```

Scope System

Input Parameters

appmt_rec	Pointer to array of AppointmentDB structures
string	Pointer to array of strings
num_appmt	total number of appointment
appmt_rec_buf	the packed buffer of the appointments
string_buf	the packed buffer of strings of appointments
num_bytes_appmt	the total number of bytes of the packed buffer of appointments
num_bytes_string	the total number of bytes of the packed buffer of the strings of the appointments

Output Parameters None

Result None

Comment None

Dev_bety.c Functions

1 BettyInit

Purpose Initialize the UCB 1200's hardware

Prototype void BettyInit()

Scope OS

Input Parameters None

Output Parameters None

Result None

Comment None

2 BettyInitSM

Purpose State machine of BettyInit
Prototype void BettyInitSM()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

3 BettyReset

Purpose Reset the UCB 1200 chip
Prototype void BettyReset()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

Dev_pen.c Functions

3 PenDownIntEnable

Purpose This function is to enable the pen down interrupt generated form UCB 1200

Prototype void PenDownIntEnable()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

4 PenDownIntEnableSM

Purpose This function is to enable the pen down interrupt generated form UCB 1200

Prototype void PenDownIntEnableSM()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

5 PenGetxy

Purpose This function will provide the coordinate of pen on touch panel

Prototype void PenGetxy()

Scope System

Input Parameters None

Output Parameters None

Result A pen-move or a pen-up message with coordinate is generated
Comment None

6 PenInit

Purpose This function is to initialize the pen input sub-system
Prototype void PenInit()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

7 PenInitSM

Purpose State machine PenInit
Prototype void PenInitSM()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

8 PenRestoreDefault

Purpose The function restore the default value of the calibrated pen data
Prototype void PenRestoreDefault()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

9 PenRotateInit

Purpose This function will inform the pen driver to rotate the pen coordinate
Prototype void PenRotateInit()

Scope System
Input Parameters None
Output Parameters None
Result None
Comment None

10 PenSetCalValue

Purpose This function is used to convert the ADC pen coordinates to display coordinates and set the calibrated pen data
Prototype WORD PenSetCalValue (WORD x1, WORD y1, WORD x2, WORD y2)
Scope System /application
Input Parameters x1, y1 : top-left ADC value of pen coordinate
x2, y2 : bottom-right ADC value of pen coordinate
Output Parameters None
Result None
Comment None

11 PenTmrIntHandler

Purpose This function is called by the timer every 20ms to sample pen coordinate
Prototype void PenTmrIntHandler()
Scope System
Input Parameters None
Output Parameters None
Result None
Comment None

Dev_pwr.c Functions

1 AHTurnLcdOff

Purpose This function will turn off the LCD on emergency (battery empty)
Prototype void AHTurnLcdOff(void)

Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

2 BatChkMainLevel

Purpose Check the battery level
Prototype WORD BatChkMainLevel()
Scope System / Application
Input Parameters None

Output Parameters None
Result Return 1 if not permitted.
Return 0 if operation permitted.
A power level message will be generated
Comment None

3 BatChkMainLevelBakup

Purpose Check the previous battery level status. This function is called when the BatChkMainLevel() is not permitted
Prototype WORD BatChkMainLevelBakup()
Scope System / Application
Input Parameters None

Output Parameters None
Result None
Comment None

4 BatChkMainLevelInt

Purpose Check the battery level within interrupt
Prototype WORD BatChkMainLevelInt()
Scope System
Input Parameters None

Output Parameters None
Result None

Comment None

5 BatDoorOpen

Purpose Check the battery compartment door status
Prototype WORD BatDoorOpen()
Scope System / Application
Input Parameters None

Output Parameters None
Result Return 1 if door is open
Return 0 if door is close
Comment None

6 BatLowHandler

Purpose This function handle the situation when battery level drop to 2.2 V.
Prototype BatLowHandler()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

7 BatLowPowerManagement

Purpose Shutdown the battery hunger device when battery is extremely low
Prototype BatLowPowerManagement()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

8 BatMapLifeTime

Purpose Convert the ADC value of battery level to life time of battery in term of percentage
Prototype int BatMapLifeTime(int v)

Scope System
Input Parameters v : ADC value of battery level

Output Parameters None
Result Return the percentage of battery life
Comment None

9 CheckBatteryEmpty

Purpose Check if the battery is out
Prototype WORD CheckBatteryEmpty()
Scope System / application
Input Parameters None

Output Parameters None
Result Return 1 if battery is out
Return 0 if battery is not out
Comment None

10 CheckBatteryLow

Purpose Check if battery is extremely low
Prototype WORD CheckBatteryLow()
Scope System / application
Input Parameters None

Output Parameters None
Result Return 1 if battery is extremely low
Return 0 if battery is not extremely low
Comment None

11 ChkBatterySM

Purpose State machine of ChkBattery
Prototype void ChkBatterySM()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

12 CpuChangeSpeed

Purpose Change the clock rate of CPU
Prototype void CpuChangeSpeed(int speed)
Scope System / application
Input Parameters speed : a number to divide 75MHz

Output Parameters None
Result None
Comment None

13 CpuChangeSpeedInt

Purpose Change the clock rate of CPU within interrupt
Prototype Void CpuChangeSpeedInt(int speed)
Scope System
Input Parameters speed : a number to divide 75MHz
Output Parameters None
Result None
Comment None

14 CpuChiOff

Purpose Initialize and turn off the CHI module of CPU
Prototype CpuChiOff()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

15 CpuChkSpeed

Purpose Check the CPU clock rate currently running
Prototype int CpuChkSpeed()
Scope System
Input Parameters None

Output Parameters None
Result Return 0 if running at 75MHz
Return 1 if running at 37MHz
Return 2 if running at 18MHz
Return 3 if running at 9MHz
Comment None

16 CpuDoze

Purpose Turn the CPU to idle mode
Prototype void CpuDoze()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

17 CpuDozeOnSemiOff

Purpose Turn the CPU to idle mode when the CPU is under a semi-power off situation
Prototype void CpuDozeOnSemiOff()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

18 CpuGotoSleep

Purpose Power down the CPU.
Prototype CpuGotoSleep()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

19 CpuIROff

Purpose Initialize and turn off the IR module of CPU
Prototype CpuIROff()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

20 CpuMBusOff

Purpose Initialize and turn off the MagicBus module of CPU
Prototype CpuMBusOff()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

21 CpuPowerOff

Purpose Initialize and turn off the CPU
Prototype CpuPowerOff()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

22 CpuPowerOn

Purpose Turn the CPU on
Prototype CpuPowerOn()
Scope Internal
Input Parameters None

Output Parameters None

Result None
Comment None

23 CpuSibOff

Purpose Initialize and turn off the SIB module of CPU
Prototype CpuSibOff()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

24 CpuSpiOff

Purpose Initialize and turn off the SPI module of CPU
Prototype CpuSpiOff()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

25 CpuUartBOff

Purpose Initialize and turn off the UART B module of CPU
Prototype CpuUartBOff()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

26 HwAudioInit

Purpose Initialize the enable io pin of CPU and turn audio amplifier off
Prototype HwAudioInit()

Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

27 HwAudioOff

Purpose Initialize the enable io pin of CPU and turn audio amplifier off
Prototype HwAudioOff()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

28 HwAudioOn

Purpose Initialize the enable io pin of CPU and turn audio amplifier on
Prototype HwAudioOn()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

29 HwDiagnosticPinOff

Purpose Initialize the io pin of CPU and turn off the diagnostic pin
Prototype HwDiagnosticPinOff()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

30 HwHotSyncInit

Purpose Initialize the hot sync io pin of CPU
Prototype HwHotSyncInit()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

31 HwHotSyncOff

Purpose Initialize the hot sync io pin of CPU and turn it off
Prototype HwHotSyncOff()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

32 HwHotSyncOn

Purpose Initialize the hot sync io pin of CPU and turn it off
Prototype HwHotSyncOn()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

33 HwSystemEmgPowerOff

Purpose Power off the hardware system when battery is out.
Prototype HwSystemEmgPowerOff()
Scope System
Input Parameters None

Output Parameters None

Result None
Comment None

34 HwSystemPowerOff

Purpose Power off the hardware system
Prototype HwSystemPowerOff()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

35 HwTringOff

Purpose Turn off the io pin of CPU connected to TRING
Prototype HwTringOff()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

36 HwUartAInit

Purpose Turn off the Uart Tx and Rx hardware
Prototype HwUartAInit()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

38 LcdTurnedOff

Purpose Check if the Lcd hardware is turned off or not
Prototype WORD LcdTurnedOff(void)
Scope System
Input Parameters None

Output Parameters None
Result Return 1 if LCD is off
Return 0 if LCD is on
Comment None

39 NegBatDoorIntISR

Purpose Interrupt service routine of battery door status
Prototype NegBatDoorIntISR()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

40 NegPwrIntISR

Purpose Interrupt service routine battery level drop to 2.2v
Prototype NegPwrIntISR()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

41 NegPwrOkIntISR

Purpose Interrupt service routine battery level drop to 2.0v
Prototype NegPwrOkIntISR()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

42 PosBatDoorIntISR

Purpose Interrupt service routine of battery door status
Prototype PosBatDoorIntISR()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

43 PosPwrIntISR

Purpose Interrupt service routine battery level drop to 2.2v
Prototype PosPwrIntISR()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

44 PosPwrOkIntISR

Purpose Interrupt service routine battery level drop to 2.0v
Prototype PosPwrOkIntISR()
Scope System
Input Parameters None

Output Parameters None

Result None
Comment None

45 PowerMgr

Purpose Power Management routine will generate power event
Prototype PowerMgr(MsgType *PwrMsg)
Scope System
Input Parameters PwrMsg: system power message

Output Parameters None
Result Power Event will be generated
Comment None

46 PowerMgrInit

Purpose Initialize the power Management routine
Prototype void PowerMgrInit()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

47 PowerMgrInit2

Purpose Initialize the power Management routine
Prototype void PowerMgrInit2()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

48 pRestoreIoReg

Purpose Restore CPU IO register’s config
Prototype pRestoreIoReg()

Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

49 pSaveIoReg

Purpose Save CPU IO register's config
Prototype PsaveIoReg()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

50 SndDriverOff

Purpose Disable snd DMA before turn off the system
Prototype SndDriverOff()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

51 UartAOff

Purpose Initialize and turn off the UART A module of CPU
Prototype UartAOff()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

52 UartAOn

Purpose Initialize and turn on the UART A module of CPU
Prototype UartAOn()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

Dev_sib.c Functions

1 BettyQueueFlush

Purpose Flush the accessing queue
Prototype BettyQueueFlush()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

2 BtyAdcAccess

Purpose UCB1200 accessing routine
Prototype WORD BtyAdcAccess(void (*calling)())
Scope System
Input Parameters calling: Function pointer to acknowledge
Output Parameters None
Result None
Comment None

3 BtyAdcAccessHandler

Purpose UCB1200 accessing routine
Prototype BtyAdcAccessHandler()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

4 BtyAdcIsr

Purpose UCB1200 ADC Interrupt service routine (Not implement)
Prototype void BtyAdcIsr()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

5 BtyIoIsr

Purpose UCB1200 ADC Interrupt service routine (Not implement)
Prototype void BtyIoIsr()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

6 dSibAccess

Purpose SIB low level access supporting routine
Prototype void dSibAccess()
Scope Internal
Input Parameters None

Output Parameters None
Result None

Comment None

8 dSibReadData

Purpose Low level read service request for SIB
Prototype WORD dSibReadData(void (*caller)())
Scope Internal
Input Parameters caller : function to be executed when complete

Output Parameters None
Result None
Comment None

9 dSibWrite

Purpose Low level write SIB
Prototype void dSibWrite(int cmd)
Scope Internal
Input Parameters cmd : register number with command

Output Parameters None
Result None
Comment None

10 dSibWriteData

Purpose Low level read service request for SIB
Prototype void dSibWriteData(void (*caller)())
Scope Internal
Input Parameters caller : function to executed when complete

Output Parameters None
Result None
Comment None

12 SF0Isr

Purpose Sub-frame 0 interrupt service routine (for Sib Communication)
Prototype void SF0Isr()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

13 SF1Isr

Purpose Sub-frame 1 interrupt service routine (for Sib Communication)
Prototype void SF1Isr()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

14 SibAccess

Purpose	UCB1200 Access routine
Prototype	WORD SibAccess(WORD cmd, void (*caller)())
Scope	System
Input Parameters	cmd : register number with command caller : function to executed when complete
Output Parameters	None
Result	None
Comment	None

16 SibCapture

Purpose	To engage the SIB port
Prototype	void SibCapture()
Scope	System / application
Input Parameters	None
Output Parameters	None
Result	None
Comment	None

17 SibInit

Purpose	Initialize the SIB interface
Prototype	void SibInit()
Scope	System
Input Parameters	None
Output Parameters	None
Result	None
Comment	None

18 SibIntHandler

Purpose Interrupt service routine for interrupt from UCB1200
Prototype void SibIntHandler()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

19 SibIrqPosISR

Purpose Sib Interrupt service routine for positive edge of SIBIRQ
Prototype void SibIrqPosISR()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

20 SibRead

Purpose Read Data from SIB
Prototype void SibRead()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

21 SibReadData

Purpose API call for reading data from SIB
Prototype int SibReadData(int BtyReg)
Scope Application
Input Parameters BtyReg : Register of UCB1200 to be read

Output Parameters None
Result None
Comment None

22 SibReadDataHandle

Purpose Handler for reading data from sib
Prototype void SibReadDataHandle(void)
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

23 SibRelease

Purpose To release the SIB port
Prototype void SibRelease()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

24 SibWrite

Purpose Write data from SIB
Prototype void SibWrite()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

25 SibWriteData

Purpose API call for writing data from SIB
Prototype void SibWriteData(int BtyReg, int cmd)
Scope Application
Input Parameters BtyReg : register number of UCB1200
cmd: command to be written to UCB1200
Output Parameters None
Result None
Comment None

26 SibWriteDataHandle

Purpose Handler for reading data from sib
Prototype void SibWriteDataHandle(void)
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

Dev_snd.c Functions

1 SndEffEnable2

Purpose Enable Sound I/O
Prototype void SndEffEnable2(WORD Channel)
Scope System
Input Parameters Channel : MIC / SPEAKER

Output Parameters None
Result None
Comment None

2 SndEnable

Purpose Enable Sound I/O
Prototype void SndEnable(WORD Channel)

Scope System
Input Parameters Channel : MIC / SPEAKER
Output Parameters None
Result None
Comment None

3 SndEnableSM

Purpose State machine of SndEnable
Prototype void SndEnableSM()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

4 SndEnable_Int

Purpose Enable Sound I/O within interrupt
Prototype Void SndEnable_Int(WORD Channel)
Scope System
Input Parameters Channel : MIC / SPEAKER
Output Parameters None
Result None
Comment None

5 SndInit

Purpose Initialize the UCB1200's audio channel
Prototype void SndInit(WORD SamplingRate, WORD DataLength, WORD InputGain, WORD OutputAttn)
Scope System
Input Parameters SamplingRate : Sampling rate (Hz)
DataLength : 8bit or 16 bit data mode
InputGain : Input Gain of MIC
OutputAttn : Output Attenuation of SPEAKER
Output Parameters None
Result None
Comment None

6 SndInitSM

Purpose State machine of SndInit
Prototype void SndInitSM()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

7 SndIntHandler

Purpose Interrupt service routine for swapping buffer
Prototype SndIntHandler()
Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

8 SndPlyVolume

Purpose Set the play volume of audio output
Prototype Void SndPlyVolume(WORD InputGain, WORD OutputAttn)
Scope System
Input Parameters InputGain: Input Gain of MIC
OutputAttn: Output Attenuation of SPEAKER
Output Parameters None
Result None
Comment None

9 SndPlyVolumeSM

Purpose State machine of SndPlyVolume()
Prototype Void SndPlyVolumeSM()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

10 SndSetBufferSize

Purpose Set the DMA buffer size
Prototype void SndSetBufferSize(WORD Size)
Scope System
Input Parameters Size: Size of dma buffer

Output Parameters None
Result None
Comment None

11 SndStartDma

Purpose To setup and start the sound DMA channel
Prototype SndStartDma(WORD Action)
Scope System
Input Parameters Action : SND_RECORDING / SND_PLAYING

Output Parameters None
Result None
Comment None

12 SndVolume

Purpose Set volume of audio output
Prototype void SndVolume (WORD InputGain, WORD OutputAttn)
Scope System
Input Parameters InputGain: Input Gain of MIC
OutputAttn: Output Attenuation of SPEAKER

Output Parameters None
Result None
Comment None

13 SndVolumeSM

Purpose State machine of SndVolume()
Prototype void SndVolumeSM()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

Dev_tel.c Functions

1 TelDisable

Purpose Disable the telecom codec of UCB1200
Prototype void TelDisable()
Scope Application / System
Input Parameters None

Output Parameters None
Result None
Comment None

2 TelEnable

Purpose Enable the telecom codec of UCB1200
Prototype void TelEnable()
Scope Application / System
Input Parameters None

Output Parameters None
Result None
Comment None

3 TelGetBufferPtr

Purpose Provide receive and transmit dma buffer address to application
Prototype void TelGetBufferPtr(SHORT **rx_buf, SHORT **tx_buf)

Scope Application / System
Input Parameters rx_buf: address of receive dma buffer
tx_buf: address of receive dma buffer
Output Parameters None
Result None
Comment None

4 TelInit

Purpose Initialize the telecom codec of UCB1200 and CPU register
Prototype void TelInit()
Scope Application / System
Input Parameters None

Output Parameters None
Result None
Comment None

5 TelOffHook

Purpose Engage the phone line
Prototype void TelOffHook()
Scope Application / System
Input Parameters None

Output Parameters None
Result None
Comment None

6 TelOnHook

Purpose Release the phone line
Prototype void TelOnHook()
Scope Application / System
Input Parameters None

Output Parameters None
Result None
Comment None

7 TelSetBufferSize

Purpose Set the dma buffer size
Prototype void TelSetBufferSize(WORD Size)
Scope Application / System
Input Parameters size : size of dma buffer

Output Parameters None
Result None
Comment None

8 TelSetDmaAddress

Purpose To set the dma buffer address
Prototype void TelSetDmaAddress(SHORT *inbuf, SHORT *outbuf)
Scope Application / System
Input Parameters inbuf : address of input buffer
outbuf : address of output buffer
Output Parameters None
Result None
Comment None

9 TelSetSamplingRate

Purpose To set the sampling rate
Prototype void TelSetSamplingRate(WORD SamplingRate)
Scope Application / System
Input Parameters SamplingRate : sampling rate of telecom codec
Output Parameters None
Result None
Comment None

10 TelSetUpBty

Purpose To setup the UCB1200 for software modem
Prototype void TelSetUpBty(WORD SamplingRate)
Scope Application / System
Input Parameters SamplingRate : sampling rate of telecom codec
Output Parameters None
Result None
Comment None

11 TelStartDma

Purpose To start the dma of telecom codec
Prototype void TelStartDma()
Scope Application / System
Input Parameters None

Output Parameters None
Result None
Comment None

12 TelStopDma

Purpose To stop the dma of telecom codec
Prototype void TelStopDma()
Scope Application / System
Input Parameters None

Output Parameters None
Result None
Comment None

dm os.c Functions

1 meminit

Purpose Initialize dynamic memory.

Prototype void meminit(void)

Scope System

Input parameters None

Output parameters None

Return None

Comment None

2 pcalloc

Purpose Allocate a chunk of memory, initialize memory content to 0x00.

Prototype void *pcalloc(UWORD num, UWORD size)

Scope OS

Input parameters

num	Number of data structure to allocate
size	Size of one data structure

Output parameters None

Return

NULL	Fail, not enough memory
Otherwise	The pointer to the allocated memory

Comment None

3 pfree

Purpose Release an allocated memory chunk.

Prototype void pfree(void *pool)

Scope OS

Input parameters

pool	Pointer of memory to release

Output parameters None

Return None

Comment None

4 pmalloc

Purpose Allocate a chunk of memory.

Prototype void *pmalloc(UWORD size)

Scope System

Input parameters

size	Memory size required, in byte

Output parameters None

Return

NULL	Fail, not enough memory
Otherwise	The pointer to the allocated memory

Comment OS will allocated memory size in multiple of 4 byte, e.g. word alignment.

dm user.c Functions

1 qcalloc

Purpose Allocate a chunk of memory, initialize memory content to 0x00.

Prototype void *qcalloc(UWORD num, UWORD size)

Scope System/Application

Input parameters

num	Number of data structure to allocate
size	Size of one data structure

Output parameters None

Return

NULL	Fail, not enough memory

	Otherwise	The pointer to the allocated memory

Comment None

2 qfree

Purpose Release an allocated memory chunk.

Prototype void qfree(void *pool)

Scope Application

Input parameters

pool	Pointer of memory to release

Output parameters None

Return None

Comment None

3 qmalloc

Purpose Allocate a chunk of memory.

Prototype void *qmalloc(UWORD size)

Scope System/Application

Input parameters

size	Memory size required, in byte

Output parameters None

Return

NULL	Fail, not enough memory
Otherwise	The pointer to the allocated memory

Comment Allocated memory size in multiple of 4, e.g. word alignment.

4 userdminit

Purpose Initialize dynamic memory.

Prototype void userdminit(void)

Scope System

Input parameters None

Output parameters None

Return None

Comment None

Dynamic Memory Allocation

System

- pmalloc()
- pfree()
- pcalloc()

Application

- qmalloc()
- qfree()
- qcalloc()

System

1 pmalloc

Purpose Allocate a chunk of memory

Prototype void *pmalloc(UWORD size)

Scope OS

Input parameters

size	Memory size required, in byte

Output parameters None

Return

NULL	Fail, not enough memory
Otherwise	The pointer to the allocated memory

Comment OS will allocated memory size in multiple of 4 byte, e.g. word alignment

2 pcalloc

Purpose Allocate a chunk of memory, initialize memory content to 0x00

Prototype void *pcalloc(UWORD num, UWORD size)

Scope OS

Input parameters

num	Number of data structure to allocate
size	Size of one data structure

Output parameters None

Return

NULL	Fail, not enough memory
Otherwise	The pointer to the allocated memory

Comment

3 pfree

Purpose	Release an allocated memory chunk	
Prototype	void pfree(void *pool)	
Scope	OS	
Input parameters	pool	Pointer of memory to release
Output parameters	None	
Return	None	
Comment		

Application

1 qmalloc

Purpose	Allocate a chunk of memory	
Prototype	void *qmalloc(UWORD size)	
Scope	All	
Input parameters	size	Memory size required, in byte
Output parameters	None	
Return	NULL Otherwise	Fail, not enough memory The pointer to the allocated memory
Comment	Allocated memory size in multiple of 4, e.g. word alignment	

2 qcalloc

Purpose	Allocate a chunk of memory, initialize memory content to 0x00

Prototype void *qcalloc(UWORD num, UWORD size)

Scope All

Input parameters

num	Number of data structure to allocate
size	Size of one data structure

Output parameters None

Return

NULL	Fail, not enough memory
Otherwise	The pointer to the allocated memory

Comment

3 qfree

Purpose Release an allocated memory chunk

Prototype void qfree(void *pool)

Scope Application

Input parameters

pool	Pointer of memory to release

Output parameters None

Return None

Comment

El.c Functions

1 CheckEL

Purpose If the EL status on, reset the EL off counter. But if the EL status Off, this function nothing to do.
Prototype void CheckEL()
Scope System / Application
Input Parameters None
Output Parameters None
Result None
Comment None

2 DisELTimerInt

Purpose Disable the EL Timer to count the EL auto on / off
Prototype void DisELTimerInt()
Scope Internal
Input Parameters None
Output Parameters None
Result None
Comment None

3 ELInit

Purpose Init the EL function
Prototype void ELInit()
Scope System
Input Parameters None
Output Parameters None
Result None
Comment None

4 ELState

Purpose Check the EL Status is ON / OFF
Prototype void ELState(UWORD *elstate)
Scope Application / System
Input Parameters UWORD

Output Parameters UWORD
Result TRUE the EL is ON
FALSE the EL is OFF
Comment None

5 EnELTimerInt

Purpose Enable the EL start to count
Prototype void EnELTimerInt()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

6 OffEL

Purpose Turn OFF the EL
Prototype void OffEL()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

7 OnEL

Purpose Turn On the EL
Prototype void OnEL()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

8 ResetELWhenOn

Purpose Check the EL Status, when EL On, reset the EL counter
Prototype void ResetELWhenOn()
Scope Internal
Input Parameters None
Output Parameters None
Result None
Comment None

Error Manager Functions

1 ErrDisplay

Purpose Display the error message to terminal
Prototype void ErrDisplay(BYTE *errMsg)
Scope Internal
Input Parameters ErrMsg – point to the display string

Output Parameters None
Result None
Comment None

Eventmgr.c Functions

1 EvtAddEventHandler

Purpose To add an event handler function into the EvtGetEvent function to handle certain types of event. For example, this function is for the use with modem connection

Prototype BOOLEAN EvtAddEventHandler(BOOLEAN *fct())

Scope System

Input Parameters

fct	function pointer of the added EventHandler function that is added to the EvtGetEvent function

Output Parameters None

Result

TRUE	Added correctly
FALSE	Cannot added

Comment None

2 EvtAppend

Purpose To append an event into event queue

Prototype void EvtAppend(EvtType event, EvtQueueType *Q)

Scope Application/System

Input Parameters

event	event
Q	Pointer to the last entry of event queue

Output Parameters None

Result None

Comment None

3 EvtAppendEvt

Purpose To append an event into event queue

Prototype void EvtAppendEvt(WORD eventType, WORD eventID, WORD para1, WORD para2, void *evtPBP)

Scope Application/System

Input Parameters

eventType	event type
eventID	event ID
para1	event parameter 1
para2	event parameter 2
evtPBP	additonal event pointer for additional information

Output Parameters None

Result None

Comment None

4 EvtAppendEvtReplace

Purpose To append an event into event queue. The event manager searches whether an event with the same event type and event ID is already in the event queue. If yes, the event already in the event queue is replaced by the input event. Otherwise, the input event is only added to the queue.

Prototype

```
void EvtAppendEvtReplace(WORD eventType, WORD eventID,
                         WORD para1, WORD para2,
                         void *evtPBP)
```

Scope Application/System

Input Parameters

eventType	event type
eventID	event ID
para1	event parameter 1
para2	event parameter 2
evtPBP	additonal event pointer for additional information

Output Parameters None

Result None

Comment None

5 EvtDummy

Purpose It is a dummy function, which only returns FALSE, for faster operation of the event queue. If there is no added event handler function, then the EvtEventHandler function is mapped to this EvtDummy.

Prototype BOOLEAN EvtDummy(void)

Scope System

Input Parameters None

Output Parameters None

Result FALSE

Comment None

6 EvtGetEvent

Purpose It is a function to get an event from event queue.

Prototype void EvtGetEvent(EvtType *eventP)

Scope Application/System

Input Parameters None

Output Parameters None

Result FALSE

Comment When there is no event in event queue and no message in message queue, then the function will loop continuously and the CPU will stay in sleep mode until event or message is put in the queue.

7 EvtHandler

Purpose To process the added event handler

Prototype BOOLEAN EvtHandler(void)

Scope System

Input Parameters None

Output Parameters None

Result

TRUE	if
FALSE	if there is no event

Comment None

8 EvtInitHandler

Purpose To initialise the EvtHandle array

Prototype void EvtInitHandler(void)

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

9 EvtQueueCreate

Purpose To initialize the event queue to empty

Prototype void EvtQueueCreate(EvtQueueType *Q)

Scope System

Input Parameters

Q	Pointer to event queue

Output Parameters None

Result None

Comment None

10 EvtQueueEmpty

Purpose To check whether the event queue is empty or not

Prototype BOOLEAN EvtQueueEmpty(EvtQueueType *Q)

Scope System

Input Parameters Q Pointer to event queue

Output Parameters None

Result None

Comment None

11 EvtQueueFull

Purpose To check whether the event queue is full or not

Prototype BOOLEAN EvtQueueFull(EvtQueueType *Q)

Scope System

Input Parameters Q Pointer to event queue

Output Parameters None

Result None

Comment None

12 EvtRemoveEventHandler

Purpose To remove a event handler function from the event handler array

Prototype BOOLEAN EvtRemoveEventHandler(BOOLEAN (*fct)())

Scope	System	
Input Parameters	fct	Function pointer
Output Parameters	None	
Result	TRUE FALSE	removed successfully added successfully
Comment	None	

13 EvtServe

Purpose	To get an event from the head of the queue	
Prototype	void EvtServe(EvtType *eventP, EvtQueueType *Q)	
Scope	System	
Input Parameters	Q	Pointer to the event queue
Output Parameters	eventP	Pointer to the event that is on the head of the queue
Result	None	
Comment	None	

Ex_rate.c Functions

1 ExchangeCheckDBInited

Purpose	To check whether the exchange rate has been init or not
Prototype	BOOLEAN ExchangeCheckDBInited()

Scope Application/OS

Input Parameters None

Output Parameters None

Result

TRUE	if the database is exists
FALSE	if the database is not exists

Comment None

2 ExchangeCloseExchangeRateDB

Purpose To close the exchange rate database

Prototype void ExchangeCloseExchangeRateDB()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

3 ExchangeGetExchangeRate

Purpose

Prototype double ExchangeGetExchangeRate(USHORT first_country,
USHORT second_country)

Scope Application/System

Input Parameters

first_country	the index of first country
second_country	the index of second country

Output Parameters None

Result the exchange rate of first country to second country

Comment None

4 ExchangeGetMappingNum

Purpose To get the mapped country number from internal array

Prototype USHORT ExchangeGetMappingNum(USHORT item_num)

Scope OS

Input Parameters item_num wanted country number

Output Parameters None

Result The mapped number

Comment None

5 ExchangeInitExchangeRateDatabase

Purpose To initialise the exchange rate database

Prototype void ExchangeInitExchangeRateDatabase()

Scope OS

Input Parameters None

Output Parameters None

Result None

Comment It takes a long time to complete

6 ExchangeSetExchangeRateMappingArray

Purpose To set the exchange rate – country mapping array

Prototype void ExchangeSetExchangeRateMappingArray()

Scope OS

Input Parameters None

Output Parameters None

Result None

Comment None

Font Functions

1 SysGetFontHeight

Purpose To request the font height of a particular font

Prototype BYTE SysGetFontHeight(BYTE font_id)

Scope OS

Input Parameters

font_id	the input font type SMALL_FONT, MEDIUM_FONT, LARGE_FONT

Output Parameters None

Result The height of the font

Comment None

Global Find Functions

1 GlobalFindAddItem

Purpose This function is called to add an item in the global find link. The added item will be displayed in the Global Find Result screen.

Prototype
```
BOOLEAN GlobalFindAddItem(AppID appid, DatabaseID dbid,
                          RecordID rec_id,
                          USHORT field_num,
                          BYTE *string, BOOLEAN override)
```

Scope Application/System

Input Parameters

appid	The application ID of the application that provides the item. It can be set to 0xFFFF if the function is called by system
dbid	The database ID of the database that provides the item
rec_id	The record ID of the record that provides the item
field_num	The field number of the field that provides the item
string	a BYTE * pointer of a string that will be shown in the Global Find Result screen to represent the provided item
override	(This input parameter is only for system use) TRUE means that item is forced to add into the Global Find link whatever the memory is enough or not FALSE means that item should not be added into the Global Find link when memory is not enough

Output Parameters None

Result

TRUE	if item is added
FALSE	if item cannot be added

Comment None

2 GlobalFindDeleteAllItems

Purpose This function is called to delete all added items in the Global Find link.

Prototype void GlobalFindDeleteAllItems()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

3 GlobalFindInit

Purpose This function is called to initialize all parameters and variables of Global Find handle functions.

Prototype void GlobalFindInit ()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment This function is called when the PDA is soft-reset or hard-reset

4 GlobalFindPopupGlobal

Purpose This function is called to popup the Global Find Input screen.

Prototype void GlobalFindPopupGlobal ()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment The screen before popping up the Global Find Input screen is saved for later restoration.

5 GlobalFindPrepareAppSeparator

Purpose This function is called by *GlobalFindSearchData* to prepare an application separator (for example, ---------- Memo ----------) for an application.

Prototype void GlobalFindPrepareAppSeparator(AppID app_id)

Scope System

Input Parameters

app_id	the application ID of the application separator

Output Parameters None

Result None

Comment The prepared application separator is automatically inserted into the Global Find item link.

6 GlobalFindPrepareGotoRecCmd

Purpose This function is called to create the command pointer for application launch event. The information inside the command pointer is determined by the clicked item.

Prototype void GlobalFindPrepareGotoRecCmd(ObjectID clicked_string_id,
void **cmd_ptr)

Scope System

Input Parameters

clicked_string_id	the object ID of the clicked string object on the Global Find Result

		screen. By knowing the string object ID, the required information of the selected item can be obtained
	cmd_ptr	Pointer reference to the command pointer

Output Parameters None

Result None

Comment None

7 GlobalFindPutDataToTable

Purpose This function is called to put the items in the Global Find item link to the table object in the Global Find Result screen

Prototype void GlobalFindPutDataToTable()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment The table object is not redrawn.

8 GlobalFindRestorePreviousFormStatus

Purpose Sometimes, menu or popup trigger objects are popup in the previous application screen. Therefore, *BatteryWarningRestorePreviousFormStatus* is called to close the popup menu object or popup trigger and the application screen can be returned to normal situation.

Prototype void GlobalFindgRestorePreviousFormStatus()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

9 GlobalFindRestoreRunningApp

Purpose This function is called to redraw the previous screen that is saved before Global Find Input screen is popup. It should be called after the Global Find process is cancelled.

Prototype void GlobalFindRestoreRunningApp()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment The previous screen (160 pixels x 160 pixels) is saved before popping up the Global Find Input screen.

10 GlobalFindScrollbarSetTable

Purpose This function is called to set data of the table again when the scrollbar is moved

Prototype void GlobalFindScrollbarSetTable(EvtType *Event)

Scope System

Input Parameters

Event	Pointer to EVT_SCROLLBAR_REPEAT or EVT_SCROLLBAR_SELECT events

Output Parameters None

Result None

Comment None

11 GlobalFindSearchData

Purpose This function is called to search the searching string in all applications

Prototype void GlobalFindSearchData()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

12 GlobalFindSearchText

Purpose This function provides an intensive algorithm to search a string in another string.

Prototype WORD GlobalFindSearchText(BYTE *short_string,
BYTE *whole_string)

Scope Application/System

Input Parameters

short_string	Pointer to the searching string
whole_string	Pointer to the string to be searched

Output Parameters None

Result The character position of the starting character in the string, which is to be search.

Comment None

13 GlobalFindSetMatchString

Purpose This function is called to set text of the STRING_GFIND_RESULT_MATCH string object. The text would be used to shown what the searching string is.

Prototype void GlobalFindSetMatchString(BYTE *matching_string)

Scope System

Input Parameters matching_string Pointer to the searching string

Output Parameters None

Result None

Comment None

14 GlobalFindSetupScrollbar

Purpose This function is called to setup the scrollbar of the Global Find Result screen when the screen is first loaded.

Prototype void GlobalFindSetupScrollbar()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

15 GlobalFindStringHandleEvent

Purpose This function is called to handle the clicking on the string objects in the table in the Global Find Result screen.

Prototype BOOLEAN GlobalFindStringHandleEvent(EvtType *Event)

Scope	System	
Input Parameters	Event	Pointer to the input event
Output Parameters	None	
Result	TRUE FALSE	if handled successfully if not handled
Comment	None	

Hookmgr.c Functions

1 IntHookMgr()

Purpose Execute the hooked routine

Prototype void IntHookMgr(MsgType *hook_msg)

Scope Internal

Input Parameters Hook_msg - carry the Int number in msgID and the input argument pointer in msgPBP

Output Parameters None

Result None

Comment None

HwInit.c Functions

1 BootUpInit

Purpose Init the hardware module

Prototype **BootUpInit()**

Scope OS

Input Parameters None

Output Parameters None
Result None
Comment None

init.c Functions

1 DummyOut

Purpose Dummy function
Prototype int DummyOut(const char *fmt, ...)
Scope OS
Input Parameters None

Output Parameters None
Result None
Comment None

2 InitAll

Purpose Init the software module
Prototype void InitAll()
Scope OS
Input Parameters None

Output Parameters None
Result None
Comment None

3 SysDisableDebug

Purpose Disable the debug message
Prototype void SysDisableDebug()
Scope OS

Input Parameters None

Output Parameters None
Result None
Comment The message is print to terminal through UART port

4 SysEnableDebug

Purpose Enable the debug message
Prototype void SysEnableDebug()
Scope OS
Input Parameters None

Output Parameters None
Result None
Comment The message is print to terminal through UART port

Inlay.c Functions

1 InlayCheckEnableStatus

Purpose This function is called to check whether the inlay region is enabled or not.

Prototype BOOLEAN InlayCheckEnableStatus ()

Scope Application/System

Input Parameters None

Output Parameters None

Result

TRUE	if inlay is enabled for clicking on
FALSE	if inlay is disabled for clicking on

Comment If inlay is enabled, then all inlay regions. Otherwise, the LEFT and RIGTH CUSTOMIZED DOTs are disabled

2 InlayGetClickedRegion

Purpose This function is called to check the clicked region by checking the coordinates of the pen.

Prototype SHORT InlayGetClickedRegion(SHORT xcoord, SHORT ycoord)

Scope System

Input Parameters

xcoord	The x-coordinate of the pen
ycoord	The y-coordinate of the pen

Output Parameters None

Result The number of the clicked region

INLAY_MENU	if clicked on the MENU region
INLAY_OK	if clicked on the OK region
INLAY_KEYBOARD	if clicked on the Keyboard ABC region
INLAY_EXIT	if clicked on the EXIT region
INLAY_MAIN_MENU	if clicked on the MAIN MENU region
INLAY_CALCULATOR	if clicked on the Keyboard 123 region
INLAY_LEFT	if clicked on the INLAY LEFT CUSTOMIZED DOT region
INLAY_RIGHT	if clicked on the INLAY RIGHT CUSTOMIZED DOT region

Comment None

3 InlayHandleEvent

Purpose This function is called to handle the pen actions and part of the selection of inlay regions.

Prototype BOOLEAN InlayHandleEvent(EvtType *Event)

Scope System

Input Parameters Event — Pointer to input event

Output Parameters None

Result TRUE — if handled successfully
FALSE — if not handled

Comment None

4 InlaySetEnableStatus

Purpose This function is called to set whether to enable or disable the inlay regions.

Prototype void InlaySetEnableStatus(BOOLEAN status)

Scope System

Input Parameters status — TRUE if enable
FALSE if disable

Output Parameters None

Result None

Comment None

intc.c Functions

1 ChkPt

Purpose Reserve

Prototype void ChkPt(UWORD a, UWORD b, UWORD c)

Scope Internal

Input Parameters n.a

Output Parameters n.a.

Result n.a.

Comment For debug purpose only

2 IntBuildPriorityMask

Purpose Reserve

Prototype Void IntBuildPriorityMask()

Scope Internal

Input Parameters n.a.

Output Parameters n.a.

Result n.a

Comment Obsolete in next version

3 IntClearHookAddr

Purpose Reserve

Prototype void IntClearHookAddr(UBYTE addr)

Scope Internal

Input Parameters n.a.

Output Parameters n.a.

Result n.a.

Comment Obsolete in next version

4 IntDisableGlobal

Purpose Reserve

Prototype void IntDisableGlobal()

Scope Internal

Input Parameters n.a.

Output Parameters n.a.

Result n.a.

Comment Obsolete in next version

5 IntEnableGlobal

Purpose Reserve

Prototype void IntEnableGlobal()

Scope Internal

Input Parameters n.a.

Output Parameters n.a.

Result n.a.

Comment Obsolete in next version

6 IntGetActive

Purpose Get the active status of the specific hook

Prototype UBYTE IntGetActive(USHORT hook_no)

Scope OS

Input Parameters Hook no the number of the hook function

Output Parameters None

Result The active status, TRUE if active

Comment None

7 IntHookISR

Purpose Hook an ISR function to a specific interrupt

Prototype USHORT IntHookISR(USHORT ISRID, USHORT int_no, UWORD offset)

Scope OS

Input Parameters

ISRID	ID of the ISR function
Int_no	Interrupt number
Offset	The function enter address (base addr = 0)

Output Parameters None

Result Hook number

Comment None

8 IntInit

Purpose Init the hook table and priority mask

Prototype void IntInit()

Scope OS

Input Parameters None

Output Parameters None

Result None

Comment None

9 IntMaskIntEnable

Purpose Mask the interrupt enable register

Prototype void IntMaskIntEnable(UWORD int_no)

Scope Internal

Input Parameters n.a.

Output Parameters n.a.

Result n.a.

Comment reserve

10 IntRunHookedISR

Purpose Execute a specific interrupt service routine(s)

Prototype void IntRunHookedISR(USHORT int_no, BYTE *arg)

Scope OS

Input Parameters

Int_no	intrrupt number
Arg	input argument pointer

Output Parameters None

Result None

Comment None

11 IntSearchFreeHookCell

Purpose Search for free hook cell

Prototype USHORT IntSearchFreeHookCell(USHORT int_no)

Scope Internal

Input Parameters Interrupt number

Output Parameters None

Result Hook number of free cell

Comment None

12 IntSearchLastHookCell

Purpose Search for last hook cell

Prototype USHORT IntSearchLastHookCell(USHORT int_no)

Scope Internal

Input Parameters Interrupt number

Output Parameters None

Result Hook number of last cell

Comment None

13 IntSearchNextPtr

Purpose Search for next hook cell

Prototype USHORT IntSearchNextPtr(USHORT hook_no)

Scope Internal

Input Parameters Hook_no hook number

Output Parameters None

Result Hook number of next cell

Comment None

14 IntSetActive

Purpose Set the Hook active

Prototype void IntSetActive(USHORT hook_no)

Scope OS

Input Parameters

Hook_no	hook number

Output Parameters None

Result None

Comment None

15 IntSetHookAddr

Purpose Set the page table and application ID of the hook address

Prototype void IntSetHookAddr(UBYTE addr, UWORD *ptable, AppID appid)

Scope OS

Input Parameters

Addr	logical address (0-255)
Ptable	point to the page table address
Appid	application ID

Output Parameters None

Result None

Comment None

16 IntSetPriority

Purpose Set the interrupt priority

Prototype void IntSetPriority(USHORT int_no, USHORT priority)

Scope OS

Input Parameters n.a.

Output Parameters n.a.

Result n.a.

Comment Reserve

17 IntUnhookISR

Purpose Unhook a hooked function according to the hook number

Prototype void IntUnhookISR(USHORT hook_no)

Scope OS

Input Parameters Hook_no hook number

Output Parameters None

Result None

Comment None

18 IntUnhookISRbyAppID

Purpose Unhook a hooked function according to the application ID

Prototype void IntUnhookISRbyAppID(USHORT appid)

Scope OS

Input Parameters App_ID application ID

Output Parameters None

Result None

Comment None

19 OtherGenExcHandle

Purpose Display the unexpected exception handling

Prototype void OtherGenExcHandle()

Scope Interrupt

Input Parameters None

Output Parameters None

Result None

Comment System stop after this function

20 OtherIntHighHandle

Purpose Display the unexpected exception handling

Prototype void OtherIntHighHandle()

Scope Interrupt

Input Parameters None

Output Parameters None

Result None

Comment System stop after this function

21 OtherIntLow1Handle

Purpose Display the unexpected exception handling

Prototype void OtherIntLow1Handle()

Scope Interrupt

Input Parameters None

Output Parameters None

Result None

Comment System stop after this function

22 OtherIntLow2Handle

Purpose Display the unexpected exception handling

Prototype void OtherIntLow2Handle()

Scope Interrupt

Input Parameters None

Output Parameters None

Result None

Comment System stop after this function

23 OtherIntLow3Handle

Purpose Display the unexpected exception handling

Prototype void OtherIntLow3Handle()

Scope Interrupt

Input Parameters None

Output Parameters None

Result None

Comment System stop after this function

24 OtherIntLow4Handle

Purpose Display the unexpected exception handling

Prototype void OtherIntLow4Handle()

Scope Interrupt

Input Parameters None

Output Parameters None

Result None

Comment System stop after this function

25 OtherIntLow5Handle

Purpose Display the unexpected exception handling

Prototype void OtherIntLow5Handle()

Scope Interrupt

Input Parameters None

Output Parameters None

Result None

Comment System stop after this function

26 PrintRA

Purpose Print the return addres

Prototype void PrintRA(UWORD a)

Scope OS

Input Parameters None

Output Parameters None

Result None

Comment Debug only

27 UtlbLoadExcHandle

Purpose Set TLB for load exception

Prototype void UtlbLoadExcHandle()

Scope Interrupt

Input Parameters None

Output Parameters None

Result None

Comment None

28 UtlbStoreExcHandle

Purpose Set TLB for store exception

Prototype void UtlbStoreExcHandle()

Scope Interrupt

Input Parameters None

Output Parameters None

Result None

Comment None

iocon.c Functions

1 CheckKey8

Purpose Check the record key when Interrupt

Prototype void CheckKey8()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

2 DisHotSynCheck

Purpose Disable the HotSync Key

Prototype void DisHotSynCheck()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

3 DisIOKeyInt

Purpose Disable all the General IO Port Interrupt

Prototype void DisIOKeyInt()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

4 DisIOKeyTimerInt

Purpose Disable General IO Key Timer Interrupt

Prototype void DisIOKeyTimerInt()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

5 DisPowerPinInt

Purpose Disable Power Key Interrupt

Prototype void DisPowerPinInt()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

6 DisPowerTimerInt

Purpose Disable Power Key Timer Interrupt

Prototype void DisPowerTimerInt()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

7 EnIOKeyInt

Purpose Enable Genernal IO key Interrupt

Prototype void EnIOKeyInt()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

8 EnIOKeyTimerInt

Purpose Enable General IO key Timer Interrupt

Prototype void EnIOKeyTimerInt(UWORD io_rep_time)

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

9 EnPowerPinInt

Purpose Enable Power Key Interrupt

Prototype void EnPowerPinInt()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

10 EnPowerTimerInt

Purpose Enable Power Key Timer Interrupt

Prototype void EnPowerTimerInt()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

11 HotSyncDet

Purpose Check the HotSync Key Press

Prototype void HotSyncDet()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

12 HotSynDebounCheck

Purpose If the HotSync key internal, check the deboun
Prototype void HotSynDebounCheck()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

13 HotSynPortMoveOut

Purpose Set the HotSync Key port without press
Prototype void HotSynPortMoveOut()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

14 IOCtrl

Purpose Check which Short-cut Key have been pressed
Prototype void IOCtrl()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

15 IOKeyCtrlInit

Purpose Init the Short-cut key and Power key
Prototype void IOKeyCtrlInit()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

16 IOKeyRepeat

Purpose Check the key press repeat
Prototype void IOKeyRepeat()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

17 KbPowerCtrl

Purpose External Driver that can power ON / OFF the pda
Prototype void KbPowerCtrl()
Scope External Driver
Input Parameters None

Output Parameters None
Result None
Comment None

18 KeyDown

Purpose Check the KeyDown Key
Prototype void KeyDown()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

19 KeyStop

Purpose Check the KeyStop Key
Prototype void KeyStop()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

20 KeyUp

Purpose Check the KeyUp Key
Prototype void KeyUp()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

21 OnPdaTxRecKey

Purpose Check the Record Key status that can power on the Pda

Prototype void OnPdaTxRecKey()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

22 PlaySoundKey

Purpose Check the Pda status On / Off to decide active the Play Sound Application

Prototype void PlaySoundKey()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

23 PowerCtrl

Purpose When Power Key press, this function will set a timer to deboun check

Prototype void PowerCtrl()

Scope Internal

Input Parameters None

Output Parameters None

Result None

Comment None

24 PowerRepeat

Purpose This function will confirm the Power Key have been pressed
Prototype void PowerRepeat()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

25 ReadKey8

Purpose Read the Record Key press or not
Prototype void ReadKey8()
Scope Internal
Input Parameters None
Output Parameters None
Result None
Comment None

26 ReadPlaySoundKey

Purpose Read the play sound key press or not
Prototype void ReadPlaySoundKey()
Scope Internal
Input Parameters None
Output Parameters None
Result None
Comment None

27 ResetIOIntFlag

Purpose Clear the general IO key interrupt flag
Prototype void ResetIOIntFlag()
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

28 ResetPowerFlag

Purpose Clear the Power key interrupt flag
Prototype void ResetPowerFlag()
Scope Internal
Input Parameters None
Output Parameters None
Result None
Comment None

ioconmsg.c Functions

1 IOKeyCtrlMgr

Purpose Transfer which key is pressed to the system
Prototype **void IOKeyCtrlMgr(MsgType* message)**
Scope Internal
Input Parameters None

Output Parameters None
Result None
Comment None

Kernel.c Functions

1 ShellHandleEvent

Purpose This function is the heart of the OS. It is used to handle the EVT_APP_LAUNCH and EVT_POWER_OFF. Therefore, when the power is shut down or application is being switched, the ShellHandleEvent handles the event.

Prototype BOOLEAN ShellHandleEvent(EvtType *Event)

Scope System

Input Parameters

Event	input event

Output Parameters None

Result

TRUE	if event is handled
FALSE	if event is not handled

Comment None

2 ShellHandleEventWhileLoop

Purpose It is a function to gather enough information (battery level, application that is being launched, battery door status) after the PDA is powered on after powering off.

Prototype BOOLEAN ShellHandleEventWhileLoop()

Scope System

Input Parameters None

Output Parameters None

Result

TRUE	if event is handled
FALSE	if event is not handled

Comment None

3 SysBuildAllIntPageTable

Purpose Rebuild the interrupt page tables for third-party installed interrupt handlers when system has been soft-reset.

Prototype void SysBuildAllIntPageTable()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment The interrupt vector information is retrieved from interrupt hook table.

4 SysBuildPageTable

Purpose Rebuild application page table if system has been soft-reset.

Prototype BOOLEAN SysBuildPageTable(USHORT index)

Scope System

Input Parameters

index	Index in Application Registry

Output Parameters None

Result

TRUE	Success
FALSE	Not enough memory

Comment Page table is built according to the start block of the application in the MAT and the links in the MAT.

5 SysCall

Purpose Jump to a system call

Prototype void SysCall(UWORD code)

Scope System/Application

Input Parameters

code	System call number

Output Parameters None

Result None

Comment None

6 SysCallGetCode

Purpose Get the system call code

Prototype UWORD SysCallGetCode()

Scope System

Input Parameters None

Output Parameters None

Result The code

Comment This function has been obsolete.

7 SysCallHooked

Purpose Jump to a hooked system call

Prototype void SysCallHooked(UWORD code)

Scope System/Application

Input Parameters code System call code number

Output Parameters None

Result None

Comment None

8 SysCheckModemExist

Purpose Get the existence of modem

Prototype BOOLEAN SysCheckModemExist()

Scope System/Application

Input Parameters None

Output Parameters None

Result

TRUE	Modem is present
FALSE	Modem is absent

Comment None

9 SysClearAppData

Purpose Clear the allocated memory when application terminate

Prototype BOOLEAN SysClearAppData(AppID app)

Scope System

Input Parameters

app	Application ID

Output Parameters None

Result Always return TRUE

Comment 2 types of memory will be released to system:
1. application data region (.data, .bss)
2. dynamic memory data region (memory allocated by *qmalloc()*)

10 SysClearAppMode1PageTable

Purpose Clear data allocated by *SysRunMode1App()* and restore previous application settings

Prototype void SysClearAppMode1PageTable(AppID app)

Scope System

Input Parameters

app	Application ID

Output Parameters None

Result None

Comment This function is called after Mode1 application exit, all memory of the Mode1 application will be released and previous application's page table will be restored and continue to execute.

11 SysClearDBData

Purpose Delete all database

Prototype BOOLEAN SysClearDBData()

Scope System

Input Parameters None

Output Parameters None

Result Always return TRUE

Comment None

12 SysClearInterruptData

Purpose Clear the allocated memory when interrupt routine terminate.

Prototype BOOLEAN SysClearInterruptData(AppID app)

Scope System

Input Parameters app Interrupt hander ID

Output Parameters None

Result Always return TRUE

Comment This function has been obsolete.

13 SysClearMode1AppData

Purpose Clear the allocated memory when Mode1 application terminate.

Prototype BOOLEAN SysClearMode1AppData(AppID app)

Scope System

Input Parameters app Application ID

Output Parameters None

Result Always return TRUE

Comment Refer to *SysClearAppData()*.

14 SysClearSysData

Purpose Release all dynamic memory allocated by system.

Prototype BOOLEAN SysClearSysData()

Scope System

Input Parameters None

Output Parameters None

Result Always TRUE

Comment All *pmalloc()* chunks will be cleared.

15 SysDefaultSyscall

Purpose Default routine for unhooked system call

Prototype void SysDefaultSyscall()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment This is the default rountine for hookable system calls.

16 SysGetActiveAppID

Purpose Get active application's Application ID.

Prototype AppID SysGetActiveAppID()

Scope System/Application

Input Parameters None

Output Parameters None

Result Application ID of the active application

Comment GetActiveAppID() will return 1 (Application ID of 'MainMenu') if no active application, GetRealActiveAppID() will return 0 (no active application) instead.

17 SysGetAppAttribute

Purpose Get the application attributes.

Prototype BOOLEAN SysGetAppAttribute(AppID app_id, UBYTE query_bit, UWORD *reserved)

Scope

app_id	Application ID

Input Parameters

query_bit	Attribute to be queried
reserved	Reserved, must be NULL

Output Parameters None

Result

TRUE	The attribute is set
TRUE	The attribute is not set

Comment Available attribute:

BIT_RUN	Active application
BIT_TYPE	Device driver
BIT_ALARM	Has alarm feature
BIT_FIND	Has global find feature
BIT_RAM_PRG	Third-party program
BIT_FLASH_PRG	Build-in program
BIT_ALL_APP	TRUE for all applications

18 SysGetAppEntryIndex

Purpose Get application's index in Application Registry.

Prototype BOOLEAN SysGetAppEntryIndex(UWORD app_id, USHORT *index)

Scope System/Application

Input Parameters

app_id	Application ID

Output Parameters

index	Index in Application Registry

Result

TRUE	Success
FALSE	Application not found

Comment None

19 SysGetAppID

Purpose Get application's Application ID.

Prototype BOOLEAN SysGetAppID(BYTE *app_name, AppID *app_id)

Scope System/Application

Input Parameters

app_name	NULL terminated string of the application's name

Output Parameters

app_id	Application ID

Result

TRUE	Success
FALSE	Application not found

Comment Application is unique identified by the application name. The Application ID served as a handle to access the application. Maximum length of application name is 20 characters (including the NULL character).

20 SysGetAppIDFromIndex

Purpose Get Application ID by a index number

Prototype AppID SysGetAppIDFromIndex(USHORT index, UBYTE type)

Scope System

Input Parameters

index	Index to count
type	Application attributes take to count

Output Parameters None

Result Application ID or 0xffff if not found

Comment The maximum value of index can be get from *SysTotalInstalledApp()*. Refer to *SysGetAppAttribute()* for valid types.

21 SysGetAppName

Purpose Get application's name.

Prototype BOOLEAN SysGetAppName(AppID app, BYTE *app_name)

Scope System

Input Parameters

app	Application ID

Output Parameters

app_name	Application name

Result

TRUE	Success
FALSE	Application not found

Comment Caller need to pre-allocate 20 bytes memory to app_name.

22 SysGetAppSize

Purpose Get application size

Prototype BOOLEAN SysGetAppSize(AppID app, UWORD *size)

Scope System/Application

Input Parameters

app	Application ID

Output Parameters	size	Size of the application, in the unit of 4K-byte
Result	TRUE FALSE	Success Application not found
Comment	None	

23 SysGetAppStatus

Purpose	Get the resource location of the application	
Prototype	BOOLEAN SysGetAppStatus(UWORD app_id, UWORD *res_addr)	
Scope	System	
Input Parameters	app_id	Application ID
Output Parameters	res_addr	Memory location of the resource
Result	TRUE FALSE	Success Application not found
Comment	None	

24 SysGetIndexAppID

Purpose	Get the application ID by index in Application Registry	
Prototype	AppID SysGetIndexAppID(USHORT index)	
Scope	System	
Input Parameters	index	Index in Application Registry
Output Parameters	None	
Result	Application ID or 0xffff if not found	
Comment	None	

25 SysGetInstallAppHandle

Purpose Get the pointer to access the third-party installed application's registry

Prototype void SysGetInstallAppHandle(InstallAppInf **handle)

Scope System/Application

Input Parameters None

Output Parameters handle Handle to access the registry

Result None

Comment This function is targeted for application installer to install application to system.

26 SysGetInstallAppIconHandle

Purpose Get the pointer to access the third-party installed application's icon registry

Prototype void SysGetInstallAppIconHandle(BYTE **handle)

Scope System/Application

Input Parameters None

Output Parameters handle Handle to access the icon registry

Result None

Comment This function is targeted for application installer to install application to system.

27 SysGetOSVersionNo

Purpose Get VT-OS version number

Prototype void SysGetOSVersionNo(BYTE **ver_no)

Scope Application

Input Parameters None

Output Parameters ver_no VT-OS version number

Result None

Comment Memory is allocated to ver_no internally, caller should call *qfree()* to free the buffer

28 SysGetPageTable

Purpose Get the application's page table

Prototype BOOLEAN SysGetPageTable(UWORD app, UWORD **page_table)

Scope System

Input Parameters app Application ID

Output Parameters page_table Application's page table

Result TRUE Success
FALSE Application not found

Comment page_table is subjected to be read only, caller should not modify the content of page_table

29 SysGetPrevAppID

Purpose Get previous active application's Application ID.

Prototype AppID SysGetPrevAppID()

Scope System/Application

Input Parameters None

Output Parameters None

Result Previous active application's Application ID

Comment None

30 SysGetRealActiveAppID

Purpose Get active application's Application ID

Prototype AppID SysGetRealActiveAppID()

Scope System/Application

Input Parameters None

Output Parameters None

Result Active application's Application ID or 0 if not active application

Comment Refer to *SysGetActiveAppID()*.

31 SysGetRegistryData

Purpose Get application information in Application Registry.

Prototype BOOLEAN SysGetRegistryData(USHORT index, REGISTRY *reg_data)

Scope System

Input Parameters

index	Index in Application Registry

Output Parameters

reg_data	Application information

Result

TRUE	Success
FALSE	Application not found

Comment None

32 SysHookIntHandler

Purpose Hook an interrupt handler.

Prototype USHORT SysHookIntHandler(USHORT int_no, UWORD *start_addr)

Scope System

Input Parameters

int_no	Interrupt number

	start_addr	Memory location of the hooked routine

Output Parameters None

Result The hook number of 0xffff if fail

Comment None

33 SysHookSyscall

Purpose Hook a system call

Prototype void SysHookSyscall(AppID app, UWORD sc_no, UWORD *address)

Scope Application

Input Parameters

app	Application ID of the application that the system call routine resident
sc_no	System call number
address	Memory location of system call routine in the application

Output Parameters None

Result None

Comment None

34 SysInit

Purpose Initialize Application Registry and interrupt hook table

Prototype void SysInit()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment Only call when system is hard-reset

35 SysLoadAppMode1PageTable

Purpose Load the page table for Mode1 application

Prototype BOOLEAN SysLoadAppMode1PageTable(USHORT index)

Scope System

Input Parameters

index	Index in Application Registry

Output Parameters None

Result

TRUE	Success
FALSE	Fail

Comment None

36 SysLoadAppPageTable

Purpose Load page table for application

Prototype BOOLEAN SysLoadAppPageTable(AppID app_id)

Scope System

Input Parameters

app_id	Application ID

Output Parameters None

Result

TRUE	Success
FALSE	Application not found

Comment None

37 SysLoadInterruptPageTable

Purpose Load page table for interrupt handler

Prototype BOOLEAN SysLoadInterruptPageTable(AppID app_id)

Scope System

Input Parameters

app_id	Application ID

Output Parameters None

Result

TRUE	Success
FALSE	Application not found

Comment This function has been obsolete.

38 SysNewApp

Purpose Register new application to Application Registry

Prototype Err SysNewApp(BYTE app_name[20], UBYTE attr, UWORD prg_addr, UBYTE run_spec, UWORD *res_addr, UWORD data_size, UWORD *page_table, AppID *app)

Scope System

Input Parameters

app_name	Application name, NULL terminated
attr	Application attribute
prg_addr	MAT entry of the first block of the application
run_spec	Application run clock speed
res_addr	Address of resource in the program image
data_size	Size of the .bss and .data region

Output Parameters

page_table	Page table of the application
app	Application ID

Result

TRUE	Success
ERR_SYS_APP_EXIST	Application already existed
ERR_SYS_REG_FULL	Registry full, cannot install any more application

Comment This function only call by MMU, to install application, use *MemoryInstallProg().*

39 SysNextAppID

Purpose Get next available Application ID

Prototype AppID SysNextAppID()

Scope System

Input Parameters None

Output Parameters None

Result Next available Application ID

Comment None

40 SysRemoveApp

Purpose Remove an application

Prototype BOOLEAN SysRemoveApp(AppID app_id)

Scope System/Application

Input Parameters

app_id	Application ID

Output Parameters None

Result

TRUE	Success
FALSE	Application not found

Comment None

41 SysRestoreActiveUserDMStatus

Purpose Restore the dynamic memory status when Mode1 application terminate

Prototype void SysRestoreActiveUserDMStatus()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

42 SysRunApp

Purpose Launch an application

Prototype BOOLEAN SysRunApp(USHORT app_id, UWORD cmd, UWORD *cmd_ptr)

Scope System

Input Parameters

app_id	Application ID
cmd	Launch code
cmd_ptr	Data passed to application

Output Parameters None

Result

TRUE	Success
FALSE	Application not found

Comment None

43 SysRunHookedIntHandler

Purpose Launch all hooked handler for specify interrupt

Prototype void SysRunHookedIntHandler(USHORT int_no)

Scope System

Input Parameters

int_no	Interrupt number

Output Parameters None

Result None

Comment None

44 SysRunMode1App

Purpose Launch Mode1 application

Prototype BOOLEAN SysRunMode1App(AppID app_id, WORD cmd, void *cmd_ptr)

Scope System/Application

Input Parameters

app_id	Application ID
cmd	Launch code
cmd_ptr	Data pass to application

Output Parameters None

Result

TRUE	Success
FALSE	Application not found

Comment LAUNCH_CMD_FIND is classified into Mode1. In Mode1 mode, application cannot display any user-interface in the screen and cannot get any action from the user.

45 SysSaveActiveUserDMStatus

Purpose Save the dynamic memory status prior to execute Mode1 application

Prototype void SysSaveActiveUserDMStatus()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

46 SysSearchFreeASID

Purpose Get a free ASID in TLB

Prototype UWORD SysSearchFreeASID()

Scope System

Input Parameters None

Output Parameters None

Result Free ASID

Comment None

47 SysSearchFreeHookCell

Purpose Get next available hook number

Prototype USHORT SysSearchFreeHookCell(USHORT int_no)

Scope System

Input Parameters int_no Interrupt number

Output Parameters None

Result Available hook number

Comment This function has been obsolete and replaced by *IntSearchFreeHookCell().*

48 SysSearchLastHookCell

Purpose Get last hook routine for specify interrupt

Prototype USHORT SysSearchLastHookCell(USHORT int_no)

Scope System

Input Parameters int_no Interrupt number

Output Parameters None

Result The cell number

Comment This function has been obsolete and replaced by *IntSearchLastHookCell()*.

49 SysSearchNextPtr

Purpose Search the previous linked hook number

Prototype USHORT SysSearchNextPtr(USHORT hook_no)

Scope System

Input Parameters hook_no Hook number

Output Parameters None

Result The previous hook number of hook_no

Comment This function has been obsolete and replaced by *IntSearchNextPtr()*.

50 SysSetActiveAppID

Purpose Set active application

Prototype SysSetActiveAppID(AppID app_id)

Scope System

Input Parameters app_id Application ID

Output Parameters None

Result None

Comment Application attribute BIT_RUN will be set

51 SysSetPageTable

Purpose Set application's default page table

Prototype BOOLEAN SysSetPageTable(UWORD app, UWORD *page_table)

Scope System

Input Parameters

app	Application ID
page_table	Page table pointer

Output Parameters None

Result

TRUE	Success
FALSE	Application not found

Comment None

52 SysTotalInstalledApp

Purpose Get number of application has specify attributes

Prototype USHORT SysTotalInstalledApp(UBYTE type)

Scope System/Application

Input Parameters

type	Application attributes

Output Parameters None

Result Number of application has the specify attributes

Comment Refer to *SysGetAppAttribute()* for available attribues.

53 SysUnhookIntHandler

Purpose Unhook a interrupt handler

Prototype void SysUnhookIntHandler(USHORT hook_no)

Scope System

Input Parameters hook_no Hook number

Output Parameters None

Result None

Comment None

54 SysUnhookSyscall

Purpose Un-hook a system call

Prototype void SysUnhookSyscall(UWORD sc_no)

Scope Application

Input Parameters sc_no System call number

Output Parameters None

Result None

Comment None

55 SysUnHookSyscallByAppID

Purpose Un-hook all system calls belong to a specify application

Prototype void SysUnHookSyscallByAppID(AppID appid)

Scope Application

Input Parameters app_id Application ID

Output Parameters None

Result None

Comment None

Lcdapi.c Functions

1 BmpQ1ToBmpQ4

Purpose To convert a 1-bit quantised bitmap to a 4-bit quantised bitmap

Prototype void BmpQ1ToBmpQ4(BitmapTemplate *bitmap_struct1,
BitmapTemplate *bitmap_struct2)

Scope Application/System

Input Parameters bitmap_struct1 Pointer to an input BitmapTemplate with 1-bit quantised bitmap

Output Parameters bitmap_struct2 Pointer to an output BitmapTemplate with 4-bit quantised bitmap

Result None

Comment None

2 BmpQ2ToBmpQ4

Purpose To convert a 2-bit quantised bitmap to a 4-bit quantised bitmap

Prototype void BmpQ2ToBmpQ4(BitmapTemplate *bitmap_struct1,
BitmapTemplate *bitmap_struct2)

Scope Application/System

Input Parameters bitmap_struct1 Pointer to an input BitmapTemplate with 2-bit quantised bitmap

Output Parameters bitmap_struct2 Pointer to an output BitmapTemplate with 4-bit quantised bitmap

Result None

Comment None

3 BmpQ4ToBmpQ1

Purpose To convert a 4-bit quantised bitmap to a 1-bit quantised bitmap

Prototype void BmpQ4ToBmpQ1(BitmapTemplate *bitmap_struct1,
BitmapTemplate *bitmap_struct2)

Scope Application/System

Input Parameters

bitmap_struct1	Pointer to an input BitmapTemplate with 4-bit quantised bitmap

Output Parameters

bitmap_struct2	Pointer to an output BitmapTemplate with 1-bit quantised bitmap

Result None

Comment None

4 FontToBmp

Purpose To convert a 4-bit quantised bitmap to a 1-bit quantised bitmap

Prototype BitmapTemplate *FontToBmp(BYTE character, BYTE color,
BYTE bg_color, BYTE font)

Scope Application/System

Input Parameters

character	ASCII code of the character that is required to convert
color	Color of the font
bg_color	Background color of the font
font	font size

Output Parameters None

Result it returns a BitmapTemplate pointer pointing to the bitmap that is converted from a font character

Comment None

5 LcdCheckInsertPt

Purpose To check whether the insert point is within the input bounds or not

Prototype BOOLEAN LcdCheckInsertPt(ObjectBounds bounds)

Scope Application/System

Input Parameters

bounds	the bounds that u want to check

Output Parameters None

Result

TRUE	if insert point is within the bounds
FALSE	if insert point is outside the bounds

Comment None

6 LcdDrawArc

Purpose To draw arc

Prototype

```
void LcdDrawArc(SHORT xcoord1, SHORT ycoord1, SHORT radius,
                BYTE style, BYTE color, BYTE bg_color,
                BYTE fill, BYTE arc_location)
```

Scope Application/System

Input Parameters

xcoord1	x-coordinate of the centre
ycoord1	y-coordinate of the centre
radius	the radius
style	line style
color	color of the drawn line
bg_color	background of the arc
fill	whether to fill the arc or not
arc_location	the location of the arc
	QUARTER_1
	QUARTER_2
	QUARTER_3
	QUARTER_4

Output Parameters None

Result None

Comment None

7 LcdDrawBitmap

Purpose To draw an image (bitmap) on a particular region

Prototype void LcdDrawBitmap(BitmapTemplate *bitmap_struct,
BOOLEAN invert)

Scope Application/System

Input Parameters None

Output Parameters

bitmap_struct	Pointer to a BitmapTemplate structure that holds the bounds and the bitmap that is to be drawn
invert	whether the bitmap is drawn inversely or not

Result None

Comment None

8 LcdDrawBlinkInsertPt

Purpose To draw blinking insert point on the screen

Prototype void LcdDrawBlinkInsertPt()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

9 LcdDrawBox

Purpose To draw blinking insert point on the screen

Prototype void LcdDrawBox(ObjectBounds *bounds, BYTE color,
BYTE bg_color, BYTE fill)

Scope Application/System

Input Parameters

bounds	Pointer to bounds structure that holds the bounds of the box
color	color of box
bg_color	background of the box
fill	whether to fill the box or not

Output Parameters None

Result None

Comment None

10 LcdDrawCharSpace

Purpose To draw a character spacing

Prototype void LcdDrawCharSpace(SHORT xcoord1, SHORT ycoord1,
BYTE color, BYTE font)

Scope Application/System

Input Parameters

xcoord1	the x-coordinate of left-top corner of the space
ycoord1	the y-coordinate of left-top corner of the space
color	color of the space
font	font size

Output Parameters None

Result None

Comment None

11 LcdDrawCharSpaceS

Purpose To draw a character spacing while the background is transparent

Prototype void LcdDrawCharSpaceS(SHORT xcoord1, SHORT ycoord1,
BYTE color, BYTE font)

Scope Application/System

Input Parameters

xcoord1	the x-coordinate of left-top corner of the space
ycoord1	the y-coordinate of left-top corner of the space
color	color of the space
font	font size

Output Parameters None

Result None

Comment None

12 LcdDrawCharacter

Purpose To draw a character on the screen

Prototype void LcdDrawCharacter(SHORT xcoord, SHORT ycoord,
BYTE character, BYTE color,
BYTE bg_color, BYTE font)

Scope Application/System

Input Parameters

xcoord	the x-coordinate of left-top corner of the character
ycoord	the y-coordinate of left-top corner of the character
character	the ASCII code of the character
color	color of the character
bg_color	the background color of the character
font	font size

Output Parameters None

Result None

Comment None

13 LcdDrawCharacterS

Purpose To draw a character on the screen while the background is transparent

Prototype
```
void LcdDrawCharacterS(SHORT xcoord, SHORT ycoord,
                       BYTE character, BYTE color,
                       BYTE bg_color, BYTE font)
```

Scope Application/System

Input Parameters

xcoord	the x-coordinate of left-top corner of the character
ycoord	the y-coordinate of left-top corner of the character
character	the ASCII code of the character
color	color of the character
bg_color	the background color of the character
font	font size

Output Parameters None

Result None

Comment None

14 LcdDrawEllipse

Purpose To draw ellipse

Prototype
```
void LcdDrawEllipse(SHORT xcoord1, SHORT ycoord1,
                    SHORT major_radius, SHORT minor_radius,
                    BYTE color, BYTE bg_color, BYTE fill)
```

Scope Application/System

Input Parameters

xcoord1	the x-coordinate of the centre of the ellipse
ycoord1	the y-coordinate of the centre of the ellipse
major_radius	major radius
minor_radius	minor radius
color	color of line of the ellipse
bg_color	the background color of the ellipse
fill	whether to fill the ellipse or not

Output Parameters None

Result None

Comment None

15 LcdDrawFixedString

Purpose To draw a label or string without multiple lines

Prototype void LcdDrawFixedString(ObjectBounds *bounds, BYTE *string, BYTE color, BYTE bg_color, BYTE font, BOOLEAN dotdot, SHORT margin)

Scope Application/System

Input Parameters

bounds	Pointer to bounds structure of the string
string	Pointer to string to be drawn
color	color of the text
bg_color	background color of the text
font	font size
dotdot	whether dots are put to the display or not if the string cannot be fully shown within the input bounds
margin	the left and right margin of the string

Output Parameters None

Result None

Comment None

16 LcdDrawFixedStringS

Purpose To draw a label or string without multiple lines while the background is transparent

Prototype void LcdDrawFixedStringS(ObjectBounds *bounds, BYTE *string, BYTE color, BYTE bg_color, BYTE font, BOOLEAN dotdot, SHORT margin)

Scope Application/System

Input Parameters

bounds	Pointer to bounds structure of the string
string	Pointer to string to be drawn
color	color of the text

	bg_color	background color of the text
	font	font size
	dotdot	whether dots are put to the display or not if the string cannot be fully shown within the input bounds
	margin	the left and right margin of the string

Output Parameters None

Result None

Comment None

17 LcdDrawHoriLine

Purpose To draw a horizontal line on the screen

Prototype void LcdDrawHoriLine(SHORT xcoord1, SHORT xcoord2, SHORT ycoord1, BYTE style, BYTE color, BYTE bg_color)

Scope Application/System

Input Parameters

xcoord1	x-coordinate of the starting point of the line
xcoord2	x-coordinate of the end point of the line
ycoord1	y-coordinate of the horizontal
style	the line style DOTTED_LINE NON_DOTTED_LINE
color	color of the line
bg_color	background color of the line

Output Parameters None

Result None

Comment xcoord1 must be smaller than xcoord2

18 LcdDrawInsertPt

Purpose To draw insert point on the screen

Prototype void LcdDrawInsertPt(SHORT xcoord, SHORT ycoord)

Scope Application/System

Input Parameters

xcoord	x-coordinate of the insert point
ycoord	y-coordinate of the insert point

Output Parameters None

Result None

Comment None

19 LcdDrawLine

Purpose To draw line on the screen

Prototype void LcdDrawLine(SHORT xcoord1, SHORT ycoord1,
SHORT xcoord2, SHORT ycoord2,
BYTE thick, BYTE style, BYTE color,
BYTE bg_color)

Scope Application/System

Input Parameters

xcoord1	x-coordinate of the starting point
ycoord1	y-coordinate of the starting point
xcoord2	x-coordinate of the end point
ycoord2	y-coordinate of the end point
thick	thickness of the line
style	style of the line
color	color of the line
bg_color	background color of the line

Output Parameters None

Result None

Comment None

20 LcdDrawVariableString

Purpose To draw multiple lines of text for a field object

Prototype void LcdDrawVariableString(Field *field_ptr)

Scope Application/System

Input Parameters

field_ptr	Pointer to field object

Output Parameters None

Result None

Comment None

21 LcdDrawVertLine

Purpose To draw vertical line

Prototype void LcdDrawVertLine(SHORT xcoord1, SHORT ycoord1,
SHORT ycoord2, BYTE style,
BYTE color, BYTE bg_color)

Scope Application/System

Input Parameters

xcoord1	x-coordinate of the vertical line
ycoord1	y-coordinate of the starting point of the line
ycoord2	y-coordinate of the end point of the line
style	style of the line
color	color of the line
bg_color	background color of the line

Output Parameters None

Result None

Comment ycoord1 must not be larger than ycoord2

22 LcdEnableInsertPt

Purpose To enable or disable insert point

Prototype void LcdEnableInsertPt(BOOLEAN enable, SHORT xcoord, SHORT ycoord, BYTE font)

Scope Application/System

Input Parameters

enable	TRUE if enable FALSE if disable
xcoord	the x-coordinate of the top-left corner of the insert point
ycoord	the y-coordinate of the bottom-right corner of the insert point
font	font size

Output Parameters None

Result None

Comment Insert point is drawn or erased at once after the function is called successfully

23 LcdEraseRegion

Purpose To erase a specified region on the screen

Prototype void LcdEraseRegion(ObjectBounds *bounds)

Scope Application/System

Input Parameters

bounds	Pointer to a bounds structure that specified the bounds of the erased region

Output Parameters None

Result None

Comment None

24 LcdGetBitmap

Purpose To get an image (bitmap) with a specified bounds

Prototype void LcdGetBitmap(BitmapTemplate *save_behind)

Scope Application/System

Input Parameters

save_behind	Pointer to BitmapTemplate structure that holds the specified bounds of the region that is wanted to be obtained

Output Parameters None

Result None

Comment None

25 LcdGetInsertPt

Purpose To get an image (bitmap) with a specified bounds

Prototype void LcdGetInsertPt(BOOLEAN *enable, SHORT *xcoord, SHORT *ycoord, BYTE *font)

Scope Application/System

Input Parameters None

Output Parameters

enable	Pointer to BOOLEAN variable that shows the insert point is enabled or not
xcoord	Pointer to x-coordinate of the left-top corner of the insert point
ycoord	Pointer to y-coordinate of the left-top corner of the insert point
font	Pointer to font size

Result None

Comment None

26 LcdGetOrientation

Purpose To get the orientation of the LCD display

Prototype BYTE LcdGetOrientation()

Scope Application/System

Input Parameters None

Output Parameters None

Result

0	not rotated
1	rotated 90 degrees

Comment None

27 LcdGetPixel

Purpose To get the color of a particular pixel

Prototype void LcdGetPixel(SHORT xcoord1, SHORT ycoord1, BYTE *color)

Scope Application/System

Input Parameters

xcoord1	x-coordinate of the pixel
ycoord1	y-coordinate of the pixel

Output Parameters

color	Pointer to the color information of the pixel

Result None

Comment None

28 LcdInvertBox

Purpose To invert drawing of a particular region

Prototype void LcdInvertBox(ObjectBounds *bounds)

Scope Application/System

Input Parameters

bounds	Pointer to bounds structure that is the region to be inverted in color

Output Parameters None

Result None

Comment None

29 LcdInvertHoriLine

Purpose To invert a line drawing

Prototype void LcdInvertHoriLine(SHORT xcoord1, SHORT xcoord2, SHORT ycoord1)

Scope Application/System

Input Parameters

xcoord1	x-coordinate of the starting point of the horizontal line
xcoord2	x-coordinate of the end point of the horizontal line
ycoord1	y-coordinate of the horizontal line

Output Parameters None

Result None

Comment None

30 LcdMsgInsertPt

Purpose To append message to message queue for updating the insert point

Prototype void LcdMsgInsertPt()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

31 LcdRot

Purpose To set the orientation of the LCD drawing and the pen orientation

Prototype void LcdRot(BYTE rot)

Scope Application/System

Input Parameters

rot	rotation	
	0	not rotated
	1	rotated 90 degrees

Output Parameters None

Result None

Comment None

32 LcdSetOrientation

Purpose To change the orientation of the LCD screen and the pen tapping

Prototype void LcdSetOrientation(BYTE rot)

Scope Application/System

Input Parameters

rot	rotation	
	0	not rotated
	1	rotated 90 degrees

Output Parameters None

Result None

Comment The screen is redrawn

33 LcdSetOrientationMsg

Purpose To append message to message queue to inform the change of the orientation of the LCD drawing

Prototype void LcdSetOrientationMsg(MsgType *msg)

Scope Application/System

Input Parameters

msg	Pointer to message structure

Output Parameters None

Result None

Comment None

34 LcdSetPixel

Purpose To set the color of a particular of pixel

Prototype void LcdSetPixel(SHORT xcoord1, SHORT ycoord1, BYTE color)

Scope Application/System

Input Parameters

xcoord1	x-coordinate of the pixel to be set
ycoord1	y-coordinate of the pixel to be set
color	color of the pixel

Output Parameters None

Result None

Comment None

35 LcdShowExceptionCode

Purpose To show exception code on the screen

Prototype void LcdShowExceptionCode(UWORD addr, UWORD code)

Scope Application/System

Input Parameters	addr	Exception in address
	code	Exception code
Output Parameters	None	
Result	None	
Comment	None	

36 LcdTextboxDrawString

Purpose	To draw string in a textbox object	
Prototype	void LcdTextboxDrawString(Textbox *tb_ptr)	
Scope	Application/System	
Input Parameters	tb_ptr	Pointer to textbox object
Output Parameters	None	
Result	None	
Comment	string is drawn at once on the screen	

lcddrv.c

1 LcdClr

Purpose	Clear the LCD memory
Prototype	void LcdClr(void)
Scope	OS
Input Parameters	None

Output Parameters None

Result None

Comment None

2 LcdInit

Purpose Init the LCD module

Prototype void LcdInit(void)

Scope OS

Input Parameters None

Output Parameters None

Result None

Comment None

3 LcdTurnOff

Purpose Turn off the LCD

Prototype void LcdTurnOff(void)

Scope OS

Input Parameters None

Output Parameters None

Result None

Comment None

4 LcdTurnOn

Purpose Turn on the LCD

Prototype void LcdTurnOn(void)

Scope OS

Input Parameters None

Output Parameters None

Result None

Comment None

5 PixelRead

Purpose Read the color of pixel at the specified location

Prototype BYTE PixelRead(SHORT xcoord, SHORT ycoord)

Scope Internal

Input Parameters Cartesian coordinate

Output Parameters None

Result Color

Comment If the location is outside the screen region, return 0

6 PixelWrite

Purpose Write the color of pixel at the specified location

Prototype void PixelWrite(SHORT xcoord, SHORT ycoord, BYTE color)

Scope Internal

Input Parameters Cartesian coordinate and color

Output Parameters None

Result None

Comment If the location is outside the screen region, do nothing

7 WordBodyPack

Purpose Pack two words into one refering to the merge point assuming former word always ends packing to its tail and latter word always starts packing from its head

Prototype UWORD WordBodyPack(UWORD word1, UWORD word2, BYTE merge_pt)

Scope Internal

Input Parameters Two words and merge point

Output Parameters Packed word

Result None

Comment p(0:31) = w1(31-mp*bg:31) || w2(0:31-mp*bg)

8 WordHead1Pack

Purpose Pack two words into one refering to the merge point assuming latter word always starts packing from its head

Prototype UWORD WordHead1Pack(UWORD word1, UWORD word2, BYTE merge_pt)

Scope Internal

Input Parameters Two words and merge point

Output Parameters Packed word

Result None

Comment p(0:31) = w1(0:mp*bg-1) || w2(0:31-mp*bg)

9 WordHead2Pack

Purpose Pack two words into one refering to two merge points assuming latter word always starts packing from its head and possibly ends packing in the middle

Prototype UWORD WordHead2Pack(UWORD word1, UWORD word2, BYTE merge_pt1, BYTE merge_pt2)

Scope Internal

Input Parameters Two words and merge point

Output Parameters Packed word

Result None

Comment p(0:31) = w1(31-mp*bg:31) || w2(0:31-mp*bg)

10 WordRead

Purpose Read the color of pixels in word form at the specified location

Prototype UWORD WordRead(SHORT xcoord, SHORT ycoord)

Scope Internal

Input Parameters Cartesian coordinate

Output Parameters None

Result Color in word form

Comment If the location is outside the screen region, return COLOR_WHITE

11 WordTail1Pack

Purpose Pack two words into one refering to former word's shift and the merge point

Prototype UWORD WordTail1Pack(UWORD word1, BYTE word1_shift, UWORD word2, BYTE merge_pt)

Scope Internal

Input Parameters Two words with former word's shift and merge point

Output Parameters None

Result Packed word

Comment p(0:31) = w1(s1*bg:(s1+mp)*bg-1) || w2(mp*bg:31)

12 WordWrite

Purpose Write the color of pixel at the specified location

Prototype void WordWrite(SHORT xcoord, SHORT ycoord, UWORD data)

Scope Internal

Input Parameters Cartesian coordinate and color

Output Parameters None

Result None

Comment If the location is outside the screen region, do nothing

Main.c Functions

1 SysEventLoop

Purpose This function is called only by _main function to act as the event loop of the system. The system loops in this function and launch different application as required.

Prototype void SysEventLoop()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

2 SysEventLoopWhileLoop

Purpose It is a function to gather the information of the battery level and the status of the battery door lock

Prototype BOOLEAN SysEventLoopWhileLoop()

Scope System

Input Parameters None

Output Parameters None

Result

TRUE	battery level is acceptable and the battery door lock is closed
FALSE	battery is empty or the battery door lock is opened

Comment None

3 _main

Purpose It is the PDA startup routine for hard-reset and soft-reset. whenever the PDA is reset, this function is called

Prototype void _main()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

memchk.c

1 SysBlockAddr

Purpose Resovle the physical address of a block in the MAT.

Prototype UWORD SysBlockAddr(MatInfoList* mat, USHORT block_num)

Scope System

Input Parameters

mat	Pointer to the MAT
block_num	Block the be resolved

Output Parameters None

Result Physical address of the block

Comment None

2 SysMemCheck

Purpose Find check sum of all MAT in the system.

Prototype BOOLEAN SysMemCheck()

Scope System

Input Parameters None

Output Parameters None

Result

TRUE	The check sum is correct
FALSE	The check sum is incorrect

Comment None

3 SysMemCheckMAT

Purpose Find the check sum of a MAT.

Prototype Err SysMemCheckMAT(MatInfoList *mat)

Scope System

Input Parameters

mat	Pointer to the MAT

Output Parameters None

Result

TRUE	Check sum is correct
ERR_MEM_INV_CHKSUM	Check sum is incorrect

Comment None

4 SysMemDataCheckSum

Purpose Find the check sum of the data blocks.

Prototype UWORD SysMemDataCheckSum()

Scope System

Input Parameters None

Output Parameters None

Result Check sum of the data blocks

Comment This function has been obsolete

5 SysMemMATCheckSum

Purpose Calculate the check sum of MAT.

Prototype UWORD SysMemMATCheckSum(MatInfoList *mat)

Scope System

Input Parameters mat Point to the MAT

Output Parameters None

Result Check sum of the mat

Comment check sum = XOR all ID and Next entries in the MAT

Memory Manager and Data Manager Functions Summary (User)

I) Memory Manager

- MemoryTotalFree
- MemoryTotalSysMem

II) Data Manager

- DataCategoryName
- DataCategoryNextFree
- DataCategorySetName
- DataCategorySort
- DataCloseDB
- DataCloseRecord
- DataDBInfo
- DataDBSize
- DataDeleteDB
- DataDeleteRecord
- DataFieldSize
- DataFindDB
- DataFindBinRecord
- DataFindRecord
- DataGetDBSortField
- DataGetField
- DataIsDBOpen
- DataIsExclusive
- DataIsRecordChange
- DataIsRecordOpen
- DataMoveCat
- DataNewDB
- DataNewRecord
- DataNewRecordWithID
- DataNumtoRecID
- DataOpenDB
- DataOpenNextRecord
- DataOpenPrevRecord
- DataOpenRecord
- DataReadField
- DataRecIDtoNum
- DataRecordInfo

- DataSeekField
- DataSetDBInfo
- DataSetRecordAttribute
- DataTotalField
- DataTotalRecord
- DataUpdateField
- DataWriteField

Memory Manager Functions

1 MemoryTotalFree

Purpose Find total free blocks in System

Prototype USHORT MemoryTotalFree()

Scope All

Input Parameters None

Output Parameters None

Return Number of free block

Comments

2 MemoryTotalSysMem

Purpose Get total memory available for memory manager

Prototype void MemoryTotalSysMem(UWORD *totalmem)

Scope All

Input Parameters None

Output Parameters	totalmem	Total available memory in byte
Return	None	
Comments		

Data Manager Functions

1 DataCategoryName

Purpose	Get a category's name	
Prototype	Err DataCategoryName(DatabaseID dbid, UBYTE cat, BYTE *name)	
Scope	All	
Input Parameters	dbid	Database ID of the database
	cat	The category numer
Output Parameters	name	The category name
Return	TRUE	Success
	ERR_DATA_DB_NOT_OPEN	Database not opened
Comments	name = NULL if category name not set The valid category range is between 0 and 254 inclusively	

2 DataCategoryNextFree

Purpose	Return next free category's number
Prototype	Err DataCategoryNextFree(DatabaseID dbid, UBYTE * next_cat)
Scope	All

Input Parameters

dbid	Database ID

Output Parameters

next_cat	Next free category's number

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_CAT_FULL	No free category available

Comments The return category is always greater than 0, even if category 0 is un-used

3 DataCategorySetName

Purpose Set a category's name

Prototype Err DataCategorySetName(DatabaseID dbid, UBYTE cat, BYTE *name)

Scope All

Input Parameters

dbid	Database ID
cat	Category number
name	Category name

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_MISS	Database not found
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_DB_RO	Database is read only

Comments

4 DataCategorySort

Purpose Sort the category name

Prototype UBYTE DataCategorySort(DatabaseID dbid, UBYTE cat[256])

Scope All

Input Parameters

dbid	Database ID

Output Parameters

cat	The sorted category's number, in ascending order

Return The number of valid entry in *cat*

Comments Category 0 will not be sorted

5 DataCloseDB

Purpose Close an opened database

Prototype Err DataCloseDB(DatabaseID dbid)

Scope All

Input Parameters

dbid	Database ID of the database to close

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened

Comments All opened records of the database will be closed by Data Manager

6 DataCloseRecord

Purpose Close an opened record

Prototype Err DataCloseRecord(DatabaseID dbid, RecordID rec_id)

Scope All

Input Parameters

rec_id	Record ID of the record to close

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_REC_NOT_OPEN	Record not opened

Comments Record also closed when the database is closed

7 DataDBInfo

Purpose Get a database's information

Prototype Err DataDBInfo(DatabaseID dbid, USHORT *version, AppID *owner, RTM *create_date, RTM *backup_date, BYTE *name, BYTE *owner_info)

Scope All

Input Parameters

dbid	Database ID of the database to check

Output Parameters

version	Database version
owner	Owner of the database
create_date	Creation date of the database
modi_date	Last modification date of the database
back_date	Last backup date of the database
name	The name of the database
owner_info	owner information of the database

Return

TRUE	Success
ERR_DATA_DB_MISS	Database not found

Comments Only first 32 bytes of database name will return

8 DataDBSize

Purpose Get the database size

Prototype Err DataDBSize(DatabaseID DBID, USHORT *total_block, UWORD *total_byte)

Scope All

Input Parameters

dbid	The Database ID of the database

Output Parameters

total_block	Total block entry in MAT
total_byte	Total byte used by the DB

Return

> 0	The size of the database
0	Database not found

Comments

9 DataDeleteDB

Purpose Delete a database

Prototype Err DataDelDB(DatabaseID dbid)

Scope All

Input Parameters

dbid	Database ID of the database to delete

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_MISS	**Database not found**
ERR_DATA_DB_BUSY	Database in use by other application

Comments If database is opened by another application, it cannot delete

10 DataDeleteRecord

Purpose Delete a record

Prototype Err DataDeleteRecord(DatabaseID dbid, RecordID rec_id, BOOLEAN archive)

Scope All

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record to be deleted
archive	Set the archive bit

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_RECORD_BUSY	Record is in used / opened by other application, can not delete
ERR_DATA_INV_RECID	Record not found
ERR_DATA_LOCKED	Record's *lock* flag is set
ERR_DATA_DB_RO	Database is read only

Comments Cannot delete locked or opened record

11 DataFieldSize

Purpose Get the field size

Prototype Err DataFieldSize(Database dbid, RecordID rec_id, USHORT field_num, UWORD *field_size)

Scope All

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record
field_num	The field number

Output Parameters field_size The field size in byte

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_REC_NOT_OPEN	Record not opened
ERR_DATA_INV_FIELD	Invalid field number

Comments

12 DataFindDB

Purpose Find a database and get its database ID

Prototype BOOLEAN DataFindDB(BYTE *name, DatabaseID *dbid)

Scope All

Input Parameters name The name of the database to find, terminated by ‘0x00’

Output Parameters dbid Database ID of the database

Return

TRUE	Success
FALSE	Database not found

Comments

13 DataFindBinRecord

Purpose Find a binary data record

Prototype Err DataFindBinRecord(DatabaseID dbid, USHORT find_field, UWORD start_pos, BYTE *buffer, USHORT compare_length, RecordID *rec_id)

Scope All

Input Parameters

dbid	Database ID of the database
find_field	Field to search
start_pos	Record number to start the search
buffer	Search string, terminated by ‘0x00’
compare_length	Data length to compare, valid range is 0 – 30

Output Parameters

rec_id	Record ID of the match record

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_NOT_FOUND	Record not found
ERR_DATA_INV_PARAM	Invalid parameter

Comments

14 DataFindRecord

Purpose Find a record

Prototype Err DataFindRecord(DatabaseID dbid, USHORT find_field, UWORD start_pos, BYTE *buffer, RecordID *rec_id)

Scope All

Input Parameters

dbid	Database ID of the database
find_field	Field to search
start_pos	Record number to start the search
buffer	Search string, terminated by ‘0x00’

Output Parameters

rec_id	Record ID of the match record

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_NOT_FOUND	Search string not found in all records
ERR_DATA_INV_PARAM	Invalid parameter

Comments

15 DataGetField

Purpose Read entire field

Prototype Err DataGetField(DatabaseID dbid, RecordID rec_id, USHORT field_num, BYTE **buffer, UWORD *byte_read)

Scope All

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record to read
field_num	Record's field to read

Output Parameters

buffer	Field content
byte_read	Size of *buffer* in byte

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_REC_NOT_OPEN	Record not opened
ERR_DATA_INV_PARAM	Invalid field number
ERR_DATA_NO_SPACE	Not enough memory to allocate buffer

Comments No need to allocate buffer in application, the function will allocate the buffer, application should call q*free(buffer)* to release the buffer

16 DataIsDBOpen

Purpose Check if the database is opened by the application

Prototype BOOLEAN DataIsDBOpen(DatabaseID dbid, DBOpenMode *open_mode, USHORT *sort_field)

Scope All

Input Parameters	dbid	Database ID of the database
Output Parameters	open_mode	Database open mode
	sort_field	Sorting field of the database
Return	TRUE	Database is opened
	FALSE	**Database is not opened**

Comments

17 DataIsExclusive

Purpose Check if the database is opened by other application in exclusive mode

Prototype Boolean DataIsExclusive(DatabaseID dbid, AppID *app)

Scope All

Input Parameters	dbid	Database ID of the database to check
Output Parameters	app	Application ID of the ex.. application
Return	TRUE	The database is opened exclusively by another application
	FALSE	The database is not opened exclusively

Comments If the database does not exist in the system, FALSE will return

18 DataIsRecordChange

Purpose Check if the record has been modified

Prototype BOOLEAN DataIsRecordChange(DatabaseID dbid, RecordID rec_id, BYTE *modi)

Scope All

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record to check
modi	The modi number returned from open record

Output Parameters

modi	The updated modi number

Return

TRUE	Record has changed since last check / record not opened
FALSE	Record has not changed

Comments The modi number of a record will increase if the record is modified by write operation

19 DataIsRecordOpen

Purpose Check if record is opened

Prototype BOOLEAN DataIsRecordOpen(DatabaseID dbid, RecordID rec_id, USHORT *last_field, UWORD *read_pos)

Scope All

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record

Output Parameters

last_field	Last read/write field
read_pos	Next read position

Return

TRUE	Record is opened
FALSE	Record is not opened

Comments

20 DataMoveCat

Purpose Move all records in one category to another category

Prototype Err DataMoveCat(DatabaseID dbid, UBYTE src_cat, UBYTE dest_cat)

Scope All

Input Parameters

dbid	DatabaseID of the database
src_cat	Source category
dest_cat	Destination category

Output Parameters None

Return

TRUE	**Success**
ERR_DATA_INV_PARAM	Invalid parameter (src_cat = dest_cat)
ERR_DATA_DB_NOT_OPEN	Database not open
DATA_DATA_DB_RO	Database is read only

Comments

21 DataNewDB

Purpose Create a new database

Prototype Err DataNewDB(BYTE *name, USHORT version, BYTE *owner_info, DatabaseID *dbid)

Scope All

Input Parameters

name	The name of the database, terminated by ‘0x00’
version	The version of the database
owner_info	Owner specified information, terminated by’0x00’

Output Parameters

dbid	DatabaseID of the new database

Return

TRUE	Success
ERR_DATA_DB_EXIST	DB with the same name already exists
ERR_DATA_NO_SPACE	Not enough ram space

Comments

22 DataNewRecord

Purpose Add a new record

Prototype Err DataNewRecord(DatabaseID dbid, UBYTE cat, USHORT num_field, RecordID *rec_id,)

Scope All

Input Parameters

dbid	Database ID of the database that the record will belong
cat	Category of the record
num_field	Number of field in the record

Output Parameters

rec_id	Record ID of the new record

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not open
ERR_DATA_DB_LOCK	The DB is opened exclusively by other application
ERR_DATA_INV_PARAM	Invalid parameter
ERR_DATA_NO_SPACE	Not enough ram space

Comments The record will be opened, call *DataCloseRecord()* to close the record

23 DataNewRecordWithID

Purpose Add a new record with specified Record ID

Prototype Err DataNewRecordWithID(DatabaseID dbid, RecordID rec_id, BYTE cat, USHORT num_field)

Scope All

Input Parameters

dbid	Database ID of the database that the

		record will belong
	rec_id	Record ID of the new record
	cat	Category of the record
	num_field	Number of field in the record

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not open
ERR_DATA_DB_LOCK	The DB is opened exclusively by other application
ERR_DATA_INV_PARAM	Invalid parameter
ERR_DATA_NO_SPACE	Not enough ram space

Comments (1) Record ID must between 0x80000000 and 0x90000000 inclusively
(2) Application can read/write the record without open it by *DataOpenRecord()* once the database has been opened.
(3) The record will not counted by *DataTotalRecord()*

24 DataNumtoRecID

Purpose Find the record ID of a record in the sort table

Prototype Err DataNumtoRecID(DatabaeID dbid, UWORD rec_num, RecordID *rec_id)

Scope Internal

Input Parameters

dbid	Database ID of the database
rec_num	Record number in the sort table

Output Parameters

rec_id	Record ID

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_INV_RECNUM	Invalid record number

Comments

25 DataOpenDB

Purpose Open a database

Prototype Err DataOpenDB(DatabaseID dbid, USHORT sort_field, DBOpenMode open_mode)

Scope All

Input Parameters

dbid	Database ID of the database to open
sort_field	Field to sort
open_mode	Open mode of the database

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_MISS	Database Not found
ERR_DATA_INV_PARAM	Invalid parameter (e.g.sort_field not exist)
ERR_DATA_DB_LOCK	The database is other by other application in exclusive mode
ERR_DATA_NO_MEM	Not enough memory

Comments (1) open_mode can be OPEN_RO (read only), OPEN_RW (read/write), OPEN_EX (exclusively).
(2) Application need to close and open the database again to change the open mode

26 DataOpenNextRecord

Purpose Open next record

Prototype Err OpenNextRecord(DatabaseID dbid, RecordID src_rec, BOOLEAN close_current, RecordID *next_rec_id, BYTE *modi)

Scope All

Input Parameters

dbid	Database ID of the database
src_rec	Record ID of Current record
close_current	if TRUE, current record will close (only if open next record success)

Output Parameters

next_rec_id	Record ID of next record
modi	Modification number

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_REC_NOT_OPEN	Source record not opened
ERR_DATA_INV_RECID	Source record not found
ERR_DATA_EOR	No next record

Comments

27 DataOpenPrevRecord

Purpose Open previous record

Prototype Err DataOpenPrevRecord(DatabaseID dbid, RecordID src_rec, BOOLEAN close_current, RecordID *prev_rec_id)

Scope All

Input Parameters

dbid	Database ID of the database
src_rec	Record ID of current record
close_current	= TRUE will close current record (only if open previous record success)

Output Parameters

prev_rec_id	Record ID of Previous record

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_REC_NOT_OPEN	Source record not opened
ERR_DATA_INV_RECID	Source record not found
ERR_DATA_BOR	No previous record

Comments

28 DataOpenRecord

Purpose Open a record

Prototype Err DataOpenRecord(DatabaseID dbid, UWORD rec_num, RecordID *rec_id, BYTE *modi)

Scope All

Input Parameters

dbid	DatabaseID of the database to open
rec_num	The record to open

Output Parameters

rec_id	RecordID of the opened record
modi	The modification number (set to NULL if unwanted)

Return

TRUE	Success
ERR_DATA_NOT_FOUND	Record not found
ERR_DATA_OPENED	Already opened by the application
ERR_DATA_DB_LOCK	Database is opened exclusively by other application
ERR_DATA_NO_MEM	Not enough memory

Comments If the record is already opened by the application, ERR_DATA_OPENED will return, open count of the record will not increase, record pointer will not reset, output parameters still valid

29 DataReadField

Purpose Read a part of the field

Prototype Err DataReadField(DatabaseID dbid, RecordID rec_id, USHORT field_num, UWORD start_pos, UWORD num_byte, BYTE **buffer, UWORD *byte_read)

Scope All

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record to read
field_num	Field to read
start_pos	Read start position (see comment)
num_byte	Number of byte to read (see comment)

Output Parameters

buffer	Record content
byte_read	Size of *buffer* in byte

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_REC_NOT_OPEN	**Record not opened**
ERR_DATA_INV_PARAM	*start_pos* > field size

Comments *start_pos*: the byte position relative to start of the field
If *start_pos* = READ_RELATIVE, will start from last read position.
If *num_byte* = READ_TO_END will read to end of the field.
Not need to allocate buffer in application, the function will allocate the buffer, application should call q*free(buffer)* to release the buffer

30 DataRecIDtoNum

Purpose Convert a Record ID to record number of the sort table

Prototype Err DataRecIDtoNum(DatabaeID dbid, RecordID rec_id, UWORD *rec_num)

Scope Internal

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record

Output Parameters

rec_num	The record's poistion in sort table

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_INV_RECID	Invalid record ID

Comments *rec_num* will only valid if no new record add/delete to the database, so do not store *rec_num* permanently

31 DataRecordInfo

Purpose Get the record information

Prototype Err DataRecordInfo(DatabaseID dbid, RecordID rec_id, UWORD *rec_size, UBYTE *cat, Attribute *secret, Attribute *lock, RTM *modi_date)

Scope All

Input Parameters

dbid	Database ID
rec_id	The Record ID of the record

Output Parameters

rec_size	Record size in byte
cat	Record category
secret	Secret flag
lock	Lock flag
modi_date	Record last modification date

(set to NULL for unwanted data)

Return

TRUE	Success
ERR_DATA_INV_RECID	Invalid record ID
ERR_DATA_DB_MISS	Database not found

Comments No need to open the record

32 DataSeekField

Purpose Move the record pointer

Prototype Err DataSeekField(DatabaseID dbid, RecordID rec_id, USHORT field_num, WORD position)

Scope All

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record to read
field_num	Field number
position	New record pointer position relative to current record pointer

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not open
ERR_DATA_REC_NOT_OPEN	Record not open
ERR_DATA_INV_FIELD	Invalid field
ERR_DATA_NO_MEM	Not enough memory

Comments position can be +ve (seek forward), -ve (seek backward) or SEEK_EOF (seek to end of field), SEEK_BOF (seek to begin of field)
If (current pointer + *position*) < 0, will position to begin of field
If (current pointer + *position*) > field length, will position to end of field

33 DataSetDBInfo

Purpose Set the information of the database

Prototype Err DataSetDBInfo(DatabaseID dbid, USHORT version, AppID owner, RTM *backup_date)

Scope Internal

Input Parameters

dbid	Database ID of the database
version	The version number of the database
owner	Application ID of the owner
backup_date	Last backup date

Output Parameters None

Return

TURE	Success
ERR_DATA_DB_MISS	Database not found

Comments To un-alter the value, set
owner = DB_OWNER_NOCHG
version = DB_VER_NOCHG
backup_date = NULL

34 DataSetRecordAttribute

Purpose Set the record attribute

Prototype Err DataSetRecordAttribute(DatabaseID dbid, RecordID rec_id, Attribute secret, Attribute lock, UBYTE cat)

Scope All

Input Parameters

dbid	Database ID
record_id	The Record ID of the record
secret	Secret flag
lock	Lock flag
cat	Category

secret and *lock* can be ATTR_SET, ATTR_CLR or ATTR_UNCHG
cat = 0xFF means unchange

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not open
ERR_DATA_REC_NOT_OPEN	Record not open

Comments

35 DataTotalField

Purpose Find the number of field in a record

Prototype Err DataTotalField(DatabaseID dbid, RecordID rec_id, USHORT *total_field)

Scope All

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record

Output Parameters

total_field	Total field of the record

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_REC_NOT_OPEN	Record not opened
ERR_DATA_DB_MISS	Database not found
ERR_DATA_INV_RECID	Invalid Record ID

Comments

36 DataTotalRecord

Purpose Get total record in the database

Prototype Err DataTotalRecord(DatabaseID dbid, UWORD *total_rec)

Scope All

Input Parameters

dbid	Database ID of the database

Output Parameters

total_rec	Total record in the database

Return

TRUE	Success
ERR_DATA_MISS	Database not found

Comments

37 DataUpdateField

Purpose Append data to a field

Prototype Err DataUpdateField(DatabaseID dbid, RecordID rec_id, USHORT field_num, UWORD num_byte, BYTE *buffer)

Scope All

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record
field_num	The field to update
num_byte	Size of *buffer* in byte
buffer	Data to append

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_REC_NOT_OPEN	Record not opened
ERR_DATA_INV_PARAM	Invalid parameter (num_byte = 0)
ERR_DATA_NO_SPACE	No memory

Comments

1. Do not append more than 4KB data to a field in one write operation.
2. The field size of last field of a record cannot exceed 4KB.If so, you need to append a dummy field to the record.

38 DataWriteField

Purpose Change the content of a field

Prototype Err DataWriteField(DatabaseID dbid, RecordID rec_id, USHORT field_num, UWORD num_byte, BYTE *buffer)

Scope All

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record
field_num	The field to update
num_byte	Size of *buffer* in byte
buffer	New content of the field

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_REC_NOT_OPEN	Record not opened

Comments

Memory Manager Functions (Sys)

1 MMUInit

Purpose Initialise memory manager

Prototype void MMUInit(void)

Scope OS

Input Parameters None

Output Parameters None

Return None

Comments Will erase all databases and memory content, rebuild the MAT and reset the dynamic memory system

2 MemoryAppStartBlock

Purpose Find the first block number of an application in the MAT

Prototype Err MemoryAppStartBlock(AppID app_id, MatInfoList **mat, USHORT *start_block)

Scope OS

Input Parameters None

Output Parameters

app_id	Application ID to be found

Return

mat	Pointer to the MAT structure where the application resident
block	The first block of the application in the MAT

Comments

3 MemoryBlockFreeSpace

Purpose Find free space in a block

Prototype UWORD MemoryBlockFreeSpace(MatInfoList *mat, USHORT block)

Scope All

Input Parameters

mat	The MAT to be checked
block	The block to be checked

Output Parameters None

Return Block free space in byte

4 MemoryBlockType

Purpose Get a block's type

Prototype BlockType MemoryBlockType((MatInfoList *mat, USHORT block)

Scope All

Input Parameters

mat	The MAT to be checked
block	The block to be checked

Output Parameters None

Return

0 – 0xFFFE	**Type of the block**
0xFFFF	**Invalid block**

Comments

5 MemoryBuildMAT

Purpose Build the MAT

Prototype Err MemoryBuildMAT(void)

Scope Internal

Input Parameters None

Output Parameters None

Return

TRUE	Success
ERR_MEM_NO_MEM	No physical memory available

Comments

6 MemoryCheck

Purpose Check the integrity of the memory

Prototype BOOLEAN MemoryCheck()

Scope All

Input Parameters None

Output Parameters None

Return

TRUE	Memory is valid
FALSE	Memory content is invalid

Comments

7 MemoryCheckMAT

Purpose Check the integrity of MAT

Prototype BOOLEAN MemoryCheckMAT(MatInfoList *mat)

Scope All

Input Parameters

mat	The MAT to be checked

Output Parameters None

Return

TRUE	The MAT is valid
ERR_MEM_INV_CHKSUM	The MAT is invalid

Comments

8 MemoryClearAppDM

Purpose Clear all dynamic allocated memory block created by the application

Prototype Err MemoryClearAppDM(AppID appid)

Scope OS

Input Parameters

appid	Application ID of the application

Output Parameters None

Return

TRUE	Success
FALS	No memory allocated by the application

Comments

9 MemoryCopyData

Purpose Copy data from one block to another

Prototype Err MemoryCopyData(MatInfoList *mat , USHORT src_block, USHORT src_offset, USHORT dest_block, USHORT dest_offset, UWORD num_byte)

Scope Internal

Input Parameters

mat	The MAT to update
src_block	Source block for copy
src_offset	Copy start position (offset of the block)
dest_block	Destination block
dest_offset	Destination start position
num_byte	Number of byte to copy

Output Parameters None

Return

TRUE	Success
ERR_MEM_INV_SRC	Invalid source block
ERR_MEM_INV_DEST	Invalid destination block
ERR_MEM_INV_PARAM	if destination cannot fit the bytes (num_byte + dest_offset >

4KB)

Comments

10 MemoryDMANew

Purpose Allocate new blocks for dynamic memory

Prototype Err MemoryDMANew(MatInfoList *mat, UBYTE type, AppID appid, USHORT num_block)

Scope Internal

Input Parameters

type	Memory type, DM_OS or DM_USER
appid	Application ID

Output Parameters None

Return

TRUE	Success
ERR_MEM_INV_TYPE	Invalid memory type
ERR_MEM_NO_SPACE	Not enough memory

Comments

11 MemoryDataCheckSum

Purpose Calculate the check sum of all memory blocks

Prototype UWORD MemoryDataCheckSum()

Scope Internal

Input Parameters None

Output Parameters None

Return The check sum

Comments

12 MemoryFindStartBlock

Purpose Find the first block of a database in the MAT

Prototype Err MemoryFindStartBlock(MatInfoList *mat, UWORD dbid, USHORT *block)

Scope Internal

Input Parameters

mat	The MAT to be found
dbid	Database ID of the database

Output Parameters

block	First block of the database in the MAT

Return

TRUE	Success
ERR_MEM_ID_MISS	Invalid block

Comments

13 MemoryGetConBlock

Purpose Get continuous block

Prototype Err MemoryGetConBlock(MatInfoList *mat, USHORT num_block, UWORD id, BlockType type, USHORT prev_block, USHORT *new_block)

Scope OS

Input Parameters

mat	The MAT to be updated
num_block	Number of continuous block to allocate
id	ID of the new block
prev_block	Previous block number

Output Parameters

first_block	First allocated block

Return

TRUE	Success
ERR_MEM_NO_MEM	Cannot allocate the blocks

Comments

14 MemoryGetPageTable

Purpose Get system, application and interrupt page table

Prototype void MemoryGetPageTable(UWORD **sys_table, UWORD **app_table, UWORD **int_table)

Scope Internal

Input Parameters None

Output Parameters

sys_table	Pointer to the system page table
app_table	Pointer to the application page table
int_table	pointer to the interrupt page table

Return None

Comments

15 MemoryGetSpace

Purpose Find block/blocks with free space > request

Prototype BOOLEAN MemoryGetSpace(MatInfoList *mat, USHORT block, UWORD size, BOOLEAN con, USHORT *new_block, UWORD *offset)

Scope All

Input Parameters

mat	The MAT to update
block	start block of the database
size	request size
con	if TRUE, if size > 4 KB, allocate block in continue memory

Output Parameters

new_block	First block of the new block(s)
offset	Offset from the block

Return Number of block used by the database / program

16 MemoryGroupUsage

Purpose Find the block usage of a database / program

Prototype USHORT MemoryGroupUsage(MatInfoList *mat, UWORD id)

Scope All

Input Parameters

id	Database ID or program ID

Output Parameters None

Return Number of block used by the database / program

Comments If the id not exist, return 0

17 MemoryInsertBlock

Purpose Allocate a block AFTER the specific block in MAT link

Prototype Err MemoryInsertBlock(MatInfoList *mat, USHORT block, USHORT num_block, USHORT *new_block)

Scope Internal

Input Parameters

mat	The MAT to be updated
block	New block will insert after this block
num_block	Number of block to be inserted

Output Parameters

new_block	First inserted block’s number

Return

TRUE	Success
ERR_MEM_INV_BLOCK	Invalid block number
ERR_MEM_NO_MEM	Not enough memory

Comments

18 MemoryMATCheckSum

Purpose Calculate the check sum of MAT

Prototype UWORD MemoryMATCheckSum(MatInfoList *mat)

Scope Internal

Input Parameters

mat	The MAT

Output Parameters None

Return Check sum of the MAT

Comments

19 MemoryMoveBlock

Purpose Move the content and MAT info. of a block to another block and mark the origin block free

Prototype Err MemoryMoveBlock(MatInfoList *mat, USHORT src_block, USHORT dest_block)

Scope Internal

Input Parameters

mat	The MAT
src_block	Source block’s number
dest_block	Destination block’s number

Output Parameters None

Return

TRUE	Success
ERR_MEM_INV_BLOCK	invalid source/destination block

Comments

20 MemoryNewBlock

Purpose Allocate new block

Prototype Err MemoryNewBlock((MatInfoList *mat, USHORT num_block, UWORD id, BlockType type, USHORT prev_block, USHORT *new_block)

Scope Internal

Input Parameters

mat	The MAT to update
num_block	Number of new block requested
id	Database/Application ID of the new block
type	Block type
prev_block	Previous block number if new block is linked to another block

Output Parameters

new_block	The new block

Return

TRUE	Success
ERR_MEM_INV_BTYPE	Invalid block type
ERR_MEM_INV_BLOCK	Invalid previous block

Comments Allocate a free entry in MAT, update MAT's link

21 MemoryNextBlock

Purpose Find next linked block

Prototype Err MemoryNextBlock((MatInfoList *mat, USHORT block, USHORT *next)

Scope Internal

Input Parameters

mat	The MAT to update
block	Block to check

Output Parameters

next	Next linked block

Return

TRUE	Success
ERR_MEM_INV_BLOCK	Invalid block
ERR_MEM_EOB	No next block

Comments

22 MemoryPrevBlock

Purpose Find previous linked block

Prototype Err MemoryPrevBlock((MatInfoList *mat, USHORT block, USHORT *prev)

Scope Internal

Input Parameters

mat	The MAT to update
block	Block to check

Output Parameters

prev	previous linked block

Return

TRUE	Success
ERR_MEM_INV_BLOCK	Invalid block
ERR_MEM_BOB	No previous block

Comments

23 MemoryReleaseBlock

Purpose Mark a block free

Prototype Err MemoryReleaseBlock((MatInfoList *mat, USHORT block, BOOLEAN check_link)

Scope Internal

Input Parameters

mat	The mat to update
block	Block to free
check_link	If TRUE, blocks forward and backward links will also free

Output Parameters None

Return

TRUE	Success
ERR_MEM_INV_BLOCK	Invalid block

Comments

24 MemoryResolveBlock

Purpose Resolve physical address to [mat number,block number] format

Prototype BOOLEAN MemoryResolveBlock(UWORD addr, USHORT *block, USHORT *mat_num)

Scope Internal

Input Parameters

addr	physical address to resolve

Output Parameters

block	The resolved block number
mat_num	The resolved MAT number

Return

TRUE	Success
FALS	Fail

Comments

25 MemorySetMATFragment

Purpose Set/clear the fragment flag of a block

Prototype Err MemorySetMatFragment((MatInfoList *mat, USHORT block, BOOLEAN flag)

Scope Internal

Input Parameters

mat	The MAT to be updated
block	Block number in the MAT
flag	If TRUE, set fragment flag otherwise clear the fragment flag

Output Parameters None

Return

TRUE	Success
ERR_MEM_INV_BLOCK	Invalid block number

Comments

26 MemoryTotalFreeBlock

Purpose Return number of free block in MAT

Prototype USHORT MemoryTotalFreeBlock(MatInfoList *mat)

Scope Internal

Input Parameters

mat	The MAT to be checked

Output Parameters None

Return Number of free block in the MAT

Comments

27 MemoryUpdateCheckSum

Purpose Update Data block’s check sum

Prototype void MemoryUpdateCheckSum()

Scope Internal

Input Parameters None

Output Parameters None

Return None

Comments

Data Manager Functions

1 DataBuildSortTable

Purpose Build a sort table

Prototype Err DataBuildSortTable(DatabaseID dbid, USHORT sort_field, SortTableLnkPtr *sort_table, RecordID *max_rec_id)

Scope Internal

Input Parameters

dbid	Database ID of the database to sort
sort_field	Field to sort or DB_NO_SORT

Output Parameters

sort_table	The sort table
max_rec_id	Largest Record ID

Return

TRUE	Success
ERR_DATA_DB_MISS	Database not found
ERR_DATA_NO_MEM	Not enough memory to hold the sort table
ERR_DATA_INV_FIELD	Invalid field

Comments Records can only sort in ascending order

2 DataFreeSortTable

Purpose Dispose and free a sort table

Prototype Err DataFreeSortTable(DatabaseID dbid, USHORT sort_field, SortTableLnk * unused_Table)

Scope Internal

Input Parameters

dbid	Database ID of the database
sort_field	Sort field of the sort table
unused_table	Poiner to the unused sort table

Output Parameters None

Return

TRUE	Success
ERR_DATA_NOT_FOUND	Sort table not found

Comments

3 DataSortData

Purpose Sort the record and inset entry into sort table

Prototype Err DataSortData(RecordID rec_id, UWORD data_addr, SortTableLnkPtr sort_table, BOOLEAN is_sort, UWORD *sort_pos)

Scope Internal

Input Parameters

rec_id	RecordID of the record to insert
data_addr	The physical address of the data of the record
sort_table	The sort table to build
is_sort	= FALSE with not sort the record, only insert in the first entry

Output Parameters

sort_pos	The position of the record in the sort table

Return

TRUE	Success
ERR_DATA_NO_SPACE	Not enough memory

Comments This function can ONLY called by *DataBuildSortTable()*

4 DataUpdateSortTable

Purpose Update sort table after a record has been modified

Prototype Err DataUpdateSortTable(DatabaseID dbid, RecordID rec_id, USHORT field_num)

Scope Internal

Input Parameters

dbid	Database ID of the database
rec_id	Record ID of the record
field_num	The modified field

Output Parameters None

Return

TRUE	Success
ERR_DATA_DB_NOT_OPEN	Database not opened
ERR_DATA_RECORD_NOT_OPEN	Record not opened

Comments

Msg.c Functions

1 MsgAppend

Purpose To append a message into message queue

Prototype void MsgAppend(MsgType msg, MsgQueueType *Q)

Scope Application/System

Input Parameters

msg	message to be added
Q	Pointer to the last entry of message queue

Output Parameters None

Result None

Comment This routine is NOT safe to be called from Interrupt level

2 MsgAppendInt

Purpose To append a message into message queue

Prototype void MsgAppendInt(MsgType msg, MsgQueueType *Q)

Scope Application/System

Input Parameters

eventType	event type
eventID	event ID
para1	event parameter 1
para2	event parameter 2
evtPBP	additonal event pointer for additional information

Output Parameters None

Result None

Comment This routine is safe to be called from Interrupt level

3 MsgAppendMsg

Purpose To append a message into message queue

Prototype void MsgAppendMsg(void (*ptr_fct)(), WORD msgID, WORD para1, WORD para2, void * msgPBP)

Scope Application/System

Input Parameters

ptr_fct	function pointer
msgID	message ID
para1	message parameter 1
para2	message parameter 2
msgPBP	additional message pointer for additional

information

Output Parameters None

Result None

Comment This routine is NOT safe to be called from Interrupt level

4 MsgAppendMsgInt

Purpose To append a message into message queue

Prototype void MsgAppendMsgInt(void (*ptr_fct)(), WORD msgID,
WORD para1, WORD para2, void * msgPBP)

Scope Application/System

Input Parameters

ptr_fct	function pointer
msgID	message ID
para1	message parameter 1
para2	message parameter 2
msgPBP	additional message pointer for additional information

Output Parameters None

Result None

Comment This routine is safe to be called from Interrupt level

5 MsgAppendMsgReplaceInt

Purpose To append a message into message queue. The message manager searches whether a message with the same message function pointer and the message ID is already in the message queue. If yes, the message already in the message queue is replaced by the input message. Otherwise, the input message is only added to the queue.

Prototype void MsgAppendMsgReplaceInt(void (*ptr_fct)(), WORD msgID,
WORD para1, WORD para2,
void * msgPBP)

Scope Application/System

Input Parameters

ptr_fct	function pointer
msgID	message ID
para1	message parameter 1
para2	message parameter 2
msgPBP	additional message pointer for additional information

Output Parameters None

Result None

Comment This routine is safe to be called from Interrupt level

6 MsgClearAppMsg

Purpose To clear all messages, which are belongs to application, in the message queue if the message

Prototype void MsgClearAppMsg()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

7 MsgDecode

Purpose To dispatch the message to the correspond handling routine

Prototype void MsgDecode(MsgQueueType *msgQ)

Scope System

Input Parameters

msgQ	Pointer to the message queue

Output Parameters None

Result None

Comment None

8 MsgQueueCreate

Purpose To create an empty message queue

Prototype void MsgQueueCreate(MsgQueueType *Q)

Scope System

Input Parameters Q Pointer to the message queue

Output Parameters None

Result None

Comment None

9 MsgQueueEmpty

Purpose To check whether the message queue is empty or not

Prototype BOOLEAN MsgQueueEmpty(MsgQueueType *Q)

Scope System

Input Parameters Q Pointer to the message queue

Output Parameters None

Result None

Comment None

10 MsgQueueFull

Purpose To check whether the message queue is full or not

Prototype BOOLEAN MsgQueueFull(MsgQueueType *Q)

Scope System

Input Parameters

Q	Pointer to the message queue

Output Parameters None

Result None

Comment None

11 MsgServe

Purpose To get a message, which is on the head of the queue, from the message queue

Prototype BOOLEAN MsgServe(MsgType *msgP, MsgQueueType *Q)

Scope System

Input Parameters

Q	Pointer to the message queue

Output Parameters

msgP	Pointer to the output message

Result

TRUE	message is gotten successfully
FALSE	message queue is empty

Comment None

resmgr.c

1 ResGetInstallIconData

Purpose Return the pointer to system variable *install_icon_res_data.*

Prototype UBYTE *ResGetInstallIconData()

Scope System/Application

Input Parameters None

Output Parameters None

Result The pointer to system variable *install_icon_res_data*

Comment *install_icon_res_data* contains the icon images which will be displayed in the PDA's main menu for user to identify applications .

2 ResClose

Purpose Close a resource object.

Prototype Err ResClose(ObjectID obj_id)

Scope System

Input Parameters

obj_id	ID of the resource object

Output Parameters None

Result

TRUE	Success
FALSE	Fail (due to the resource object not opened)

Comment None

3 ResFieldInfo

Purpose Get the size and location of a resource field.

Prototype Err ResFieldInfo(AppID app, ObjectID obj_id, USHORT field_num, UWORD *field_size, UWORD *field_addr)

Scope System

Input Parameters

app	Application ID
obj_id	ID of the resource object
field_num	Field number

Output Parameters

field_size	Size of the field
field_addr	Memory location of the field

Result

TRUE	Success
ERR_RES_NOT_OPEN	Resource object not opened

Comment None

4 ResFieldSize

Purpose Get the field size of a resource object.

Prototype Err ResFieldSize(ObjectID obj_id, USHORT field_num, UWORD *field_size)

Scope System

Input Parameters

obj_id	ID of the resource object
field_num	Field number

Output Parameters

field_size	Size of the field

Result

TRUE	Success
ERR_RES_NOT_OPEN	Resource object not opened

Comment None

5 ResGetField

Purpose Read a resource object’s field.

Prototype Err ResGetField(ObjectID obj_id, USHORT field_num, BYTE **buffer, UWORD *byte_read)

Scope System

Input Parameters

obj_id	ID of the resource object
field_num	Field number

Output Parameters

buffer	Data of the resource object
byte_read	Number of byte read into buffer

Result

TRUE	Success
ERR_RES_NOT_OPEN	Resource object not open
ERR_RES_NO_MEM	Not enough memory

Comment Buffer is allocated by *pmalloc()* internally, caller should call *qfree()* to free the buffer.

6 ResGetPointer

Purpose Get a memory pointer to the resource object's field.

Prototype Err ResGetPointer(ObjectID obj_id, USHORT field_num, USHORT offset, void **pointer)

Scope System

Input Parameters

obj_id	ID of the resource object
field_num	Field number
offset	Offset to the field

Output Parameters

pointer	Pointer to the resource object's field

Result

TRUE	Success
ERR_RES_NOT_OPEN	Resource object not opened

Comment pointer = memory address of the field + offset

7 ResInit

Purpose Initialize Resource Manager

Prototype void ResInit()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

8 ResObjInfo

Purpose Check if the resource object is opened and return the object structure.

Prototype BOOLEAN ResObjInfo(AppID app, ObjectID obj_id, ResLnk **res_ptr)

Scope System

Input Parameters

app	Application ID
obj_id	ID of the resource object

Output Parameters

Res_ptr	Pointer to the object structure in Resource Table

Result

TRUE	Success
FALSE	Resource object not opened

Comment None

9 ResObjLocate

Purpose Get the memory location of the resource object.

Prototype BOOLEAN ResObjLocate(UWORD res_addr, ObjectID obj_id, UWORD *obj_addr)

Scope System

Input Parameters

res_addr	Memory location of the resource
obj_id	ID of the resource object

Output Parameters

obj_addr	Memory location of the resource object

Result

TRUE	Success
FALSE	Object not found in the resource

Comment None

10 ResOpen

Purpose Open a resource object.

Prototype Err ResOpen(ObjectID obj_id)

Scope System

Input Parameters

obj_id	ID of the resource object

Output Parameters None

Result

TRUE	Success
ERR_OBJ_MISS	Object not found in the resource
ERR_RES_NO_MEM	Not enough memory

Comment

1. Object must be opened before any read operation
2. Call ResClose() to close an opened object

11 ResReadByte

Purpose Read specify number of byte.

Prototype void ResReadByte(UWORD start_addr, UWORD num_byte, BYTE *data, UWORD *last_addr)

Scope System

Input Parameters

start_addr	Source memory location
num_byte	Number of byte to read

Output Parameters

data	Pre-allocated destination buffer
last_addr	Last read memory location

Result None

Comment This function is similar to *memcpy()* with last read address returned.

12 ResReadField

Purpose Read part of a resource object's field

Prototype Err ResReadField(ObjectID obj_id, USHORT field_num, UWORD start_pos, UWORD num_byte, BYTE **buffer, UWORD *byte_read)

Scope System

Input Parameters

obj_id	ID of the resource object
field_num	Field number
start_pos	First offset to read
num_byte	Number of byte to read

Output Parameters

buffer	Field data
byte_read	Number of byte in buffer

Result

TRUE	Success
ERR_RES_NOT_OPEN	Resource object not opened
ERR_RES_NO_MEM	Not enough memory

Comment None

13 ResSkipByte

Purpose Skip bytes in resource.

Prototype void ResSkipByte(UWORD start_addr, UWORD num_byte, UWORD *last_addr)

Scope	System	
Input Parameters	start_addr num_byte	Source memory locaton Number of byte to skip
Output Parameters	last_addr	Destination memory location
Result	None	
Comment	last_addr = start_addr + num_byte	

sio_rxtx.c

1 SIORecieve

Purpose	Receive the external serial data or command to the system
Prototype	Void SIORecieve(UWORD *rx_data)
Scope	Internal
Input Parameters	None
Output Parameters	None
Result	None
Comment	None

2 SIOTransmit

Purpose	Transmit the serial data or command to the external device
Prototype	void SIOTransmit(UWORD tx_command_data)
Scope	Internal

Input Parameters None
Output Parameters None
Result None
Comment None

sioloop1.c

1 ActiveSIORdy

Purpose Enable the SIO serial port
Prototype void ActiveSIORdy()

Scope Internal
Input Parameters None
Output Parameters None
Result None
Comment None

2 DisSioDevDet

Purpose Disable the SIO serial port
Prototype void DisSioDevDet()

Scope Internal
Input Parameters None
Output Parameters None
Result None
Comment None

3 GetIconSlot

Purpose Get the external add-in Application icon
Prototype UWORD GetIconSlot()

Scope System / internal

Input Parameters None
Output Parameters None
Result None
Comment None

4 GetInstallInfEntry

Purpose Get the external add-in program information entry
Prototype UWORD GetInstallInfEntry()

Scope System/ internal
Input Parameters None
Output Parameters None
Result None
Comment None

5 HwSioInit

Purpose Init the SIO hardware part
Prototype void HwSioInit()

Scope Internal / System
Input Parameters None
Output Parameters None
Result None
Comment None

6 HwSioOff

Purpose Setting the SIO Hardware when the Pda power down
Prototype void HwSioOff()

Scope System

Input Parameters None
Output Parameters None
Result None
Comment None

7 HwSioOn

Purpose Setting the SIO hardware when the Pda power On
Prototype void HwSioOn()

Scope System
Input Parameters None
Output Parameters None
Result None
Comment None

8 SIODebounCheck

Purpose Before enable the SIO serial port, check the deboun first
Prototype void SIODebounCheck()

Scope Internal
Input Parameters None
Output Parameters None
Result None
Comment None

9 SioDevDet

Purpose When external device plug-in, this interrupt will occur to check
Prototype void SioDevDet()

Scope Internal

Input Parameters None
Output Parameters None
Result None
Comment None

10 SioDevMoveOut

Purpose Disable the SIO port when external device move out
Prototype void SioDevMoveOut()

Scope Internal
Input Parameters None
Output Parameters None
Result None
Comment None

11 SIOInit

Purpose Init the SIO driver
Prototype void SIOInit()

Scope System
Input Parameters None
Output Parameters None
Result None
Comment None

12 SIORdy

Purpose Read the external serial data
Prototype void SIORdy()

Scope Internal
Input Parameters None
Output Parameters None
Result None
Comment None

13 SwSIOInit

Purpose Software init the SIO

Prototype void SwSIOInit()

Scope System / internal
Input Parameters None
Output Parameters None
Result None
Comment None

14 Tdelay

Purpose Delay timer
Prototype void Tdelay(int count)

Scope Internal
Input Parameters None
Output Parameters None
Result None
Comment None

std.c

1 Delay

Purpose Time delay

Prototype void Delay(int count)

Scope System

Input Parameters count : delay period

Output Parameters None

Result None

Comment None

Strapi.c Functions

1 StrAnalyzeLine

Purpose To calculate the line information of each line of text in a field object

Prototype void StrAnalyzeLine(Field *field_ptr)

Scope Application/System

Input Parameters field_ptr Pointer to a field object

Output Parameters None

Result None

Comment None

2 StrAnalyzeWord

Purpose To calculate the number of characters of the input string that can be displayed within the input width and the total number of pixels that are required to display the characters

Prototype void StrAnalyzeWord(BYTE *string, BYTE font,
SHORT width, WORD *len, SHORT *wordw)

Scope Application/System

Input Parameters string Pointer to the input string
font the font size

	width	the maximum number of pixel for displaying the string
Output Parameters	len	Pointer to the number of characters ,which can be displayed within the width, in the string
	wordw	Pointer to the total number of pixel to display the calculated number *len of characters

Result None

Comment The input string must be terminated

3 StrCharPosToXY

Purpose To calculate the x-coordinate and y-coordinate of the left-top corner of a character that is being displayed within the bounds of the field object

Prototype
```
void StrCharPosToXY(Field *field_ptr, WORD char_pos,
                    SHORT *xcoord, SHORT *ycoord)
```

Scope Application/System

Input Parameters	field_ptr	Pointer to a field object
	char_pos	character position
Output Parameters	xcoord	Pointer to the value of the x-coordinate of the left-top corner of the character
	ycoord	Pointer to the value of the y-coordinate of the left-top corner of the character

Result None

Comment The input character position must be lager than or equal to the character position of the character that is being displayed on the top-left corner of the field object. Also, the input character position must be smaller than or equal to the character position of the character that is being displayed on the bottom-right corner of the field object.

4 StrCharPosToXYForTable

Purpose To calculate the x-coordinate and y-coordinate of the left-top corner of a character that is being displayed within the bounds of the field object. If the character, which is pointed by the character position, is not being displayed on the screen, then the output x-coordinate and output y-coordinate are set to –100 and –100.

Prototype void StrCharPosToXYForTable(Field *field_ptr, WORD char_pos, SHORT *xcoord, SHORT *ycoord)

Scope Application/System

Input Parameters

field_ptr	Pointer to a field object
char_pos	character position

Output Parameters

xcoord	Pointer to the value of the x-coordinate of the left-top corner of the character
ycoord	Pointer to the value of the y-coordinate of the left-top corner of the character

Result None

Comment The input character position must be lager than or equal to the character position of the character that is being displayed on the top-left corner of the field object. Also, the input character position must be smaller than or equal to the character position of the character that is being displayed on the bottom-right corner of the field object.

5 StrChopString

Purpose To chop the input string according to the bounds of the displaying area

Prototype void StrChopString(ObjectBounds *bounds, BYTE *string, BYTE font, BOOLEAN dotdot, SHORT margin)

Scope Application/System

Input Parameters

bounds	Pointer to bounds for displaying the string
string	The input string
font	font size
dotdot	whether dots should be displayed or not if the input string cannot be fully displayed
margin	the number of pixel of margin on the left and right

Output Parameters None

Result None

Comment None

6 StrExtract

Purpose This function is called to extract a portion of string from a longer input string by specifying the starting character and the number of the characters to be extracted

Prototype
```
void StrExtract(BYTE *string1, WORD start, WORD length,
                BYTE *string2)
```

Scope Application/System

Input Parameters

string1	Pointer to the input string
start	the character position of the starting character of the extracted string in the input string
length	the number of characters to be extracted

Output Parameters

string2	Pointer to the extracted string

Result None

Comment None

7 StrGetInsertPtPos

Purpose To calculate the insert point character position, the corrected x-coordinate of insert point and the corrected y-coordinate of the insert point by giving the expected x-coordinate and y-coordinate of the insert point

Prototype
```
void StrGetInsertPtPos(Field *field_ptr, SHORT xcoord,
                       SHORT ycoord, WORD *insert_pos,
                       SHORT *insert_pt_x, SHORT *insert_pt_y)
```

Scope Application/System

Input Parameters

field_ptr	Pointer to a field object
xcoord	Expected x-coordinate of the insert point

	ycoord	Expected y-coordinate of the insert point
Output Parameters	insert_pos	Corrected insert point character position
	insert_pt_x	Corrected x-coordinate of the insert point
	insert_pt_y	Corrected y-coordinate of the insert point

Result None

Comment If the expected x and y-coordinate of the insert point is out of the bounds of the field object, then the corrected insert point will be set at the left-top corner of the field object or the bottom-right corner of the field object depending on the expected y-coordinate of the insert point.

8 StrGetWidth

Purpose To calculate the total number of pixel that is required to display the string on screen

Prototype UWORD StrGetWidth(BYTE *string, BYTE font)

Scope Application/System

Input Parameters	string1	Pointer to the input string
	font	the font size

Output Parameters None

Result The total number of pixel that is required to display the string on screen

Comment the input string must be null-terminated

9 StrTextboxAnalyze

Purpose To calculate the line information of text in a textbox object

Prototype void StrTextboxAnalyze(Textbox *tb_ptr)

Scope Application/System

Input Parameters	tb_ptr	Pointer to a textbox object

Output Parameters None

Result None

Comment None

10 StrTextboxCharPosToXY

Purpose To calculate the x-coordinate and y-coordinate of the left-top corner of a character that is being displayed within the bounds of the field object

Prototype void StrTextboxCharPosToXY(Textbox *tb_ptr, WORD char_pos, SHORT *xcoord, SHORT *ycoord)

Scope Application/System

Input Parameters

tb_ptr	Pointer to a txtbox object
char_pos	character position

Output Parameters

xcoord	Pointer to the value of the x-coordinate of the left-top corner of the character
ycoord	Pointer to the value of the y-coordinate of the left-top corner of the character

Result None

Comment The input character position must be lager than or equal to the character position of the character that is being displayed on the left of the field object. Also, the input character position must be smaller than or equal to the character position of the character that is being displayed on the right of the field object.

11 StrTextboxGetInsertPtPos

Purpose To calculate the insert point character position, the corrected x-coordinate of insert point and the corrected y-coordinate of the insert point by giving the expected x-coordinate and y-coordinate of the insert point

Prototype void StrTextboxGetInsertPtPos(Textbox *tb_ptr, SHORT xcoord, SHORT ycoord, WORD *insert_pos, SHORT *insert_pt_x,

SHORT *insert_pt_y)

Scope Application/System

Input Parameters

tb_ptr	Pointer to a textbox object
xcoord	Expected x-coordinate of the insert point
ycoord	Expected y-coordinate of the insert point

Output Parameters

insert_pos	Corrected insert point character position
insert_pt_x	Corrected x-coordinate of the insert point
insert_pt_y	Corrected y-coordinate of the insert point

Result None

Comment If the expected x and y-coordinate of the insert point is out of the bounds of the field object, then the corrected insert point will be set at the left edge of the textbox object or the right edge of the textbox object depending on the expected x-coordinate of the insert point.

12 StrTextboxXYToCharPos

Purpose To calculate the character position of the character that is pointed by the input x-coordinate and the input y-coordinate

Prototype void StrTextboxXYToCharPos(Textbox *tb_ptr, SHORT xcoord, SHORT ycoord, WORD *char_pos)

Scope Application/System

Input Parameters

tb_ptr	Pointer to a textbox object
xcoord	The input x-coordinate
ycoord	The input y-coordinate

Output Parameters

char_pos	Pointer to the value of the calculated character position

Result None

Comment None

13 StrXYToCharPos

Purpose To calculate the character position of the character that is pointed by the input x-coordinate and the input y-coordinate

Prototype void StrXYToCharPos(Field *field_ptr, SHORT xcoord,
SHORT ycoord, WORD *char_pos)

Scope Application/System

Input Parameters

field_ptr	Pointer to a field object
xcoord	The input x-coordinate
ycoord	The input y-coordinate

Output Parameters

char_pos	Pointer to the value of the calculated character position

Result None

Comment None

System.c Functions

1 SysApplicationHandleEvent

Purpose To perform system-form switching and set the active form of the system to a system-form. The SystemFormDispatchEvent function is mapped to a corresponding form event handler function.

Prototype BOOLEAN SysApplicationHandleEvent(EvtType *Event)

Scope System

Input Parameters

Event	input event

Output Parameters None

Result

TRUE	if event is handled
FALSE	if event is not handled

Comment This function is only called by SysHandleEvent

2 SysTranslateCustomiseButtons

Purpose To translate the customized short-cut keys or inlay regions to pre-defined functions. When user press one of the short-cut key, event is sent to this function. This function checks the key mapping in System Setup application and performs the corresponding functions.

Prototype BOOLEAN SysTranslateCustomiseButtons(EvtType *Event)

Scope System

Input Parameters

Event	input event

Output Parameters None

Result

TRUE	if event is handled
FALSE	if event is not handled

Comment None

3 SysTranslateCustomiseButtonsWithoutPowerOff

Purpose This function is same as SysTranslatecustomiseButtons, except that it does not handle the POWER OFF button

Prototype BOOLEAN SysTranslateCustomiseButtonsWithoutPowerOff(EvtType *Event)

Scope System

Input Parameters

Event	input event

Output Parameters None

Result

TRUE	if event is handled
FALSE	if event is not handled

Comment None

4 SysApplicationInit

Purpose It is a function only called by the Kernel to reset a number of system parameters and status during application switching

Prototype BOOLEAN SysApplicationInit ()

Scope System

Input Parameters None

Output Parameters None

Result

TRUE	handled
FALSE	not handled

Comment The parameters and status, that are reset, include:

1. keyboard is set back to be visible
2. the short-cut key is set back to clickable
3. alarm alert is turned back on
4. the insert point on the screen is removed
5. battery warning is allowed

5 SystemButtonTranslation

Purpose To translate the customized short-cut keys or inlay regions to pre-defined functions. When user press one of the short-cut key, event is sent to this function. This function checks the key mapping in System Setup application and performs the corresponding functions.

Prototype BOOLEAN SystemButtonTranslation(BYTE key)

Scope System

Input Parameters

key	the mapped-to key	
	SYSETUP_APP_NONE	0
	SYSETUP_ANNIVERSARIES	1
	SYSETUP_CALCULATOR	2
	SYSETUP_EMAIL	3

	SYSETUP_EXPENSE	4
	SYSETUP_GLOBAL_FIND	5
	SYSETUP_MEMO	6
	SYSETUP_PHONEBOOK	7
	SYSETUP_TO_DO_LIST	8
	SYSETUP_SCHEDULER	9
	SYSETUP_SKETCH	10
	SYSETUP_SYSTEM_SETUP	11
	SYSETUP_PC_SYNC	12
	SYSETUP_VOICE_MEMO	13
	SYSETUP_JOT	14

Output Parameters None

Result

TRUE	handled
FALSE	not handled

Comment None

6 SystemButtonTranslationWithoutPowerOff

Purpose This function is same as SystemButtonTranslationWithoutPowerOff, except that it does not handle the POWER OFF button

Prototype BOOLEAN SystemButtonTranslationWithoutPowerOff(BYTE key)

Scope System

Input Parameters

key	the mapped-to key	
	SYSETUP_APP_NONE	0
	SYSETUP_ANNIVERSARIES	1
	SYSETUP_CALCULATOR	2
	SYSETUP_EMAIL	3
	SYSETUP_EXPENSE	4
	SYSETUP_GLOBAL_FIND	5
	SYSETUP_MEMO	6
	SYSETUP_PHONEBOOK	7
	SYSETUP_TO_DO_LIST	8
	SYSETUP_SCHEDULER	9
	SYSETUP_SKETCH	10
	SYSETUP_SYSTEM_SETUP	11
	SYSETUP_PC_SYNC	12
	SYSETUP_VOICE_MEMO	13
	SYSETUP_JOT	14

Output Parameters None

Result

TRUE	handled
FALSE	not handled

Comment None

7 SystemCheckMappedHardwareKeyStatus

Purpose This function is called to get the status (whether clicking on the short-cut key is possible or not) of the short-cut keys

Prototype BOOLEAN SystemCheckMappedHardwareKeyStatus()

Scope System

Input Parameters None

Output Parameters None

Result

TRUE	handled
FALSE	not handled

Comment None

8 SystemHandleEvent

Purpose This function includes the system foreground functions. Inside the *SystemHandleEvent*, all the system foreground functions are placed in hierarchy too.

Prototype BOOLEAN SystemHandleEvent(EvtType *Event)

Scope System

Input Parameters

Event	input event

Output Parameters None

Result

TRUE	handled

	FALSE	not handled

Comment The SystemHandleEvent includes FormHandleEvent, KeyboardHandleEvent, InlayHandleEvent, etc…

9 SystemSetMappedHardwareKeyStatus

Purpose This function is called to set whether to enable or disable the usaof the short-cut keys

Prototype void SystemSetMappedHardwareKeyStatus(BOOLEAN status)

Scope System

Input Parameters	status	TRUE if enable the use of short-cut keys FALSE if disable the use of short-cut keys

Output Parameters None

Result None

Comment None

tlbc.c

1 TlbDirtyEntry

Purpose Dirty TLB content of specific ASID

Prototype void TlbDirtyEntry(UWORD sys_asid)

Scope OS

Input Parameters	Sys_asid	process ID

Output Parameters None

Result None

Comment None

2 TlbGetASID

Purpose Get current ASID

Prototype UWORD TlbGetASID()

Scope OS

Input Parameters None

Output Parameters None

Result Current ASID

Comment None

3 TlbGetBadVPN

Purpose Get the bad VPN

Prototype UWORD TlbGetBadVPN()

Scope OS

Input Parameters None

Output Parameters None

Result Bad VPN

Comment None

4 TlbGetBVAddr

Purpose Get bad virtual address

Prototype UWORD TlbGetBVAddr()

Scope OS

Input Parameters None

Output Parameters None

Result Bad virtual address

Comment None

5 TlbGetRand

Purpose Get Random number

Prototype UWORD TlbGetRand()

Scope OS

Input Parameters None

Output Parameters None

Result Random number : 8 – 63

Comment None

6 TlbInit

Purpose Initialize (Clear) TLB content of all entries

Prototype void TlbInit()

Scope OS

Input Parameters None

Output Parameters None

Result None

Comment None

7 TlbSetASID

Purpose Set current ASID

Prototype void TlbSetASID(UWORD sys_asid)

Scope OS

Input Parameters Current ASID value

Output Parameters None

Result None

Comment None

tmrapi.c

1 AlmDecode

Purpose To decode the alarm message

Prototype void AlmDecode(MsgType *msg)

Scope System

Input Parameters msg Pointer to system message

Output Parameters None

Result None

Comment None

2 AlmGetTime

Purpose To get alarm time

Prototype void AlmGetTime(RTM *time)

Scope System

Input Parameters time Pointer to system time structure

Output Parameters None

Result None

Comment None

3 AlmISR

Purpose alarm interrupt service routine

Prototype void AlmISR()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

4 AlmSetStatus

Purpose To enable or disable the alarm

Prototype void AlmSetStatus(BOOLEAN status)

Scope System

Input Parameters

status	TRUE if alarm is enabled FALSE if alarm is disabled

Output Parameters None

Result None

Comment None

5 AlmSetTime

Purpose To set alarm time value

Prototype void AlmSetTime(RTM *time)

Scope None

Input Parameters

time	Pointer to alarm time to be set

Output Parameters None

Result None

Comment None

6 ClearAppTmr

Purpose To clear application created timer

Prototype void ClearAppTmr()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

7 Hcf

Purpose To compute HCF of two numbers

Prototype UWORD Hcf(UWORD n1, UWORD n2)

Scope System

Input Parameters

n1	first number
n2	second number

Output Parameters None

Result HCF of the number

Comment None

8 Lcm

Purpose To compute LCM of two numbers

Prototype UWORD Lcm(UWORD n1, UWORD n2)

Scope System

Input Parameters

n1	first number
n2	second number

Output Parameters None

Result the LCM of the two input numbers

Comment None

9 LcmCnt

Purpose To find the LCM of periods of all interrupts

Prototype UWORD LcmCnt()

Scope System

Input Parameters None

Output Parameters None

Result LCM of periods of all interrupts

Comment None

10 LLAdd

Purpose To add two numbers of type long long

Prototype LLONG LLAdd(LLONG n1, LLONG n2)

Scope System

Input Parameters

n1	first long long type number
n2	second long long type number

Output Parameters None

Result the sum of the two input long long type number

Comment None

11 LLDiv

Purpose To divide one number of type long long to another of type word

Prototype LLONG LLDiv(LLONG n1, UWORD n2, UWORD *n3_r)

Scope System

Input Parameters

n1	the first long long type number
n2	the second long long type number

Output Parameters	n3_r	Pointer to the remainder of the result
Result	the result of long long type	
Comment	None	

12 LLMin

Purpose	To subtract one number of type long long from another	
Prototype	LLONG LLMin(LLONG n1, LLONG n2)	
Scope	System	
Input Parameters	n1 n2	the first number the second number
Output Parameters	None	
Result	the result (n1 – n2)	
Comment	None	

13 LLMul1

Purpose	To multiple one number of type long long to another of type word	
Prototype	LLONG LLMul1(UWORD n1, UWORD n2)	
Scope	System	
Input Parameters	n1 n2	the first input long long type number the second input UWORD type number
Output Parameters	None	
Result	the result (n1 * n2)	
Comment	None	

14 LLMul2

Purpose To multiple two numbers of type long long

Prototype LLONG LLMul2(LLONG n1, LLONG n2)

Scope System

Input Parameters

n1	the first long long type number
n2	the second long long type number

Output Parameters None

Result the result (n1 * n2)

Comment None

15 RtcCheckYMDValid

Purpose To check whether the input date is valid or not

Prototype

```
BOOLEAN RtcCheckYMDValid(SHORT year,
                         SHORT mon,
                         SHORT mday)
```

Scope Application/System

Input Parameters

year	the year of the date to be checked
mon	the month of the date to be checked
mday	the day of month of the date to be checked

Output Parameters None

Result

TRUE	valid
FALSE	invalid

Comment None

16 RtcClr

Purpose To clear real time clock

Prototype void RtcClr()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

17 RtcCompareTime

Purpose To compare two input times

Prototype BOOLEAN RtcCompareTime(RTM *time_1, RTM *time_2)

Scope Application/System

Input Parameters

time_1	Pointer to RTM structured time
time_2	Pointer to RTM structured time

Output Parameters None

Result

TRUE	if time_1 > time_2
FALSE	otherwise

Comment

18 RtcDaysShift

Purpose To calculate the correct date with a number of days shifted

Prototype void RtcDaysShift(SHORT *year, SHORT *mon, SHORT *mday,
SHORT day)

Scope Application/System

Input Parameters

*year	Pointer to the input year
*mon	Pointer to the input month
*mday	Pointer to the input day of month

	day	Number of days to be shifted
Output Parameters	*year	Pointer to the output year
	*mon	Pointer to the output month
	*mday	Pointer to the output day of month

Result None

Comment None

19 RtcDecode

Purpose To decode the RTC message

Prototype void RtcDecode(MsgType *msg)

Scope System

Input Parameters	msg	Pointer to system message

Output Parameters None

Result None

Comment None

20 RtcDiffTime

Purpose To calculate the difference of the two times of RTM format

Prototype LLONG RtcDiffTime(RTM *time_1, RTM *time_2)

Scope Application/System

Input Parameters	time_1	Pointer to first RTM input time
	time_2	Pointer to second RTM input_time

Output Parameters None

Result Elapsed time in seconds, from time1 to time2

Comment None

21 RtcFormatDate

Purpose To output a formatted date string

Prototype
```
SHORT RtcFormatDate(RTM *in_date,
                    BYTE *date_pref,
                    BYTE *out_buf)
```

Scope Application/System

Input Parameters

in_date	RTM date to be formatted

Output Parameters

date_pref	Pointer to output format
out_buf	Pointer to output date buffer

Result

FALSE	if can't format the input date

Comment The following is the date format:

NULL	system default
M	month, 1 or 2 digit
MM	month, 2 digit
D	day, 1 or 2 digit
DD	day, 2 digit
Y	year, 2 digit
YY	year, 4 digit
h	hour, 24 time format, 1 or 2 digit
hh	hour, 24 time format, 2 digit
H	hour, 12 time format, 1 or 2 digit
HH	hour, 12 time format, 2 digit
m	minute, 1 or 2 digit
mm	minute, 2 digit
s	second, 1 or 2 digit
ss	second, 2 digit
e	hundred-second, 1 or 2 digit
ee	hundred-second, 2 digit
A	AM / PM
a	am / pm
~	force to print next character, not check keyword
W	weekday, 3 character, first character in capital
WW	weekday, full name, first character in capital
w	weekday, 3 character
ww	weekday, full name
O	month name, 3 character, first character in capital

OO month name, full name, first character in capital
o month name, 3 character
oo month name, full name

22 RtcGetDaysInMonth

Purpose To get the number of days in a particular year and month

Prototype SHORT RtcGetDaysInMonth(SHORT year, SHORT mon)

Scope Application/System

Input Parameters

year	input year
mon	input month

Output Parameters None

Result the number of days

Comment None

23 RtcGetTime

Purpose To get the current time and data of the system

Prototype void RtcGetTime(RTM *time)

Scope Application/System

Input Parameters

time	Pointer to the RTM time structure

Output Parameters None

Result None

Comment None

24 RtcGetTimeEvt

Purpose To get the current time and event through sending event

Prototype void RtcGetTimeEvt()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment The current time and data is sent to the application through event

25 RtcISR

Purpose To real time clock interrupt service routine

Prototype void RtcISR()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

26 RtcRollOver

Purpose To force the real time clock to roll over

Prototype void RtcRollOver()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

27 RtcSetTime

Purpose To set time value for real time clock

Prototype void RtcSetTime(RTM *time)

Scope Application/System

Input Parameters time Pointer to the input RTM structure

Output Parameters None

Result None

Comment None

28 RtcToTime

Purpose Convert binary counter into time of RTM format

Prototype void RtcToTime(LLONG cnt, RTM *time)

Scope System

Input Parameters cnt binary long long type counter

Output Parameters time Pointer to output time format

Result None

Comment None

29 RtcWday1ToYMD

Purpose To calculate the day of month

Prototype BOOLEAN RtcWday1ToYMD(SHORT year, SHORT mon,
SHORT wday, SHORT nth_week,

SHORT *mday)

Scope Application/System

Input Parameters

year	the input year
mon	the input month
wday	the input day of week
nth_week	the input nth of week in the input month

Output Parameters

mday	Pointer to the value of day of month

Result

TRUE	calculate successfully
FALSE	cannot be calculated out

Comment None

30 RtcWday2ToYMD

Purpose To calculate the day of month

Prototype BOOLEAN RtcWday2ToYMD(SHORT year, SHORT mon,
SHORT wday, SHORT nth_wday,
SHORT *mday)

Scope Application/System

Input Parameters

year	the input year
mon	the input month
wday	the input day of week
nth_week	the input nth of week in the input month

Output Parameters

mday	Pointer to the value of day of month

Result

TRUE	calculate successfully
FALSE	cannot be calculated out

Comment None

31 RtcWday3ToYMD

Purpose To calculate the day of month and month

Prototype
```
void RtcWday3ToYMD(SHORT year, SHORT nth_wday,
                   SHORT wday,
                   SHORT *mon, SHORT *mday)
```

Scope Application/System

Input Parameters

year	the input year
nth_wday	the input nth week day
wday	the input week day

Output Parameters

mon	Pointer to the output month
mday	Pointer to the output day of month

Result None

Comment None

32 RtcWday4ToYMD

Purpose To calculate the day of month and month

Prototype
```
BOOLEAN RtcWday4ToYMD(SHORT year, SHORT nth_week,
                      SHORT wday,
                      SHORT *mon, SHORT *mday)
```

Scope Application/System

Input Parameters

year	the input year
nth_week	the input nth of week
wday	the week day

Output Parameters

mon	Pointer to month
mday	Pointer to day of month

Result

TRUE	calculate successfully
FALSE	cannot calculate out

Comment None

33 RtcWday5ToYMD

Purpose To calculate month and day of month

Prototype
```
BOOLEAN RtcWday5ToYMD(SHORT year, SHORT nth_week,
                      SHORT wday,
                      SHORT *mon, SHORT *mday)
```

Scope Application/System

Input Parameters

year	the input year
nth_week	the input nth week

Output Parameters

mon	Pointer to output month
mday	Pointer to output day of month

Result

TRUE	calculate successfully
FALSE	cannot calculate out

Comment None

34 RtcYMDToWday1

Purpose To calculate nth number of week and day of week

Prototype
```
void RtcYMDToWday1(SHORT year, SHORT mon,
                   SHORT mday,
                   SHORT *wday, SHORT *nth_week)
```

Scope Application/System

Input Parameters

year	the input year
mon	the input month
mday	the input day of month

Output Parameters

wday	Pointer to day of week
nth_week	Pointer to nth week

Result None

Comment None

35 RtcYMDToWday2

Purpose To calculate day of month, day of week and nth number of day of week

Prototype
```
void RtcYMDToWday2(SHORT year, SHORT mon,
                   SHORT mday, SHORT *wday,
                   SHORT *nth_wday)
```

Scope Application/System

Input Parameters

year	the input year
mon	the input month
mday	the input day of month

Output Parameters

wday	Pointer to day of week
nth_wday	Pointer to the nth of day of week

Result None

Comment None

36 RtcYMDToWday3

Purpose To calculate the day of week and nth number of day of week

Prototype
```
void RtcYMDToWday3(SHORT year, SHORT mon,
                   SHORT mday,
                   SHORT *wday, SHORT *nth_wday)
```

Scope Application/System

Input Parameters

year	the input year
mon	the input month
mday	the input day of month

Output Parameters

wday	Pointer to the day of week
nth_wday	Pointer to the nth number of day of week

Result None

Comment None

37 RtcYMDToWday4

Purpose To calculate day of week and nth number of week

Prototype void RtcYMDToWday4(SHORT year, SHORT mon, SHORT mday, SHORT *wday, SHORT *nth_week)

Scope Application/System

Input Parameters

year	the input year
mon	the input month
mday	the input day of month

Output Parameters

wday	Pointer to day of week
nth_week	Pointer to the nth number of week

Result None

Comment None

38 TimeToRtc

Purpose To convert time of RTM format into binary counter

Prototype LLONG TimeToRtc(RTM time)

Scope Application/System

Input Parameters

time	the RTM time structure

Output Parameters None

Result a long long type value

Comment None

39 TmrComputeTimeBase

Purpose To calculate the time base of the timer

Prototype void TmrComputeTimeBase()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

40 TmrInit

Purpose To initialise the timer manager

Prototype void TmrInit()

Scope System

Input Parameters None

Output Parameters None

Result None

Comment None

41 TmrIntDisable

Purpose To disable a particular timer interrupt

Prototype void TmrIntDisable(UWORD label)

Scope System

Input Parameters label timer label

Output Parameters None

Result None

Comment None

42 TmrIntDisableAll

Purpose To disable all timer interrupts

Prototype void TmrIntDisableAll()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

43 TmrIntEnable

Purpose To enable timer interrupt

Prototype UWORD TmrIntEnable(UWORD period, void *fptr)

Scope Application/System

Input Parameters

period	timer period
fptr	callback function

Output Parameters None

Result timer label

Comment None

44 TmrIntEnableInt

Purpose To enable timer interrupt

Prototype UWORD TmrIntEnableInt(UWORD period, void *fptr)

Scope Application/System

Input Parameters

period	timer period
fptr	callback function

Output Parameters None

Result timer label

Comment This function should be called in interrupt service routine

45 TmrIntEnableIntPen

Purpose To enable the timer interrupt for pen

Prototype UWORD TmrIntEnableIntPen(UWORD period, void *fptr)

Scope System

Input Parameters

period	the period of the timer
fptr	callback function

Output Parameters None

Result timer label

Comment None

46 TmrISR

Purpose Timer interrupt service routine

Prototype void TmrISR()

Scope Aplication\System

Input Parameters None

Output Parameters None

Result None

Comment None

47 TmrQuantizePeriod

Purpose To quantise the time period

Prototype UWORD TmrQuantizePeriod(UWORD period)

Scope Application\System

Input Parameters period timer period

Output Parameters None

Result quantised period

Comment None

48 TmrWaitTime

Purpose timer waits for a interval

Prototype void TmrWaitTime(UWORD interval)

Scope System

Input Parameters interval a number in seconds

Output Parameters None

Result None

Comment None

49 WordBreak

Purpose	To break the number of type word into long long	
Prototype	LLONG WordBreak(UWORD n1)	
Scope	System	
Input Parameters	n1	the UWORD input number
Output Parameters	None	
Result	result long long	
Comment	None	

Uart Manager Functions List (Sys)

1 ComGetUartBuffer

Prototype	void ComGetUartBuffer(WORD port)	
Purpose	Get uart data from device driver	
Scope	Internal	
Input Parameters	port	Uart port number, UART_A or UART_B
Output Parameters	port_hdle	The handle to access the port
Return	None	
Comments		

2 ComReset

Prototype	void ComReset(void)
Purpose	Release all uart port (set to free)

Scope	Internal
Input Parameters	None
Output Parameters	None
Return	None
Comments	

3 ComWriteOutData

Prototype	void ComWriteOutData(WORD port)	
Purpose	Flush uart port data	
Scope	Internal	
Input Parameters	port	Uart port, UART_A or UART_B
Output Parameters	None	
Return		
Comments		

4 UartMgrInit

Prototype	void UartMgrInit(void)	
Purpose	Initialise uart manager	
Scope	OS	
Input Parameters	port	Uart port, UART_A or UART_B
Output Parameters	None	

Return None

Comments

Uart Manager Functions Summary (User)

1 ComCapture

Prototype Err ComCapture(WORD port_hdle, ComMode mode, UWORD time_out, UWORD buffer_size, BYTE *buffer)

Purpose Start to capture data from uart port

Scope All

Input Parameters

port_hdle	Handle to the uart port
mode	Port mode, COM_MODE_CHAR or COM_MODE_BUFFER
time_out	Time out in millisecond if in buffer mode
buffer_size	Size of data buffer if in buffer mode
buffer	Pointer to the application allocated buffer

Output Parameters None

Return

TRUE	Success
ERR_COM_INV_HANDLE	Invalid handle
ERR_COM_INV_BUFFER	Invalid buffer pointer or buffer size
ERR_COM_INV_PARAM	Invalid capture mode

Comments Application need to:
Call ComConfigPort() to configure the port
Call ComStart() to capture uart data
Call *ComClosePort()* to release the port

2 ComClosePort

Prototype	Err ComClosePort(WORD port_hdle)	
Purpose	Close an uart port	
Scope	All	
Input Parameters	port_hdle	Handle of the port
Output Parameters	None	
Return	TRUE	Success
	ERR_COM_INV_HANDLE	Invalid handle
Comments	Application MUST call ComClosePort() to release the port or other application cannot use the port	

3 ComConfigPort

Prototype	Err ComConfigPort(WORD port_hdle, WORD baud, WORD config)	
Purpose	Set uart port configuration	
Scope	All	
Input Parameters	port_hdle	Handle of the port
	baud	Baud rate (e.g. UART_BAUD9600)
	config	Data bit, parity and stop bit (e.g. UART_PARITY_NONE | UART_DATABIT8 | UART_STOPBIT1)
Output Parameters	None	
Return	TRUE	Success
	ERR_COM_INV_HANDLE	Invalid handle
	ERR_COM_INV_PARAM	Invalid baud rate

Comments
1. Should call *ComConfigPort*() after open the port
2. Available baud rate:
300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
3. Available parity option:
UART_PARITY_NONE, UART_PARITY_EVEN,
UART_PARITY_ODD
4. Available data bit option:
UART_DATABIT7, UART_DATABIT8
5. Available stop bit option:
UART_STOPBIT1, UART_STOPBIT2

4 ComGetConfig

Prototype: Err ComGetConfig(WORD port_hdle, WORD *baud, WORD *config)

Purpose: Get uart port configuration

Scope: All

Input Parameters

port_hdle	Handle of the port

Output Parameters

baud	Baud rate
config	Data bit, parity and stop bit

Return

TRUE	Success
ERR_COM_INV_HANDLE	Invalid handle

Comments

5 ComGetMode

Prototype: Err ComGetMode(WORD port_hdle, ComMode *mode)

Purpose: Get the port mode

Scope: All

Input Parameters

port_hdle	Handle of the port

Output Parameters	mode	Current modeCOM_MODE_BUFFER, or COM_MODE_CHAR
Return	TRUE	Success
	ERR_COM_INV_HANDLE	Invalid handle
Comments		

6 ComGetUartBuffer

Prototype	void ComGetUartBuffer(WORD port)	
Purpose	Get uart data from device driver	
Scope	Internal	
Input Parameters	port	Uart port number, UART_A or UART_B
Output Parameters	port_hdle	The handle to access the port
Return	None	
Comments		

7 ComIsHandleValid

Prototype	BOOLEAN ComIsHandleValid(WORD port_hdle)	
Purpose	Check the handle is valid or not	
Scope	All	
Input Parameters	port_hdle	Handle to check
Output Parameters	None	
Return	TRUE	Handle is valid

	FALSE	Handle is invalid

Comments

8 ComOpenPort

Prototype: Err ComOpenPort(WORD port, WORD *port_hdle)

Purpose: Open an uart port

Scope: All

Input Parameters	port	Uart port number, UART_A or UART_B
Output Parameters	port_hdle	The handle to access the port
Return	TRUE	Success
	ERR_COM_PORT_BUSY	Port is opened by another application
	ERR_COM_INV_PARAM	Invalid port number

Comments:
Application need to:
Call ComConfigPort() to configure the port
Call ComCapture() to capture uart data
Call *ComClosePort()* to release the port

10 ComSendChar

Prototype: Err ComSendChar(WORD port_hdle, BYTE data)

Purpose: Send a 8-bit data

Scope: All

Input Parameters	port_hdle	Handle of the port
	data	Data to send
Output Parameters	None	

Return	TRUE	Success
	ERR_COM_INV_HANDLE	Handle is invalid

Comments

11 ComSendData

Prototype: Err ComSendData(WORD port_hdle, BYTE **buffer, UWORD buffer_size)

Purpose: Send data through uart port

Scope: All

Input Parameters:

port_hdle	Handle of the port
buffer	Data to send
buffer_size	Size of buffer in byte

Output Parameters: None

Return	TRUE	Success
	ERR_COM_INV_HANDLE	Handle is invalid

Comments

uart.c

XXXX UartADmaFullIsr

Purpose Interrupt service routine of transmit dma interrupt

Prototype void UartADmaFullIsr ()

Scope System

Input Parameters port : UART Port Number

Output Parameters None

Result Content of UART Control register

Comment None

1 UartChkCfg

Purpose To get the UART configuration

Prototype WORD UartChkCfg(WORD port)

Scope System

Input Parameters port : UART Port Number

Output Parameters None

Result Content of UART Control register

Comment None

2 UartChkRxDmaBusy

Purpose Check if receive dma is ready or not

Prototype WORD UartChkRxDmaBusy(WORD port)

Scope System

Input Parameters port : UART Port Number

Output Parameters None

Result 1: Busy
0: Ready

Comment

3 UartChkStatus

Purpose To check UART status

Prototype WORD UartChkStatus(WORD port)

Scope System
Input Parameters port: UART port number
Output Parameters None
Result Status of UART port
Comment None

4 UartChkTxDmaBusy

Purpose Check if transmit dma is ready or not
Prototype WORD UartChkTxDmaBusy(WORD port)

Scope System
Input Parameters port: UART port number
Output Parameters None
Result 1: Busy
0: Ready
Comment

5 UartDisable

Purpose Disable UART port
Prototype void UartDisable(WORD port)

Scope System / Application
Input Parameters port: UART port number
Output Parameters None
Result None
Comment None

6 UartEnable

Purpose To enable UART port
Prototype void UartEnable(WORD port)

Scope System / Application

Input Parameters port: UART port number
Output Parameters None
Result None
Comment None

8 UartInit

Purpose Initialize the UART port
Prototype void UartInit(WORD port, WORD baud, WORD config)

Scope System / Application
Input Parameters port: UART port number
baud: e.g. 9600 – 115200
config: e.g. “N81”
Output Parameters None
Result None
Comment None

9 UartIsrRx

Purpose Interrupt service routine of transmit dma interrupt
Prototype void UartIsrRx()

Scope System
Input Parameters None

Output Parameters None
Result None
Comment None

10 UartReadBuf

Purpose Retrieve a buffer of data from UART port input buffer
Prototype WORD UartReadBuf(WORD port, BYTE *buf)

Scope System
Input Parameters port: UART port number
buf: address of input buffer
Output Parameters buf: buffered data
Result None
Comment None

11 UartWriteBuf

Purpose Write a buffer of data to the UART transmit buffer
Prototype void UartWriteBuf(WORD port, BYTE *buf, WORD length)

Scope System
Input Parameters port: UART port number
buf: address of output buffer
length: length of buffer to be sent
Output Parameters None
Result None
Comment None

uartmgr.c

1 ComCapture

Purpose To start to capture the uart port data

Prototype
```
Err ComCapture(WORD port_hdle, ComMode mode,
               UWORD time_out,
               UWORD buffer_size,
               BYTE *buffer)
```

Scope Application/System

Input Parameters

port_hdle	handle of the port
mode	port mode, COM_MODE_CHAR or COM_MODE_BUFFER
time_out	time out if in buffer mode
buffer_size	buffer size, if in buffer mode
buffer	the buffer pointer, if in buffer mode

Output Parameters None

Result

TRUE	Success
ERR_COM_INV_HANDLE	invalid handle
ERR_COM_INV_PARAM	mode is invalid
ERR_COM_INV_BUFFER	buffer = NULL or buffer_size = 0, if in buffer mode

Comment uart data will not capture with ComOpenPort(), call ComCapture to start capture uart data

2 ComClosePort

Purpose To close an uart port

Prototype Err ComClosePort(WORD port_hdle)

Scope Application/System

Input Parameters port_hdle handle of the port

Output Parameters None

Result TRUE Success
ERR_COM_INV_HANDLE
invalid handle

Comment None

3 ComConfigPort

Purpose To set the baud, parity, stop bit of the port

Prototype Err ComConfigPort(WORD port_hdle, WORD baud, WORD config)

Scope Application/system

Input Parameters

port_hdle	handle of the port
baud	the baud rate to set e.g. UART_BAUD115200
config	Parity | Databit | Stop bit e.g. UART_PARITY_NONE | UART_DATABIT8 | UART_STOPBIT1

Output Parameters None

Result TRUE Success
ERR_COM_INV_HANDLE
invalid handle
ERR_COM_INV_BUAD_RATE
invalid baud rate

Comment call this after ComOpenPort()

4 ComGetConfig

Purpose To get uart port configuration

Prototype Err ComGetConfig(WORD port_hdle, WORD *baud, WORD *config)

Scope Application/System

Input Parameters port_hdle - handle of the port

Output Parameters

baud	baud rate, e.g. UART_BAUD38400
config	Parity | Databit | Stop bit

Result None

Comment None

5 ComGetDataMode

Purpose To get the data mode

Prototype int ComGetDataMode()

Scope Application/system

Input Parameters None

Output Parameters None

Result the data mode

Comment None

6 ComGetMode

Purpose To get the port mode

Prototype Err ComGetMode(WORD port_hdle, ComMode *mode)

Scope Application/System

Input Parameters

port_hdle	handle of the port

Output Parameters

mode	port mode

Result

TRUE	Success
ERR_COM_INV_HANDLE	invalid handle

Comment None

7 ComGetUartBuffer

Purpose To get uart data from device driver

Prototype void ComGetUartBuffer(WORD port)

Scope Application/System

Input Parameters

port	uart port, UART_A or UART_B

Output Parameters None

Result None

Comment None

8 ComIsHandleValid

Purpose To check if the handle is valid or not

Prototype BOOLEAN ComIsHandleValid(WORD port_hdle)

Scope Application/System

Input Parameters

port_hdle	the handle to check

Output Parameters None

Result

TRUE	the handle is valid
FALSE	the handle is invalid

Comment None

9 ComOpenPort

Purpose To open an uart port

Prototype Err ComOpenPort(WORD port, WORD *port_hdle)

Scope Application/System

Input Parameters

port	port to open, UART_A or UART_B

Output Parameters

port_hdle	the handle to access this connection

Result

TRUE	Success
ERR_COM_PORT_BUSY	opened by other app
ERR_COM_INV_PORT	invalid port number

Comment None

10 ComReset

Purpose To reset uart ports and initilize Uart Manager

Prototype void ComReset()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

11 ComSendChar

Purpose To send a 8-bit data

Prototype Err ComSendChar(WORD port_hdle, BYTE data)

Scope Application/System

Input Parameters

port_hdle	handle of the port
data	data to send

Output Parameters None

Result

TRUE	Success
ERR_COM_INV_HANDLE	invalid handle
ERR_MEM_NO_MEM	not enough memory

Comment None

12 ComSendData

Purpose To send data

Prototype

```
Err ComSendData(WORD port_hdle,
                BYTE **buffer,
                UWORD buffer_size)
```

Scope Application/System

Input Parameters

port_hdle	handle of the port
buffer	data to send
buffer_size	size of buffer

Output Parameters None

Result

TRUE	Success
ERR_COM_INV_HANDLE	invalid handle
ERR_MEM_NO_MEM	not enough memory

Comment None

13 ComSetDataMode

Purpose To set data mode

Prototype

```
void ComSetDataMode(int mode)
```

Scope Application/System

Input Parameters

mode	data mode

Output Parameters None

Result None

Comment None

14 ComWriteOutData

Purpose To send data through Device driver

Prototype void ComWriteOutData(WORD port)

Scope Application/System

Input Parameters port port number

Output Parameters None

Result None

Comment None

15 DecodeData

Purpose To decode data

Prototype UBYTE DecodeData(UBYTE in, UBYTE *out)

Scope Application/System

Input Parameters in a byte

Output Parameters out Pointer to a byte

Result None

Comment None

16 HParam

Purpose To get the upper 2-byte of a 4-byte data

Prototype USHORT HParam(UWORD word)

Scope Application/System

Input Parameters word the 4-byte data

Output Parameters None

Result The upper 2-byte

Comment None

17 IsrInit

Purpose To initialize the interrupt

Prototype void IsrInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

18 Lparam

Purpose To get the lower 2-byte of a 4-byte data

Prototype USHORT LParam(UWORD word)

Scope Application/System

Input Parameters word the 4-byte data

Output Parameters None

Result The lower 2-byte

Comment None

19 SetHParam

Purpose To set the value of upper 2-byte of a 4-byte data

Prototype void SetHParam(UWORD *word, USHORT hdata)

Scope Application/System

Input Parameters

word	the new data
hdata	the 2-byte data

Output Parameters None

Result None

Comment None

20 SetLParam

Purpose To set the value of lower 2-byte of a 4-byte data

Prototype void SetLParam(UWORD *word, USHORT ldata)

Scope Application/System

Input Parameters

word	the new data
ldata	the 2-byte data

Output Parameters None

Result None

Comment None

21 UartGetc

Purpose To get a byte from uart port

Prototype BYTE UartGetc(WORD port)

Scope Application/System

Input Parameters port port number

Output Parameters None

Result byte return

Comment None

22 UartMgrInit

Purpose To initialize UartMgr

Prototype void UartMgrInit()

Scope Application/System

Input Parameters None

Output Parameters None

Result None

Comment None

23 UartPrintf

Purpose To printf

Prototype void UartPrintf(WORD port, BYTE *buffer)

Scope Application/System

Input Parameters port uart port
buffer byte pointer

Output Parameters None

Result None
Comment None

24 UartPrints

Purpose To print a byte pointer data to uart

Prototype void UartPrints(BYTE *c)

Scope Application/System

Input Parameters

c	Pointer to byte string

Output Parameters None

Result None

Comment None

25 UartPutc

Purpose To put a character to uart

Prototype void UartPutc(WORD port, BYTE data)

Scope Application/System

Input Parameters

port	the uart port
data	a byte

Output Parameters None

Result None

Comment None

26 UartRxData

Purpose To uart receive data

Prototype WORD UartRxData(WORD port, BYTE *data)

Scope Application/System

Input Parameters

port	uart port
data	the received byte string pointer

Output Parameters None

Result

UART_SUCCESS	success
UART_BUF_EMPTY	nothing in the buffer

Comment None

27 UartTxData

Purpose To transmit the data

Prototype WORD UartTxData(WORD port, BYTE data)

Scope Application/System

Input Parameters

port	uart port
data	a byte

Output Parameters None

Result

UART_SUCCESS
UART_BUF_FULL

Comment None

welcome.c Functions

1 Lowbatscr

Purpose To draw a Low Battery screen when the PDA is just powered on

Prototype BOOLEAN Lowbatscr()

Scope System

Input Parameters None

Output Parameters None

Result

TRUE	if battery level of the PDA is not empty
FALSE	if battery level of the PDA is empty

Comment None

2 Welscr

Purpose To draw a Welcome screen when the PDA is just powered on after hard-reset

Prototype BOOLEAN Welscr()

Scope System

Input Parameters None

Output Parameters None

Result

TRUE	if battery level of the PDA is not empty
FALSE	if battery level of the PDA is empty

Comment None

3 WelscrTmr

Purpose To process the welcome screen timer

Prototype void WelscrTmr()

Scope None

Input Parameters None

Output Parameters None

Result None

Comment After the welcome screen is drawn, a 5 seconds timer is set. After 5 seconds, this routine is called to erase the welcome screen