W90N745

System Library

User's Manual

Publication Release Date: 06/14/2006
Revision 1.0

This page intentionally left blank.

CONTENTS

W90N745 Supported C Library Functions Listing

Special C Library Functions:

1. Timer

sysGetTicks	get current selected timer in ticks
sysResetTicks	reset selected timer's tick counter
sysUpdateTickCount	update selected timer's tick counter
sysSetTimerReferenceClock	set selected timer's reference clock
sysStartTimer	setup own timer
sysStopTimer	stop timer
sysClearWatchDogTimerCount	clear Watchdog timer reset counter
sysClearWatchDogTimerInterruptStatus	clear Watchdog timer interrupt status
sysDisableWatchDogTimer	disable Watchdog timer
sysDisableWatchDogTimerReset	disable Watchdog timer reset function
sysEnableWatchDogTimer	enable Watchdog timer
sysEnableWatchDogTimerReset	enable Watchdog timer reset function
sysInstallWatchDogTimerISR	setup own Watchdog timer ISR
sysSetWatchDogTimerInterval	set Watchdog timer time-out interval
sysSetTimerEvent	set event of selected timer
sysClearTimerEvent	clear event of selected timer
sysSetLocalTime	set local time
sysGetCurrentTime	get current time which count by timer0
sysDelay	delay specific time which unit is 10ms

2. UART

sysGetChar	get a character from UART
sysInitializeUART	Initializes the COM port
sysPrintf	display message through RS-232 to terminal by interrupt mode
sysprintf	display message through RS-232 to terminal by polling mode
sysPutChar	put a character into UART

3. AIC

sysDisableInterrupt	disable interrupt source of mask register
sysEnableInterrupt	enable interrupt source of mask register

sysInstallExceptionHandler	setup own exception handler into interrupt vector table
sysInstallFiqHandler	setup own Fiq handler into interrupt vector table
sysInstallIrqHandler	setup own Irq handler into interrupt vector table
sysInstallISR	setup own ISR into AIC interrupt table
sysSetGlobalInterrupt	enable/disable all interrupts
sysSetInterruptPriorityLevel	set interrupt priority level
sysSetInterruptType	set interrupt source type
sysSetLocalInterrupt	enable/disable CPSR interrupts
sysSetAIC2SWMode	set AIC to software mode
sysGetInterruptEnableStatus	return specific interrupt source enable/disable status

4. Cache

sysEnableCache	enable system cache
sysDisableCache	disable system cache
sysFlushCache	flush system cache

5. External IO

sysSetExternalIO	initial external IO control
sysSetExternalIOTiming1	set up external IO access cycles
sysSetExternalIOTiming2	set up external IO chip selection time

6. Power Management

sysDisableAllPM_IRQ	Clean the PM IRQ status
sysEnablePM_IRQ	Save specific PM IRQ for wake up system
sysPMStart	Enter IDLE/MIDLE/PD power saving mode

NAME

sysGetTicks – get current selected timer in ticks

SYNOPSIS

UINT32 sysGetTicks(INT32 nTimeNo);

DESCRIPTION

This function gets the Timer 0 or Timer 1's current tick count.

PARAMETER

nTimeNo	TIMER0, TIMER1

RETURN VALUE

The current selected timer tick count.

EXAMPLE

```
/* Get current timer 0 tick count */
UINT32 btime;
btime = sysGetTicks(TIMER0);
```

NAME

sysResetTicks – reset the selected timer's tick count

SYNOPSIS

INT32 sysResetTicks(INT32 nTimeNo);

DESCRIPTION

This function used to reset Timer 0 or Timer 1's global tick counter.

PARAMETER

nTimeNo	TIMER0, TIMER1

RETURN VALUE

Successful

EXAMPLE

```
/* Reset timer 0 tick count */
INT32 status;
status = sysResetTicks(TIMER0);
```

NAME

sysUpdateTickCount – update selected timer's tick counter

SYNOPSIS

INT32 sysUpdateTickCount(INT32 nTimeNo, UINT32 uCount);

DESCRIPTION

This function used to update Timer 0 or Timer 1's global tick counter.

PARAMETER

nTimeNo	TIMER0, TIMER1
uCount	tick counter value

RETURN VALUE

Successful

EXAMPLE

```
/* update  timer 0's tick counter as 3000 */
sysUpdateTickCount (TIMER0, 3000);
```

NAME

sysSetTimerReferenceClock – set timer reference clock

SYNOPSIS

INT32 sysSetTimerReferenceClock(UINT32 nTimeNo, UINT32 uClockRate);

DESCRIPTION

This function used to set the timer's reference clock. The default reference clock is system clock (15MHz).

PARAMETER

nTimeNo	TIMER0, TIMER1
uClockRate	reference clock

RETURN VALUE

Successful

EXAMPLE

```
/* Set 20MHz to be timer 0's reference clock */
INT32 status;
status = sysSetTimerReferenceClock(TIMER0, 20000000);
```

NAME

sysStartTimer – setup own timer

SYNOPSIS

INT32 sysStartTimer(INT32 nTimeNo, UINT32 uTicksPerSecond, INT32 nOpMode);

DESCRIPTION

sysStartTimer will start Timer 0 or Timer 1. *nTimeNo* is used to select timer 0 or timer 1. Because of W90N745 timer has three operation modes, the *nOpMode* is used to set the operation mode. *uTicksPerSecond* indicates that how many ticks per second.

PARAMETER

nTimeNo	TIMER0, TIMER1
nTickPerSecond	tick number per second
nOpMode	ONE_SHOT_MODE, PERIODIC_MODE, TOGGLE_MODE

RETURN VALUE

Successful

EXAMPLE

```
/* Start the timer 1, and set it to periodic mode and 100 ticks per second */
INT32 status;
status = sysStartTimer(TIMER1, 100, PERIODIC_MODE);
```

NAME

sysStopTimer – stop own timer

SYNOPSIS

INT32 sysStopTimer(INT32 nTimeNo);

DESCRIPTION

sysStopTimer will stop Timer 0 or Timer 1. *nTimeNo* is used to select timer 0 or timer 1. After disabling timer, this function will restore the interrupt service routine.

PARAMETER

nTimeNo	TIMER0, TIMER1

RETURN VALUE

Successful

EXAMPLE

```
/* Stop the timer 1 */
INT32 status;
status = sysStopTimer(TIMER1);
```

NAME

sysClearWatchDogTimerCount – clear watch dog timer reset counter

SYNOPSIS

VOID sysClearWatchDogTimerCount(VOID);

DESCRIPTION

This function is used to clear watch dog timer reset count. When interrupt occurred, the system will reset after 1024 clock cycles. Clear the reset counter, the system will not be reset.

PARAMETER

None

RETURN VALUE

None

EXAMPLE

sysClearWatchDogTimerCount();

NAME

sysClearWatchDogTimerInterruptStatus – clear watch dog timer interrupt status

SYNOPSIS

VOID sysClearWatchDogTimerInterruptStatus(VOID);

DESCRIPTION

This function is used to clear watch dog timer interrupt status. When interrupt occurred, the watch dog timer interrupt flag will be set. Clear this flag, the interrupt will occur again.

PARAMETER

None

RETURN VALUE

None

EXAMPLE

sysClearWatchDogTimerInterruptStatus();

NAME

sysDisableWatchDogTimer – disable watch dog timer

SYNOPSIS

VOID sysDisableWatchDogTimer(VOID);

DESCRIPTION

This function is used to disable watch dog timer.

PARAMETER

None

RETURN VALUE

None

EXAMPLE

sysDisableWatchDogTimer();

NAME

sysDisableWatchDogTimerReset – disable watch dog timer reset function

SYNOPSIS

VOID sysDisableWatchDogTimerReset(VOID);

DESCRIPTION

This function is used to disable watch dog timer reset function.

PARAMETER

None

RETURN VALUE

None

EXAMPLE

sysDisableWatchDogTimerReset();

NAME

sysEnableWatchDogTimer – enable watch dog timer

SYNOPSIS

VOID sysEnableWatchDogTimer(VOID);

DESCRIPTION

This function is used to enable watch dog timer.

PARAMETER

None

RETURN VALUE

None

EXAMPLE

sysEnableWatchDogTimer();

NAME

sysEnableWatchDogTimerReset – enable watch dog timer reset function

SYNOPSIS

VOID sysEnableWatchDogTimerReset(VOID);

DESCRIPTION

This function is used to enable watch dog timer reset function. The system will be reset when this function is enabled.

PARAMETER

None

RETURN VALUE

None

EXAMPLE

sysEnableWatchDogTimerReset();

NAME

sysInstallWatchDogTimerISR – setup own watch dog timer interrupt service routine

SYNOPSIS

PVOID sysInstallWatchDogTimerISR(INT32 nIntTypeLevel, PVOID pvNewISR);

DESCRIPTION

This function is used to set up own watch dog timer interrupt service routine. *nIntTypeLevel* is select interrupt to be FIQ or IRQ, and level group 0 ~ 7. *pvNewISR* is the own interrupt service routine's pointer.

PARAMETER

nIntTypeLevel	FIQ_LEVEL_0, IRQ_LEVEL_1 ~ IRQ_LEVEL_7
pvNewISR	the pointer of watch dog timer interrupt service routine

RETURN VALUE

a pointer which point to old ISR

EXAMPLE

```
/* Set watch dog timer interrupt to be IRQ and group level 1 */
PVOID oldVect;
oldVect = sysInstallWatchDogTimerISR(IRQ_LEVEL_1, myWatchDogISR);
```

NAME

sysSetWatchDogTimerInterval – set watch dog timer interval

SYNOPSIS

INT32 sysSetWatchDogTimerInterval(INT32 nWdtInterval);

DESCRIPTION

This function is used to set the watch dog timer interval. The default is 0.5 minutes. You can select interval to be 0.5, 1, 2, and 4 minutes.

PARAMETER

nWdtInterval WDT_INTERVAL_0, WDT_INTERVAL_1, WDT_INTERVAL_2, WDT_INTERVAL_3.

The watch dog timer interval is shown as follows.

nWdtInterval	Interrupt Timeout	Reset Timeout	Real Time Interval
WDT_INTERVAL_0	2^{14} clocks	2^{14} + 1024 clocks	0.28 sec.
WDT_INTERVAL_1	2^{16} clocks	2^{16} + 1024 clocks	1.12 sec.
WDT_INTERVAL_2	2^{18} clocks	2^{18} + 1024 clocks	4.47 sec.
WDT_INTERVAL_3	2^{20} clocks	2^{20} + 1024 clocks	17.9 sec.

RETURN VALUE

Successful

EXAMPLE

```
/* Set watch dog timer interval to WDT_INTERVAL_0 */
INT32 status;
status = sysSetWatchDogTimerInterval(WDT_INTERVAL_0);
```

NAME

sysSetTimerEvent –set event of selected timer

SYNOPSIS

INT32 sysSetTimerEvent(UINT32 nTimeNo, UINT32 nTimeTick, PVOID pvFun);

DESCRIPTION

This function is used to set the event of selected timer. *nTimeNo* is used to select timer 0 or timer 1. The event function which pointed by *pvFun* shall be executed after *nTimeTick* system timer tick.

PARAMETER

nTimeNo	TIMER0, TIMER1
nTimeTick	tick count before event executed
pvFun	event function pointer

RETURN VALUE

event number

EXAMPLE

```
/* Set event function “hello” after 100 tick */
INT nEventNo;
VOID hello(VOID)
{
   sysPrintf(“Hello World!\n”);
}
nEventNo = sysSetTimerEvent (TIMER0, 100, (PVOID)hello);
```

NAME

sysClearTimerEvent –clear event of selected timer

SYNOPSIS

VOID sysClearTimerEvent(UINT32 nTimeNo, UINT32 uTimeEventNo);

DESCRIPTION

This function is used to clear the event of selected timer. *nTimeNo* is used to select timer 0 or timer 1. The event function which indicated by *uTimeEventNo* shall be cleared.

PARAMETER

nTimeNo	TIMER0, TIMER1
uTimeEventNo	event number which want to clear

RETURN VALUE

None

EXAMPLE

```
/* clear event NO 5*/

sysClearTimerEvent (TIMER0, 5);
```

NAME

sysSetLocalTime – set local time

SYNOPSIS

VOID sysSetLocalTime(DateTime_T ltime);

DESCRIPTION

This function is used to set local time. *ltime* is a structure which contains year, month, day, hour, minute, and second information.

PARAMETER

ltime — structure which contains the following information

```
typedef struct datetime_t
{
        UINT32      year;
        UINT32      mon;
        UINT32      day;
        UINT32      hour;
        UINT32      min;
        UINT32      sec;
} DateTime_T;
```

RETURN VALUE

None

EXAMPLE

```
/* set local time*/
DateTime_T      TimeInfo;
TimeInfo.year = 2006;
TimeInfo.mon = 6;
TimeInfo.day = 12
TimeInfo.hour = 9;
TimeInfo.min = 0;
TimeInfo.sec = 30;
sysSetLocalTime(TimeInfo);
```

NAME

sysGetCurrentTime –get current time

SYNOPSIS

VOID sysGetCurrentTime(DateTime_T *curTime);

DESCRIPTION

This function is used to get local time. *curTime* is a structure pointer which contains year, month, day, hour, minute, and second information.

PARAMETER

*curTime structure pointer which contains the following information

```
typedef struct datetime_t
{
        UINT32      year;
        UINT32      mon;
        UINT32      day;
        UINT32      hour;
        UINT32      min;
        UINT32      sec;
} DateTime_T;
```

RETURN VALUE

None

EXAMPLE

```
/* set local time*/
DateTime_T      TimeInfo;
sysGetCurrentTime(TimeInfo);
```

NAME

sysDelay –delay specific time which unit is 10ms

SYNOPSIS

VOID sysDelay(UINT32 uTicks);

DESCRIPTION

This function is used to delay a specific period. *uTicks* is the length of delay time which unit is ten milliseconds. Please notice that the delay period has an extent of error which less than ten milliseconds.

PARAMETER

uTicks	delay period which unit is ten milliseconds

RETURN VALUE

None

EXAMPLE

```
/* delay 1s*/
sysDelay(100);
```

NAME

sysGetChar – get a character from UART without echo

SYNOPSIS

CHAR sysGetChar(VOID);

DESCRIPTION

This function is user to obtain the next available character from the UART. Nothing is echoed. When no available characters are found, the function waits until a character from UART is found.

PARAMETER

None

RETURN VALUE

a character from UART

EXAMPLE

```
/* get user's input*/
CHAR cUserInput;
cUserInput = sysGetChar();
```

NAME

sysInitializeUART – initialize the UART

SYNOPSIS

INT32 sysInitializeUART(WB_UART *uart);

DESCRIPTION

WB_UART is the device initialization structure. The definition is as following:

```
typedef struct UART_INIT_STRUCT
{
    UINT32      freq;
    UINT32      baud_rate;
    UINT32      data_bits;
    UINT32      stop_bits;
    UINT32      parity;
    UINT32      rx_trigger_level;
} WB_UART;
```

uart->freq is UART reference clock. Default is 15MHz. If user have different reference clock, used this parameter to change it.
uart->baud_rate is used to set the COM port baud rate. The range is from 9600 to 230400.
The UART data bit can be 5, 6, 7, or 8. Use *uart->data_bits* to set the suitable data bits.
The UART stop bit can be 1, or 2. Use *uart->stop_bits* to set the suitable stop bits.
uart->parity is used to set the suitable parity check.
uart->rx_trigger_level is used to set the suitable trigger level.

PARAMETER

uart->data_bits	WB_DATA_BITS_5 ~ WB_DATA_BITS_8
uart->stop_bits	WB_STOP_BITS_1, WB_STOP_BITS_2
uart->parity	WB_PARITY_NONE, WB_PARITY_ODD, WB_PARITY_EVEN
uart->rx_trigger_level	LEVEL_1_BYTE, LEVEL_4_BYTES, LEVEL_8_BYTES, LEVEL_14_BYTES

RETURN VALUE

Successful/ WB_INVALID_PARITY/ WB_INVALID_DATA_BITS/ WB_INVALID_STOP_BITS/ WB_INVALID_BAUD

EXAMPLE

```
WB_UART_T uart;
```

```
uart.uiFreq = APB_SYSTEM_CLOCK;
uart.uiBaudrate = 115200;
uart.uiDataBits = WB_DATA_BITS_8;
uart.uiStopBits = WB_STOP_BITS_1;
uart.uiParity = WB_PARITY_NONE;
uart.uiRxTriggerLevel = LEVEL_1_BYTE;
sysInitializeUART(&uart);    WB_UART_T uart;
```

NAME

sysPrintf – display message through RS232 to terminal by interrupt mode

SYNOPSIS

VOID sysPrintf(PCHAR pcStr, …);

DESCRIPTION

The function sends the specified *str* to the terminal through the RS-232 interface by interrupt mode.

PARAMETER

pcStr	pointer of string which want to display

RETURN VALUE

None

EXAMPLE

sysPrintf(“Hello World!\n”);

NAME

sysprintf – display message through RS232 to terminal by polling mode

SYNOPSIS

VOID sysPrintf(PCHAR pcStr, …);

DESCRIPTION

The function sends the specified *str* to the terminal through the RS-232 interface by polling mode.

PARAMETER

pcStr	pointer of string which want to display

RETURN VALUE

None

EXAMPLE

sysprintf(“Hello World!\n”);

NAME

sysPutChar – put a character out to UART

SYNOPSIS

VOID sysPutChar(UCHAR ch);

DESCRIPTION

The function sends the specified *ch* to the UART.

PARAMETER

ch character which want to display

RETURN VALUE

None

EXAMPLE

sysPutChar(“A”);

NAME

sysDisableInterrupt – disable interrupt source

SYNOPSIS

INT32 sysDisableInterrupt(UINT32 intNo);

DESCRIPTION

This function is used to disable interrupt source.

PARAMETER

intNo interrupt source number

RETURN VALUE

Successful or Fail.

EXAMPLE

```
/* Disable timer 0 interrupt (source number is 7) */
INT32 status;
status = sysDisableInterrupt(7);
```

NAME

sysEnableInterrupt – enable interrupt source

SYNOPSIS

INT32 sysEnableInterrupt(UINT32 intNo);

DESCRIPTION

This function is used to enable interrupt source.

PARAMETER

intNo interrupt source number

RETURN VALUE

Successful or Fail.

EXAMPLE

```
/* Enable timer 0 interrupt (source number is 7) */
INT32 status;
status = sysEnableInterrupt(7);
```

NAME

sysInstallExceptionHandler – setup own exception handler to interrupt vector table

SYNOPSIS

PVOID sysInstallExceptionHandler(INT32 exceptType, PVOID pNewHandler);

DESCRIPTION

This function is used to install *pNewHandler* into *exceptType* exception.

PARAMETER

exceptType	WB_SWI, WB_D_ABORT, WB_I_ABORT, WB_UNDEFINE
pNewHandler	pointer of the new handler

RETURN VALUE

a pointer which point to old handler

EXAMPLE

```
/* Setup own software interrupt handler */
PVOID oldVect;
oldVect = sysInstallExceptionHandler(WB_SWI, pNewSWIHandler);
```

NAME

sysInstallFiqHandler – setup own FIQ ISR into interrupt vector table

SYNOPSIS

PVOID sysInstallFiqHandler(PVOID pNewISR);

DESCRIPTION

Use this function to install FIQ handler into interrupt vector table.

PARAMETER

pNewISR pointer of the new ISR handler

RETURN VALUE

a pointer which point to old ISR

EXAMPLE

```
/* Setup own FIQ handler */
PVOID oldVect;
oldVect = sysInstallFiqHandler(pNewFiqISR);
```

NAME

sysInstallIrqHandler – setup own IRQ ISR to interrupt vector table

SYNOPSIS

PVOID sysInstallIrqHandler(PVOID pNewISR);

DESCRIPTION

Use this function to install FIQ handler into interrupt vector table.

PARAMETER

pNewISR pointer of the new ISR handler

RETURN VALUE

a pointer which point to old ISR

EXAMPLE

```
/* Setup own IRQ handler */
PVOID oldVect;
oldVect = sysInstallIrqHandler(pNewIrqISR);
```

NAME

sysInstallISR – setup own interrupt service routine to AIC interrupt vector table

SYNOPSIS

PVOID sysInstallISR(INT32 intTypeLevel, INT32 intNo, PVOID pNewISR, PVOID pParam);

DESCRIPTION

W90N745 interrupt group level is 0 ~ 7. Level 0 is FIQ, and level 1 ~ 7 are IRQ. The highest priority is 0, and the lowest priority is 7. Use this function to set up interrupt source (*intNo*) *pNewISR* handler to AIC interrupt vector table.

PARAMETER

intTypeLevel	FIQ_LEVEL_0, IRQ_LEVEL_1 ~ IRQ_LEVEL_7
intNo	interrupt source number
pNewISR	function pointer of new ISR
pParam	parameter for ISR

RETURN VALUE

a pointer which point to old ISR

EXAMPLE

```
/* Setup timer 0 handler */
PVOID oldVect;
oldVect = sysInstallISR(IRQ_LEVEL_1, 7, pTimerISR, param);
```

NAME

sysSetGlobalInterrupt – enable / disable all interrupts

SYNOPSIS

INT32 sysSetGlobalInterrupt(INT32 intState);

DESCRIPTION

Enable / disable all interrupt sources.

PARAMETER

intState	ENABLE_ALL_INTERRUPTS, DISABLE_ALL_INTERRUPTS

RETURN VALUE

Successful

EXAMPLE

```
/* Disable all interrupt */
INT32 status;
status = sysSetGlobalInterrupt(DISABLE_ALL_INTERRUPTS);
```

NAME

sysSetInterruptPriorityLevel – set interrupt priority level

SYNOPSIS

INT32 sysSetInterruptPriorityLevel(UINT32 intNo, UINT32 intLevel);

DESCRIPTION

W90N745 interrupt has 8 group levels. The highest is 0, and the lowest is 7. Use this function can change the priority level after install ISR.

PARAMETER

intNo	interrupt source number
intLevel	FIQ_LEVEL_0, IRQ_LEVEL_1 ~ IRQ_LEVEL_7

RETURN VALUE

Successful or Fail.

EXAMPLE

```
/* Change timer 0 priority to level 4 */
INT32 status;
status = sysSetInterruptPriorityLevel(7, 4);
```

NAME

sysSetInterruptType – set interrupt source type

SYNOPSIS

INT32 sysSetInterruptType(UINT32 intNo, UINT32 intSourceType);

DESCRIPTION

W90N745 has four kinds of interrupt source types. They are low level sensitive, high level sensitive, negative edge trigger, and positive edge trigger. The default is high level sensitive. This function is used to change the interrupt source type.

PARAMETER

intNo	interrupt source number
intSourceType	LOW_LEVEL_SENSITIVE, HIGH_LEVEL_SENSITIVE, NEGATIVE_EDGE_TRIGGER, POSITIVE_EDGE_TRIGGER

RETURN VALUE

Successful or Fail.

EXAMPLE

```
/* Change timer 0 source type to be positive edge trigger */
INT32 status;
status = sysSetInterruptType(7, POSITIVE_EDGE_TRIGGER);
```

NAME

sysSetLocalInterrupt – enable / disable CPSR interrupt

SYNOPSIS

INT32 sysSetLocalInterrupt(INT32 intState);

DESCRIPTION

The CPSR I bit and F bit need to be enabled or disabled, when using interrupt. This function is used to enable / disable I bit and F bit.

PARAMETER

intState	ENABLE_IRQ, ENABLE_FIQ, ENABLE_FIQ_IRQ, DISABLE_IRQ, DISABLE_FIQ, DISABLE_FIQ_IRQ

RETURN VALUE

Successful

EXAMPLE

```
/* Enable I bit of CPSR */
INT32 state;
state = sysSetLocalInterrupt(ENABLE_IRQ);
```

NAME

sysSetAIC2SWMode –set AIC to software mode

SYNOPSIS

INT32 sysSetAIC2SWMode(VOID);

DESCRIPTION

This function is used to set AIC as software mode. When the system AIC in software mode, the priority of each interrupt source shall be handled by software.

PARAMETER

intState	ENABLE_IRQ, ENABLE_FIQ, ENABLE_FIQ_IRQ, DISABLE_IRQ, DISABLE_FIQ, DISABLE_FIQ_IRQ

RETURN VALUE

Successful

EXAMPLE

```
/* Set AIC as software mode */
sysSetAIC2SWMode();
```

NAME

sysGetInterruptEnableStatus –return interrupt source enable/disable status

SYNOPSIS

UINT32 sysGetInterruptEnableStatus(VOID);

DESCRIPTION

This function is used to get the enable/disable status of interrupt which save in AIC_IMR register.

PARAMETER

None

RETURN VALUE

value of AIC_IMR register

EXAMPLE

```
/* Set AIC as software mode */
UINT32 uIMRValue;
uIMRValue = sysGetInterruptEnableStatus();
```

NAME

sysEnableCache –enable system cache

SYNOPSIS

VOID sysEnableCache(VOID);

DESCRIPTION

This function is used to enable cache.

PARAMETER

None

RETURN VALUE

None

EXAMPLE

```
/* enable cache */
sysEnableCache();
```

NAME

sysDisableCache –disable system cache

SYNOPSIS

VOID sysDisableCache(VOID);

DESCRIPTION

This function is used to disable cache.

PARAMETER

None

RETURN VALUE

None

EXAMPLE

```
/* disabled cache */
sysDisableCache();
```

NAME

sysFlushCache – flush system cache

SYNOPSIS

VOID sysFlushCache(INT32 cacheType);

DESCRIPTION

This function is used to flush system cache. The parameter, cacheType is used to select cache which needs to be flushed.

PARAMETER

cacheType	I_CACHE, D_CACHE, I_D_CACHE

RETURN VALUE

None

EXAMPLE

```
/* flush cache */
sysFlushCache(I_D_CACHE);
```

NAME

sysSetExternalIO – initial external IO control

SYNOPSIS

VOID sysSetExternalIO (INT extNo, UINT32 extBaseAddr, UINT32 extSize, INT extBusWidth);

DESCRIPTION

This function is used to initial external IO control, include setup base address, size, and bus width.

PARAMETER

extNo	EXT0, EXT1, EXT2, EXT3
extBaseAddr	base address for external IO used
extSize	SIZE_256K, SIZE_512K, SIZE_1M, SIZE_2M, SIZE_4M, SIZE_8M, SIZE_16M, SIZE_32M
extBusWidth	BUS_DISABLE, BUS_BIT_8, BUS_BIT_16, BUS_BIT_32

RETURN VALUE

None

EXAMPLE

```
/* Set external IO 0 – base:0xC0000000, size:16M, bus width:16-bit */
sysSetExternalIO (EXT0, 0xC0000000, SIZE_16M, BUS_BIT_16);
```

NAME

sysSetExternalIOTiming1– set up external IO access cycles

SYNOPSIS

VOID sysSetExternalIOTiming1 (INT extNo, INT tACC, INT tACS);

DESCRIPTION

This function is used to change the access cycles and address set-up time.

PARAMETER

extNo	EXT0, EXT1, EXT2, EXT3
tACC	Range is from 0x0 to 0xF
tACS	Range is from 0x0 to 0x7

RETURN VALUE

None

EXAMPLE

```
/* Set external IO 2 access cycles */
sysSetExternalIOTiming1 (EXT2, 0x7, 0x3);
```

NAME

sysSetExternalIOTiming2– set up external IO chip selection time

SYNOPSIS

VOID sysSetExternalIOTiming2 (INT extNo, INT tCOH, INT tCOS);

DESCRIPTION

This function is used to set up the chip selection hold-on time and chip selection set-up time.

PARAMETER

extNo	EXT0, EXT1, EXT2, EXT3
tCOH	Range is from 0x0 to 0x7
tCOS	Range is from 0x0 to 0x7

RETURN VALUE

None

EXAMPLE

```
/* Set external IO 1 hold-on time */
sysSetExternalIOTiming2 (EXT1, 0x3, 0x3);
```

NAME

sysDisableAllPM_IRQ – Clean the PM IRQ status

SYNOPSIS

VOID sysDisableAllPM_IRQ(VOID);

DESCRIPTION

This function cleans the PM IRQ status. The PM IRQ status records the specific IRQ source number which use to wake up system.

PARAMETER

None

RETURN VALUE

None

EXAMPLE

```
/* Reset PM IRQ status*/
sysDisableAllPM_IRQ();
```

NAME

sysEnablePM_IRQ – Save specific PM IRQ for wake up system

SYNOPSIS

INT sysEnablePM_IRQ(INT irq_no);

DESCRIPTION

This function saves the PM IRQ status. This status indicates the IRQ source numbers which need enable before system enter IDLE/MIDLE/PD mode. On the other word, user should use this function to save the specific IRQ source numbers which use to wake up system.

PARAMETER

irq_no	IRQ source number(0 ~ 31)

RETURN VALUE

Successful	Operation finishes successfully
WB_PM_INVALID_IRQ_NUM	IRQ source number is not correct

EXAMPLE

```
#define IRQ_KEYPAD 29
sysEnablePM_IRQ(IRQ_KEYPAD);// use KPI to wake up system
```

NAME

sysPMStart –Enter IDLE/MIDLE/PD power saving mode

SYNOPSIS

INT sysPMStart(INT pd_type);

DESCRIPTION

This function starts PM procedure according to the parameter, pd_type. Please notice that the sysPMStart function must not called in any ISR. Besides, this function will disable and flush cache before enter power saving mode since it uses system SRAM.

PARAMETER

pd_type	WB_PM_IDLE/WB_PM_PD/WB_PM_MIDLE

RETURN VALUE

Successful	Operation finishes successfully
WB_PM_PD_IRQ_Fail	Power down IRQ setting error
WB_PM_Type_Fail	Power saving type error

EXAMPLE

```
INT status;
status=sysPMStart(WB_PM_PD);
```

CORPORATE HEADQUARTERS:
NO. 9, Li Hsin Rd.
Science-Based Industrial Park
Hsinchu, Taiwan, R.O.C.
TEL: 886-03-5678168
FAX: 886-03-5665535
WWW:http://www.winbond.com.tw/

INFORMATION CONTACTS:
Hui-Ping Chen
Micro controller R&D Division Dept. II (NS22)
TEL: 886-03-5678168 Ext. 7980
E-MAIL: HPChen0@winbond.com