ComwireLite Library User's Manual V1.0

04/26/2013

0 Table of Content

1 Revision History

1.1 Document History

Revision	Date	By	Remark
V1.0	04/26/2013	Porter Yang	First Version

2 ComwireLite Summary

The ComwireLite algorithm provides the ability to transmit-receive commands through the audio wire. This algorithm can be used in communication areas. Below are some recommended considerations developers should consider.

3 Library Service loops

The ComwireLite library can support the foreground service loops. In foreground service loop, users have to put the service loop in the main routine to keep entering the service regularly. Inside the ComwireLite service loop, there will be a mechanism to decide any task should be carried on. Some overhead will produced inevitably. The amount of overhead varies and depends on the payload of CPU.

Example:

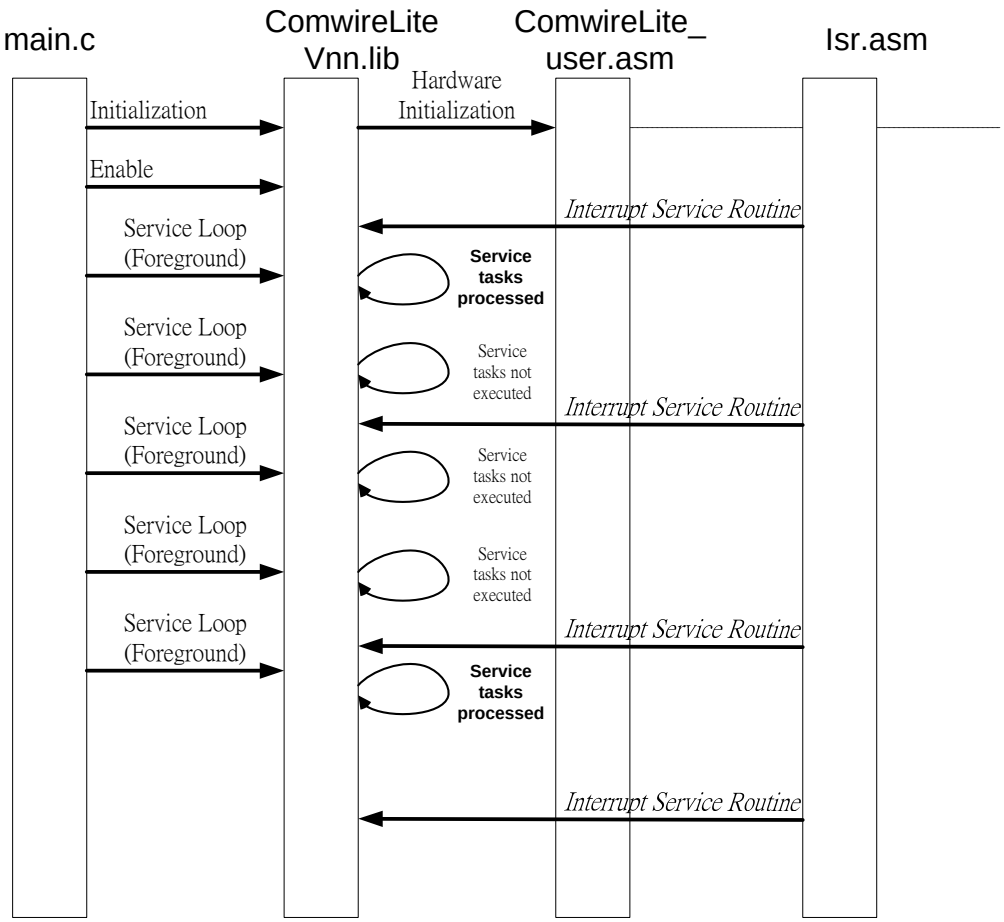
In main.c

```
int main()
{
    System_Initial();                       // System initial
    ComwireLite_Init(16);
    ComwireLite_Enable();
    while(1)
    {
        Key = SP_GetKeyOn();
        if(Key != 0)
        {
            ComwireLite_Send_Cmd(Key);
        }

        if(ComwireLite_Is_Cmd_Rcvd())
        {
            Cmd = ComwireLite_Get_Cmd();
            *pP_IOB_Buffer = Cmd;
        }
        ComwireLite_ServiceLoop();          // Foreground Service loop
        System_ServiceLoop();               // Service loop for watchdog clear
    } // end of while
    return 0;
} // end of main
```

In isr.asm:

```
_IRQ0:
    push R1, R5 to [SP];                    // save registers
    call F_ISR_Service_ComwireLite          // Interrupt service routine
    R1 = C_IRQ0_TMA;
    [P_INT_Clear] = R1;
    pop R1, R5 from [SP];                   // restore registers
    reti;
```



Timing diagram: Foreground service loop

4 API of ComwireLite

4.1 Hardware Dependent Functions Function: Initializes ComwireLite

4.1.1 Function: Initialize ComwireLite module, sets Interrupt source and Timer before running

Syntax:

C: void ComwireLite_Init (int DataLength)

ASM: R1 = 16
Call F_ComwireLite_Init

Parameters: R1: Data Length

Return Value: None

Library: ComwireLite_vXXX

Remark: 1. Data Length is valid bit number

2. This function initializes the Kernel of ComwireLite. It also initializes the Timer A, IR and enables the Timer A IRQ at the sample rate on 8KHz.

3. To produce 6KHz and 12KHz square wave

4. The ComwireLite receiving pin must be set to floating, so that it can determine the input signal.

5. The hardware setting is opened for user's reference (see F_ComwireLite_HW_Init in ComwireLite_User.asm).

4.2 ComwireLite Control

4.2.1 Function: Enable ComwireLite to start transmitting and receiving

Syntax:

C: void ComwireLite_Enable (void)

ASM: Call F_ComwireLite_Enable

Parameters: None

Return Value: None

Library: ComwireLite_vXXX

Remark: None

4.2.2 Function: Disable ComwireLite to stop transmitting and receiving

Syntax:

C: void ComwireLite_Disable(void);

ASM: Call F_ ComwireLite_Disable

Parameters: None

Return Value: None

Library: ComwireLite_vXXX

Remark: None

4.2.3 Function: Get Tx Busy Flag

Syntax:

C: int ComwireLite_Is_Tx_Busy (void);

ASM: Call F_ ComwireLite_Is_Tx_Busy

Parameters: None

Return Value: R1: 0 is Idel
1 is Busy

Library: ComwireLite_vXXX

Remark: None

4.2.4 Function: Send Transmit Command

Syntax:

C: void ComwireLite_Send_Cmd (int command);

ASM: R1: command
Call ComwireLite_Send_Cmd

Parameters: R1: 16-bit command

Return Value: None

Library: ComwireLite_vXXX

Remark: None

4.2.5 Function: Get Rx Busy Flag

Syntax:

C: int ComwireLite_Is_Rx_Busy (void);

ASM: Call F_ ComwireLite_Is_Rx_Busy

Parameters: None

Return Value: R1: 0 is Idel
1 is Busy

Library: ComwireLite_vXXX

Remark: None

4.2.6 Function: Get Command Received Flag

Syntax:

C: int ComwireLite_Is_Cmd_Rcvd (void);

ASM: Call F_ ComwireLite_Is_Cmd_Rcvd

Parameters: None

Return Value: R1: 0 means no command,
1 means command received.

Library: ComwireLite_vXXX

Remark: None

4.2.7 Function: Get Received Command

Syntax:

C: int ComwireLite_Get_Cmd (void);

ASM: Call F_ ComwireLite_Get_Cmd

Parameters: None

Return Value: R1: 16-bit command

Library: ComwireLite_vXXX

Remark: None

4.2.8 Function: ComwireLite Service Loop

Syntax:

C: void ComwireLite_ServiceLoop (void);

ASM: Call F_ ComwireLite_ServiceLoop

Parameters: None

Return Value: None

Library: ComwireLite_vXXX

Remark: None

4.3 ISR Functions: Interrupt service routine for ComwireLite

Syntax:

C: None

ASM: Call F_ISR_Service_ComwireLite

Parameters: None

Return Value: None

Library: ComwireLite_vXXX

Remark: 1. This function is used in assembly only and it can be hooked on the _FIQ, _IRQ1 or _IRQ2: label. (See isr.asm for details)

2. The F_ISR_Service_ComwireLite will not take up any time to process the Interrupt routine except minor overheads if the program is not in beat tracking mode. It is possible for users to place user-define function in the same FIQ or IRQ.

EX:

```
_IRQ0:
        push r1,r5 to [sp];
        call   F_ISR_Service_ComwireLite
        call   F_User_ISR
        R1 = C_IRQ0_TMA;
        [P_INT_Clear] = R1;
        pop r1,r5 from [sp]
        reti
```

3. This function will not destroy the content in R1-R5 register. Programmers have not to protect

the registers externally.
4. The Timer A IRQ is working on 8KHz.

4.4 User Functions: for ComwireLite

4.4.1 Function: Hardware Initial for ComwireLite

Syntax:

C: void ComwireLite_HW_Init(void)

ASM: Call F_ComwireLite_HW_Init

Parameters: None

Return Value: None

Library: ComwireLite_User.asm

Remark: 1. IOA1: GPCE Tx, Hardware IR
2. IOA2: GPCE Rx, I/O
3. TimerA as 8KHz

4.4.2 Function: Enable for ComwireLite

Syntax:

C: None

ASM: Call F_ComwireLite_HW_Enable

Parameters: None

Return Value: None

Library: ComwireLite_User.asm

Remark: The user can use this function to control external circuit here for power saving

4.4.3 Function: Disable for ComwireLite

Syntax:

C: None

ASM: Call F_ComwireLite_HW_Enable

Parameters: None

Return Value: None

Library: ComwireLite_User.asm

Remark: The user can use this function to control external circuit here for power saving

4.4.4 Function: Get I/O for ComwireLite

Syntax:

C: None

ASM: Call F_ComwireLite_HW_GetIO

Parameters: None

Return Value: R1: I/O data

Library: ComwireLite_User.asm

Remark: Get I/O data

4.4.5 Function: Set Timer for IR

Syntax:

C: None

ASM: Call F_ComwireLite_HW_IR_Timer

Parameters: R1

Return Value: None

Library: ComwireLite_User.asm

Remark: None

4.4.6 Function: Turn On IR

Syntax:

C: None

ASM: Call F_ComwireLite_HW_IR_ON

Parameters: None

Return Value: None

Library: ComwireLite_User.asm

Remark: None

4.4.7 Function: Turn Off IR

Syntax:

C: None

ASM: Call F_ComwireLite_HW_IR_OFF

Parameters: None

Return Value: None

Library: ComwireLite_User.asm

Remark: None

4.4.8 Function: Turn On INT

Syntax:

C: None

ASM: Call F_ComwireLite_HW_INT_ON

Parameters: None

Return Value: None

Library: ComwireLite_User.asm

Remark: None

4.4.9 Function: Turn Off INT

Syntax:

C: None

ASM: Call F_ComwireLite_HW_INT_OFF

Parameters: None

Return Value: None

Library: ComwireLite_User.asm

Remark: None

5 Resources List of ComwireLite

5.1 TABLE 1: RAM Size (Unit: Decimal Word)

	IRAM	ISRAM	RAM	SRAM	ORAM	OSRAM
ComwireLte					88	

5.2 TABLE 2: ROM Size (Unit: Decimal Word)

	TEXT	CODE	DATA	USER DEFINE
ComwireLite	44	438		

5.3 TABLE 3: Hardware Resources VS Library

	Interrupt	Timer Setting	Audio
ComwireLite	TMA IRQ	8 KHz	

5.4 TABLE 5: CPU Usage Rate (approximate)

	GPCE001A CPU Usage Rate (48 MHz) while ComAir
ComwireLte	5.12% (6.4us/125us)

5.5 TABLE 6: Name of Overlap RAM in the library

Table: Name of Overlap RAM in the library

	Overlap RAM definition	
Algorithm	**Overlap RAM Label**	**Size(word)**
ComwireLte	OVERLAP_COMWIRELITE_API_BLOCK	88