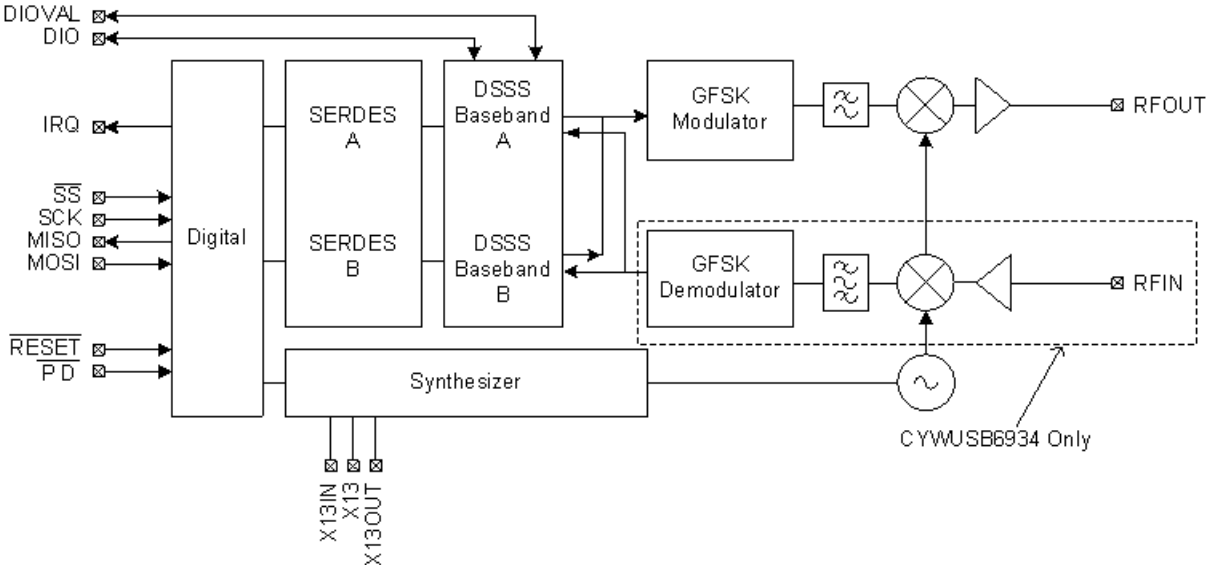
WirelessUSBLS Radio Datasheet WirelessUSBLS V 1.0

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	flash	RAM	
CYWUSB6953						
Non-Blocking Calls Enabled	0	0	0	1320	10	0
Non-Blocking Calls Disabled	0	0	0	1150	2	0

Features and Overview

- 2.4 GHz radio transceiver
- Operates in the unlicensed Industrial, Scientific, and Medical (ISM) band (2.4 GHz-2.483 GHz)
- -90 dBm receive sensitivity
- Up to 0 dBm output power
- Range of up to 10 meters
- Data throughput up to 62.5 Kbs

Figure 1. WirelessUSBLS Block Diagram



Functional Description

The WirelessUSB LS Radio User Module is a firmware interface to the WirelessUSB LS Radio hardware. The radio does not use any PSoC digital or analog blocks. The WirelessUSB LS Radio API provides functions callable from both C and assembly to start the radio, send and receive data, change channels, change PN codes, etc.

Parameters and Resources

DataRate(Kbps)

DataRate specifies the raw data rate as described in WirelessUSB LS 2-Way HID Systems. The table below describes the available data rates.

Data Rate (Kbps)	Use
16	The 16 kbps data rate uses 64chips/bit PN Codes providing higher processing gain and thus longer range than 32 and 64 kbps data rates. It is ideal for long range, high data integrity, and low data rate applications. The maximum data throughput for this data rate is 15.7 kbps.
64	The 64 kbps data rate uses two 32chips/bit PN Codes allowing two data bits to be represented by the transmission of a single bit. The 64 kbps data rate is ideal for short range, moderate data rate applications. The maximum data throughput for this data rate is 62.5 kbps.

NonBlockingCalls

NonBlockingCalls is used to enable or disable the receive ISR. Slave devices disable NonBlocking-Calls; master devices typically require that you enable NonBlockingCalls.

ChipErrorThreshold

Use ChipErrorThreshold to determine the number of chip errors allowed for each PN code when attempting to correlate a single data bit. For example, if the default ChipErrorThreshold is set to three, this means that there are, at most, three wrong chips for each PN code correlation.

DefaultTxPowerLevel

TxPowerLevel controls the transmit power of the radio. You can select any value between 0 (lowest power level) and 7 (highest power level). The default power level is 7.

EofBits

EofBits are the number of consecutive invalid bits before an EOF interrupt occurs. Select any value between 1 and 7.

CrystalStartupTime(msec)

CrystalStartupTime is the amount of time specified for the crystal to stabilize. Different crystals take different amounts of time to stabilize. You can adjust this value to match the crystal you are using. It is given in milliseconds. Anywhere from 0 to 20 msec can be selected The typical value is 3. See WirelessUSB Crystal Guidelines.

CrystalAdjust

CrystalAdjust is given for calibrating the on chip load capacitance supplied to the crystal. Any value between 0 and 31 is possible. Adjust this setting only if necessary. See WirelessUSB Crystal Guidelines and WirelessUSB™ LS 2.4GHz DSSS Radio SoC Datasheet. The USB Device User Module does not use the PSoC Designer User Module Parameter Grid Display for personalization. Instead, it uses a form driven USB Setup Wizard to define the USB descriptors for the application. From the descriptors, the Wizard personalizes the User Module.

Application Programming Interface

The Application Programming Interface (API) routines are provided as part of the user module to allow the designer to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the "include" files.

Note In this, as in all user module APIs, the values of the A and X register may be altered by calling an API function. It is the responsibility of the calling function to preserve the values of A and X prior to the call if those values are required after the call. This "registers are volatile" policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer™. The C compiler automatically takes care of this requirement. Assembly language programmers must ensure their code observes the policy, too. Though some user module API function may leave A and X unchanged, there is no guarantee they will do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

The following are the API programming routines provided for WirelessUSBLS. All functions adhere to fastcall16 as described in the PSoC Technical Reference Manual.

WirelessUSBLS_Start

Description:

Initializes and configures the radio by resetting the radio, delaying CrystalStartupTime(msec) to stabilize the crystal, initializing the SPI, and loading the radio configuration data.

C Prototype:

```
void WirelessUSBLS_Start(void)
```

Assembler:

```
lcall WirelessUSBLS_Start
```

Parameters:

None.

Return Value:

None.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_SetChannel

Description:

Sets the frequency of the radio. Channels are spaced at 1 MHz intervals with channel 0 centered at 2.402 GHz and channel 78 centered at 2.480 GHz.

C Prototype:

```
void WirelessUSBLS_SetChannel(BYTE bChannel)
```

Assembler:

```
mov  A, [bChannel]
lcall WirelessUSBLS_SetChannel
```

Parameters:

bChannel: An 8 bit value used to define the channel of the radio. Channels that are used are the valid channels in the 2.4GHz frequency band. Valid channels include channels 0 through 77 (2.402 – 2.479). (See WirelessUSB LS Theory of Operation)

Return Value:

None.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_SetPnCode

Description:

Loads the 8-byte PN code pointed to by pbPnCode into the correlator.

C Prototype:

```
void WirelessUSBLS_SetPnCode(const BYTE *pbPnCode)
```

Assembler:

```
mov  A, >pbPnCode  ; Load MSB into A
mov  X, <pbPnCode  ; Load LSB into X
lcall WirelessUSBLS_SetPnCode
```

Parameters:

pbPnCode: A pointer to an array of 8 bytes used to define the PN code of the radio. (See WirelessUSB LS Theory of Operation)

Return Value:

None.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_SendData

Description:

Transmits data from the radio.

C Prototype:

```
void WirelessUSBLS_SendData(BYTE bLength, BYTE *pbData)
```

Assembler:

```
mov  A, pbData
push A           ; Push MSB onto stack
```

```
mov  A, pbData+1
push A           ; Push LSB onto stack
mov  A, [bLength]
push A           ; Push bLength onto stack
lcall WirelessUSBLS_SetPnCode
```

Parameters:

bLength: An 8 bit value used to define the length of the data being transmitted. pbData: A pointer to an array of bytes to be transmitted.

Return Value:

None.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_bReadData

Description:

Receives bytes of data from the radio until an EOF has been received or the timeout is reached. The Timeout is specified in 15usec units.

C Prototype:

```
BYTE WirelessUSBLS_bReadData(BYTE bLength, BYTE *pbData, BYTE *pbValid, WORD wTimeout)
```

Assembler:

```
mov  A, [wTimeout]
push A            ; Push MSB of wTimeout onto stack
mov  A, [wTimeout+1]
push A            ; Push LSB of wTimeout onto stack
mov  A, pbValid
push A            ; Push MSB of pbValid onto stack
mov  A, pbValid+1
push A            ; Push LSB of pbValid onto stack
mov  A, pbData
push A            ; Push MSB of pbData onto stack
mov  A, pbData+1
push A            ; Push LSB of pbData onto stack
mov  A, [bLength]
push A            ; Push bLength onto stack
lcall WirelessUSBLS_bReadData
```

Parameters:

bLength: An 8 bit value used to define the length of the receive buffer. pbData: A pointer to a buffer to store the received data. pbValid: A pointer to a buffer to store the valid mask of the received data. This buffer is useful for error detection and correction. wTimeout: A 16 bit timeout counter used in case an acknowledge is never received. (15 usec units)

Return Value:

bRxDataLength: An 8 bit value of the length of the data (in bytes) that was received.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_StartNonBlockingRead

Description:

Initializes receive mode and enables the interrupt service routine. The ISR will be called each time a byte is received. In order for non-blocking receive mode to function properly psocgpioint.asm must call the receive ISR routine.

C Prototype:

```
BYTE WirelessUSBLS_StartNonBlockingRead(BYTE bLength, BYTE *pbData, BYTE *pbValid)
```

Assembler:

```
mov  A, pbValid
push A            ; Push MSB of pbValid onto stack
mov  A, pbValid+1
push A            ; Push LSB of pbValid onto stack
mov  A, pbData
push A            ; Push MSB of pbData onto stack
mov  A, pbData+1
push A            ; Push LSB of pbData onto stack
mov  A, [bLength]
push A            ; Push bLength onto stack
lcall WirelessUSBLS_StartNonBlockingRead
```

Parameters:

bLength: An 8 bit value used to define the length of the receive buffer. pbData: A pointer to a buffer that stores the received data. pbValid: A pointer to a buffer that checks the validity of the received packet.

Return Value:

bRxDataLength: An 8 bit value of the length of the data (in bytes) that was received.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_bCheckRxInt

Description:

Checks for an EOF to be reached during non-blocking receive mode.

C Prototype:

```
BYTE WirelessUSBLS_bCheckRxInt(void)
```

Assembler:

```
lcall WirelessUSBLS_bCheckRxInt     ;Return Value is in the Accumulator
```

Parameters:

None.

Return Value:

bIntRxDataLength: An 8 bit value of the length of the data (in bytes) that was received.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_bCheckRxInProgress

Description:

Checks to see if a non-blocking receive is in progress.

C Prototype:

```
BYTE WirelessUSBLS_bCheckRxInProgress(void)
```

Assembler:

```
lcall WirelessUSBLS_bCheckRxInProgress
```

Parameters:

None.

Return Value:

Status: 1 if in progress, else 0.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same isTrue for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_StopNonBlockingRead

Description:

Disables interrupts and puts the radio in idle mode. The ISR will no longer be called.

C Prototype:

```
void WirelessUSBLS_StopNonBlockingRead(void)
```

Assembler:

```
lcall WirelessUSBLS_StopNonBlockingRead
```

Parameters:

None.

Return Value:

None.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_Sleep

Description:

Puts the radio into sleep mode, which reduces the current to conserve power.

C Prototype:

```
void WirelessUSBLS_Sleep(void)
```

Assembler:

```
lcall WirelessUSBLS_Sleep
```

Parameters:

None.

Return Value:

None.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_Wakeup

Description:

Brings the radio out of sleep mode; this function has a delay of CrystalStartUpTime msec. If Crystal-Adjust is non-zero there is an additional delay of 2 msec.

C Prototype:

```
void WirelessUSBLS_Wakeup(void)
```

Assembler:

```
lcall WirelessUSBLS_Wakeup
```

Parameters:

None.

Return Value:

None.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_GetMid

Description:

Manufacturing ID (MID)

C Prototype:

```
void WirelessUSBLS_GetMid(BYTE *pbMid)
```

Assembler:

```
mov  A, pbMid    ; Load MSB into A
mov  X, pbMid+1  ; Load LSB into X
lcall WirelessUSBLS_GetMid
```

Parameters:

pbMid: A pointer to a four byte buffer to store the MID of the radio.

Return Value:

None.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_bGetRssi

Description:

Returns the signal strength of the channel noise floor.

C Prototype:

```
BYTE WirelessUSBLS_bGetRssi(void)
```

Assembler:

```
lcall WirelessUSBLS_bGetRssi  ;Return value is in the Accumulator
```

Parameters:

None.

Return Value:

An 8 bit value of the current valid RSSI reading. (Range: 0x00 – 0x1F)

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_SetTxPowerLevel

Description:

Sets the power amplitude level for transmitting data.

C Prototype:

```
void WirelessUSBLS_SetTxPowerLevel(BYTE bPowerLevel)
```

Assembler:

```
mov  A, [bPowerLevel]
lcall WirelessUSBLS_SetTxPowerLevel
```

Parameters:

bPowerLevel: An 8-bit value used to define the power amplitude level when transmitting data. (Range: 0-7)

Return Value:

None.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_bGetTxPowerLevel

Description:

Retrieves the current power amplitude level used for transmitting data.

C Prototype:

```
BYTE WirelessUSBLS_bGetTxPowerLevel(void)
```

Assembler:

```
lcall WirelessUSBLS_bGetTxPowerLevel  ;Return value is in the Accumulator
```

Parameters:

None.

Return Value:

An 8 bit value of the current power amplitude level. (Range: 0-7)

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_bReadRegister

Description:

Gets the data stored in the radio register that is passed in through A.

C Prototype:

```
BYTE TestWirelessUSBLS_bReadRegister(BYTE bRegister)
```

Assembler:

```
lcall WirelessUSBLS_bReadRegister
```

Parameters:

Address: Address of the register that contains the requested data.

Return Value:

Data: Data that was stored in the register.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

WirelessUSBLS_WriteRegister

Description:

Writes data to a radio register.

C Prototype:

```
void TestWirelessUSBLS_WriteRegister(BYTE bRegister, BYTE bData)
```

Assembler:

```
lcall WirelessUSBLS_WriteRegister
```

Parameters:

Address: Address of the register that contains the requested data. Data: Data that is written to the register.

Return Value:

None.

Side Effects:

The A and X registers may be modified by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C27x34). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

Sample Firmware Source Code

The following is a simple C and assembly example for transmitting a byte of data from the radio.

```
//-------------------------------------------------------------------
// Sample C code for WirelessUSBLS
//
//  Transmit one byte of data (55h) from the radio
//
//-------------------------------------------------------------------
#include <m8c.h>          // part specific constants and macros
#include "PSoCAPI.h"      // PSoC API definitions for all User Modules

const BYTE abPnCodeTable[] =
{
    0x36, 0xF3, 0x8C, 0xB5, 0x11, 0x4A, 0xCA, 0x1F,
   0x36, 0xF3, 0x8C, 0xB5, 0x11, 0x4A, 0xCA, 0x1F,
   0x36, 0xF3, 0x8C, 0xB5, 0x11, 0x4A, 0xCA, 0x1F,
   0x36, 0xF3, 0x8C, 0xB5, 0x11, 0x4A, 0xCA, 0x1F,
   0x36, 0xF3, 0x8C, 0xB5, 0x11, 0x4A, 0xCA, 0x1F,
   0x36, 0xF3, 0x8C, 0xB5, 0x11, 0x4A, 0xCA, 0x1F,
   0x36, 0xF3, 0x8C, 0xB5, 0x11, 0x4A, 0xCA, 0x1F,
   0x36, 0xF3, 0x8C, 0xB5, 0x11, 0x4A, 0xCA, 0x1F,
```

```
};

void main(void)
{
   BYTE bTxBuffer[2] = { 0x55, 0xAA };     //Transmit buffer with two bytes

    WirelessUSBLS_Start();                  //Initialize the radio

    WirelessUSBLS_SetChannel(0x07);                  //Set to channel 7
    WirelessUSBLS_SetPnCode(&abPnCodeTable[8 * 1]);  //Set to PN code 1
    WirelessUSBLS_SendData(2, (BYTE *)&bTxBuffer);   //Transmit the data
}
```

A sample project written in assembly is as follows.

```
;;-------------------------------------------------------------------
;; Sample asm WirelessUSB Code
;;
;;  Transmit one byte of data (55h) from the radio
;;-------------------------------------------------------------------
include "m8c.inc"        ; part specific constants and macros
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

export _main

area text (ROM, REL)

_main:
  call   WirelessUSBLS_Start        ; Initialize WirelessUSBLS
  mov    A, 07h                     ; Set to channel 7
  call   WirelessUSBLS_SetChannel

   mov    A, PnCodeTable             ; load MSB of PN code index into A
   mov    X, PnCodeTable+1           ; load LSB of PN code index into X
   call   WirelessUSBLS_SetPnCode
  mov    A, TxBuffer                ; push MSB of TxBuffer pointer
   push   A
  mov    X, TxBuffer+1              ; push LSB of TxBuffer pointer
  push   A
  mov    A, 1                       ; Length of transfer data (in bytes) is 1
   push   A
  call   WirelessUSBLS_SendData

.LITERAL
TxBuffer:
    db 55h
EndTxBuffer:
.ENDLITERAL

.LITERAL
PnCodeTable:
    db 36h, F3h, 8Ch, B5h, 11h, 4Ah, CAh, 1Fh
   db 36h, F3h, 8Ch, B5h, 11h, 4Ah, CAh, 1Fh
   db 36h, F3h, 8Ch, B5h, 11h, 4Ah, CAh, 1Fh
   db 36h, F3h, 8Ch, B5h, 11h, 4Ah, CAh, 1Fh
```

```
   db 36h, F3h, 8Ch, B5h, 11h, 4Ah, CAh, 1Fh
    db 36h, F3h, 8Ch, B5h, 11h, 4Ah, CAh, 1Fh
    db 36h, F3h, 8Ch, B5h, 11h, 4Ah, CAh, 1Fh
   db 36h, F3h, 8Ch, B5h, 11h, 4Ah, CAh, 1Fh
EndPnCodeTable:
.ENDLITERAL
```

Version History

Version	Originator	Description
1.0	DHA	Initial version

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.