Application Note: JN-AN-1252

High Power Support

Supporting High Power Mode on a JN5189/QN9090/K32W device.

1 Application Note Overview

The following devices

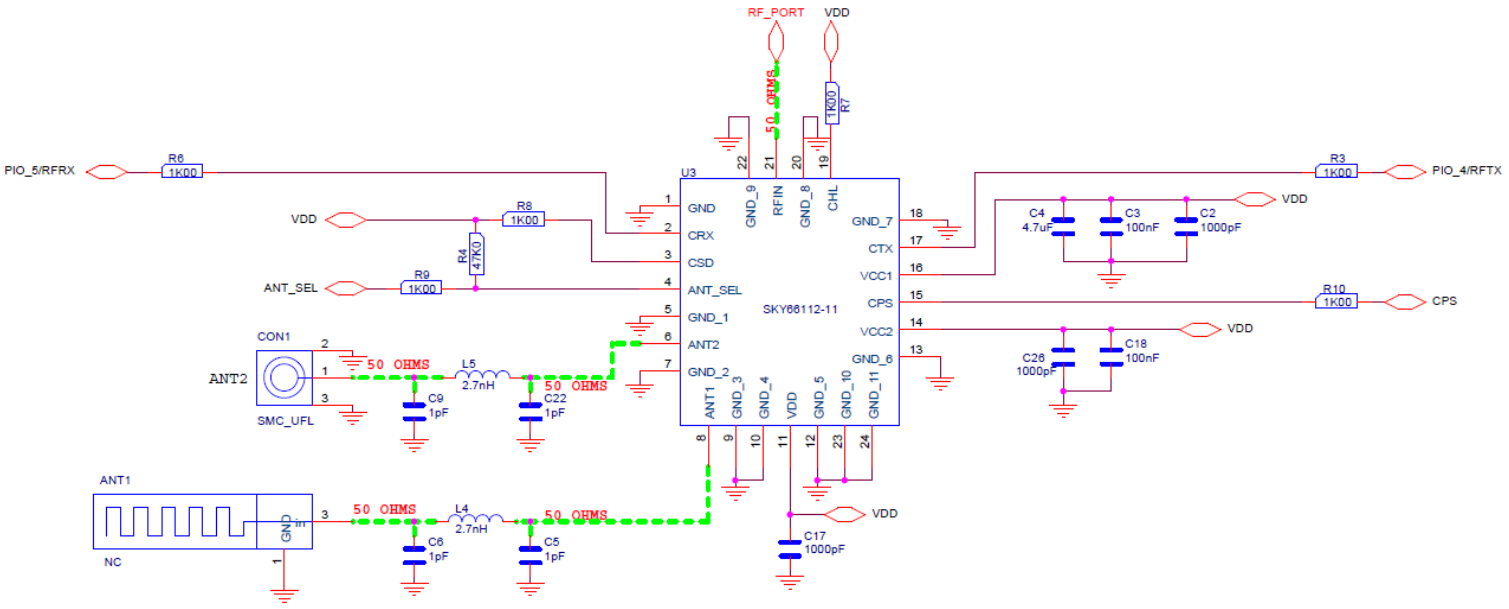
- JN5189(T)/JN5188(T)
- QN9090(T)/QN9030(T)
- K32W061/K32W041

feature an integrated radio, which can be used with an external Front End. This application note is concerned with setting up the device for hardware designs that uses a FEM (Front-End module).

1.1 Contents

2 FEM Design

The parameters used in this example are suitable for a test design based on a Skyworks SKY66112-11. The exact parameters needed for compliance with different Front Ends or different layouts will require additional testing to confirm that the parameters used are compliant. The examples design used is shown below:



3 What is Required

To support a Front-End module the following is required

1) Set up RX and TX DIO control
2) Set the compliance limit parameter on the radio

3.1 Setting up the DIO

Some of the PIO lines can be used to generate RFRX and RFTX signals. These signals are connected directly from the MODEM with the device and indicate if the device is in RX or TX mode and, by implication, if the MODEM is idle. Any of the PIOS that have RFRX and RFTX options can be used from the table below

I/O Names	FUNC0	FUNC1	FUNC2	FUNC3	FUNC4	FUNC5	FUNC6	FUNC7	Default at reset / during boot	Default internal pull-up / pull-down	ISP mode
PIO0_0	GPIO 0	USART0_SCK	USART1_TXD	GPIOSEC 0	PWM0-PU	SPI1_SCK	PTA0	PDM0_DATA	GPIO0 (I)	pull-up	
PIO0_1	GPIO 1	FLICKER_CTRL	USART1_RXD	GPIOSEC 1	PWM1-PD	SPI1_MISO	PTA1	PDM0_CLK	GPIO1 (I)	pull-down	
PIO0_2	GPIO 2	SPI0_SCK	USART0_RXD	GPIOSEC 2	PWM2-PD	SPI1_MOSI	ISO7816_RST	MCLK	GPIO2 (I)	pull-down	
PIO0_3	GPIO 3	SPI0_MISO	USART0_TXD	GPIOSEC 3	PWM3-PU	SPI1_SSELN0	ISO7816_CLK	PTA2	GPIO3 (I)	pull-up	
PIO0_4	GPIO 4	SPI0_MOSI	USART0_CTS	GPIOSEC 4	PWM4-PU	SPI1_SSELN1	ISO7816_IO	RFTX	GPIO4 (I)	pull-up	
PIO0_5	GPIO 5	SPI0_SSELN	USART0_RTS	GPIOSEC 5	SPI1_MISO	SPI1_SSELN2	PTA3	RFRX	GPIO5 (I)	pull-up	ISP_ENTRY
PIO0_6	GPIO 6	SPI0_SCK	USART0_RTS	CT32B1_MAT0	PWM6-PD	I2C1_SCL	USART1_TXD	ADE	GPIO6 (I)	pull-down	
PIO0_7	GPIO 7	SPI0_MISO	USART0_CTS	CT32B1_MAT1	PWM7-PD	I2C1_SDA	USART1_RXD	ADO	GPIO7 (I)	pull-down	
PIO0_8	GPIO 8	SPI0_MOSI	USART0_TXD	CT32B0_MAT0	PWM8-PU	ANA_COMP_OUT	RFTX	PDM1_DATA	USART0_TXD	pull-up	UART_ISP
PIO0_9	GPIO 9	SPI0_SSELN	USART0_RXD	CT32B1_CAP1	PWM9-PU	USART1_SCK	ADO	PDM1_CLK	USART0_RXD	pull-up	UART_ISP
PIO0_10	GPIO 10	CT32B0_CAP0	USART1_TXD	GPIOSEC 10	RFTX	I2C0_SCL	SPI0_SCK	PDM0_DATA	GPIO10 (I)	external pull-up	
PIO0_11	GPIO 11	CT32B1_CAP0	USART1_RXD	GPIOSEC 11	RFRX	I2C0_SDA	SPI0_MISO	PDM0_CLK	GPIO11 (I)	external pull-up	
PIO0_12	GPIO 12	IR_BLASTER	SWCLK	GPIOSEC 12	PWM0-PU	I2C1_SCL	SPI0_MOSI	ANA_COMP_OUT	SWCLK	pull-up	
PIO0_13	GPIO 13	SPI1_SSELN2	SWDIO	GPIOSEC 13	PWM2-PU	I2C1_SDA	SPI0_SSELN	FLICKER_CTRL	SWDIO	pull-up	
PIO0_14/ADC	GPIO 14	SPI1_SSELN1	USART0_SCK	CT32B0_CAP1	PWM1-PU	SWO	MCLK	RFTX	GPIO14 (I)	pull-up	
PIO0_15/ADC	GPIO 15	SPI1_SCK	ANA_COMP_OUT	FLICKER_CTRL	PWM3-PU	I2C0_SCL	PDM1_DATA	RFRX	GPIO15 (I)	pull-up	
PIO0_16/ADC	GPIO 16	SPI1_SSELN0	ISO7816_RST	GPIOSEC 16	PWM5-PU	I2C0_SDA	PDM1_CLK	SPIFI_CSN	GPIO16 (I)	pull-up	
PIO0_17/ADC	GPIO 17	SPI1_MOSI	ISO7816_CLK	SWO	PWM6-PD	CLK_OUT	PTA0	SPIFI_IO3	GPIO17 (I)	pull-down	
PIO0_18/ADC	GPIO 18	SPI1_MISO	ISO7816_IO	CT32B0_MAT1	PWM7-PD	USART0_TXD	PTA1	SPIFI_CLK	GPIO18 (I)	pull-down	
PIO0_19/ADC	GPIO 19	ADO	USART1_RXD	CLK_IN	PWM4-PD	USART0_RXD	PTA2	SPIFI_IO0	GPIO19 (I)	pull down	
PIO0_20/ACM	GPIO 20	IR_BLASTER	USART1_TXD	GPIOSEC 20	PWM8-PD	RFTX	PTA3	SPIFI_IO2	GPIO20 (I)	pull-down	
PIO0_21/ACP	GPIO 21	IR_BLASTER	USART1_SCK	FLICKER_CTRL	PWM9-PU	RFRX	SWO	SPIFI_IO1	GPIO21 (I)	pull-up	

Special PIO functions are setup using IOCON_PinMuxSet. To use this function, include the following header

#include "fsl_iocon.h"

In our example, the required functions are on DIO4 and DIO5 using IOCON_FUNCTION 7. This is done with the following two lines of code:

```
IOCON_PinMuxSet(IOCON, 0, 4, IOCON_MODE_INACT | IOCON_FUNC7 | IOCON_DIGITAL_EN);
IOCON_PinMuxSet(IOCON, 0, 5, IOCON_MODE_INACT | IOCON_FUNC7 | IOCON_DIGITAL_EN);
```

4 Zigbee 3.0 Applications

4.1 Setting up the Radio Parameters

There are two functions that are defined to set up the radio. Both should be used in conjunction to configure the radio. These functions are defined in AppApi.h.

The first function is

PUBLIC void vAppApiSetRadioTxModes(teRadioTxMode eTxMode, teRadioTxMode eTxModeCh26);

This is concerned with changing the radios internal settings to set the radio is in different proprietary modes affect currents and setting within the radio. The following are defined

E_RADIO_TX_MODE_STD = 0
E_RADIO_TX_MODE_PROP_1 = 1
E_RADIO_TX_MODE_PROP_2 = 2

The standard way to use this would be to set the radio XT mode to standard. If there are issues on radio channel 26 with band edge due to the output power from the front end then using proprietary mode for channel 26 may be required. In our example, the following is used:

```
vAppApiSetRadioTxModes(E_RADIO_TX_MODE_STD, E_RADIO_TX_MODE_PROP_2);
```

The second function is

PUBLIC void vAppApiSetComplianceLimits(int8 i8TxMaxPower,int8 i8TxMaxPowerCh26, uint8 u8CcaThreshold);

This function deals with standard parameters for the radio such as CCA threshold. These parameters should be evaluated using CMET.

In our example, the radio is set to output 4db and the CCA threshold is adjusted to 83 to deal the RX gain in the LNA and still be compliant with the IEEE802.15.4 specification. The following is used:

```
vAppApiSetComplianceLimits(4, 4, 83);
```


4.2 Editing the Zigbee 3.0 Application Notes

The code need to be added before **BEFORE** ZPS_eAplAfInit() is called. In 'JN-AN-1244-ZigBee-3-0-Light-Bulbs-for-JN518x', the most suitable location is in APP_vInitialiseNode() in app_zlo_light_node.c. Add the following code before ZPS_eAplAfInit() is called:

```
// Start of HP Support Initialisation
IOCON_PinMuxSet(IOCON, 0, 4, IOCON_MODE_INACT | IOCON_FUNC7 | IOCON_DIGITAL_EN);
IOCON_PinMuxSet(IOCON, 0, 5, IOCON_MODE_INACT | IOCON_FUNC7 | IOCON_DIGITAL_EN);
vAppApiSetComplianceLimits(4, 4, 83);
vAppApiSetRadioTxModes(E_RADIO_TX_MODE_STD, E_RADIO_TX_MODE_PROP_2);
// End of HP Support Initialisation
…
// Existing Code
ZPS_u32MacSetTxBuffers (4);
/* Initialise ZBPro stack */
ZPS_eAplAfInit();
```

5 BLE Applications

5.1 Editing the BLE Example Applications

The Code need to be added within the BleApp_Init function. For example, in 'beacon_bm', the code should be added at the end of void BleApp_Init(void) in beacon.c :

```
// Start of HP Support Initialisation
IOCON_PinMuxSet(IOCON, 0, 4, IOCON_MODE_INACT | IOCON_FUNC7 | IOCON_DIGITAL_EN);
IOCON_PinMuxSet(IOCON, 0, 5, IOCON_MODE_INACT | IOCON_FUNC7 | IOCON_DIGITAL_EN);
```

Revision History

Version	Notes
1V0	First release
1V1	Added BLE support

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 12/2019
Document identifier: JN-AN-1252