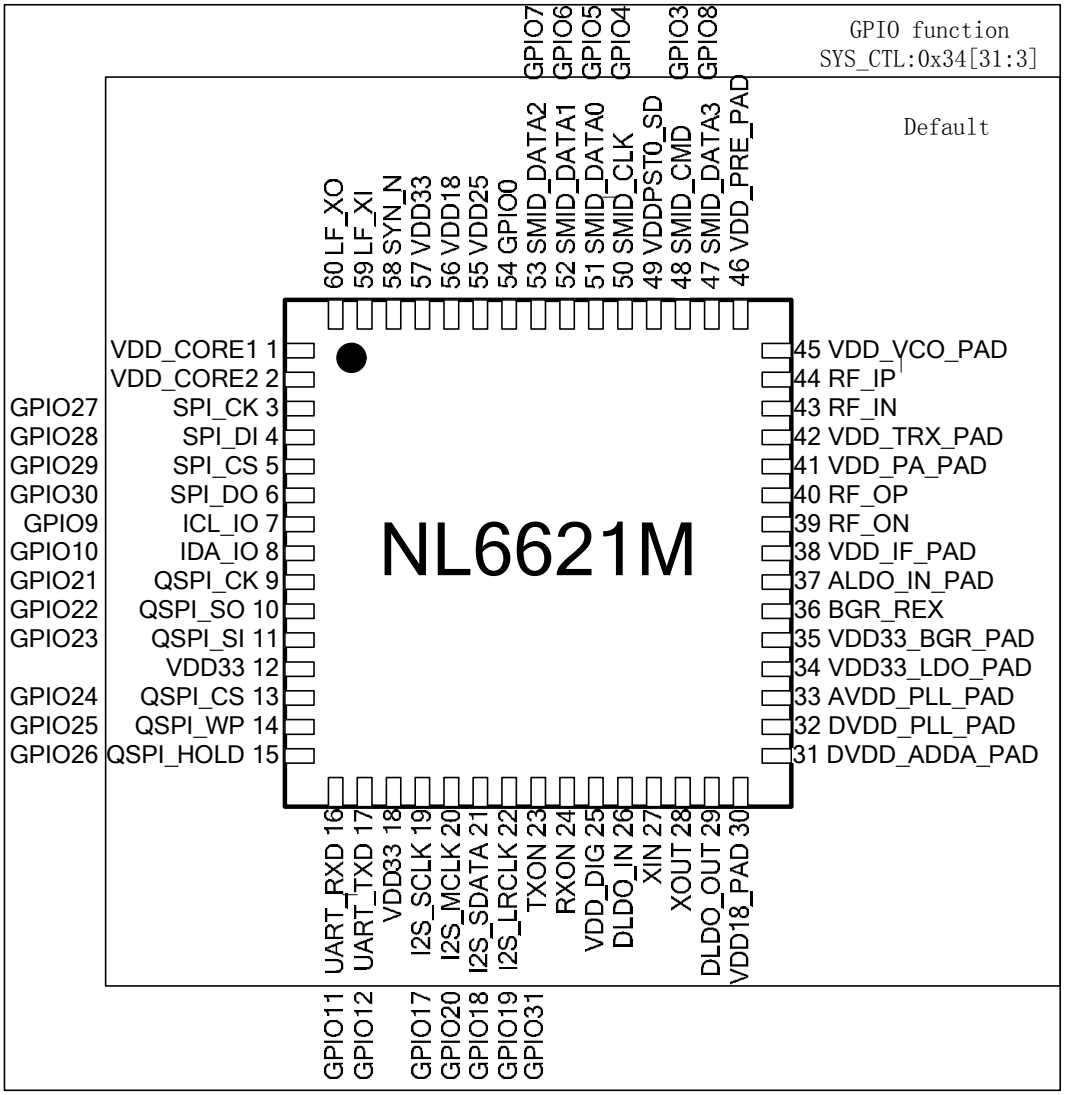
NL6621 QFN60 GPIO 使用说明
V1.0

修订记录：

Version	Description	Date	Author
V1.0	初稿	2014-9-1	刘健生

目录

一、QFN60 GPIO pin 脚对照图：



二、QFN60 GPIO register 说明：

5， 管脚描述

5.1.管脚类型

管脚类型	描述
IPU	内部上拉
IPD	内部下拉
I	输入
O	输出
IO	双向
P	电源
G	地
A	模拟
NC	未连接

5.2.管脚描述

管脚	编号.	IO 类型	描述
VDD_CORE1	1	O	1.2V LDO output , Bypass with a capacitor as close to the pin as possible.
VDD_CORE2	2	O	1.2V LDO output, Bypass with a capacitor as close to the pin as possible.
SPI_CK	3	I/O	SPI clock when NL6621M as SPI master, can also be configured as GPIO
SPI_DI	4	I/O	SPI data in when NL6621M as SPI master, can also be configured as GPIO
SPI_CS	5	I/O	SPI chip select when NL6621M as SPI master, can also be configured as GPIO
SPI_DO	6	I/O	SPI data out when NL6621M as SPI master, can also be configured as GPIO
ICL	7	I/O	I2C clock when NL6621M as I2C master, pull up outside, can also be configured as GPIO
IDA	8	I/O	I2C data when NL6621M as I2C master, pull up outside, can also

			be configured as GPIO
QSPI_CK	9	I/O	QSPI clock when NL6621M as QSPI master, can also be configured as GPIO
QSPI_SO	10	I/O	QSPI data out when NL6621M as QSPI master, can also be configured as GPIO, strapping pin, see blow description
QSPI_SI	11	I/O	QSPI data input when NL6621M as QSPI master, can also be configured as GPIO
VDD33	12	P	3.3V power supply for IO interface
QSPI_CS	13	I/O	QSPI chip select when NL6621M as QSPI master, can also be configured as GPIO
QSPI_WP	14	I/O	QSPI write protect when NL6621M as QSPI master, can also be configured as GPIO, strapping, see below description
QSPI_HOLD	15	I/O	QSPI hold when NL6621M as QSPI master, can also be configured as GPIO, strapping with QSPI_DO,QSPI_WP as: {QSPI_HOLD, QSPI_WP, QSPI_DO} 000: load firmware from SDIO/SPI 001, load firmware from i2c EEPROM 010: load firmware from QSPI Flash 011: load firmware from UART 100: SW Debug Mode only when QFN88 Package 101: QSPI flash execution directly 110: efuse burning mode 111: reserved

UART_RX	16	I/O	UART RXD, can also be configured as GPIO
UART_TX	17	I/O	UART TXD, can also be configured as GPIO
VDD33	18	P	3.3V power supply for IO interface
I2S_SCLK	19	I/O	I2S SCLK, can also be configured as GPIO
I2S_MCLK	20	I/O	I2S master clock, can also be configured as GPIO
I2S_SDATA	21	I/O	I2S DATA, can also be configured as GPIO
I2S_RLCLK	22	I/O	I2S R/L clock, can also be configured as GPIO
TXON	23	O	TX mode enable digital input ,set high to enable TX
RXON	24	O	RX mode enable digital input,set high to enable RX
VDD_DIG	25	P	3.3V power supply for Digital IO post-drive voltage
DLDO_IN	26	P	1.8V supply for AFE-LDO's
XIN	27	Crystal Input	40 MHz crystal oscillator input or external clock input
XOUT	28	Crystal Output	40 MHz crystal oscillator output
DLDO_OUT	29	O	IF-LDO's 1.2V output, Bypass with a capacitor as close to the pin as possible.
VDD18_PAD	30	P	1.8V power supply for RF-LDOs'

DVDD_ADDA_PAD	31	O	LDO's output. Bypass with a capacitor as close to the pin as possible.
DVDD_PLL_PAD	32	O	LDO's output. Bypass with a capacitor as close to the pin as possible.
AVDD_PLL_PAD	33	O	LDO's output. Bypass with a capacitor as close to the pin as possible.
VDD33_LDO_PAD	34	P	3.3V power supply
VDD33_BGR_PAD	35	P	3.3V power supply
BGR_REX	36	O	24Kom resister
ALDO_IN_PAD	37	P	1.8V power supply
VDD_IF_PAD	38	O	LDO's output. Bypass with a capacitor as close to the pin as possible.
RF_ON	39	O	PA's negative output
RF_OP	40	O	PA's positive output
VDD_PA_PAD	41	P	3.3V power supply
VDD_TRX_PAD	42	O	LDO's output. Bypass with a capacitor as close to the pin as possible.
RF_IN	43	I	LNA's negative port input
RF_IP	44	I	LNA's positive port input
VDD_VCO_PAD	45	O	LDO's output. Bypass with a capacitor as close to the pin as possible.
VDD_PRE_PAD	46	O	LDO's output. Bypass with a capacitor as close to the pin as possible.
SD_DATA3	47	I/O	SDIO data0 pin, can also be configured as GPIO
SD_CMD	48	I/O	SDIO CMD pin, can also be configured as GPIO
VDDPST_SD	49	P	3.3V power supply for SDIO interface
SD_CLK	50	I	SDIO CLK pin, can also be configured as GPIO
SD_DATA0	51	I/O	SDIO data0 pin, can also be configured as GPIO
SD_DATA1	52	I/O	SDIO data1 pin, can also be configured as GPIO
SD_DATA2	53	I/O	SDIO data2 pin, can also be configured as GPIO

GPIO0	54	I/O	Multi-function multiplexed pin. **Funciont 1:** GPIO0 for general purpose IO usage after system reset or power on reset (Default). It can be used as input wake up signal from host to make the chip recovers from sleeping state. **Function 2:** Input pin for strapping register **NO_32K_MODE.** The input value (Usually by external pull up/down) before power on reset or system reset released is latched to register **NO_32K_MODE** at the time when the reset is releasing. After reset released, the latched value in register **NO_32K_MODE** will keep unchanged until the next reset happens and the pin is ready for use in function 1 or function 2. The register **NO_32K_MODE** is defined as: **1'b1:** No external 32.768 KHz crystal/ oscillator mode. The 32.768 Khz clock for active clock in sleeping state is generated by divided clock from 40MHz in this mode. It's the lowest system cost design by saving a 32.768 Khz crystal/ oscillator. **1'b0:** External 32.768 KHz crystal/ oscillator mode. The 32.768

			Khz clock for active clock in sleeping state is from external 32.768 Khz crystal/ oscillator. 40 MHz oscillator can be powered off to achieve the lowest power consumption in sleeping state.
VDD25	55	P	2.5V power supply for EFUSE write; Normal Condition，this pin is floating
VDD18	56	P	1.8V power supply for LDO
AVDD33	57	P	3.3V power supply for IO
RSTN	58	I	Chip reset input pin. Tie this pin HIGH if only use on chip power on reset. Connect this pin to system reset if you want to reset the chip from other components in the system.
LF_XIN	59	Crystal Input	32.768 KHz crystal input or external clock input
LF_XOUT	60	Crystal Output	32.768 KHz crystal output

34	GPIO_PIN_MUX_CTRL	WR	31:0	Bit [31:3]： 29个GPIO的管脚复用的GPIO使能信号。bit为1控制相应的管脚为GPIO。 Bit[2]: PAON 使能信号。0：GPIO2;1：PAON。 Bit[1]: I2S 和JTAG的MUX 选择。0：I2S接口；1：JTAG 接口。 Bit[0]：SDIO mux 成UART和蓝牙共存接口。0：SDIO 接口；1：UART和蓝牙共存接口。	32'h0004

位宽应为 32!!

23, GPIO 寄存器

寄存器名称	偏 移地址	RW	位宽	Default	Description
gpio_swporta_dr	0x00	RW	16	0	Values written to this register are output on the I/O signals for GPIO Port if the corresponding data direction bits for Port A are set to Output mode and the corresponding control bit for Port A is set to Software mode. The value read back is equal to the last value written to this

					register.
gpio_swporta_ddr	0x04	RW	16	0	Values written to this register independently control the direction of the corresponding data bit in Port A. The default direction can be configured as input or output after system reset through the GPIO_DFLT_DIR_A parameter. 0 – Input (default) 1 – Output
gpio_inten	0x30	RW	16	0	Allows each bit of Port to be configured for interrupts. By default the generation of interrupts is disabled. Whenever a 1 is written to a bit of this register, it configures the corresponding bit on Port to become an interrupt, otherwise,Port operates as a normal GPIO signal. Interrupts are disabled on the corresponding bits of Port if the corresponding data direction register is set to Output or if Port A mode is set to Hardware. 0 – Configure Port A bit as normal GPIO signal (default) 1 – Configure Port A bit as interrupt

gpio_intmask	0x34	RW	16	0	Controls whether an interrupt on Port can create an interrupt for the interrupt controller by not masking it. By default, all interrupts bits are unmasked. Whenever a 1 is written to a bit in this register, it masks the interrupt generation capability for this signal; otherwise interrupts are allowed through. The unmasked status can be read as well as the resultant status after masking. 0 – Interrupt bits are unmasked (default) 1 – Mask interrupt
gpio_inttype_level	0x 38	RW	16	0	Controls the type of interrupt that can occur on Port Whenever a 0 is written to a bit of this register, it configures the interrupt type to be level-sensitive; otherwise, it is edge-sensitive. 0 – Level-sensitive (default) 1 – Edge-sensitive

gpio_int_polarity	0x3c	RW	16	0	Controls the polarity of edge or level sensitivity that can occur on input of Port A. Whenever a 0 is written to a bit of this register, it configures the interrupt type to falling-edge or active-low sensitive; otherwise, it is rising-edge or active-high sensitive. 0 – Active-low (default) 1 – Active-high
gpio_intstatus	0x40	R	16	0	Interrupt status of GPIO port
gpio_raw_intstatus	0x44	R	16	0	Raw interrupt of status of GPIO port (premasking bits)
gpio_porta_eoi	0x4c	W	16	0	Controls the clearing of edge type interrupts from Port. When a 1 is written into a corresponding bit of this register, the interrupt is cleared. All interrupts are cleared when Port A is not configured for interrupts. 0 – No interrupt clear (default) 1 – Clear interrupt
gpio_ext_porta	**0x50**	R	16	0	When Port is configured as Input, then reading this location reads the values on the signal. When the data direction of Port is set as Output, reading this location reads the data register for Port A. **Reset Value:** 0x0
gpio_ls_sync	**0x60**	RW	16	0	Writing a 1 to this register results in all level-sensitive interrupts being synchronized to pclk_intr. 0 – No synchronization to pclk_intr (default) 1 – Synchronize to pclk_intr
gpio_id_code	**0x64**	R	16	0	This is a user-specified code that a system can read. It can be used for chip identification, and so on.
gpio_ver_id_code	**0x6c**	R	32		ASCII value for each number in the version, followed by *. For example 32_30_31_2A represents the version

三、QFN60 GPIO 例子说明：

例子，下面的函数封装了设置成 gpio 口的功能，以及对 gpio 口的操作。
相关的宏在代码中可找到， 有些可能不能找到， 下面给出相关定义。

```
#define SWPORTA_DDR                    (0x04)
#define EXT_PORTA                      (0x50)
#define SWPORTA_DR                     (0x0)
```

```
static void set_gpio_as_output_(UINT8 gpio_number)
{
    NST_WR_PWM_REG(ADDR_GPIO_PIN_MUX_CTRL,
NST_RD_PWM_REG(ADDR_GPIO_PIN_MUX_CTRL) | (1 << gpio_number));
    NST_WR_GPIO_REG(SWPORTA_DDR,  NST_RD_GPIO_REG(SWPORTA_DDR)  |  (1  <<
gpio_number));
}

static void set_gpio_as_input_(UINT8 gpio_number)
{
    NST_WR_PWM_REG(ADDR_GPIO_PIN_MUX_CTRL,
NST_RD_PWM_REG(ADDR_GPIO_PIN_MUX_CTRL) | (1 << gpio_number));
    NST_WR_GPIO_REG(SWPORTA_DDR,  NST_RD_GPIO_REG(SWPORTA_DDR)  &  ~(1  <<
gpio_number));
}

static UINT8 get_gpio_value_(UINT8 gpio_number)   //设置为输入时读 gpio 口
{
    return ((NST_RD_GPIO_REG(EXT_PORTA) & ( 1 << gpio_number ))==0) ? 0 : 1;
}

static UINT8 get_out_gpio_value_(UINT8 gpio_number)    //设置为输出时读 gpio 口
{
    return ((NST_RD_GPIO_REG(SWPORTA_DR) & (1 << gpio_number))==0) ? 0 : 1;
}

static void set_gpio_value_(UINT8 gpio_number, UINT8 level)
{
    if(level == 0) {// low level
        NST_WR_GPIO_REG(0x00,(NST_RD_GPIO_REG(0x00)& ~(1 << gpio_number)));
    } else if(level == 1) {
        NST_WR_GPIO_REG(0x00,(NST_RD_GPIO_REG(0x00)|(1 << gpio_number)));
    }
}
```