BK2433

Datasheet

2.4-GHz Wireless System On-Chip

V1.1

Beken Corporation
3A,1278 Keyuan Rd,Zhangjiang High-Tech Park
Pudong new Distrinct,Shanghai,201203,china
Tel： (86)21 51086811
Fax： (86)21 60871277

Revision History

Version	Date	Author(s)	Description
1.0	Oct 18,2011	Lizhen,Qxu	Initial Release
1.1	Jul 3,2012	Lizhen,Qxu	Modify Package Information

Table of Contents

List of Figures

List of tables

1 Introduction

The BK2433 is a single chip embedded BK51 processor, compact USB dongle and BK2423 2.4GHz RF transceiver. It includes three package sizes: BK2433KB(QFN56 7x7mm), BK2433MB(QFN32 5x5mm),and BK2423QB(QFN24 4x4mm).It supports a wide range of application including:

- PC peripherals
  - Mouse
  - Keyboard
  - Game control
- Remote control
  - Audio video
  - Entertainment
  - Home application
- Security systems
  - Payment
  - Alarm

2 Key feature

- Worldwide 2.4 GHz ISM band operation
- FSK enables better sensitivity and GFSK gives better spectrum efficiency
- 250 kbps, 1 Mbps or 2 Mbps air data rate
- Programmable output power: from 5 to -40 dBm
- Tolerate +/- 60ppm 16 MHz crystal
- 6 data pipes for 1:6 star networks
- 1.9 V to 3.6 V power supply
- USB2.0 full speed controller(12MHz)
- Bk51 MCU compatible with 8051
- One instruction one period, and 70% instruction no more than 2 period
- 32k bytes OTP for program
- Protective MASK ROM (OTP)
- 1k bits MTP to store customer data (minimum 20k program/erase cycle)
- 256 Bytes IRAM and 2k Bytes SRAM
- Embedded three Timer/Counter
- Support I2C, SPI, UART
- DES/3DES encryption/decryption Co-processor
- 16-32bit multiplication/division Co-processor
- Two 10bits PWM controller
- Eight channel 10bits ADC. the highest sample: 29k in continues mode
- Programmable watch dog timer
- low power consumption in Standby mode, embedded with 32k oscillator
- Low power detect circuit
- Total 40 GPIO available
- Three compact package sizes: BK2433KB(QFN56 7x7mm), BK2433MB(QFN32 5x5mm),and BK2423QB(QFN24 4x4mm).

3 Block Diagram

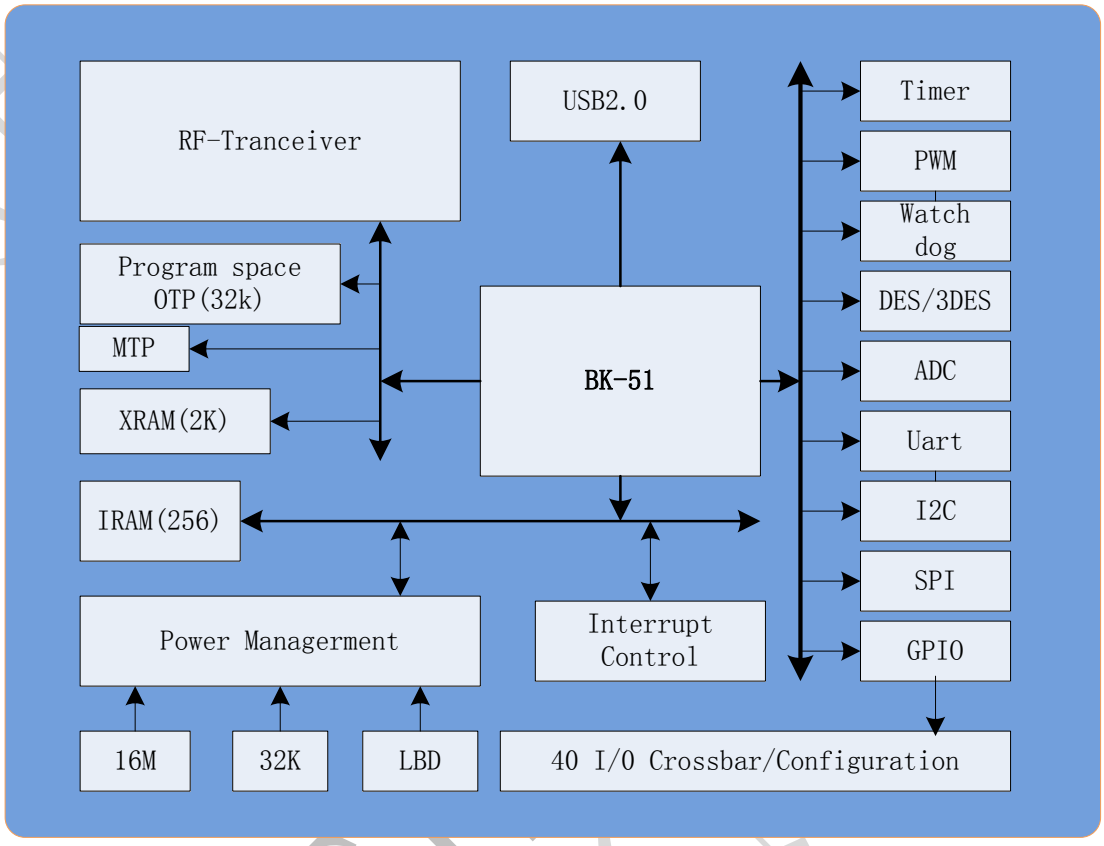


Figure 1BK2433 Block Diagram

4 PIN information

The BK2433 includes three package sizes: BK2433KB(QFN56 7x7mm), BK2433MB(QFN32 5x5mm),and BK2423QB(QFN24 4x4mm).

4.1 BK2433KB(QFN56) pin package

The next diagram shows BK2433KB(QFN56) format for the full functions usage. It can be used as keyboard MCU and total 40 GPIO available.

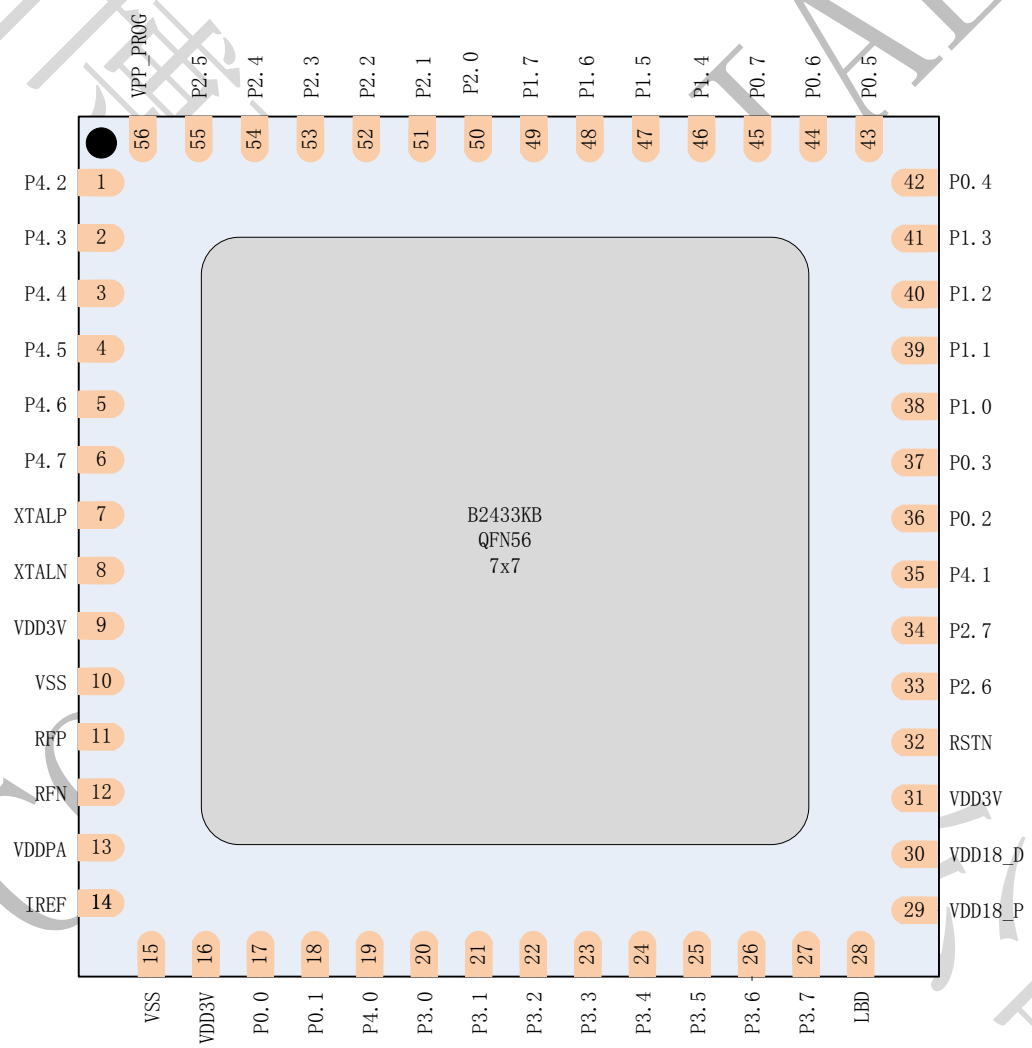


Figure 2 BK2433KB QFN56 pin assignment

PIN	Name	Pin Function	Description
1	P4.2	Digital I/O	General I/O
2	P4.3	Digital I/O	General I/O
3	P4.4	Digital I/O	General I/O
4	P4.5	Digital I/O	General I/O
5	P4.6	Digital I/O	General I/O
6	P4.7	Digital I/O	General I/O
7	XTALP	Analog output	Oscillator output
8	XTALN	Analog input	Oscillator input
9	VDD3V	Power supply	3v supply
10	VSS	ground	
11	RFP	RF port	output(PA) /LNA input p
12	RFN	RF port	output(PA) /LNA input n
13	VDDPA	Power output	1.8v supply to PA
14	IREF	Analog output	Current reference
15	VSS	ground	
16	VDD3V	Power supply	3v supply
17	P0.0	Digital I/O	General I/O，or input for timer0
18	P0.1	Digital I/O	General I/O，or input for timer1
19	P4.0	Digital I/O	General I/O
20	P3.0	Digital I/O or analog input	General I/O，or input of ADC0
21	P3.1	Digital I/O or analog input	General I/O，or input of ADC1
22	P3.2	Digital I/O or analog input	General I/O，or input of ADC2
23	P3.3	Digital I/O or analog input	General I/O，or input of ADC3
24	P3.4	Digital I/O or analog input	General I/O，or input of ADC4
25	P3.5	Digital I/O or analog input	General I/O，or input of ADC5
26	P3.6	Digital I/O or analog input	General I/O，or input of ADC6
27	P3.7	Digital I/O or analog input	General I/O，or input of ADC7
28	LBD	Analog input	Low battery detector
29	VDD18_P	Analog output	power output, connected with decoupling CAP
30	VDD18_D	Analog output	power output, connected with decoupling CAP
31	VDD3V	Power supply	3v supply
32	RSTN	Digital input	Reset for chip, active low
33	P2.6	Digital I/O	General I/O，or enable for PWM0
34	P2.7	Digital I/O	General I/O，or enable for PWM1
35	P4.1	Digital I/O	General I/O
36	P0.2	Digital I/O	General I/O，or input for timer2
37	P0.3	Digital I/O	General I/O，or acquire/reload trigger signal for timer2

38	P1.0	Digital I/O	General I/O，or input for external interrupt 0, active low
39	P1.1	Digital I/O	General I/O，or input for external interrupt 1, active low
40	P1.2	Digital I/O	General I/O
41	P1.3	Digital I/O	General I/O
42	P0.4	Digital I/O	General I/O，or MOSI for SPI
43	P0.5	Digital I/O	General I/O，or MISO for SPI
44	P0.6	Digital I/O	General I/O，or SCK for SPI
45	P0.7	Digital I/O	General I/O，or chip select for SPI
46	P1.4	Digital I/O	General I/O
47	P1.5	Digital I/O	General I/O
48	P1.6	Digital I/O	General I/O
49	P1.7	Digital I/O	General I/O
50	P2.0	Digital I/O	General I/O，or input for UART
51	P2.1	Digital I/O	General I/O，or output for UART
52	P2.2	Digital I/O	General I/O，Serial Port data output for mode 0，only used for mode 0
53	P2.3	Digital I/O	General I/O，or clock for SMBUS (I2C)
54	P2.4	Digital I/O	General I/O，or data I/O for SMBUS (I2C)
55	P2.5	Digital I/O	General I/O
56	VPP_PROG	Program/VPP	VPP supply/Program mode for OTP

Table 1 BK2433KB QFN56 pin description

4.2 BK2433MB(QFN32) pin package

The next diagram shows BK2433MB(QFN32) format for mouse application.

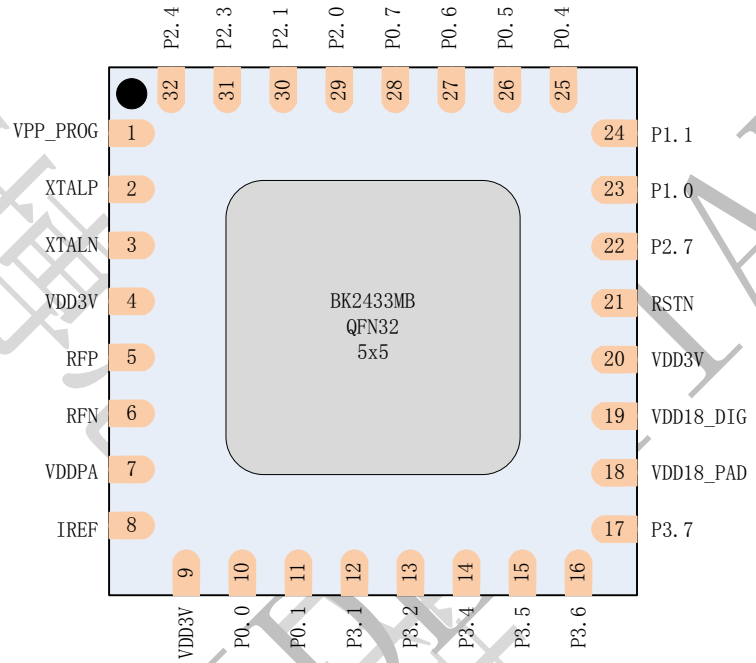


Figure 3 BK2433MB QFN32 pin assignment

The pin description of QFN32 is same as QFN56, exclude the removed pin.

4.3 BK2423QB(QFN24) pin package

The next diagram shows BK2423QB(QFN24) pin package used for dongle part.

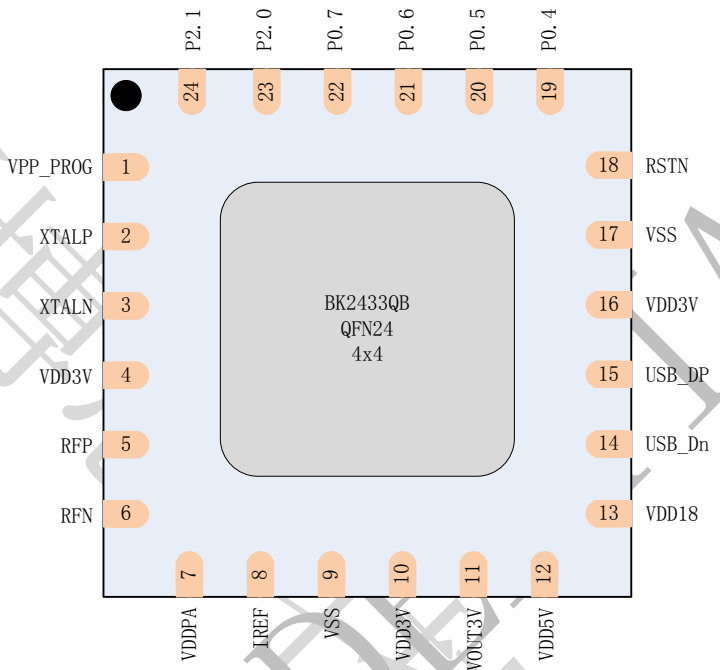


Figure 4 BK2423QB QFN24 pin assignment

PIN	Name	Pin Function	Description
1	VPP_PROG	Program/VPP	VPP supply/Program mode for OTP
2	XTALP	Analog output	Oscillator output
3	XTALN	Analog input	Oscillator input
4	VDD3V	Power supply	3v supply
5	RFP	RF port	output(PA) /LNA input p
6	RFN	RF port	output(PA) /LNA input n
7	VDDPA	Power output	1.8v supply to PA
8	IREF	Analog output	Current reference
9	VSS	ground	
10	VDD3V	Power supply	3v supply
11	VOUT3V	Analog output	3v power output, connected with decoupling CAP
12	VDD5V	Power	5V supply for USB
13	VDD18	Analog output	power output, connected with decoupling CAP
14	USB_DN	Digital I/O	USB input N
15	USB_DP	Digital I/O	USB input P
16	VDD3V	Power supply	3v supply

17	VSS	ground	
18	RSTN	Digital input	Reset for chip, active low
19	P0.4	Digital I/O	General I/O，or MOSI for SPI
20	P0.5	Digital I/O	General I/O，or MISO for SPI
21	P0.6	Digital I/O	General I/O，or SCK for SPI
22	P0.7	Digital I/O	General I/O，or chip select for SPI
23	P2.0	Digital I/O	General I/O，or input for UART
24	P2.1	Digital I/O	General I/O，or output for UART

Table 2 BK2423QB QFN24 pin description

5 BK51 Micro-Controller

5.1 Instruction Set

The core of MCU is BK-51 processor。All BK-51 instructions are binary code compatible and perform the same functions that they do in the industry standard 8051. The effects of these instructions on bits, flags, and other status functions are identical to the industry-standard 8051. However, the timing of the instructions is different, both in terms of number of clock cycles per instruction cycle and timing within the instruction cycle.

BK-51 has comprised all the peripheral equipment of the standard 8051.It include three 16bits timer, one duplex UART, one enhanced SPI, 256 bytes internal RAM, 128 bytes SFR and 32 general I/O port. But it does not include JTAG.

Besides these standard architecture and peripheral equipment, BK-51 has other custom-build peripheral equipment and function, which increase its ability greatly.

BK-51 feature：

- Compatible with MCS-51
- 16MHz frequency， can work at 16M/8M/4M/2M Hz frequency
- 80% instruction consume 1~3 clock period only
- 256 bytes internal RAM
- expanded interrupt deal system
- Pwer managerment mode
- Security for program and storage

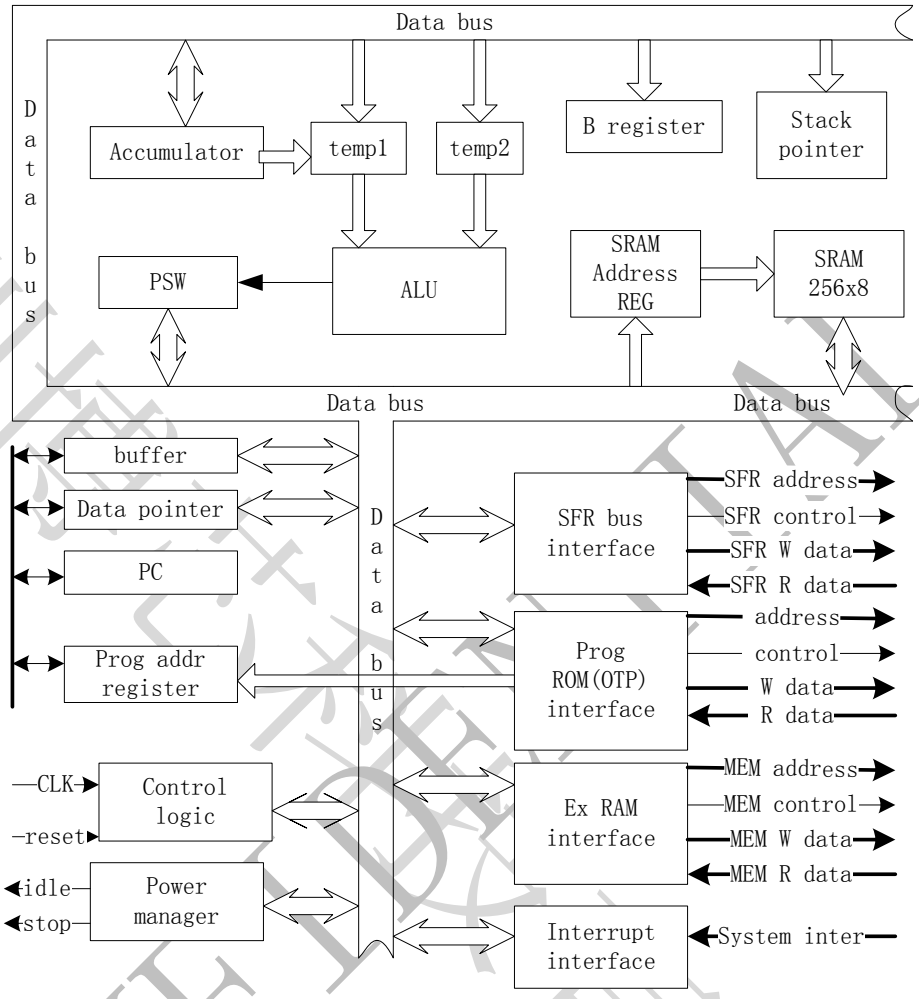


Figure 5 BK-51 architecture

Mnemonic	description	Byte	clock period
Arithmetic			
ADD A,Rn	Add register to A	1	1
ADD A,direct	Add direct byte to A	2	2
ADD A,@Ri	Add data memory to A	1	3
ADD A,#data	Add immediate to A	2	2
ADDC A,Rn	Add register to A with carry	1	1
ADDC A,direct	Add direct byte to A with carry	2	2
ADDC A,@Ri	Add data memory to A with carry	1	2
ADDC A,#data	Add immediate to A with carry	2	2
SUBB A,Rn	Subtract register from A with	1	1

	borrow		
SUBB A,direct	Subtract direct byte from A with borrow	2	2
SUBB A,@Ri	Subtract data memory from A with borrow	1	3
SUBB A,#data	Subtract immediate A with borrow	2	2
INC A	Increment A	1	1
INC Rn	Increment register	1	2
INC direct	Increment direct byte	2	3
INC @Ri	Increment data memory	1	3
DEC A	Decrement A	1	1
DEC Rn	Decrement register	1	2
DEC direct	Increment direct byte	2	3
DEC @Ri	Decrement data memory	1	3
INC DPTR	Increment data pointer	1	1
MUL AB	Multiply A by B	1	2
DIV AB	Divide A by B	1	8
DA A	Decimal adjust A	1	1
Logical			
ANL A,Rn	AND register to A	1	1
ANL A,direct	AND direct byte to A	2	2
ANL A,@Ri	AND data memory to A	1	2
ANL A,#data	AND immediate to A	2	2
ANL direct,A	AND A to direct byte	2	3
ANL direct,#data	AND immediate data to direct byte	3	4
ORL A,Rn	OR register to A	1	1
ORL A,direct	OR direct byte to A	2	2
ORL A,@Ri	OR data memory to A	1	2
ORL A,#data	OR immediate to A	2	2
ORL direct,A	OR A to direct byte	2	3
ORL direct,#data	OR immediate data to direct byte	3	4
XRL A,Rn	Exclusive-OR register to A	1	1
XRL A,direct	Exclusive-OR direct byte to A	2	2
XRL A,@Ri	Exclusive-OR data memory to A	1	2
XRL A,#data	Exclusive-OR immediate to A	2	2
XRL direct,A	Exclusive-OR A to direct byte	2	3
XRL direct,#data	Exclusive-OR immediate to	3	4

	direct byte		
CLR A	Clear A	1	1
CPL A	Complement A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through carry	1	1
SWAP A	Swap nibbles of A	1	1
Data transfer			
MOV A,Rn	Move register to A	1	1
MOV A,direct	Move direct byte to A	2	2
MOV A,@Ri	Move data memory to A	1	3
MOV A,#data	Move immediate to A	2	2
MOV Rn,A	Move A to register	1	2
MOV Rn,direct	Move direct byte to register	2	3
MOV Rn,#data	Move immediate to register	2	3
MOV direct,A	Move A to direct byte	2	3
MOV direct,Rn	Move register to direct byte	2	3
MOV direct,direct	Move direct byte to direct byte	3	4
MOV direct,@Ri	Move data memory to direct byte	2	3
MOV direct,#data	Move immediate to direct byte	3	4
MOV @Ri,A	MOV A to data memory	1	2
MOV @Ri,direct	Move direct byte to data memory	2	4
MOV @Ri,#data	Move immediate to data memory	2	4
MOV DPTR,#data16	Move 16 bits constant into DPTR	3	3
MOVC A,@A+DPTR	Move code byte relative DPTR to A	1	4
MOVC A,@A+PC	Move code byte relative PC to A	1	4
MOVX A,@Ri	Move external data (A8) to A	1	2
MOVX @Ri,A	Move A to external data (A8)	1	3
MOVX A,@DPTR	Move external data (A16) to A	1	1
MOVX @DPTR,A	Move A to external data (A16)	1	1
PUSH direct	Push direct byte onto stack	2	5
POP direct	Pop direct byte from stack	2	5
XCH A,Rn	Exchange A and register	1	2

XCH A,direct	Exchange A and direct byte	2	3
XCH A,@Ri	Exchange A and data memory	1	3
XCHD A,@Ri	Exchange A and data memory nibble	1	3
Boolean			
CLR C	Clear carry	1	1
CLR bit	Clear direct bit	2	3
SETB C	Set carry	1	1
SETB bit	Set direct bit	2	3
CPL C	Complement carry	1	1
CPL bit	Complement direct bit	2	3
ANL C,bit	AND direct bit to carry	2	2
ANL C,/bit	AND direct bit inverse to carry	2	2
ORL C,bit	OR direct bit to carry	2	2
ORL C,/bit	OR direct bit inverse to carry	2	2
MOV C,bit	Move direct bit to carry	2	2
MOV bit,C	Move carry to direct bit	2	3
JC rel	Jump on carry =1	2	3/4
JNC rel	Jump on carry =0	2	3/4
JB bit,rel	Jump on direct bit= 1	3	4/5
JNB bit,rel	Jump on direct bit= 0	3	4/5
JBC bit,rel	Jump on direct bit= 1 and clear	3	4/5
Branching			
ACALL addr11	Absolute call to subroutine	2	3
LCALL addr16	Long call to subroutine	3	4
RET	Return from subroutine	1	6
RETI	Return from interrupt	1	6
AJMP addr11	Absolute jump unconditional	2	3
LJMP addr16	Long jump unconditional	3	4
SJMP rel	Short jump (relative address)	2	4
JMP @A+DPTR	Jump indirect relative DPTR	1	4
JZ rel	Jump on accumulator= 0	2	3/4
JNZ rel	Jump on accumulator /= 0	2	3/4
CJNE A,direct,rel	Compare A, direct JNE relative	3	3/4
CJNE A,#data,rel	Compare A, immediate JNE relative	3	3/4
CJNE Rn,#data,rel	Compare reg, immediate JNE relative	3	3/4
CJNE @Ri,#data,rel	Compare ind, immediate JNE relative	3	5/6

DJNZ Rn,rel	Decrement register, JNZ relative	2	3/4
DJNZ direct,rel	Decrement direct byte, JNZ relative	3	4/5
NOP	No operation	1	1

80% instruction cycles in the BK-51 are 1~3 clock cycles in length, as opposed to the12 clock cycles per instruction cycle in the standard 8051. This translates to a more than 4X improvement in execution time for most instructions.

The architecture eliminates redundant bus states and implements parallel execution of fetch and execution phases. Most of the one-byte instructions are performed in one single cycle. The MCU uses 1 clock per cycle.

BK-51 can executes the normal 8051 instruction set, so you can use the standard 8051 tools to develop your application. All the instructions are equal to 8051's binary codes and function, includes operate code, addressing mode, affect on PSW. But the timing of instruction is different with 8051.

In many 8051 products, the machine period is different with clock period, one machine period last on 2~12 clock period, but BK-51 only has clock period, all the instructions are based on the clock period.

5.2 Memory organization

The memory organization of BK-51 is similar with 8051. It has two independent spaces: data memory and program memory. You can access them with different instructions.

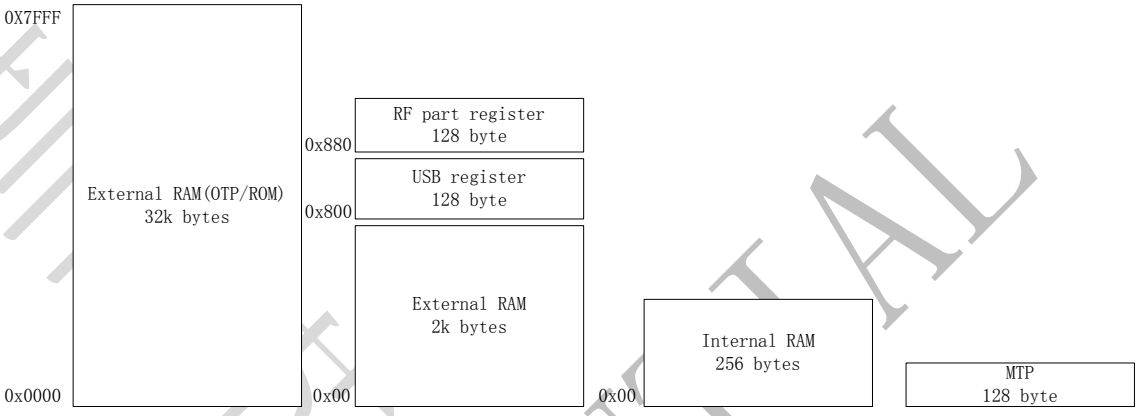


Figure 6 Memory Spaces

5.2.1 Program memory

32k bytes OTP (one time program) memory is used for storaging program. The OTP doesn't support read out for security to customer.

There are four GPIO pins are used for program OTP. P0.4, P0.5, P0.6 and P0.7 are used for this purpose in program mode. We will provide the download board and software to customer.

Also, RAM version will be provided to customer for development. You can download program into external flash firstly, then BK2433 will read the program into RAM from flash when power up.

5.2.2 Data memory

There are 2k bytes external memories and 256 bytes internal memory in BK-51 as showed in Figure 6.

The internal RAM (Figure 7) consists of 128 bytes of registers and scratchpad memory accessible through direct or indirect addressing (iram_addr addresses 00h–7Fh) Upper 128 bytes of scratchpad memory accessible through indirect addressing (iram_addr addresses 80h–FFh). 128 special function registers (SFRs) accessible through direct addressing (sfr_addr addresses 80h–FFh)

The SFRs and the upper 128bytes of RAM share the same address range (80h-FFh). However, the actual address space is separate and is differentiated by the type of addressing. Direct addressing accesses the SFRs on the sfr_bus, indirect addressing accesses the upper 128bytes of RAM on the iram_bus.

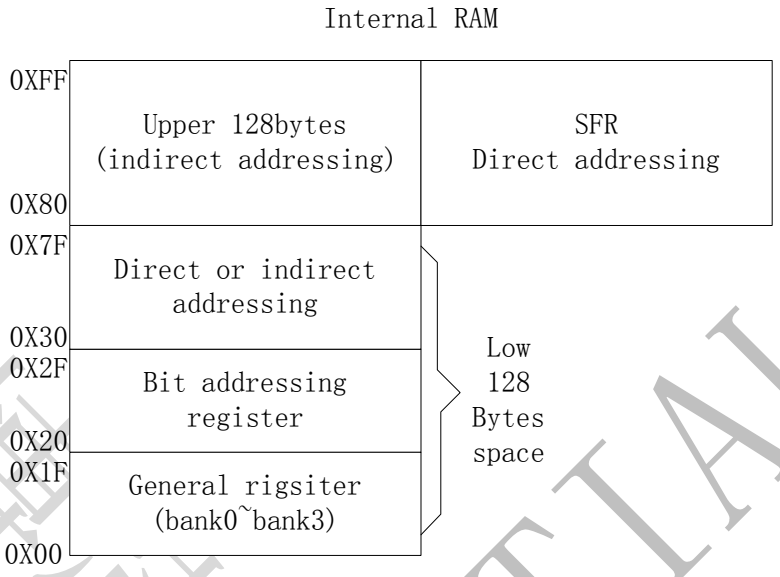


Figure 7 Internal memory spaces

2k bytes external ram are embedded in BK2433, they are mapped to the XRAM address from 0x0000 to 0x07FF. Also, any part of this XRAM can be configured as the buffer for USB endpoints. The configured USB space can be accessed by USB module or MCU part, and the hardware arbiter deal the conflict between them without customer operation.

5.2.3 General register

The lower 128 bytes are organized as shown in Figure 4. The lower 32bytes form four banks of eight registers (R0–R7). Two bits on the program status word (RS0（PSW.3）and RS1（PSW.4）) select which bank is in use. This allow fast switch when entering subroutine or interrupt program. R0 and R1 are used for indirect addressing register.

5.2.4 Bit address space

The next 16bytes form a block of bit-addressable memory space at bit addresses 00h–7Fh. All of the bytes in the lower 128 bytes are accessible through direct or indirect addressing on the iambus. Every bit has a bit address from 0x00 to 0x7F. For example, the bit 0 of address 0x20 has a bit address 0x00, the bit 7 of address 0x20 has a bit address 0x07, the bit 7 of address 0x2F has a bit address 0x7F. The type of instruction decides the addressing mode (bit addressing or byte addressing).

MCS-51™ use XX.B to represent bit address, XX means byte address，and B means bit address. For example:

MOV C， 23h . 0

Move the Boolean variable in 0x23 to carry flag. (The bit 0 of 0x23h)

5.2.5 Stack

The stack for program can locate any position of the 256 bytes memory. Stack pointer is used to point the stack area, SP point to the last position. The next stack data is placed at SP+1, and then SP increased by 1. The initial stack pointer address is 0x07, so the first stack data is placed at 0x08, which is the first register R0. The max depth of the stack can be 256 bytes.

5.2.6 SFR

REGISTER	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	addr	reset value
P0	-	-	-	-	-	-	-	-	0x80	0xff
SP	-	-	-	-	-	-	-	-	0x81	
DPL0	-	-	-	-	-	-	-	-	0x82	
DPH0	-	-	-	-	-	-	-	-	0x83	
DPL1	-	-	-	-	-	-	-	-	0x84	
DPH1	-	-	-	-	-	-	-	-	0x85	
DPS								SEL	0x86	
PCON	SMOD	EUSB	/	/	/	CMD_RST	STOP	IDLE	0x87	0x02
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0x88	
TMOD	GATE	C/T	M1	M0	GATE	C/T	M1	M0	0x89	
TL0	-	-	-	-	-	-	-	-	0x8A	
TL1	-	-	-	-	-	-	-	-	0x8B	
TH0	-	-	-	-	-	-	-	-	0x8C	
TH1	-	-	-	-	-	-	-	-	0x8D	
CKCON	CKDIV1	CKDIV0	T2M	T1M	T0M	/	/	/	0x8E	0
CLK_EN_CFG	adc_en	timer_en	uart_en	pwm_en	spi_en	i2c_en	des_en	mdu_en	0x8F	0
P1	-	-	-	-	-	-	-	-	0x90	0xFF
EXIF	IE9	IE8	IE7	IE6	IE5	IE4	IE3	IE2	0x91	0x08
MPAGE	-	-	-	-	-	-	-	-	0x92	
EICON	/	/	EPFI	PFI	/	/	/	/	0x93	0
ADC_DATL	-	-	-	-	-	-	-	-	0x94	0
ADC_DATH	setting	sample_rate[4:0]					adc_data[9:8]		0x95	0
ADC_CTL	mode[1:0]		channel[2:0]			Intr_En	Chann_en	Ready	0x96	0
WDT	/	/	EN	CLR	IDLE	PS2	PS1	PS0	0x97	0
SCON0	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	0x98	0
SBUF0	-	-	-	-	-	-	-	-	0x99	0
P0_PU									0x9A	0
P1_PU									0x9B	0
P2_PU									0x9C	0
P3_PU									0x9D	0
P4_PU									0x9E	0
P2	-	-	-	-	-	-	-	-	0xA0	0xFF
PWM0C0	PWM0_PERIOD [7:0]								0xA1	0
PWM0C1	PWM0_DUTY [5:0]						PWM0_PERIOD [9:8]		0xA2	0

PWM0C2	PWM0_EN	PWM0_PRESCAL[2:0]			PWM0_DUTY [9:6]				0xA3	0
PWM1C0	PWM1_PERIOD [7:0]								0xA4	0
PWM1C1	PWM1_DUTY [5:0]						PWM1_PERIOD [9:8]		0xA5	0
PWM1C2	PWM1_EN	PWM1_PRESCAL[2:0]			PWM1_DUTY [9:6]				0xA6	0
PWMICTL	pwm1_inte	pwm0_inte	pwm1_mod	pwm0_mod	/	/	pwm1_intf	pwm0_intf	0xA7	0
IE	EA		ET2	ES0	ET1	EX1	ET0	EX0	0xA8	0
IP		PPFI	PT2	PS0	PT1	PX1	PT0	PX0	0xA9	0x80
P0_IOSEL	-	-	-	-	-	-	-	-	0xAA	0
P1_IOSEL	-	-	-	-	-	-	-	-	0xAB	0
P2_IOSEL	-	-	-	-	-	-	-	-	0xAC	0
P3_IOSEL	-	-	-	-	-	-	-	-	0xAD	0
P4_IOSEL									0xAE	0
P3	-	-	-	-	-	-	-	-	0xB0	0xFF
P0_PD	-	-	-	-	-	-	-	-	0xB1	0
P1_PD	-	-	-	-	-	-	-	-	0xB2	0
P2_PD	-	-	-	-	-	-	-	-	0xB3	0
P3_PD	-	-	-	-	-	-	-	-	0xB4	0
P4_PD	-	-	-	-	-	-	-	-	0xB5	0
SMB_DAT	-	-	-	-	-	-	-	-	0xB6	0
SMB_CF	EN SMB	INH	BUSY	EXT HOLD	SMB TOE	SMB FTE	SMB CS1	SMB CS0	0xB7	0
SMB_CN	MASTER	TXMODE	STA	STO	ACK RQ	ARB LOST	ACK	SI	0xB8	0
SMB_TMCTL	SCL_CR			IDLE_CR			-	SCL_TMOT	0XB9	0xEC
P0_OPDR	-	-	-	-	-	-	-	-	0xBA	0xFF
P1_OPDR	-	-	-	-	-	-	-	-	0xBB	0xFF
P2_OPDR	-	-	-	-	-	-	-	-	0xBC	0xFF
P3_OPDR	-	-	-	-	-	-	-	-	0xBD	0xFF
P4_OPDR	-	-	-	-	-	-	-	-	0xBE	0xFF
P4	-	-	-	-	-	-	-	-	0xC0	0xFF
DS_WUEN									0xC3	
P0_WUEN									0xC4	0
P1_WUEN									0xC5	0
P2_WUEN									0xC6	0
P3_WUEN									0xC7	0
T2CON	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2	0xC8	0
P4_WUEN									0xC9	0
RCAP2L	-	-	-	-	-	-	-	-	0xCA	0
RCAP2H	-	-	-	-	-	-	-	-	0xCB	0
TL2	-	-	-	-	-	-	-	-	0xCC	0
TH2	-	-	-	-	-	-	-	-	0xCD	0

PSW	CY	AC	F0	RS1	RS0	OV	F1	P	0xD0	0
DES_CTL	DES_EN	DES_TDES	DEC_ENC	MAC	/	ADDR_RST	OUT_RST	DES_RST	0xD1	0
DES_INT	-	-	-	-	-	-	-	DES_INTF	0xD2	0
DES_KEY1									0xD3	0
DES_KEY2									0xD4	0
DES_KEY3									0xD5	0
DES_IN									0xD6	0
DES_OUT									0xD7	0
ACC	-	-	-	-	-	-	-	-	0xE0	0
									0XE6	
PALT	S_S0	S_T0	S_T1	S_T2	/	/	/	/	0xE7	0
EIE	EX9	EX8	EX7	EX6	EX5	EX4	EX3	EX2	0xE8	0xF0
EIP	PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2	0xE9	0xF0
EXINT_MOD	IT9	IT8	IT7	IT6	IT5	IT4	IT3	IT2	0xEA	0xFF
RNG_DAT									0xEB	
RNG_CTL	RNG_PWD		RNG_RDY						0xEC	
MTP_CTL									0xED	
MTP_ADR									0xEE	
MTP_DAT									0xEF	
B	-	-	-	-	-	-	-	-	0xF0	0
MD0	-	-	-	-	-	-	-	-	0xF1	0
MD1	-	-	-	-	-	-	-	-	0xF2	0
MD2	-	-	-	-	-	-	-	-	0xF3	0
MD3	-	-	-	-	-	-	-	-	0xF4	0
MD4	-	-	-	-	-	-	-	-	0xF5	0
MD5	-	-	-	-	-	-	-	-	0xF6	0
MDCTL									0xF7	
SPI_CN	SPIF	WCOL	MODF	RX OVRN	NSS MD1	NSS MD0	TX BMT	SPIEN	0xF8	
SPI_CFG	SPI BSY	MST EN	CK PHA	CK POL	SLV SEL	NSSIN	WCOL EN	RXB MT	0xF9	0x01
SPI_CKR	-	-	-	-	-	-	-	-	0xFA	0
SPI_DAT	-	-	-	-	-	-	-	-	0xFB	0

Table 3 SFR Register

5.3 Power management

PCON	7	6	5	4	3	2	1	0
0x87	SMOD	EUSB	CMD_RST	Latch_en	Deep_sleep	OSC32K_sel	RC32k_sel	IDLE

Table 4 power management register

SMOD: SMOD0 – Serial Port 0 baud rate doublers enable. When SMOD0=1, the baud rate for Serial Port 0 is doubled.

EUSB: R/W by software only. USB enable, the 48MHz clock will exist when EUSB=1.

CMD_RST: Write 1 to reset MCU (not include RF part).

RC32k_sel: System will select RC32k clock when write 1 to this position. Interrupts and software can clear it.（when reset, this bit will be reset to1and system enter slow clock mode）. In this state, the system clock changed to 32K RC clock, so the power consumption is very low.

OSC32k_sel: System will select OSC32k clock when write 1 to this position. Interrupts and software can clear it. In this state, the system clock changed to 32K OSC clock, so the power consumption will decrease evidently. Pls note that OSC32k clock is more accurate than RC 32k clock, but need more power consumption.

IDLE: When set by software, system enter stop mode, and it can only be wake up by enabled interrupt.(Clear it to 0). In this state, the most of clocks are shut down for power saving.

Note: set RC32k_sel and IDLE bit simultaneously can get the lowest power consumption, and in this state, all the register settings are retained.

CLK_EN_CFG	7	6	5	4	3	2	1	0
0x8F	**adc_en**	**timer_en**	**uart_en**	**pwm_en**	**spi_en**	**i2c_en**	**des_en**	**mdu_en**

CLK_EN_CFG: this register can use to power on or off all the peripheral equipment clocks for saving power.

adc_en : ADC clock enable or not (1 enable)
timer_en : TIMER 0 /1 /2 clock enable or not (1 enable)
uart_en : UART clock enable or not (1 enable)
pwm_en : PWM clock enable or not (1 enable)
spi_en : SPI clock enable or not (1 enable)
i2c_en : I2C clock enable or not (1 enable)
des_en : DES clock enable or not (1 enable)
mdu_en : MDU clock enable or not (1 enable)

5.3.1 Work State

5.3.1.1 IDLE MODE

An instruction that sets the IDLE bit (PCON.0) causes the BK-51 to enter idle mode when that instruction completes. In idle mode, CPU processing is suspended, internal registers maintain their current data, and the idle_mode_n output is activated. However, unlike the standard 8051, the clock is not disabled internally.

Activation of any enabled interrupt causes the hardware to clear the IDLE bit and

terminate idle mode.

In idle mode, the power consumption is decreased evidently.

5.3.1.2 SLEEP MODE

An instruction that sets the STOP bit (PCON.1) causes the BK-51 to enter stop mode when that instruction completes. In sleep mode, the system clock change from 16MHz to 32KHz, and the clock 48MHz used by USB stopped. The power consumption decreases to a lower level.

If write 1 to PCON.1 at this time, the system will enter deep sleep mode, and the power consumption of MCU will decreased evidently.

After reset, the system will enter sleep mode immediately.

Note: You should always clear PCON.6 to 0 to save current when USB module doesn't need work at any time.

As showed above, the IDLE bit decide CPU run or not, the SLEEP bit decide the system clock source. (16MHz or 32KHz)

5.3.1.3 DEEP SLEEP MODE

If you want to get the lowest power consumption, you can let BK2433 enter deep sleep mode. Firstly, you should set the 8 GPIO's setting, then, set latch_en (pcon[4]) to latch the register setting, lastly, set deep sleep (pcon[3]) to enter deep sleep status. In this state, the power supplied to digital would be shut down.

5.3.2 Wake Up

5.3.2.1 Wake Up from sleep moe

S_WU EN	port	7	6	5	4	3	2	1	0
0xc4	0	En/dis							
0xc5	1								
0xc6	2								
0xc7	3								
0xc9	4								

When the MCU entered IDLE/SLEEP mode, all the 40 ports pin can be used to wake up the MCU separately. Configure the corresponding SFR bit can enable or disenable the wake up function.

The process wake up from sleep mode is showed in next figure.

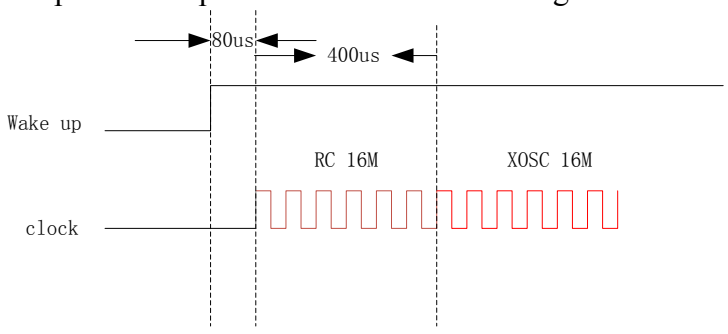


Figure 8 wake up process

After wake up, RC 16M clock will spend 80us to wake up, and after 400us, the clock source will switch to XOSC 16M automatically. RC 16M clock is not very accurate, so, during this period, you can only run ordinary MCU instruction, but can not send or receive RF package.

Pls note that: The RF part will also resume 120us for PLL locking after power up RF part. So, if you want to send/receive package through RF, you should wait 600us after wake up from sleep mode.

5.3.2.2 Wake up from deep sleep mode

DS_WUEN	7	6	5	4	3	2	1	0
0xc4	En/dis	En/dis	En/dis	En/dis	En/dis	En/dis	En/dis	En/dis
port	P3.2	P3.1	P2.4	P2.3	P2.1	P2.0	P1.1	P1.0

There are eight port can be set to wake up MCU from deep sleep status, also, you can enable or disable them separately. After wake up from deep sleep, a POR will be generated to reset the whole system.

5.4 Interrupt

BK-51 has an advanced interrupt controller with 15 sources. Software can set any interrupt source through write 1 to the corresponding position. When an enabled interrupt occurs, the MCU vectors to the address of the interrupt service routine (ISR) associated with that interrupt. The MCU executes the ISR to completion unless another interrupt of higher priority occurs.

The interrupt source, interrupt address, interrupt priority and control bit are showed in next table. Please refer to the relevant chapter to get the detail interrupt condition and interrupt status of external interrupt.

5.4.1 Interrupt priority

Interrupt name	description	Flag bit	Enable bit	Priority control	Interrupt vector	Interrupt number
int_0n	External interrupt 0	TCON.1	IE.0	IP.0	03h	0
TF0	Timer 0 interrupt	TCON.5	IE.1	IP.1	0bh	1
int1_n	External interrupt 1	TCON.3	IE.2	IP.2	13h	2
TF1	Timer 1 interrupt	TCON.7	IE.3	IP.3	1bh	3
TI or RI	Serial port interrupt	SCON.0(RI) SCON.1(TI)	IE.4	IP.4	23h	4

TF2 or EXF2	Timer 2 interrupt	T2CON.7(TF2) T2CON.6(EXF2)	IE.5	IP.5	2bh	5
PFI	Votage too low interrupt	EICON[4]	EICON[5]	IP.6	33h	6
int2	SPI0 interrupt	EXIF[0]	EIE.0	EIP[0]	43h	8
int3_n	I2C interrupt	EXIF[1]	EIE.1	EIP[1]	4bh	9
int4	USB interrupt	EXIF[2]	EIE.2	EIP[2]	53h	10
int5_n	RF interrupt	EXIF[3]	EIE.3	EIP[3]	5bh	11
int6	Not available	EXIF[4]	EIE.4	EIP[4]	63h	12
int7_n	PWM interrupt	EXIF[5]	EIE.5	EIP[5]	6bh	13
int8	ADC interrupt	EXIF[6]	EIE.6	EIP[6]	73h	14
int9_n	DES interrupt	EXIF[7]	EIE.7	EIP[7]	7bh	15

Table 5 interrupt priority

You can assign priority level (high or low) to each interrupt as listed in Table 5. Simultaneous interrupts with the same priority level (for example, both high) are resolved according to their arrived time.

Once an interrupt is being serviced, only an interrupt of higher priority level can interrupt the service routine of the interrupt currently being serviced.

5.4.2 Interrupt sampling

The int0_n and int1_n signals are both active low and can be programmed to be either edge-sensitive or level-sensitive, through the IT0 and IT1 bits in the TCON SFR. For example, when IT0= 0, int0_n is level-sensitive and the BK51 sets the IE0 flag when the int0_n pin is sampled low. When IT0 = 1, int0_n is edge-sensitive and the BK51 sets the IE0 flag when the int0_n pin is sampled high then low on consecutive samples.

TF0, TF1 and TF2 are the interrupt used for timer. TI (or RI) is used for UART. PFI used for low battery detector circuit.

The remaining external interrupts can be configured as edge-sensitive interrupt or level-sensitive interrupt; int2, int4, int6 and int8 are active high, int3_n, int5_n, int7_n and int9_n are active low.

Note: External interrupts 2, 3, 4, 5, 6, 7, 8, 9 had been used by BK2433 function module, so customer can't use them again. Customer can configure their trigger mode only.

EXINT_MOD	7	6	5	4	3	2	1	0
0xEA	IT9	IT8	IT7	IT6	IT5	IT4	IT3	IT2

Table 6 interrupt trigger mode

IT9 : when IT9 = 1, int9_n(DES) edge sensitive; when IT9 = 0，level sensitive.
IT8 : when IT8 = 1, int8(ADC) edge sensitive; when IT8 = 0，level sensitive.
IT7 : when IT7 = 1, int7_n(PWM) edge sensitive; when IT7 = 0，level sensitive.
IT6 : not available now.
IT5 : when IT5 = 1, int5_n(RF) edge sensitive; when IT5 = 0，level sensitive.
IT4 : when IT4 = 1, int4(USB1.1) edge sensitive; when IT4 = 0，level sensitive.
IT3 : when IT3 = 1, int3_n(I2C) edge sensitive; when IT3 = 0，level sensitive.
IT2 : when IT2 = 1, int2(SPI) edge sensitive; when IT2 = 0，level sensitive.

5.4.3 Clear interrupt

When interrupt happens, the interrupt flag of Timer0, timer1, int0 and int1 would be cleared by hardware automatically after the CPU vectors to the address of the interrupt service routine (ISR) associated with that interrupt. The other interrupt must be cleared by customer, or the CPU will respond to the interrupt always.

5.4.4 Register relevant to interrupt

The following SFRs are associated with interrupt control:
IE – SFR A8h
IP – SFR A9h
EXIF – SFR 91h
EICON – SFR 93h
EIE – SFR E8h
EIP – SFR E9h

The IE and IP SFRs provide interrupt enable and priority control for the standard interrupt unit, as with the standard 8051.

The EXIF, EICON, EIE, and EIP registers provide flags, enable control, and priority control for the optional extended interrupt unit.

IE register

bit	definition
IE.7	EA – Global interrupt enable. Controls masking of all interrupts except power-fail interrupt (pfi). EA=0 disables all interrupts (EA overrides individual interrupt enable bits). When EA=1, each interrupt is enabled or masked by its individual enable bit.
IE.6	/
IE.5	ET2 – Enable Timer 2 interrupt.

	ET2=0 disables Timer 2 interrupt (TF2). ET2=1 enables interrupts generated by the TF2 or EXF2 flag. If Timer 2 is not implemented (timer2=0), ET2 is present but not used.
IE.4	ES0 – Enable Serial Port interrupt. ES0=0 disables Serial Port interrupts (TI and RI). ES0=1 enables interrupts generated by the TI or RI flag.
IE.3	ET1 – Enable Timer1 interrupt. ET1=0 disables Timer1 interrupt (TF1). ET1=1 enables interrupts generated by the TF1 flag.
IE.2	EX1 – Enable external interrupt 1. EX1=0 disables external interrupt 1 (int1_n). EX1=1 enables interrupts generated by the int1_n pin.
IE.1	ET0 – Enable Timer 0 interrupt. ET0=0 disables Timer 0 interrupt (TF0). ET0=1 enables interrupts generated by the TF0 flag.
IE.0	EX0 – Enable external interrupt 0. EX0=0 disables external interrupt 0 (int0_n). EX0=1 enables interrupts generated by the int0_n pin.

Table 7 IE register

IP register

bit	definition
IP.7	/
IP.6	PPFI： voltage too low interrupt priority control 0： PFI low priority 1： PFI high priority
IP.5	PT2 – Timer 2 interrupt priority control. PT2= 0 sets Timer 2 interrupt (TF2) to low priority. PT2=1 sets Timer 2 interrupt to high priority. If Timer 2 is not implemented (timer2= 0), PT2 is present but not used.
IP.4	PS0 – Serial Port interrupt priority control. PS0= 0 sets Serial Port interrupt (TI or RI) to low priority. PS0=1 sets Serial Port interrupt to high priority.
IP.3	PT1 – Timer1 interrupt priority control. PT1= 0 sets Timer1 interrupt (TF1) to low priority. PT1=1 sets Timer1 interrupt to high priority.
IP.2	PX1 – External interrupt 1 priority control. PX1= 0 sets external interrupt 1 (int1_n) to low priority. PT1= 1 sets external interrupt 1 to high priority.
IP.1	PT0 – Timer 0 interrupt priority control. PT0= 0 sets Timer 0 interrupt (TF0) to low priority. PT0=1 sets Timer 0 interrupt to high priority.

IP.0	PX0 – External interrupt 0 priority control. PX0= 0 sets external interrupt 0 (int0_n) to low priority. PT0= 1 sets external interrupt 0 to high priority.

Table 8 IP register

EXIF register

bit	definition
EXIF.7	IE9 – External interrupt 9 flag. IE9=1 indicates that an interrupt was detected at the int9_n pin. IE9 must be cleared by software. Setting IE9 in software generates an interrupt, if enabled.
EXIF.6	IE8 – External interrupt 8 flag. IE8=1 indicates that an interrupt was detected at the int8 pin. IE8 must be cleared by software. Setting IE8 in software generates an interrupt, if enabled.
EXIF.5	IE7– External interrupt 7 flag. IE7=1 indicates that a interrupt was detected at the int7_n pin. IE7 must be cleared by software. Setting IE7 in software generates an interrupt, if enabled.
EXIF.4	IE6 – External interrupt 6 flag. IE6=1 indicates that an interrupt was detected at the int6 pin. IE6 must be cleared by software. Setting IE6 in software generates an interrupt, if enabled.
EXIF.3	IE5– External interrupt 5 flag. IE5=1 indicates that an interrupt was detected at the int5_n pin. IE5 must be cleared by software. Setting IE5 in software generates an interrupt, if enabled.
EXIF.2	IE4 – External interrupt 4 flag. IE4=1 indicates that an interrupt was detected at the int4 pin. IE4 must be cleared by software. Setting IE4 in software generates an interrupt, if enabled.
EXIF.1	IE3– External interrupt 3 flag. IE3=1 indicates that an interrupt was detected at the int3_n pin. IE3 must be cleared by software. Setting IE3 in software generates an interrupt, if enabled.
EXIF.0	IE2 – External interrupt 2 flag. IE2=1 indicates that an interrupt was detected at the int2 pin. IE2 must be cleared by software. Setting IE2 in software generates an interrupt, if enabled.

Table 9 EXIF register

EICON （0x93）

bit	definition
EICON.7	/
EICON.6	/
EICON.5	EPFI – Enable power-fail interrupt. EPFI= 0 disables power-fail interrupt (pfi). EPFI= 1 enables interrupts generated by LBD.
EICON.4	PFI – Power-fail interrupt flag(voltage too low). PFI =1 indicates a power-fail interrupt was detected at the pfi pin. PFI must

	be cleared by software before exiting the interrupt service routine. Otherwise, the interrupt occurs again. Setting PFI in software generates a power-fail interrupt, if enabled.
EICON.3	/
EICON.2	/
EICON.1	/
EICON.0	/

Table 10 EICONregister

EIE register（0xE8）

bit	definition
EIE.7	EX9： Enable external interrupt 9 EX9 = 0 disables external interrupt 9 (int9_n) EX9 = 1 enable interrupts generated by 9(int9_n)
EIE.6	EX8： Enable external interrupt 8 EX8 = 0 disables external interrupt 8 (int8) EX8 = 1 enable interrupts generated by 8 (int8)
EIE.5	EX7： Enable external interrupt 7 EX7 = 0 disables external interrupt 7 (int7_n) EX7 = 1 enable interrupts generated by 7 (int7_n)
EIE.4	EX6： Enable external interrupt 6 EX6 = 0 disables external interrupt 6 (int6) EX6 = 1 enable interrupts generated by 6 (int6)
EIE.3	EX5： Enable external interrupt 5 EX5 = 0 disables external interrupt 5 (int5_n) EX5 = 1 enable interrupts generated by 5 (int5_n)
EIE.2	EX4： Enable external interrupt 4 EX4 = 0 disables external interrupt 4 (int4) EX4 = 1 enable interrupts generated by 4 (int4)
EIE.1	EX3： Enable external interrupt 3 EX3 = 0 disables external interrupt 3 (int3_n) EX3 = 1 enable interrupts generated by 3 (int3_n)
EIE.0	EX2： Enable external interrupt 2 EX2 = 0 disables external interrupt 2 (int2) EX2 = 1 enable interrupts generated by 2 (int2)

Table 11 EIE register

EIP register（0xe9）

bit	definition
EIP.7	PX9： External interrupt 9 priority control. PX9= 0 sets external interrupt 9 to low priority(int9_n) PX9 = 1 sets external interrupt 9 to high priority (int9_n)

EIP.6	PX8： External interrupt 8 priority control. PX8 = 0 sets external interrupt 8 to low priority (int8) PX8 = 1 sets external interrupt 8 to high priority (int8)
EIP.5	PX7： External interrupt 7 priority control. PX7= 0 sets external interrupt 7 to low priority(int7_n) PX7 = 1 sets external interrupt 7 to high priority (int7_n)
EIP.4	PX6： External interrupt 6 priority control. PX6 = 0 sets external interrupt 6 to low priority (int6) PX6 = 1 sets external interrupt 6 to high priority (int6)
EIP.3	PX5： External interrupt 5 priority control. PX5= 0 sets external interrupt 5 to low priority(int5_n) PX5 = 1 sets external interrupt 5 to low priority (int5_n)
EIP.2	PX4： External interrupt 4 priority control. PX4 = 0 sets external interrupt 4 to low priority (int4) PX4 = 1 sets external interrupt 4 to high priority (int4)
EIP.1	PX3： External interrupt 3 priority control. PX3= 0 sets external interrupt 3 to low priority(int3_n) PX3 = 1 sets external interrupt 3 to low priority (int3_n)
EIP.0	PX2： External interrupt 2 priority control. PX2 = 0 sets external interrupt 2 to low priority (int2) PX2 = 1 sets external interrupt 2 to high priority (int2)

Table 12 EIP register

6 Reset system

There are three active low reset source in BK2433, they are power on reset, reset pin, watch dog reset. After reset, the MCU will re-start from address 0.

7 Clock system

There are three clock domains in BK2433, clock for MCU, clock for USB, and clock for RF part.

48MHz clock is used by USB, and you can shut down USB clock for power saving by setting SFR register USB_PWR_CN.1.

The clock frequency for RF module is always 16MHz.

There are two sources for MCU clock, one is XOSC16M, and another is RCOSC32KHz.

32KHz is used for sleep mode. There are two 32k source in the chip, one is divided from 16M, and another is generated by RC circuit. The previos clock source is more accurate than the RC 32k clock, but need more power.

16MHz is used for normal work mode. In normal work mode, the clock for MCU is divided by 16MHz clock source, it can be set by setting register CKCON[7:6]. (SFR 0X8E)

00： 16Mhz

01： 8Mhz

10： 4Mhz

11： 2Mhz

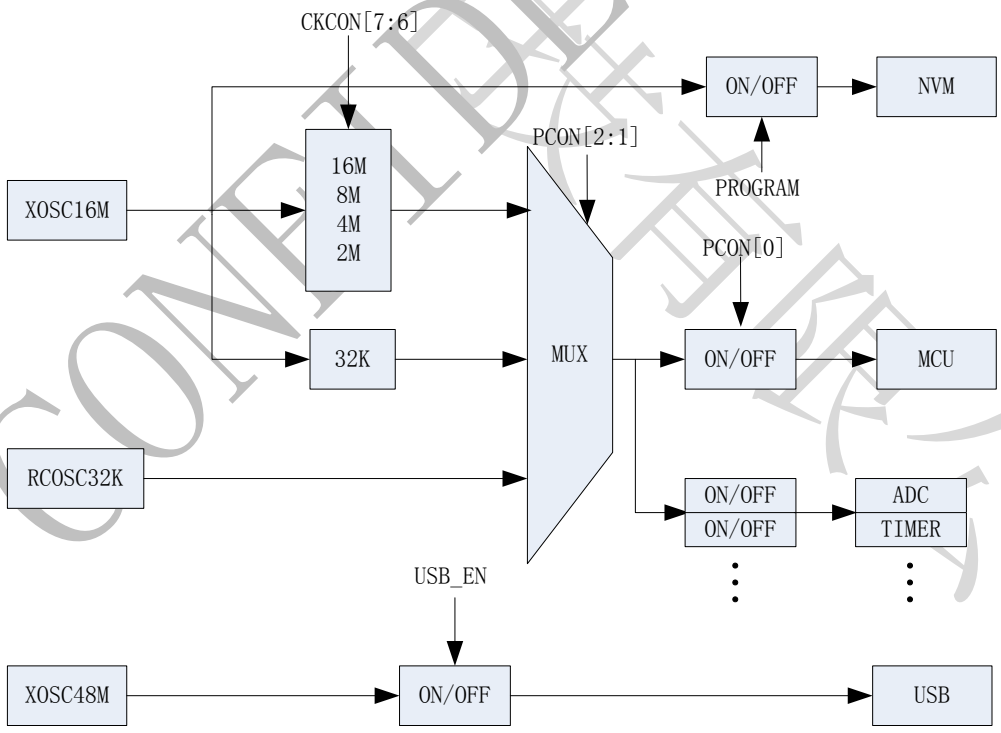


Figure 9 Clock System

8 MTP Memory

128bitsX8 MTP(multi-time program) macro is embedded in BK2433. MTP space can be accessed by MCUalso. It can be read out, writen and erased seperately.

There are 128 bytes in the MTP, and it divided to 64 sectors. Every two address comprise one sector. When sector erase, address n and address n+1 point to the same sector. (n is even)

addr	register	7	6	5	4	3	2	1	0
0xED	MTP_CTL	write	read	Sector_erase	Mase_erase				
0xEE	MTP_ADR	Addr							
0xEF	MTP_DAT	data							

MTP_CTL

Bit7: 1: need write now;

Bit6: 1: need read now;

Bit5: 1: need erase sector now;

Bit4: 1: need mass erase now;

When the operation is finished, all the bits in MTP_CTL (register 0xed) will clear to zeros.

Note1: only one operation can run at the same time.

For example: read data from MTP address 0x20

First step: write 0X20 to SFR address 0xEE;

Second step: write 1 to SFR address 0xED[7];

Wait until 0xED[7] changed to 0;

Finally, read the data from SFR address 0xEF.

Note2: For better reliability, we recommend the following write steps:

First step: Write all zeros into the space you want to write

Second step: Sector erase the address you want to write (address n and n+1 will be erased simutaneously)

Third step: write the data into the address you want to write

9 General I/O

9.1 Basic function

There are four general ports P0, P1, P2, and P3 in BK2433. All the four port can be used for general I/O with selectable direction for each bit, or these lines can be used for specialized functions.

When the four ports are configured as general I/O, the detail function of them can be set by twenty SFR register described next.

Note: the default setting for all the ports is general I/O.

SFR Pn (n=0-3): general IO data register

SFR Pn_IOSEL (n=0-3): input/output select register

1： INPUT

0：OUPUT

Note: this bit is valid only when the port is configured as general IO port

SFR Pn_OPDR (n=0-3); open_drain select register

1：open-drain，

0：Not open-drain

Note: this bit is valid only when the port is configured as general IO port

SFR Pn_PU (n=0-3); pull up enable or not

1: pull up enable

0: pull up disable

SFR Pn_PD (n=0-3): pull down enable or not

1: pull down enable

0: pull down disable

SFR Pn_WUEN (n=0-3); wake up enable register

1：wake up enable，

0：wake up disable

Note: this bit is valid only when the port is configured as general IO port

NOTE: Another eight special GPIO ports (P4) are contained in BK2433. P4 have the basic GPIO function, so you can acquire 40 GPIO totally.

9.2 Second function of port

9.2.1 Sencond function

When one port is configured as second function, the direction (input or output) is decided by the second function and is irrespective of Pn_IOSEL.

Port0	7	6	5	4	3	2	1	0
Second function	NSS	SCK	MISO	MOSI	T2EX	t2	t1	t0

Table 13 PORT0 Second Function

T0 : Timer0 external input (when (EA &ET0) || PALT.6 ==1)
T1 : Timer1 external input (when (EA &ET1) || PALT.5 ==1)
T2 : Timer2 external input (when (EA &ET2) || PALT.4 ==1)
T2EX: timer2 external capture/reload trigger input (when (EA &ET2) || PALT.4 ==1)

MOSI: The output of SPI host and the input of SPI slave (when SPI_CN.0==1)
MISO: The input of SPI host and the output of SPI slave (when SPI_CN.0==1)
SCK: The clock for SPI, it is output when configured as host and it is input when configured as slave.
NSS: It is valid only when SPI is configured as 4-wire mode. It is chip select signal located at P0.7. It is output when configured as host and it is input when configured as slave. (when SPI_CN.0==1)

Port1	7	6	5	4	3	2	1	0
Second function	/	/	/	/	/	/	int1_n	int0_n

Table 14 PORT1 Second Function

Int0_n : external interrupt 0 input pin, active low (when EA&IE0 == 1)
Int1_n : external interrupt 1 input pin, active low (when EA&IE1 == 1)

Port2	7	6	5	4	3	2	1	0
Second function	PWM 1	PWM0	/	SDA	SCL	rxd_out	txd	rxd_in

Table 15 PORT2 Second Function

rxd_in : the input for UART (when (EA & ES0) || PALT.7 ==1)
txd : the output for UART (when (EA & ES0) || PALT.7 ==1)
rxd_out : the output pin pin for received data, only used for mode0 (when(EA & ES0) || PALT.7 ==1)
SCL : clock pin for SMBUS (I2C). When SFR SMB_CF.ENSMB=1, this pin is used as SCL, or it is general IO. (When SMB_CF.7==1)
SDA: the data line for SMBUS (I2C). When SFR SMB_CF.ENSMB=1, this pin is used as SDA, or it is general IO. (When SMB_CF.7==1)
PWM0: When PWM0 is enabled, this pin used as the output of PWM0, or it is used as general IO.当PWM0 (When PWM0C2.7==1)
PWM1: When PWM1 is enabled, this pin used as the output of PWM1, or it is used as general IO.当PWM0 (When PWM1C2.7==1)

Port3	7	6	5	4	3	2	1	0

Second function	ADC_7	ADC_6	ADC_5	ADC_4	ADC_3	ADC_2	ADC_1	ADC_0

Table 16 PORT3 Second Function

ADC_n：When ADC is enabled by setting channel_en register, the corresponding port n of port3 is configured as the input for ADC. Other seven port can used general IO independently.

The second function of port is enabled when the corresponding interrupt is enabled or the corresponding bit of register PALT is valid. (When the port is used as second function, IOSEL is invalid.)

For example:

When SETB EA

SETB ET0

The PORT0.0 is configured as the input timer0. Also, you can set PALT |= 0x40 to get the same result.

Note: EA and ET0 should be valid at the same time, or the corresponding port is general IO still.

9.2.2 GPIO for 32-pin package

There are 18 (OR 19 GPIO (no LBD)) GPIO provided for 32-pin package. The next table shows the GPIO port and their second or extended functions.

port	No.	Basic Function1	Second Function	Program mode (when Program=1)	Wake up From deep sleep
P0.0	1	GPIO	Timer0	spi_cs =p0.7	
P0.1	2	GPIO	Timer1	spi_sck=p0.6	
P0.4	3	GPIO	SPI	spi_miso=p0.5	
P0.5	4	GPIO		spi_mosi=p0.4	
P0.6	5	GPIO			
P0.7	6	GPIO			
P1.0	7	GPIO	INT		y
P1.1	8	GPIO			y
P2.0	10	GPIO	UART		y
P2.1	11	GPIO			y
P2.3	12	GPIO	I2C		y
P2.4	13	GPIO			y
P2.7	13	GPIO	PWM		
P3.1	14	GPIO	ADC		y
P3.2	15	GPIO	ADC		y
P3.5	16	GPIO	ADC		
P3.6	17	GPIO	ADC		
P3.7	18	GPIO	ADC		

Table 17 GPIO for 32-pin package

10 Watch Dog

WDCON	7	6	5	4	3	2	1	0
	/	/	En_wdt	Clr_wdt	Idle_wdt	ps2	ps1	ps0

Table 18 Watch Dog Register

En_wdt: enable watch dog. 1: enable 0:disable

Clr_wdt: write 1 into this bit to clear the counter of watch dog. This bit is cleared by hardware automatically.

Idle_wdt :

1: watch dog counter is active when CPU in IDLE mode.

0: watch dog counter is inactive when CPU in IDLE mode.

Ps2, ps1, ps0: the prescale of watch dog clock.

PS2	PS1	PS0	PRE_scale
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

Table 19 The Prescale of Watch Dog clock

The overflow time of watch dog：

$$WatchdogOverflowTime = \frac{PRE\_scale \times 32768}{\texttt{system clock}}$$

When overflow occur, the whole system will be reset.

11 Timer

The BK51 includes three timer/counters (Timer 0 and Timer 1 Timer2). Each timer/counter can operate as either a timer with a clock rate based on the clk input, or as an event counter clocked by the t0 pin (Timer 0), t1pin (Timer 1) or the t2 pin (Timer 2).

11.1 Timer0 and Time1

Timers 0 and 1 each operate in four modes, as controlled through the TMOD SFR and the TCON SFR. The four modes are:
-13-bit timer/counter (mode 0)
-16-bit timer/counter (mode 1)
-8-bit counter with auto-reload (mode 2)
-Two 8-bit counters (mode 3, Timer 0 only)
The next is the register description：

Bits	Definition
TMOD.7	GATE – Timer1 gate control. When GATE=1, Timer1 will clock only when int1_n= 1 and TR1 (TCON.6) =1. When GATE=0, Timer1 will clock only when TR1= 1, regardless of the state of int1_n.
TMOD.6	C/T – Counter/Timer select. When C/T=0, Timer 1 is clocked by clk/4 or clk/12, depending on the state of T1M (CKCON.4). When C/T=1, Timer1 is clocked by the t1(P0.1) pin.
TMOD.5	M1：Timer1 mode selects bit1.
TMOD.4	M0: Timer1 mode selects bit0. M1M0: 00 mode0 01 mode1 10 mode2 11 mode3
TMOD.3	GATE – Timer 0 gate control. When GATE=1, Timer 0 will clock only when int0_n= 1 and TR0 (TCON.4)=1. When GATE= 0, Timer 0 will clock only when TR0= 1, regardless of the state of int0_n.
TMOD.2	C/T – Counter/Timer select. When C/T=0, Timer 0 is clocked by clk/4 or clk/12, depending on the state of T0M (CKCON.3). When C/T=1, Timer 0 is clocked by the t0 pin
TMOD.1	M0: Timer0 mode select bit1.
TMOD.0	M0: Timer0 mode select bit0. M1M0: 00 mode0 01 mode1 10 mode2 11 mode3

Table 20 Timer TMOD Register

Bits	Definition
TCON.7	TF1 – Timer1 overflow flag. Set to1 when the Timer1 count overflows and cleared when the CPU vectors to the interrupt service routine.
TCON.6	TR1 – Timer1 run control. Set to1 to enable counting on Timer1.
TCON.5	TF0 – Timer 0 overflow flag. Set to1 when the Timer 0 count overflows and cleared when the CPU vectors to the interrupt service routine.
TCON.4	TR0 – Timer 0 run control. Set to1 to enable counting on Timer 0.
TCON.3	IE1 – Interrupt 1 detect. IE1 is set by hardware when an interrupt is detected on the int1_n pin and is automatically cleared when the CPU vectors to the corresponding interrupt service routine.
TCON.2	IT1 – Interrupt 1 type select. When IT1=1, the BK51 detects int1_n on the falling edge (edge-sensitive). When IT1=0, the BK51 detects int1_n as a low level (level-sensitive).
TCON.1	IE0 – Interrupt 1 detect. IE0 is set by hardware when a interrupt is detected on the int0_n pin and is automatically cleared when the CPU vectors to the corresponding interrupt service routine.
TCON.0	IT0 – Interrupt 0 type select. When IT0=1, the BK51 detects int0_n on the falling edge (edge-sensitive). When IT0=0, the BK51 detects int0_n as a low level (level-sensitive).

Table 21 Timer TCON Register

11.1.1 Mode0

Mode 0 operation, illustrated in next Figure, is the same for Timer 0 and Timer 1. In mode 0, the timer is configured as a 13-bit counter that uses bits0–4 of TL0 (or TL1) and all 8 bits of TH0 (or TH1). The timer enable bit (TR0/TR1) in the TCON SFR starts the timer. The C/T bit selects the timer/counter clock source, MCU clk or t0/t1 pin.

When the 13-bit count increments from1FFFh (all ones), the counter rolls over to all zeros, the TF0 (or TF1) bit is set in the TCON SFR.

11.1.2 Mode1

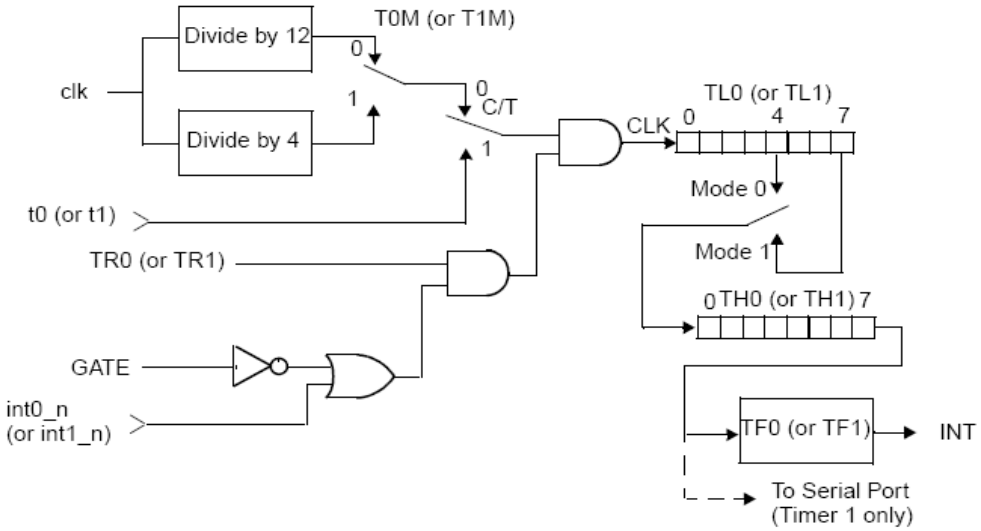


Figure 10 Timer 0/1 – Modes 0 and 1

Mode 1 operation is the same for Timer 0 and Timer 1. In mode 1, the timer is configured as a 16-bit counter. All 8 bits of the LSB register (TL0 or TL1) are used. The counter rolls over to all zeros when the count increments from FFFFh. Otherwise, mode 1 operation is the same as mode 0.

11.1.3 Mode2

Mode 2 operation is the same for Timer 0 and Timer 1. In mode 2, the timer is configured as an 8-bit counter, with automatic reload of the start value. The LSB register (TL0 or TL1) is the counter and the MSB register (TH0 or TH1) stores the reload value.

Mode 2 counter control is the same as for mode 0 and mode 1. However, in mode 2, when TLn increments from FFh, the value stored in THn is reloaded into TLn.

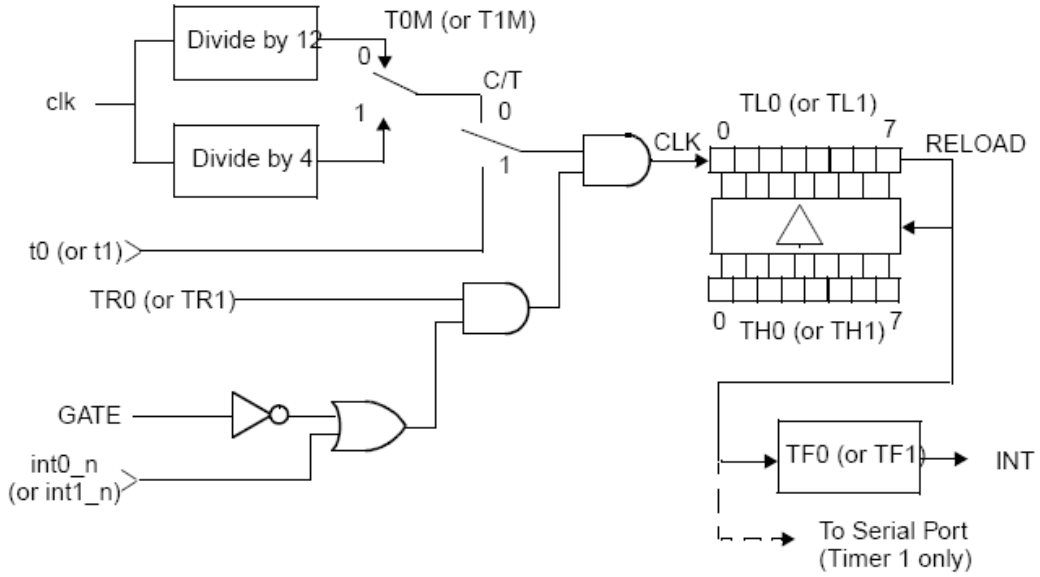


Figure 11 Timer 0/1 – Mode 2

11.1.4 Mode3

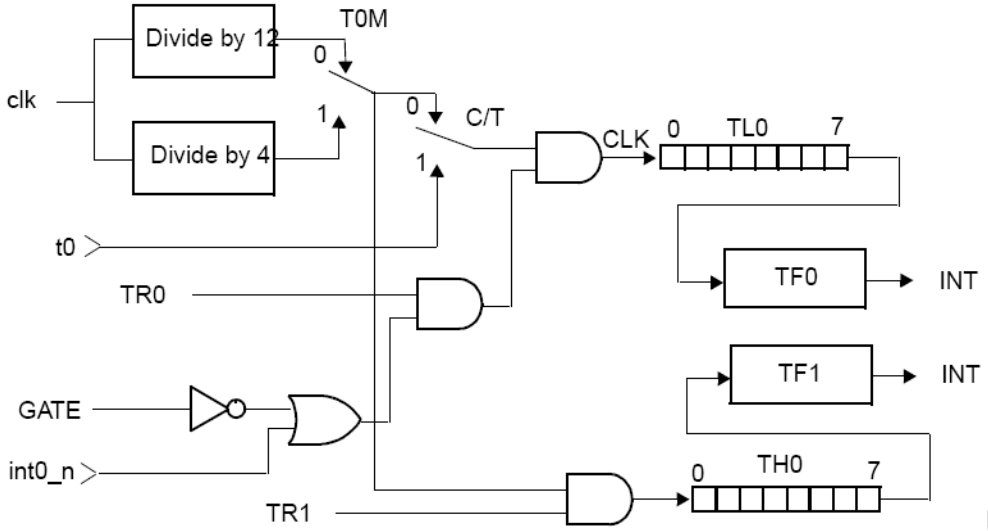


Figure 12 Timer 0 – Mode 3

In mode 3, Timer 0 operates as two 8-bit counters and Timer 1 stops counting and holds its value.

As shown in above figure, TL0 is configured as an 8-bit counter controlled by the normal Timer 0 control bits. TL0 can either count clk cycles (divided by 4 or by 12) or high-to-low transitions on t0, as determined by the C/T bit. The GATE function can be used to give counter enable control to the int0_n signal.

TH0 functions as an independent 8-bit counter. However, TH0 can only count clk

cycles (divided by 4 or by12). The Timer 1 control and flag bits (TR1 and TF1) are used as the control and flag bits for TH0.

When Timer 0 is in mode 3, Timer 1 has limited usage because Timer 0 uses the Timer 1 control bit (TR1) and interrupt flag (TF1). Timer 1 can still be used for baud rate generation and the Timer 1 count values are still available in the TL1 and TH1 registers.

11.2 Timer – Rate Conrol

Using the default rate (12 clocks per timer increment) allows existing application code with real-time dependencies, such as baud rate, to operate properly. However, applications that require fast timing can set the timers to increment every 4clk cycles by setting bits in the Clock Control register (CKCON) at SFR.

CKCON.5 control Timer2 clock rate
CKCON.4 control Timer1 clock rate
CKCON.3 control Timer0 clock rate

When a CKCON register bit is set to1, the associated counter increments at 4-clk intervals. When a CKCON bit is cleared, the associated counter increments at 12-clk intervals. The timer controls are independent of each other. The default setting for all the three timers is 0 (12-clk intervals).

Note: when setting timer 2 as Baud-Rate generator, the frequency is half of CPUclock, and regardless of the value in CKCON.5.

11.3 Timer2

Timer 2 runs only in 16-bit mode and offers several capabilities not available with Timers 0 and1. The modes available with Timer 2 are:

- 16-bit timer/counter
- 16-bit timer with capture
- 16-bit auto-reload timer/counter
- Baud rate generator

The SFRs associated with Timer 2 are:

T2CON

Bits	Definition
T2CON.7	TF2 – Timer 2 overflow flag. Hardware will set TF2 when Timer 2 overflows from FFFFh. TF2 must be cleared to 0 by the software. TF2 will only be set to a 1 if RCLK and TCLK are both cleared to 0. Writing a1 to TF2 forces a Timer 2 interrupt if enabled.
T2CON.6	EXF2 – Timer 2 external flag. Hardware will set EXF2 when a reload or capture is caused by a high-to-low transition on the t2ex pin, and EXEN2 is set. EXF2 must be cleared to 0 by the software. Writing a 1 to EXF2 forces a Timer 2 interrupt if enabled.
T2CON.5	RCLK – Receive clock flag. Determines whether Timer1 or Timer 2 is used for Serial Port 0 timing of received data in serial mode 1 or 3. RCLK =1 selects Timer 2 overflow as the receive clock. RCLK=0 selects Timer1 overflow as the receive clock.

T2CON.4	TCLK – Transmit clock flag. Determines whether Timer 1 or Timer 2 is used for Serial Port 0 timing of transmit data in serial mode 1 or 3. TCLK =1 selects Timer 2 overflow as the transmit clock. TCLK=0 selects Timer1 overflow as the transmit clock.
T2CON.3	EXEN2 – Timer 2 external enable. EXEN2=1 enables capture or reload to occur as a result of a high-to-low transition on t2ex, if Timer 2 is not generating baud rates for the serial port. EXEN2= 0 causes Timer 2 to ignore all external events at t2ex.
T2CON.2	TR2 – Timer 2 run control flag. TR2=1 starts Timer 2. TR2=0 stops Timer 2.
T2CON.1	C/T2 – Counter/timer select. C/T2= 0 selects a timer function for Timer 2. C/T2=1 selects a counter of falling transitions on the t2 pin. When used as a timer, Timer 2 runs at 4 clocks per increment or12 clocks per increment as programmed by CKCON.5, in all modes except baud rate generator mode. When used in baud rate generator mode, Timer 2 runs at 2 clocks per increment, independent of the state of CKCON.5.
T2CON.0	CP/RL2 – Capture/reload flag. When CP/RL2= 1, Timer 2 captures occur on high-to-low transitions of t2ex, if EXEN2= 1. When CP/RL2=0, auto-reloads occur when Timer 2 overflows or when high-to-low transitions occur on t2ex, if EXEN2= 1. If either RCLK or TCLK is set to1, CP/RL2 will not function and Timer 2 will operate in auto-reload mode following each overflow.

Table 22Timer2 T2CON Register

Next table summarizes how the SFR bits determine the Timer 2 mode.

RCLK ,TCLK	CP/RL2	TR2	Mode
00	0	1	16-bit timer/counter with auto-reload
00	1	1	16-bit timer/counter with capture
1x or x1	x	1	Baud rate generator
xx	x	0	off

Table 23 Timer 2 Mode Control Summary

RCAP2L: Used to capture the TL2 value when Timer 2 is configured for capture mode, or as the LSB of the 16-bit reload value when Timer 2 is configured for auto-reload mode.

RCAP2H: Used to capture the TH2 value when Timer 2 is configured for capture mode, or as the MSB of the 16-bit reload value when Timer 2 is configured for auto-reload mode.

TL2: Lower 8 bits of the 16-bit count.

TH2: Upper 8 bits of the 16-bit count.

11.3.1 16-Bit Timer/Counter Mode

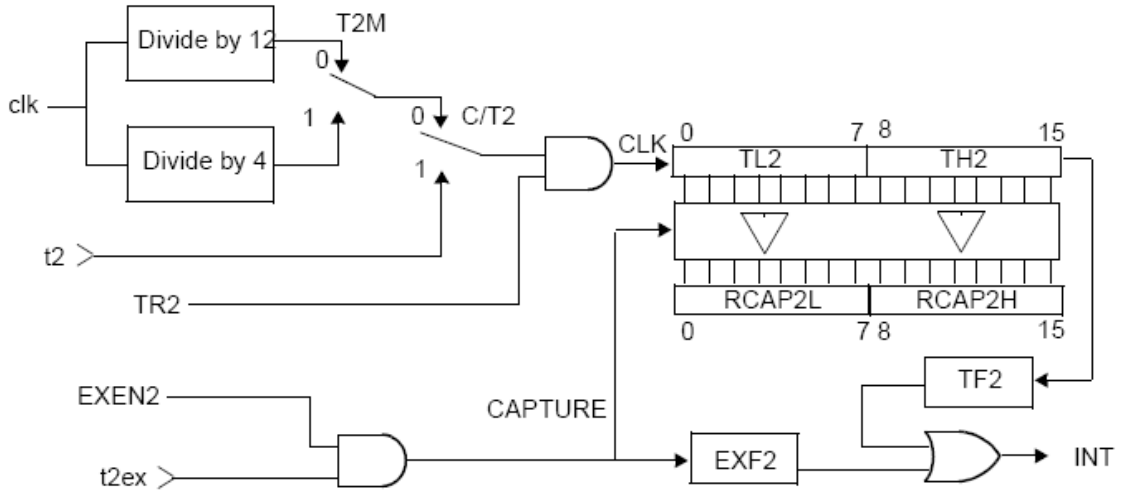


Figure 13 Timer 2 – Timer/Counter with Capture

The above figure illustrates how Timer 2 operates in timer/counter mode with the optional capture feature. The C/T2 bit determines whether the 16-bit counter counts clk cycles (divided b 4 or 12), or high-to-low transitions on the t2 pin. The TR2 bit enables the counter. When the count increments from FFFFh, the TF2 flag is set, and t2_out goes high for one clk cycle.

The Timer 2 capture mode is the same as the 16-bit timer/counter mode, with the addition of the capture registers and control signals. The CP/RL2 bit in the T2CON SFR enables the capture feature. When CP/RL2 = 1, a high-to-low transition on t2ex when EXEN2= 1 causes the Timer 2 value to be loaded into the capture registers (RCAP2L and RCAP2H).

11.3.2 16-Bit Timer/Counter Mode with Auto-Reload

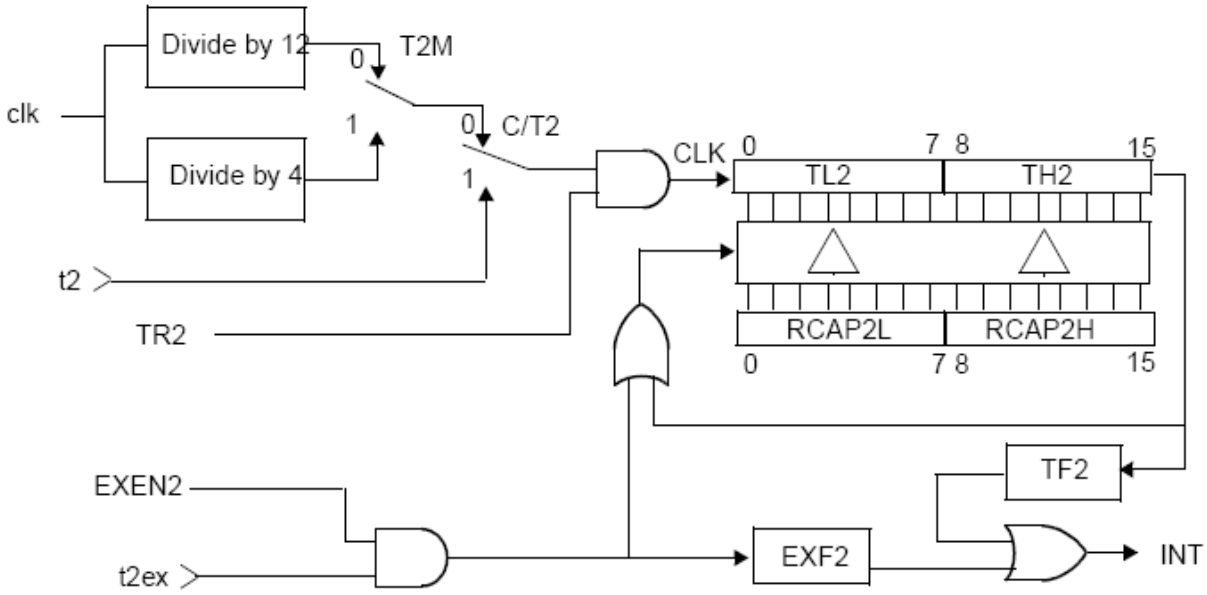


Figure 14 Timer 2 – Timer/Counter with Auto-Reload

When CP/RL2= 0, Timer 2 is configured for the auto-reload mode illustrated in above figure . Control of counter input is the same as for the other 16-bit counter modes. When the count increments from FFFFh, Timer 2 sets the TF2 flag and the starting value is reloaded into TL2 and TH2. The software must preload the starting value into the RCAP2L and RCAP2H registers.

When Timer 2 is in auto-reload mode, a reload can be forced by a high-to-low transition on the t2ex pin, if enabled by EXEN2 = 1.

11.3.3 Baud Rate Generator Mode

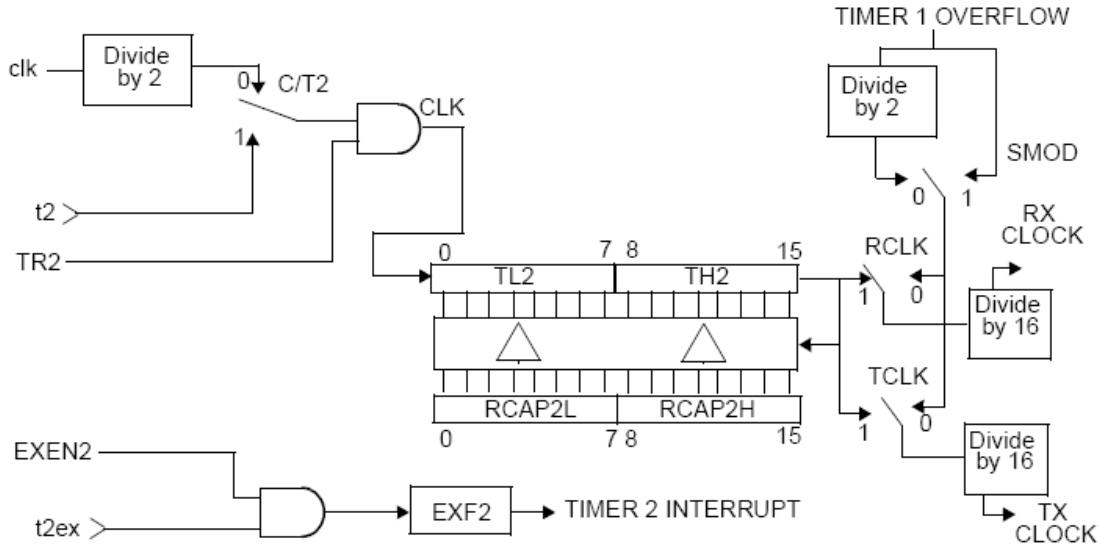


Figure 15 Timer 2 – Baud Rate Generator Mode

Setting either RCLK or TCLK to1 configures Timer 2 to generate baud rates for Serial Port in serial mode 1 or 3. In baud rate generator mode, Timer 2 functions in autoreload mode. However, instead of setting the TF2 flag, the counter overflow generates a shift clock for the serial port function. As in normal auto-reload mode, the overflow also causes the preloaded start value in the RCAP2L and RCAP2H registers to be reloaded into the TL2 and TH2 registers.

When either TCLK = 1 or RCLK = 1, Timer 2 is forced into auto-reload operation, regardless of the state of the CP/RL2 bit.When operating as a baud rate generator, Timer 2 does not set the TF2 bit. In this mode, a Timer 2 interrupt can only be generated by a high-to-low transition on the t2ex pin setting the EXF2 bit, and only if enabled by EXEN2 = 1.

The counter time base in baud rate generator mode is clk/2. To use an external clock source, set C/T2 to 1 and apply the desired clock source to the t2 pin.

12 UART

Serial Port of BK51 is identical in operation to the standard 8051 serial port. The data sent includes start bit, data bits (LSB send first) and stop bit at least.

The register inculdes：

SBUF: Serial port R/W register

SCON：Serial port control register

bits	Description
SCON.7	SM0-Serial port mode bit0.
SCON.6	SM1-Serial port mode bit 1, decoded as: SM0-SM1 mode 0 0 0 0 1 1 1 0 2 1 1 3
SCON.5	SM2 – Multiprocessor communication enable. In modes 2 and 3, SM2 enables the multiprocessor communication feature. If SM2=1 in mode 2 or 3, RI will not be activated if the received 9th bit is 0. If SM2=1 in mode 1, RI will only be activated if a valid stop is received.
SCON.4	REN – Receive enable. When REN=1, reception is enabled.
SCON.3	TB8 – Defines the state of the 9th data bit transmitted in modes 2 and 3.
SCON.2	RB8 – In modes 2 and 3, RB8 indicates the state of the 9th bit received. In mode1, RB8 indicates the state of the received stop bit. In mode 0, RB8 is not used.
SCON.1	TI – Transmit interrupt flag. Indicates that the transmit data word has been shifted out. In mode 0, TI is set at the end of the 8th data bit. In all other modes, TI is set when the stop bit is placed on the txd0 pin. TI must be cleared by the software.
SCON.0	RI – Receive interrupt flag. Indicates that a serial data word has been received. In mode 0, RI_1 is set at the end of the 8th data bit. If serial port interrupt enabled(EA=1 and ES=1), it will generate interrupt when RI=1.

Table 24 UART SCON Register

12.1 Mode0

Serial mode 0 provides synchronous, half-duplex serial communication.

The serial mode 0 baud rate is either clk/12 or clk/4, depending on the state of the SM2 bit. When SM2= 0, the baud rate is clk/12; when SM2 = 1, the baud rate is clk/4.

Mode 0 operation is identical to the standard 8051. Data transmission begins when an instruction writes to the SBUF SFR. The UART shifts the data out, LSB first, at the

selected baud rate, until the 8-bit value has been shifted out.

Mode 0 data reception begins when the REN bit is set and the RI bit is cleared in the corresponding SCON SFR. The shift clock is activated and the UART shifts data in on each rising edge of the shift clock until 8 bits have been received. One machine cycle after the 8th bit is shifted in, the RIbit is set and reception stops until the software clears the RI bit.

The next two figure illustrate Serial Port Mode 0 transmit and receive timing.

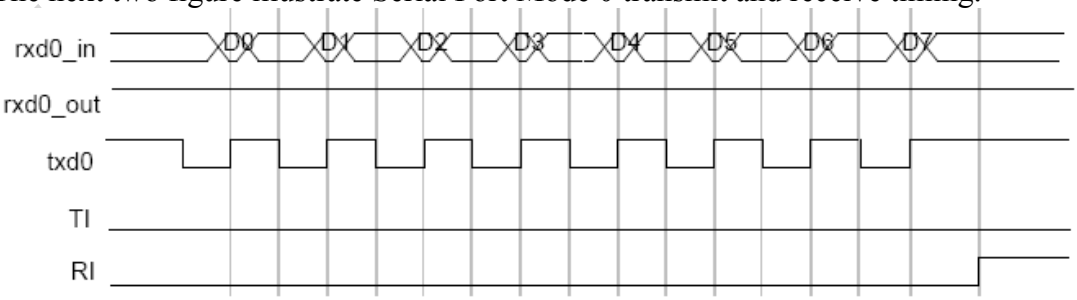


Figure 16 Uart Mode0 Tramsmit Timing

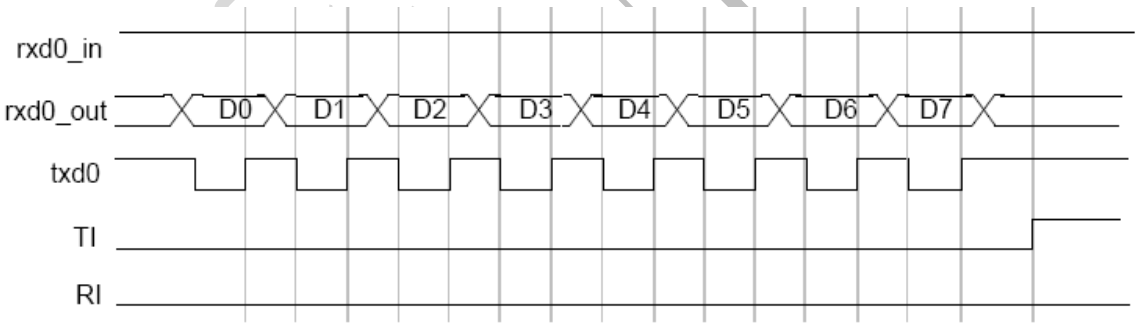


Figure 17 Uart Mode0 Receive Timing

12.2 Mode1

When SM=01, serial port works as mode 1. Mode 1 provides standard asynchronous, full-duplex communication, using a total of 10 bits: 1 start bit, 8data bits, and 1 stop bit. For receive operations, the stop bit is stored in RB8. Data bits are received and transmitted LSB first.

12.2.1 Mode1 Baud Rate

The mode 1 baud rate is a function of timer overflow. Serial Port can use either Timer 1 or Timer 2 to generate baud rates.

Each time the timer increments from its maximum count (FFh for Timer 1 or FFFFh for Timer 2), a clock is sent to the baud rate circuit. The clock is then divided by16 to generate the baud rate.

(1) using Timer1

When using Timer 1, the SMOD (the MSB bit of PCON) bit selects whether or not to divide the Timer 1 rollover rate by 2. Therefore, when using Timer 1, the baud rate is determined by the equation:

$$Baud-Rate=\frac{2^{SMOD}}{32}\times \text{Timer1 Overflow}$$

To use Timer 1 as the baud rate generator, it is best to use Timer 1 mode 2 (8-bit counter with auto-reload), although any counter mode can be used. The Timer 1 reload value is stored in the TH1 register, which makes the complete formula for Timer 1:

$$Baud-Rate=\frac{2^{SMOD}}{32}\times\frac{clk}{12\times(256-TH1)}$$

The 12 in the denominator in the above equation can be changed to 4 by setting the T1M bit in the CKCON SFR. To derive the required TH1 value from a known baud rate (when TM1 = 0), use the equation:

$$TH1=256-\frac{2^{SMOD}\times clk}{384\times Baud-Rate}$$

(2) When using Timer2

When using Timer 2, the baud rate is determined by the equation:

$$Baud-Rate=\frac{\text{Timer2 Overflow}}{16}$$

To use Timer 2 as the baud rate generator, configure Timer 2 in auto-reload mode and set the TCLK and/or RCLK bits in the T2CON SFR. TCLK selects Timer 2 as the baud rate generator for the transmitter; RCLK selects Timer 2 as the baud rate generator for the receiver. The 16-bit reload value for Timer 2 is stored in the RCAP2L and RCA2H SFRs, which makes the equation for the Timer 2 baud rate:

$$Baud-Rate=\frac{clk}{32\times(65536-RCAP2H,RCAP2L)}$$

Where RCAP2H,RCAP2L is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned number.

To derive the required RCAP2H and RCAP2L values from a known baud rate, use the equation:

$$RCAP2H,RCAP2L=65536-\frac{clk}{32\times Baud-Rate}$$

The 32 in the denominator is the result of the clk being divided by2 and the Timer 2 overflow being divided by16.

12.2.2 Mode1 tramsmit

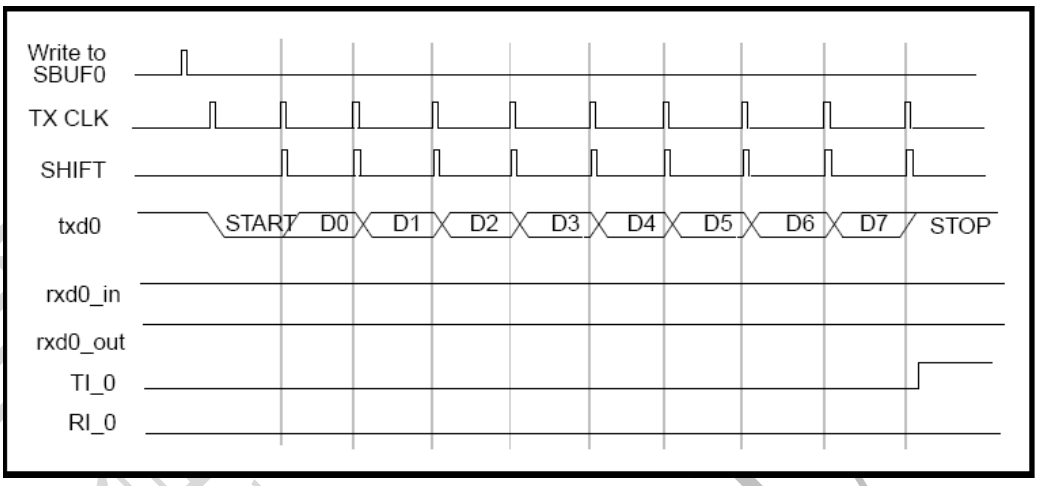


Figure 18 Uart Mode1 Transmit Timing

In mode 1, the UART begins transmitting after the first rollover of the divide-by-16 counter after the software writes to the SBUF0 (or SBUF1) register. The UART transmits data on the txd pin in the following order: start bit, 8data bits (LSB first), stop bit. The TIbit is set 2 clk cycles after the stop bit is transmitted.

12.2.3 Mode1 Receive

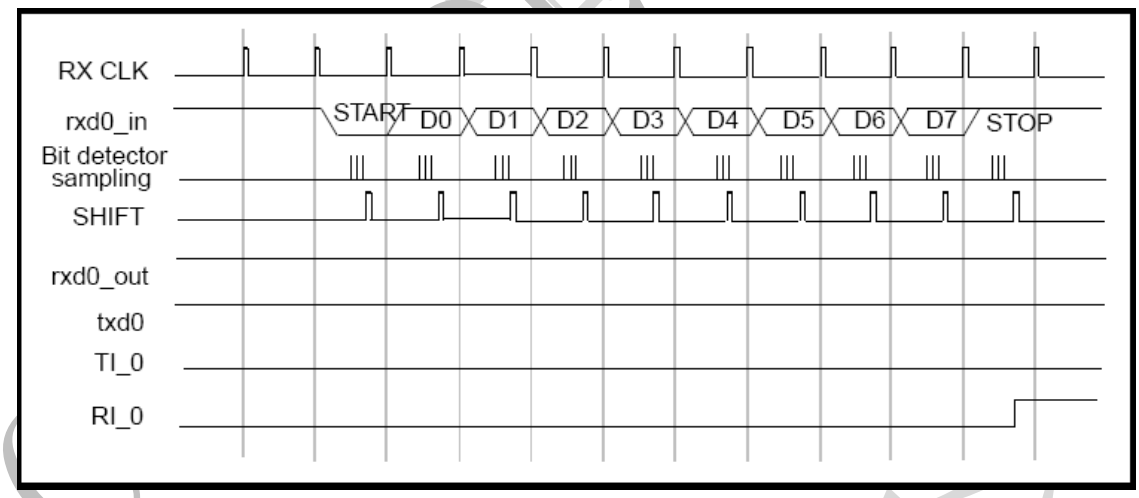


Figure 19 Uart Mode1Receive Timing

Reception begins at the falling edge of a start bit received on rxd_in, when enabled by the REN bit. For this purpose, rxd_in is sampled sixteen times per bit for any baud rate. When a falling edge of a start bit is detected, the divide-by-16 counter used to generate the receive clock is reset to align the counter rollover to the bit boundaries.

For noise rejection, the serial port establishes the content of each received bit by a majority decision of three consecutive samples in the middle of each bit time. This is especially true for the start bit. If the falling edge on rxd_in is not verified by a majority decision of three consecutive samples (low), then the serial port stops reception and waits

for another falling edge on rxd_in.

At the middle of the stop bit time, the serial port checks for the following conditions:

- RI= 0,
- and If SM2= 1, the state of the stop bit is1.

(If SM2= 0, the state of the stop bit doesn't matter.)

If the above conditions are met, the serial port then writes the received byte to the SBUF register, loads the stop bit into RB8, and sets the RI bit. If the above conditions are not met, the received data is lost, the SBUF register and RB8 bit are not loaded, and the RI bit is not set. After the middle of the stop bit time, the serial port waits for another high-to-low transition on the (rxd_in) pin.

12.3 Mode2

Mode 2 provides asynchronous, full-duplex communication, using a total of 11bits:

- One start bit
- Eight data bits
- One programmable 9th bit
- One stop bit.

The data bits are transmitted and received LSB first. For transmission, the 9thbit is determined by the value in TB8. To use the 9th bit as a parity bit, move the value of the P bit (SFR PSW.0) to TB8.

Transmission begins after the first rollover of the divide-by-16 counter following a software write to SBUF. The UART shifts data out on the txd pin in the following order: start bit, data bits (LSB first), 9th bit, stop bit. The TI bit is set when the stop bit is placed on the txd pin.

Reception begins at the falling edge of a start bit received on rxd_in, when enabled by the RENbit. For this purpose, rxd_in is sampled sixteen times per bit for any baud rate. When a falling edge of a start bit is detected, the divide-by-16 counter used to generate the receive clock is reset to align the counter rollover to the bit boundaries.

The next figure is the timing of receive and transmit in mode 2.

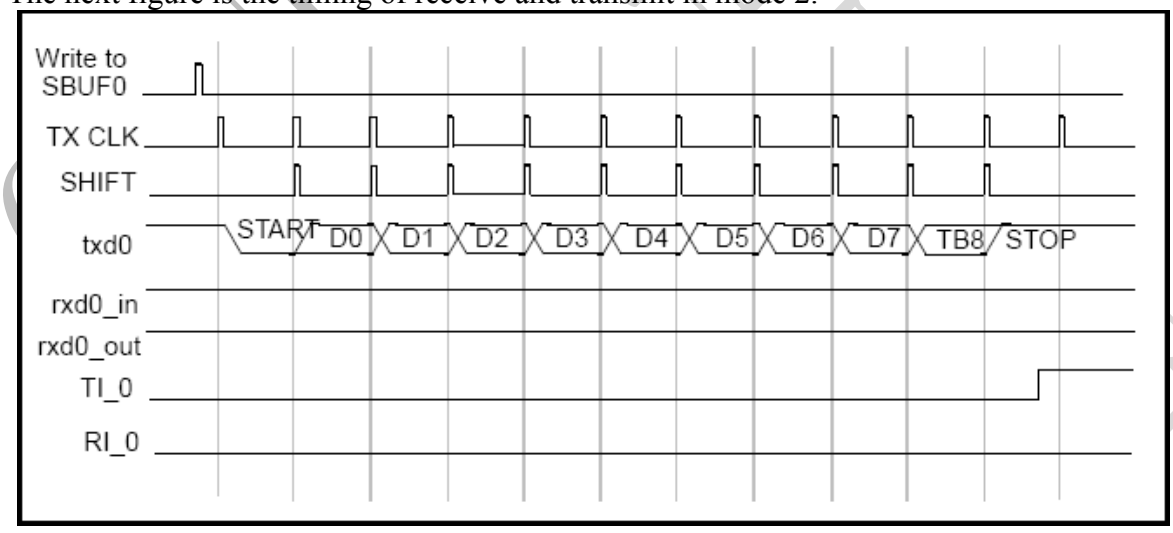


Figure 20 Uart Mode2 Transmit Timing

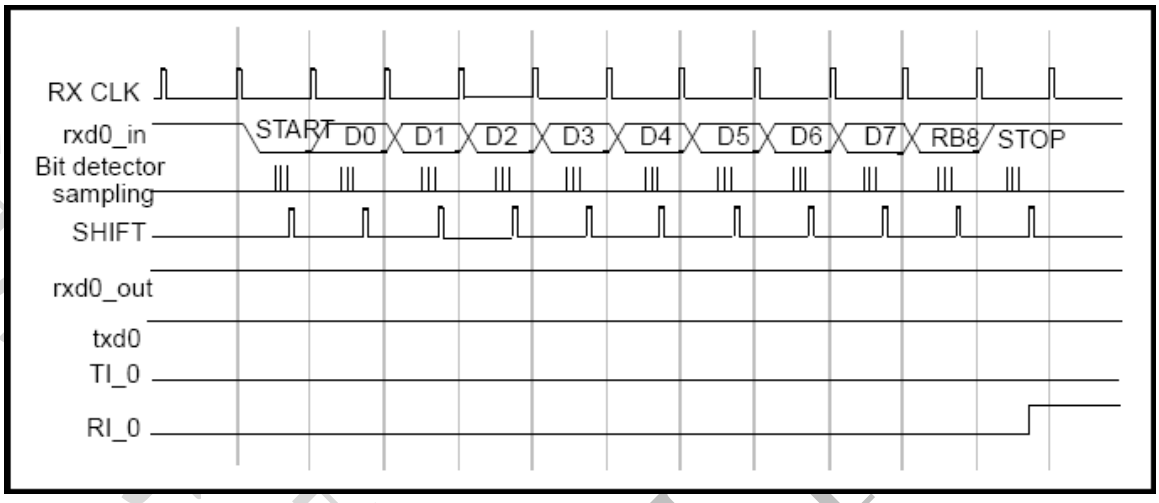


Figure 21 Uart Mode2 Receive Timing

12.4 Mode3

Mode 3 provides asynchronous, full-duplex communication, using a total of 11bits:

- One start bit
- Eight data bits
- One programmable 9th bit
- One stop bit. The data bits are transmitted and received LSB first.

The mode 3 transmits and operations are identical to mode 2. The mode 3 baud rate generation is identical to mode 1. That is, mode 3 is a combination of mode 2 protocol and mode 1 baud rate.

13 SPI

The Enhanced Serial Peripheral Interface (SPI) provides access to a flexible, full-duplex synchronous serial bus. SPI can operate as a master or slave device in both 3-wire or 4-wire modes, and supports multiple masters and slaves on a single SPI bus. The slave-select (NSS) signal can be configured as an input to select SPI in slave mode, or to disable Master Mode operation in a multi-master environment, avoiding contention on the SPI bus when more than one master attempts simultaneous data transfers. NSS can also be configured as a chip-select output in master mode, or disabled for 3-wire operation. Additional general purpose port I/O pins can be used to select multiple slave devices in master mode.

13.1 Signal Description

The four signals used by SPI (MOSI, MISO, SCK, NSS) are described below.

13.1.1 Master Out, Slave In（MOSI）

The master-out, slave-in (MOSI) signal is an output from a master device and an input to slave devices. It is used to serially transfer data from the master to the slave. This signal is an output when SPI is operating as a master and an input when SPI is operating as a slave. Data is transferred most-significant bit first. When configured as a master, MOSI is driven by the MSB of the shift register in both 3- and 4-wire mode.

13.1.2 Master In, Slave Out（MISO）

The master-in, slave-out (MISO) signal is an output from a slave device and an input to the master device. It is used to serially transfer data from the slave to the master. This signal is an input when SPI is operating as a master and an output when SPI is operating as a slave. Data is transferred most-significant bit first. The MISO pin is placed in a high-impedance state when the SPI module is disabled and when the SPI operates in 4-wire mode as a slave that is not selected. When acting as a slave in 3-wire mode, MISO is always driven by the MSB of the shift register.

13.1.3 Serial Clock（SCK）

The serial clock (SCK) signal is an output from the master device and an input to slave devices. It is used to synchro-nize the transfer of data between the master and slave on the MOSI and MISO lines. SPI generates this signal when operating as a master. The SCK signal is ignored by a SPI slave when the slave is not selected (NSS = 1) in 4-wire slave mode.

13.1.4 Slave Select（NSS）

The function of the slave-select (NSS) signal is dependent on the setting of the NSSMD1 and NSSMD0 bits in the SPICN register. There are three possible modes that can be selected with these bits:

1. NSSMD[1:0] = 00: 3-Wire Master or 3-Wire Slave Mode: SPI operates in 3-wire mode, and NSS is disabled. When operating as a slave device, SPI is always selected in 3-wire mode. Since no select signal is present, SPI must be the only slave on the bus in 3-wire mode. This is intended for point-to-point commu-nication between a master and one slave.

2. NSSMD[1:0] = 01: 4-Wire Slave or Multi-Master Mode: SPI operates in 4-wire mode, and NSS is enabled as an input. When operating as a slave, NSS selects the SPI device. When operating as a master, a 1-to-0 transition of the NSS signal disables the master function of SPI so that multiple master devices can be used on the same SPI bus.

3. NSSMD[1:0] = 1x: 4-Wire Master Mode: SPI operates in 4-wire mode, and NSS is enabled as an output. The setting of NSSMD0 determines what logic level the NSS pin will output. This configuration should only be used when operating SPI as a master device.

The function of the four pin(MOSI、MISO、SCK 、NSS) is the second function of the port. Whether use the second function is decieded by the config register(SPIEN、MSTEN、NSSMD1、NSSMD0), also it decides the input or output direction of the port. For example, when SPIEN=0, all the four pin MOSI, MISO, SCK, NSS are disabled, the four port are used as general port. When SPIEN=1, the SPI is enabled, then it is four wire mode or three wire mode is decided by the value of MSTEN、NSSMD1、NSSMD0. If it is used as master three wire mode, then config the port as MOSI、MISO、SCK, and MISO is input port, MOSI and SCK is output port.

NSS is used at 4-wire mode, and it is disabled at 3-wire mode.

13.2 Master-Slave connection method

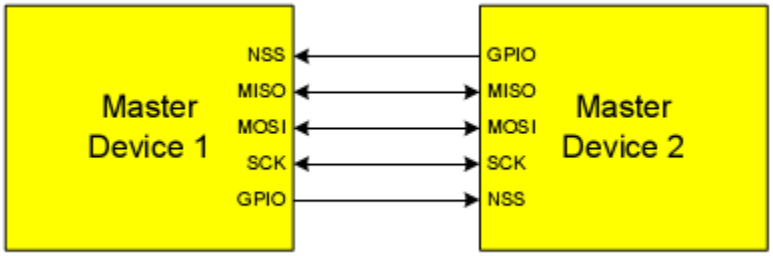


Figure 22 Multiple-Master Mode Connection Diagram

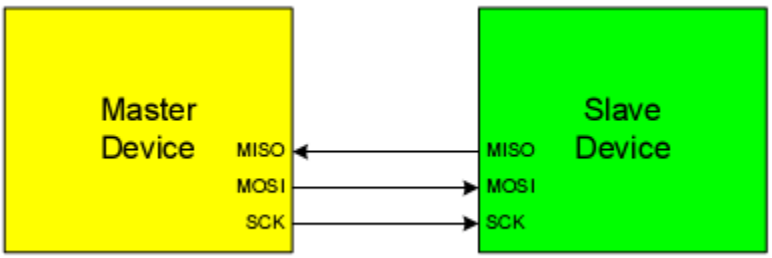


Figure 23 3-Wire Single Master and 3-Wire Single Slave Mode Connection Diagram

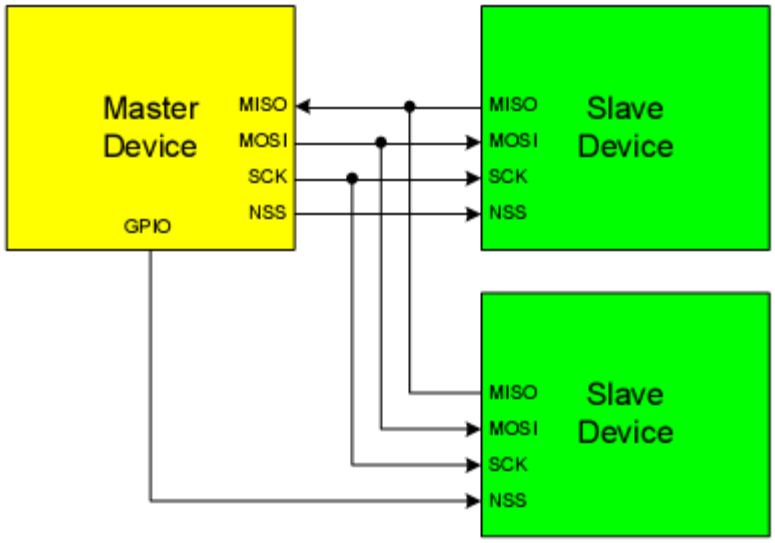


Figure 24 4-Wire Single Master Mode and 4-Wire Slave Mode Connection Diagram

13.3 SPI Master Mode Operation

A SPI master device initiates all data transfers on a SPI bus. SPI is placed in master mode by setting the Master Enable flag (MSTEN, SPICN.6). Writing a byte of data to the SPI data register (SPIDAT) when in master mode writes to the transmit buffer. If the SPI shift register is empty, the byte in the transmit buffer is moved to the shift reg-ister, and a data transfer begins. The SPI master immediately shifts out the data serially on the MOSI line while pro-viding the serial clock on SCK. The SPIF (SPICN.7) flag is set to logic 1 at the end of the transfer. If interrupts are enabled, an interrupt request is generated when the SPIF flag is set. While the SPI master transfers data to a slave on the MOSI line, the addressed SPI slave device simultaneously transfers the contents of its shift register to the SPI master on the MISO line in a full-duplex operation. Therefore, the SPIF flag serves as both a transmit-complete and receive-data-ready flag. The data byte received from the slave is transferred MSB-first into the master's shift register. When a byte is fully shifted into the register, it is moved to the receive buffer where it can be read by the processor by reading SPIDAT.

When configured as a master, SPI can operate in one of three different modes: multi-master mode, 3-wire single-master mode, and 4-wire single-master mode. The default, multi-master mode is active when NSSMD1 (SPICN.3) = 0 and NSSMD0 (SPICN.2) = 1. In this mode, NSS is an input to the device, and is used to disable the master SPIwhen another master is accessing the bus. When NSS is pulled low in this mode, MSTEN (SPICN.6) and SPIEN (SPICN.0) are set to 0 to disable the SPI master device, and a Mode Fault is generated (MODF, SPICN.5 = 1). Mode Fault will generate an interrupt if enabled. SPI must be manually re-enabled in software under these circum-stances. In multi-master systems, devices will typically default to being slave devices while they are not acting as the system master device. In multi-master mode, slave devices can be addressed individually (if needed) using general-purpose I/O pins.

3-wire single-master mode is active when NSSMD1 (SPICN.3) = 0 and NSSMD0

(SPICN.2) = 0. In this mode, NSS is not used, and is not mapped to an external port pin through the crossbar. Any slave devices that must be addressed in this mode should be selected using general-purpose I/O pins.

4-wire single-master mode is active when NSSMD1 (SPICN.3) = 1. In this mode, NSS is configured as an output pin, and can be used as a slave-select signal for a single SPI device. In this mode, the output value of NSS is con-trolled (in software) with the bit NSSMD0 (SPICN.2). Additional slave devices can be addressed using general-pur-pose I/O pins.

13.4 SPI Slave Mode Operation

When SPI is enabled and not configured as a master, it will operate as a SPI slave. As a slave, bytes are shifted in through the MOSI pin and out through the MISO pin by a master device controlling the SCK signal. A bit counter in the SPI logic counts SCK edges. When 8 bits have been shifted through the shift register, the SPIF flag is set to logic 1, and the byte is copied into the receive buffer. Data is read from the receive buffer by reading SPIDAT. A slave device cannot initiate transfers. Data to be transferred to the master device is pre-loaded into the shift register by writ-ing to SPIDAT. Writes to SPIDAT are double-buffered, and are placed in the transmit buffer first. If the shift regis-ter is empty, the contents of the transmit buffer will immediately be transferred into the shift register. When the shift register already contains data, the SPI will load the shift register with the transmit buffer's contents after the last SCK edge of the next (or current) SPI transfer.

When configured as a slave, SPI can be configured for 4-wire or 3-wire operation. The default, 4-wire slave mode, is active when NSSMD1 (SPICN.3) = 0 and NSSMD0 (SPICN.2) = 1. In 4-wire mode, the NSS signal is routed to a port pin and configured as a digital input. SPI is enabled when NSS is logic 0, and disabled when NSS is logic 1. The bit counter is reset on a falling edge of NSS. Note that the NSS signal must be driven low at least 2 system clocks before the first active edge of SCK for each byte transfer.

3-wire slave mode is active when NSSMD1 (SPICN.3) = 0 and NSSMD0 (SPICN.2) = 0. NSS is not used in this mode, and is not mapped to an external port pin through the crossbar. Since there is no way of uniquely addressing the device in 3-wire slave mode, SPI must be the only slave device present on the bus. It is important to note that in 3-wire slave mode there is no external means of resetting the bit counter that determines when a full byte has been received. The bit counter can only be reset by disabling and re-enabling SPI with the SPIEN bit.

13.5 SPI Interrupt Sources

When SPI interrupts are enabled, the following four flags will generate an interrupt when they are set to logic 1:

1. The SPI Interrupt Flag, SPIF (SPICN.7) is set to logic 1 at the end of each byte transfer. This flag can occur in all SPI modes.

2. The Write Collision Flag, WCOL (SPICN.6) is set to logic 1 if a write to SPIDAT is attempted when the transmit buffer has not been emptied to the SPI shift register. When this occurs, the write to SPIDAT will be ignored, and the transmit buffer will not be written.This flag can occur in all SPI modes. Note that the interrupt is disabled when

SPI_CFG.1=0.

3. The Mode Fault Flag MODF (SPICN.5) is set to logic 1 when SPI is configured as a master, and for multi-master mode and the NSS pin is pulled low. When a Mode Fault occurs, the MSTEN and SPIEN bits in SPICN are set to logic 0.

4. The Receive Overrun Flag RXOVRN (SPICN.4) is set to logic 1 when configured as a slave, and a transfer is completed and the receive buffer still holds an unread byte from a previous transfer. The new byte is not transferred to the receive buffer, allowing the previously received data byte to be read. The data byte which caused the overrun is lost.

13.6 SPI Timing

Four combinations of serial clock phase and polarity can be selected using the clock control bits in the SPI Configu-ration Register (SPICFG). The CKPHA bit (SPICFG.5) selects one of two clock phases (edge used to latch the data). The CKPOL bit (SPICFG.4) selects between an active-high or active-low clock. Both master and slave devices must be configured to use the same clock phase and polarity. SPI should be disabled (by clearing the SPIEN bit, SPICN.0) when changing the clock phase or polarity.

The SPI Clock Rate Register (SPICKR) controls the master mode serial clock frequency. This register is ignored when operating in slave mode. When the SPI is configured as a master, the maximum data transfer rate (bits/sec) is one-half the system clock frequency or 12.5 MHz, whichever is slower. When the SPI is con-figured as a slave, the maximum data transfer rate (bits/sec) for full-duplex operation is 1/10 the system clock fre-quency, provided that the master issues SCK, NSS (in 4-wire slave mode), and the serial input data synchronously with the slave's system clock. If the master issues SCK, NSS, and the serial input data asynchronously, the maximum data transfer rate (bits/sec) must be less than 1/10 the system clock frequency. In the special case where the master only wants to transmit data to the slave and does not need to receive data from the slave (i.e. half-duplex operation), the SPI slave can receive data at a maximum data transfer rate (bits/sec) of 1/4 the system clock frequency. This is provided that the master issues SCK, NSS, and the serial input data synchronously with the slave's system clock.

Note: The system clock frequency is MCU work frequency, which is the input 16M clock divided by { CKCON.7,CKCON.6}.

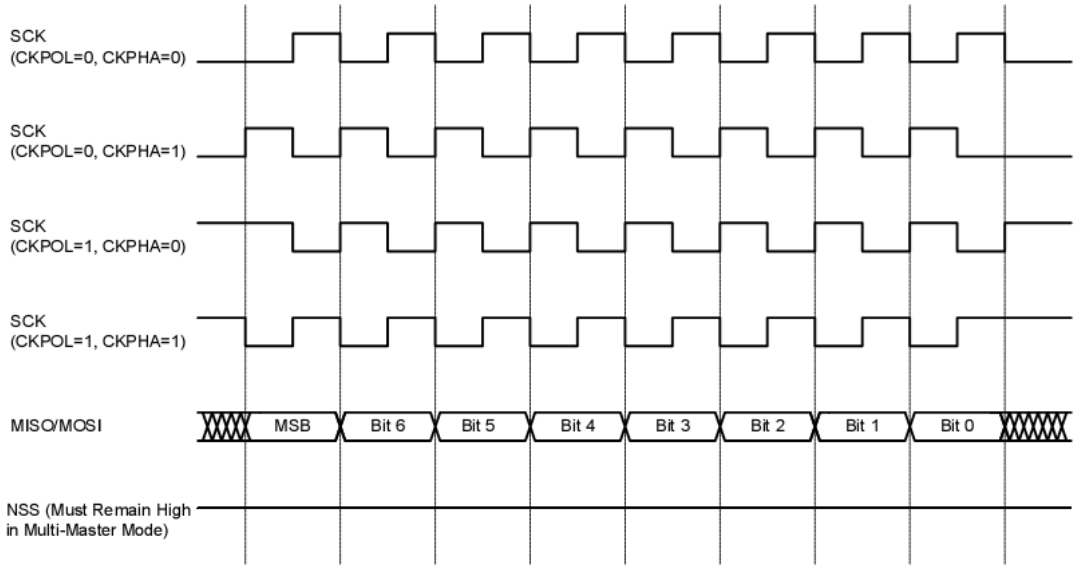


Figure 25 SPI Master Mode Data/Clock Timing

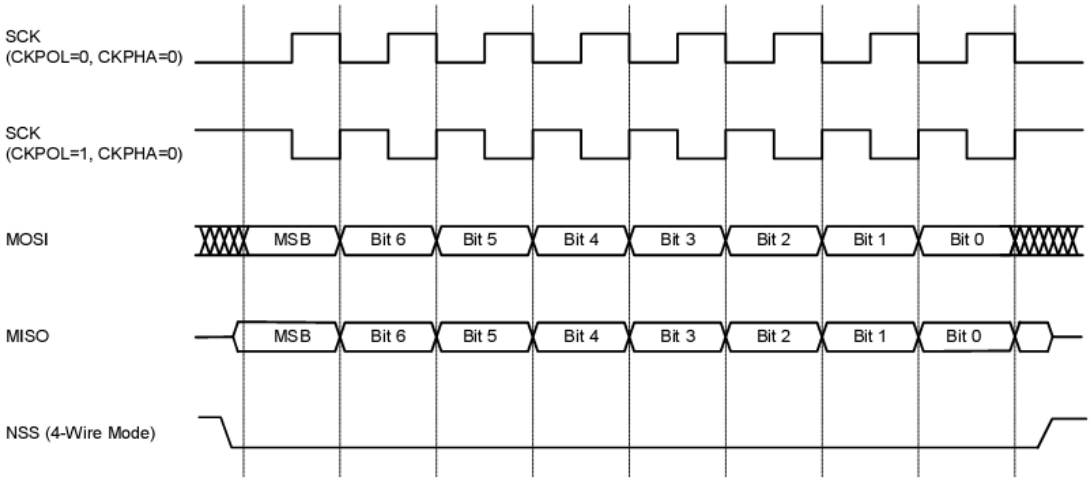


Figure 26 SPI Slave Mode Data/Clock Timing (CKPHA = 0)

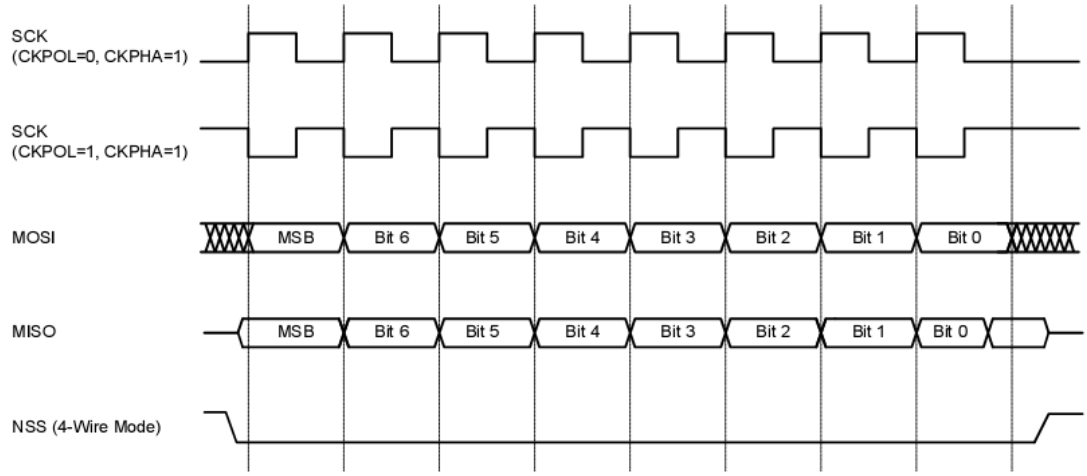


Figure 27 SPI Slave Mode Data/Clock Timing (CKPHA = 1)

Note: In slave mode, the data on MOSI are sampled at the middle of period of every bit. In master mode, the data on MISO are sampled at the last clock period to acquire the maximal setup time.

13.7 SPI Special Function Register

SPI is accessed and controlled through four special function registers in the system controller: SPICN Control Reg-ister, SPIDAT Data Register, SPICFG Configuration Register, and SPICKR Clock Rate Register. The four special function registers related to the operation of the SPI Bus are described in the following figures.

13.7.1 SPI_CFG: SPI Configuration Register

Bits	Definition
SPI_CFG.7	SPIBSY: SPI Busy (read only). This bit is set to logic 1 when a SPI transfer is in progress (Master or slave Mode).
SPI_CFG.6	MSTEN: Master Mode Enable. 0: Disable master mode. Operate in slave mode. 1: Enable master mode. Operate as a master.
SPI_CFG.5	CKPHA: SPI Clock Phase. This bit controls the SPI clock phase. 0: Data centered on first edge of SCK period. 1: Data centered on second edge of SCK period.
SPI_CFG.4	CKPOL: SPI Clock Polarity. This bit controls the SPI clock polarity. 0: SCK line low in idle state. 1: SCK line high in idle state.
SPI_CFG.3	SLVSEL: Slave Selected Flag (read only).

	This bit is set to logic 1 whenever the NSS pin is low indicating SPI is the selected slave. It is cleared to logic 0 when NSS is high (slave not selected). This bit does not indicate the instantaneous value at the NSS pin, but rather a de-glitched version of the pin input.
SPI_CFG.2	NSSIN: NSS Instantaneous Pin Input (read only). This bit mimics the instantaneous value that is present on the NSS port pin at the time that the register is read. This input is not de-glitched.
SPI_CFG.1	WCOL_EN: WCOL enble bit 1：WCOL=1; enable WCOL interrupt 0：WCOL=0; disable WCOL interrupt. SPI only have three interrupt sources now.
SPI_CFG.0	RXBMT: Receive Buffer Empty (Valid in Slave Mode, read only). This bit will be set to logic 1 when the receive buffer has been read and contains no new information. If there is new information available in the receive buffer that has not been read, this bit will return to logic 0. NOTE: RXBMT = 1 when in Master Mode.

Table 25 SPI_CFG Register

13.7.2 SPI_CN: SPI Control Register

Bits	Definition
SPI_CN.7	SPIF: SPI Interrupt Flag. This bit is set to logic 1 by hardware at the end of a data transfer. If interrupts are enabled, setting this bit causes the CPU to vector to the SPI interrupt service routine. This bit is not automatically cleared by hardware. It must be cleared by software.
SPI_CN.6	WCOL: Write Collision Flag. This bit is set to logic 1 by hardware (and generates a SPI interrupt) to indicate a write to the SPI data register was attempted while a data transfer was in progress. It must be cleared by software.
SPI_CN.5	MODF: Mode Fault Flag. This bit is set to logic 1 by hardware (and generates a SPI interrupt) when a master mode collision is detected (NSS is low, MSTEN = 1, and NSSMD[1:0] = 01). This bit is not automatically cleared by hardware. It must be cleared by software.
SPI_CN.4	RXOVRN: Receive Overrun Flag (Slave Mode only). This bit is set to logic 1 by hardware (and generates a SPI interrupt) when the receive buffer still holds unread data from a previous transfer and the last bit of the current transfer is shifted into the SPI shift register. This bit is not automatically cleared by hardware. It must be cleared by software.
SPI_CN.3	NSSMD1-NSSMD0: Slave Select Mode.

SPI_CN.2	00: 3-Wire Slave or 3-wire Master Mode. NSS signal is not routed to a port pin. 01: 4-Wire Slave or Multi-Master Mode (Default). NSS is always an input to the device. 1x: 4-Wire Single-Master Mode. NSS signal is mapped as an output from the device and will assume the value of NSSMD0.
SPI_CN.1	TXBMT: Transmit Buffer Empty. This bit will be set to logic 0 when new data has been written to the transmit buffer. When data in the transmit buffer is transferred to the SPI shift register, this bit will be set to logic 1, indicating that it is safe to write a new byte to the transmit buffer.
SPI_CN.0	SPIEN: SPI Enable. This bit enables/disables the SPI. 0: SPI disabled. 1: SPI enabled.

Table 26 SPI_CN Register

13.7.3 SPI_CKR: SPI Clock Rate Register

These bits determine the frequency of the SCK output when the SPI module is configured for master mode operation. The SCK clock frequency is a divided version of the system clock, and is given in the following equation, where SYSCLK is the system clock frequency and SPICKR is the 8-bit value held in the SPICKR register.

$$f_{SCK} = \frac{SYSCLK}{2 \times (SPI\_CKR + 1)}$$

$$(0 \le SPI\_CKR \le 255)$$

Example：if {CKCON.7,CKCON.6}=3，SYSCLK=2M,and SPI_CKR=0x04；

then $f_{SCK} = 200kHz$。

13.7.4 SPI SPI_DAT: SPI Data Register

The SPIDAT register is used to transmit and receive SPI data. Writing data to SPIDAT places the data into the transmit buffer and initiates a transfer when in Master Mode. A read of SPIDAT returns the contents of the receive buffer.

14 SMBUS (I2C)

The SMBus I/O interface is a two-wire, bi-directional serial bus. The SMBus is compliant with the System Management Bus Specification, version 1.1, and compatible with the I C serial bus. Reads and writes to the interface by the system controller are byte oriented with the SMBus interface autonomously controlling the serial transfer of the data.

Data can be transferred at up to 1/10th of the system clock as a master or slave (this can be faster than allowed by the SMBus specification, depending on the system clock used). A method of extending the clock-low duration is available to accommodate devices with different speed capabilities on the same bus.

The SMBus interface may operate as a master and/or slave, and may function on a bus with multiple masters. The SMBus provides control of SDA (serial data), SCL (serial clock) generation and synchronization, arbitration logic, and START/STOP control and generation.

It is assumed the reader is familiar with the I2C-Bus Specification -- Version 2.0 and system Management Bus Specification -- Version 1.1.

The bi-directional SCL (serial clock) and SDA (serial data) lines must be connected to a positive power supply voltage through a pull-up resistor or similar circuit. Every device connected to the bus must have an open-drain or open-collector output for both the SCL and SDA lines, so that both are pulled high (recessive state) when the bus is free.

14.1 SMBUS Operation

Two types of data transfers are possible: data transfers from a master transmitter to an addressed slave receiver (WRITE), and data transfers from an addressed slave transmitter to a master receiver (READ). The master device initiates both types of data transfers and provides the serial clock pulses on SCL. The SMBus interface may operate as a master or a slave, and multiple master devices on the same bus are supported. If two or more masters attempt to initiate a data transfer simultaneously, an arbitration scheme is employed with a single master always winning the arbitration. Note that it is not necessary to specify one device as the Master in a system; any device who transmits a START and a slave address becomes the master for the duration of that transfer.

A typical SMBus transaction consists of a START condition followed by an address byte (Bits7-1: 7-bit slave address; Bit0: R/W direction bit), one or more bytes of data, and a STOP condition. Each byte that is received (by a master or slave) must be acknowledged (ACK) with a low SDA during a high SCL (see Figure 28). If the receiving device does not ACK, the transmitting device will read a NACK (not acknowledge), which is a high SDA during a high SCL.

The direction bit (R/W) occupies the least-significant bit position of the address byte. The direction bit is set to logic 1 to indicate a "READ" operation and cleared to logic 0 to indicate a "WRITE" operation.

All transactions are initiated by a master, with one or more addressed slave devices as the target. The master generates the START condition and then transmits the slave address and direction bit. If the transaction is a WRITE operation from the master to the slave, the master transmits the data a byte at a time waiting for an ACK from the slave at

the end of each byte. For READ operations, the slave transmits the data waiting for an ACK from the master at the end of each byte. At the end of the data transfer, the master generates a STOP condition to terminate the transaction and free the bus. The next figure illustrates a typical SMBus transaction.

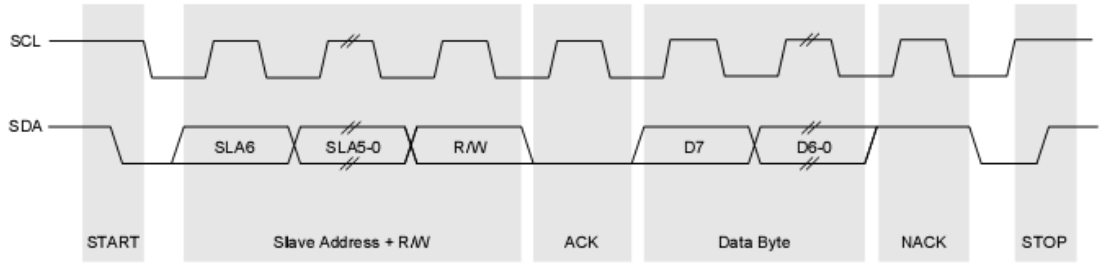


Figure 28 SMBUS Transaction

14.1.1 Clock Low Extension

A clock-low extension is used during a transfer in order to allow slower slave devices to communicate with faster masters. The slave may temporarily hold the SCL line LOW to extend the clock low period, effectively decreasing the serial clock frequency.

14.1.2 Bus Arbitration

A master may start a transfer only if the bus is free. The bus is free after a STOP condition or after the SCL and SDA lines remain high for a specified time. In the event that two or more devices attempt to begin a transfer at the same time, an arbitration scheme is employed to force one master to give up the bus. The master devices continue transmitting until one attempts a HIGH while the other transmits a LOW. Since the bus is open-drain, the bus will be pulled LOW. The master attempting the HIGH will detect a LOW SDA and lose the arbitration. The winning master continues its transmission without interruption; the losing master becomes a slave and receives the rest of the transfer if addressed. This arbitration scheme is nondestructive: one device always wins, and no data is lost.

14.1.3 SCL Low Timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than 25 ms as a “timeout” condition. Devices that have detected the timeout condition must reset the communication no later than 10 ms after detecting the timeout condition.

When the SMBTOE bit in SMBCF is set, Timer 3 is used to detect SCL low timeouts. Timer 3 is forced to reload when SCL is high, and allowed to count when SCL is low. With Timer 3 enabled and configured to overflow after 25 ms (and SMBTOE set),

the Timer 3 interrupt service routine can be used to reset (disable and re-enable) the SMBus in the event of an SCL low timeout.

14.1.4 SMBS Fress

The SMBus specification stipulates that if the SCL and SDA lines remain high for more that 50 μs, the bus is designated as free. When the SMBFTE bit in SMBCF is set, the bus will be considered free if SCL and SDA remain high for more than 2^{IDLE_CR} SMBus clock source periods. If the SMBus is waiting to generate a Master START, the START will be generated following this timeout. Note that a clock source is required for free timeout detection, even in a slave-only implementation.

14.2 Using SMBus

14.2.1 SMBus Configuration Register

SMB_CF

Bits	Definition
SMB_CF.7	ENSMB: SMBus Enable. This bit enables/disables the SMBus interface. When enabled, the interface constantly monitors the SDA and SCL pins. 0: SMBus interface disabled. 1: SMBus interface enabled.
SMB_CF.6	INH: SMBus Slave Inhibit. When this bit is set to logic 1, the SMBus does not generate an interrupt when slave events occur. This effectively removes the SMBus slave from the bus. Master Mode interrupts are not affected. 0: SMBus Slave Mode enabled. 1: SMBus Slave Mode inhibited.
SMB_CF.5	BUSY: SMBus Busy Indicator. This bit is set to logic 1 by hardware when a transfer is in progress. It is cleared to logic 0 when a STOP or free-timeout is sensed.
SMB_CF.4	EXTHOLD: SMBus Setup and Hold Time Extension Enable. This bit controls the SDA setup and hold times according to. 0: SDA Extended Setup and Hold Times disabled. 1: SDA Extended Setup and Hold Times enabled.
SMB_CF.3	SMBTOE: SMBus SCL Timeout Detection Enable. This bit enables SCL low timeout detection. If it set to logic 1, SMB will detect timeout of SCL. It would generate interrupt when timeout happened.
SMB_CF.2	SMBFTE: SMBus Free Timeout Detection Enable. When this bit is set to logic 1, the bus will be considered free if SCL and SDA remain high for more than 2^{IDLE_CR} SMBus clock source periods.
SMB_CF.1 SMB_CF.0	SMBCS1-SMBCS0: SMBus Clock Source Selection. These two bits select the SMBus clock source, which is used to generate

	the SMBus bit rate. SMBCS1 SMBCS0 SMBUS clock source 0 0 Timer 0 overflow 0 1 Timer 1 overflow 1 x Timer 2 overflow

Talbe 27 SMBus Config Register

The SMBus Configuration register (SMBCF) is used to enable the SMBus Master and/or Slave modes, select the SMBus clock source, and select the SMBus timing and timeout options. When the ENSMB bit is set, the SMBus is enabled for all master and slave events. Slave events may be disabled by setting the INH bit. With slave events inhibited, the SMBus interface will still monitor the SCL and SDA pins; however, the interface will NACK all received addresses and will not generate any slave interrupts. When the INH bit is set, all slave events will be inhibited following the next START (interrupts will continue for the duration of the current transfer).

The SMBCS1-0 bits select the SMBus clock source, which is used only when operating as a master or when the free timeout detection is enabled. When operating as a master, overflows from the selected source determine the absolute minimum SCL low and high times as defined in following Equation. Note that the selected clock source may be shared by other peripherals so long as the timer is left running at all times. For example, Timer 1 overflows may generate the SMBus and UART baud rates simultaneously.

$$T_{high\min} = T_{low\min} = \frac{1}{f_{timeroverflowrate}}$$

$T_{high\min}$: the minimum high voltage time of SCL;

$T_{low\min}$: the minimum low voltage time of SCL;

$f_{timeroverflowrate}$:the clock source of SMBUS. (timer overfolow rate)

The selected clock source should be configured to establish the minimum SCL High and Low times as per Equation. When the interface is operating as a master (and SCL is not driven or extended by any other devices on the bus), the typical SMBus bit rate is approximated by next Equation.

$$bitrate = \frac{f_{timeroverflowrate}}{3};$$

Notice that T_{high} is typically twice as large as T_{low}. The actual SCL output may vary due to other devices on the bus (SCL may be extended low by slower slave devices, or driven low by contending master devices).

Setting the EXTHOLD bit extends the minimum setup and hold times for the SDA line. The minimum SDA setup time defines the absolute minimum time that SDA is stable before SCL transitions from low-to-high. The minimum SDA hold time defines the absolute minimum time that the current SDA value remains stable after SCL transitions from high-to-low. EXTHOLD should be set so that the minimum setup and hold times meet the SMBus Specification requirements of 250 ns and 300 ns, respectively. Setup and hold time extensions are typically necessary when SYSCLK is above 10 MHz.

With the SMBTOE bit set, Timer 3 should be configured to overflow after 25 ms in order to detect SCL low timeouts. The SMBus interface will force Timer 3 to reload while SCL is high, and allow Timer 3 to count when SCL is low. The Timer 3 interrupt service routine should be used to reset SMBus communication by disabling and re-enabling the SMBus.

SMBus Free Timeout detection can be enabled by setting the SMBFTE bit. When this bit is set, the bus will be considered free if SDA and SCL remain high for more than 10 SMBus clock source periods. When a Free Timeout is detected, the interface will respond as if a STOP was detected (an interrupt will be generated, and STO will be set).

14.2.2 SMBus Control Register

SMBCN is used to control the interface and to provide status information. The higher four bits of SMBCN (MASTER, TXMODE, STA, and STO) form a status vector that can be used to jump to service routines. MASTER and TXMODE indicate the master/slave state and transmit/receive modes, respectively.

STA and STO indicate that a START and/or STOP has been detected or generated since the last SMBus interrupt. STA and STO are also used to generate START and STOP conditions when operating as a master. Writing a '1' to STA will cause the SMBus interface to enter Master Mode and generate a START when the bus becomes free (STA is not cleared by hardware after the START is generated). Writing a '1' to STO while in Master Mode will cause the interface to generate a STOP and end the current transfer after the next ACK cycle. If STO and STA are both set (while in Master Mode), a STOP followed by a START will be generated.

As a receiver, writing the ACK bit defines the outgoing ACK value; as a transmitter, reading the ACK bit indicates the value received on the last ACK cycle. ACKRQ is set each time a byte is received, indicating that an outgoing ACK value is needed. When ACKRQ is set, software should write the desired outgoing value to the ACK bit before clearing SI. A NACK will be generated if software does not write the ACK bit before clearing SI. SDA will reflect the defined ACK value immediately following a write to the ACK bit; however SCL will remain low until SI is cleared. If a received slave address is not acknowledged, further slave events will be ignored until the next START is detected.

The SI bit (SMBus Interrupt Flag) is set at the beginning and end of each transfer, after each byte frame, or when an arbitration is lost.

Pls Note About the SI Bit: The SMBus interface is stalled while SI is set; thus SCL is held low, and the bus is stalled until software clears SI.

The ARBLOST bit indicates that the interface has lost an arbitration. This may occur anytime the interface is transmitting (master or slave). A lost arbitration while operating as a slave indicates a bus error condition. ARBLOST is cleared by hardware each time SI is cleared.

Bits	Definition
SMB_CN.7	MASTER: SMBus Master/Slave Indicator. This read-only bit indicates when the SMBus is operating as a master. 0: SMBus operating in Slave Mode. 1: SMBus operating in Master Mode.

SMB_CN.6	TXMODE: SMBus Transmit Mode Indicator. This read-only bit indicates when the SMBus is operating as a transmitter. 0: SMBus in Receiver Mode. 1: SMBus in Transmitter Mode.
SMB_CN.5	STA: SMBus Start Flag. Write: 0: No Start generated. 1: When operating as a master, a START condition is transmitted if the bus is free (If the bus is not free, the START is transmitted after a STOP is received or a timeout is detected). If STA is set by software as an active Master, a repeated START will be generated after the next ACK cycle. Read: 0: No Start or repeated Start detected. 1: Start or repeated Start detected.
SMB_CN.4	STO: SMBus Stop Flag. Write: 0: No STOP condition is transmitted. 1: Setting STO to logic 1 causes a STOP condition to be transmitted after the next ACK cycle. When the STOP condition is generated, hardware clears STO to logic 0. If both STA and STO are set, a STOP condition is transmitted followed by a START condition. Read: 0: No Stop condition detected. 1: Stop condition detected (if in Slave Mode) or pending (if in Master Mode).
SMB_CN.3	ACKRQ: SMBus Acknowledge Request This read-only bit is set to logic 1 when the SMBus has received a byte and needs the ACK bit to be written with the correct ACK response value.
SMB_CN.2	ARBLOST: SMBus Arbitration Lost Indicator. This read-only bit is set to logic 1 when the SMBus loses arbitration while operating as a transmitter. A lost arbitration while a slave indicates a bus error condition.
SMB_CN.1	ACK: SMBus Acknowledge Flag. This bit defines the out-going ACK level and records incoming ACK levels. It should be written each time a byte is received (when ACKRQ=1), or read after each byte is transmitted. 0: A "not acknowledge" has been received (if in Transmitter Mode) OR will be transmitted (if in Receiver Mode). 1: An "acknowledge" has been received (if in Transmitter Mode) OR will be transmitted (if in Receiver Mode).

SMB_CN.0	SI: SMBus Interrupt Flag. This bit is set by hardware under the conditions listed in Table 29. SI must be cleared by software. While SI is set, SCL is held low and the SMBus is stalled.

Table 28 SMBusControl Register

Next table descript the hardware sources for changes to SMBCN.

Bits	Set by Hardware When:	Cleared by Hardware When:
MASTER	-A START is generated.	-A STOP is generated. -Arbitration is lost
TXMODE	-START is generated. -SMBDAT is written before the start of an SMBus frame.	-A START is detected. -Arbitartion is lost. - SMBDAT is not written before the start of a SMBus frame.
STA	-A START followed by an address byte is received.	-Must be cleared by software.
STO	-A STOP is detected while addressed as a slave. -Arbitration is lost due to a detected STOP.	-A pending STOP is generated.
ACKRQ	-A byte has been received and an ACK response value is needed.	-After each ACK cycle.
ARBLOST	-A repeated START is detected as a MASTER when STA is low (unwanted repeated START). -SCL is sensed low while attempting to generate a STOP or repeated START condition. -SDA is sensed low while transmitting a '1' (excluding ACK bits).	- Each time SI is cleared.
ACK	-The incoming ACK value is low (ACKNOWLEDGE).	-The incoming ACK value is high (NOT ACKNOWLEDGE).
SI	- A START has been generated. - Lost arbitration. -A byte has been transmitted and an ACK/NACK received. - A byte has been received. - A START or repeated START followed by a slave address + R/W has been received. -A STOP has been received.	- Must be cleared by software.

Table 29 Sources for Hardware Changes to SMBCN

14.2.3 SMB_TMCTL Timeout Detect Control Register

SMB_TMCTL register used to detect IDLE state on bus, also, it detect the error condition for SCL=0 lasted more than 25ms.

SMB_TMCTL	7	6	5	4	3	2	1	0
function	SCL_CR[2:0]			IDLE_CR[2:0]			-	SCL_TMOT

Table 30 SMB_TMCTL Time out Detect Register

SCL_CR: When SMBTOE=1, enable the function for detecting SCL=0. If SCL=0 lasted more than 2^{SCL_CR} SMBUS clock period (the timer overflow rate), SCL_TMOT will be set to trigger interrupt. For SCL_TMOT and SI use the same interrupt line int3_n, so the software should query which interrupt source when interrupt happened.

SMBUS protocol stipulates the time is 25ms. It is adviced that disable SMBUS first and then enable SMBUS When interrupt happened to reset the interrupt. The default value of SCL_CR is 7, that is say that if SCL=0 hold for 128 SMBUS clock period will trigger the interrupt.

IDLE_CR[2:0]: When SMBFTE=1, enable the function for detecting the period of SCL=1 and SDA=1. If SCL=1 and SDA=1 lasted more than 2^{SCL_CR} SMBUS clock period (the timer overflow rate), the internal state would be forced to IDLE state. The default value of IDLE _CR is 3, it means that after 8 SMBUS clock period the state of SMBUS will enter IDLE state.

SCL_TMOT: SCL=0 timeout will set this bit to trigger interrupt. This bit can be writen and read by software.

14.2.4 SMB_DAT Data Register

The SMBus Data register SMBDAT holds a byte of serial data to be transmitted or one that has just been received. Software may safely read or write to the data register when the SI flag is set. Software should not attempt to access the SMBDAT register when the SMBus is enabled and the SI flag is cleared to logic 0, as the interface may be in the process of shifting a byte of data into or out of the register.

Data in SMBDAT is always shifted out MSB first. After a byte has been received, the first bit of received data is located at the MSB of SMBDAT. While data is being shifted out, data on the bus is simultaneously being shifted in. SMBDAT always contains the last data byte present on the bus. In the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data or address in SMBDAT.

14.3 SMBUS Transfer Mode

The SMBus interface may be configured to operate as master and/or slave. At any particular time, it will be operating in one of the following four modes: Master Transmitter, Master Receiver, Slave Transmitter, or Slave Receiver. The SMBus

interface enters Master Mode any time a START is generated, and remains in Master Mode until it loses an arbitration or generates a STOP. An SMBus interrupt is generated at the end of all SMBus byte frames; however, note that the interrupt is generated before the ACK cycle when operating as a receiver, and after the ACK cycle when operating as a transmitter.

14.3.1 Master Transmitter Mode

Serial data is transmitted on SDA while the serial clock is output on SCL. The SMBus interface generates the START condition and transmits the first byte containing the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 0 (WRITE). The master then transmits one or more bytes of serial data. After each byte is transmitted, an acknowledge bit is generated by the slave. The transfer is ended when the STO bit is set and a STOP is generated. Note that the interface will switch to Master Receiver Mode if SMB0DAT is not writ-ten following a Master Transmitter interrupt. Figure 16.8 shows a typical Master Transmitter sequence. Two transmit data bytes are shown, though any number of bytes may be transmitted. Notice that the 'data byte transferred' interrupts occur after the ACK cycle in this mode.

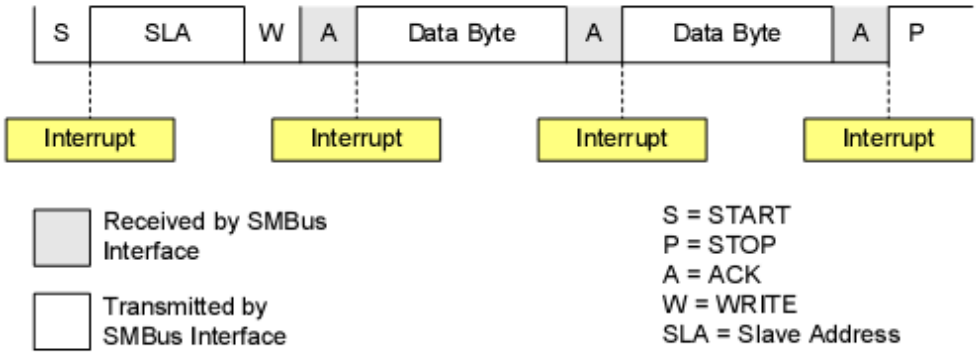


Figure 29 SMBUS Master Transmitter Sequence

14.3.2 Master Receiver Mode

Serial data is received on SDA while the serial clock is output on SCL. The SMBus interface generates the START condition and transmits the first byte containing the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 1 (READ). Serial data is then received from the slave on SDA while the SMBus outputs the serial clock. The slave transmits one or more bytes of serial data. After each byte is received, ACKRQ is set to '1' and an interrupt is generated. Software must write the ACK bit (SMB0CN.1) to define the out-going acknowledge value (Note: writing a '1' to the ACK bit generates an ACK; writing a '0' generates a NACK). Software should write a '0' to the ACK bit after the last byte is received, to transmit a NACK. The interface exits Master Receiver Mode after the STO bit is set and a STOP is generated. Note that the interface will switch to Master Transmitter Mode if SMB0DAT

is written while an active Master Receiver. Figure 16.9 shows a typical Master Receiver sequence. Two received data bytes are shown, though any number of bytes may be received. Notice that the ‘data byte transferred’ interrupts occur before the ACK cycle in this mode.

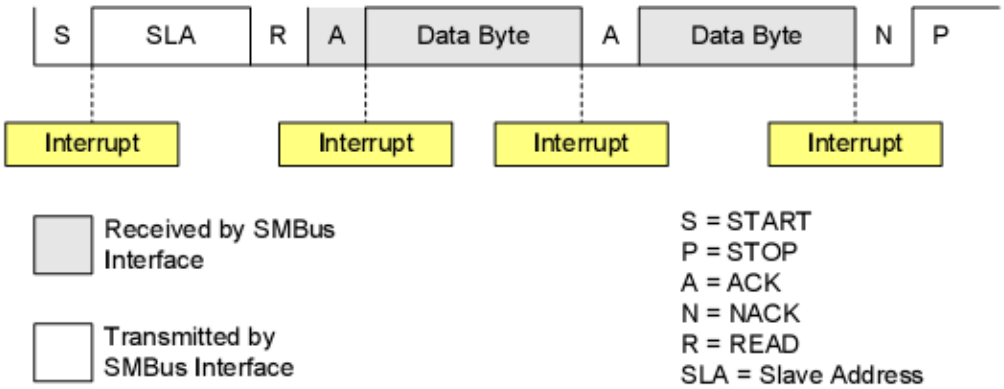


Figure 30 SMBUS Master Receiver Sequence

14.3.3 Slave Transmitter Mode

Serial data is transmitted on SDA and the clock is received on SCL. When slave events are enabled (INH = 0), the interface enters Slave Receiver Mode (to receive the slave address) when a START followed by a slave address and direction bit (READ in this case) is received. Upon entering Slave Transmitter Mode, an interrupt is generated and the ACKRQ bit is set. Software responds to the received slave address with an ACK, or ignores the received slave address with a NACK. If the received slave address is ignored, slave interrupts will be inhibited until a START is detected. If the received slave address is acknowledged, data should be written to SMB0DAT to be transmitted. The interface enters Slave Transmitter Mode, and transmits one or more bytes of data. After each byte is transmitted, the master sends an acknowledge bit; if the acknowledge bit is an ACK, SMB0DAT should be written with the next data byte. If the acknowledge bit is a NACK, SMB0DAT should not be written to before SI is cleared (Note: an error condition may be generated if SMB0DAT is written following a received NACK while in Slave Transmitter Mode). The interface exits Slave Transmitter Mode after receiving a STOP. Note that the interface will switch to Slave Receiver Mode if SMB0DAT is not written following a Slave Transmitter interrupt. The next figure shows a typical Slave Transmitter sequence. Two transmitted data bytes are shown, though any number of bytes may be transmitted. Notice that the ‘data byte transferred’ interrupts occur after the ACK cycle in this mode.

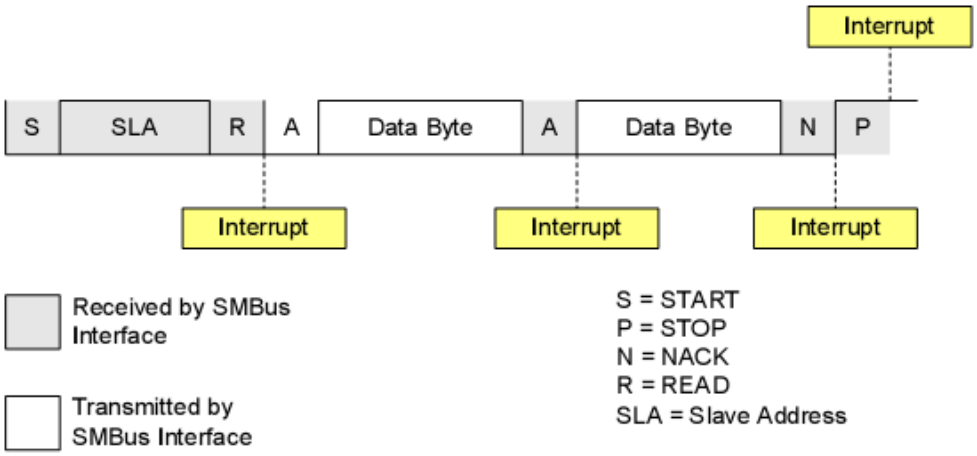


Figure 31 SMBUS Slave Transmitter Sequence

14.3.4 Slave Receiver Mode

Serial data is received on SDA and the clock is received on SCL. When slave events are enabled (INH = 0), the interface enters Slave Receiver Mode when a START followed by a slave address and direction bit (WRITE in this case) is received. Upon entering Slave Receiver Mode, an interrupt is generated and the ACKRQ bit is set. Software responds to the received slave address with an ACK, or ignores the received slave address with a NACK. If the received slave address is ignored, slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, zero or more data bytes are received. Software must write the ACK bit after each received byte to ACK or NACK the received byte. The interface exits Slave Receiver Mode after receiving a STOP. Note that the interface will switch to Slave Transmitter Mode if SMB0DAT is written while an active Slave Receiver. The next figure shows a typical Slave Receiver sequence. Two received data bytes are shown, though any number of bytes may be received. Notice that the 'data byte transferred' interrupts occur before the ACK cycle in this mode.

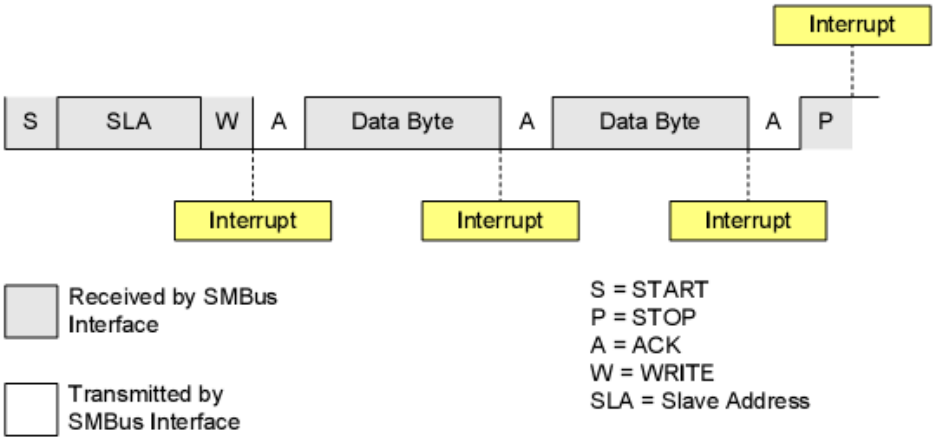


Figure 32 SMBUS Slave Receiver Sequence

14.4 SMBUS Status Decoding

The current SMBus status can be easily decoded using the SMB0CN register. In the table below, STATUS VECTOR refers to the four upper bits of SMB0CN: MASTER, TXMODE, STA, and STO. Note that the shown response options are only the typical responses; application-specific procedures are allowed as long as they conform to the SMBus specification. Highlighted responses are allowed but do not conform to the SMBus specification.

MODE	VALUES READ				CURRENT SMBUS STATE	TYPICAL RESPONSE OPTIONS	VALUES WRITTEN		
	STATUS VECTOR	ACKRQ	ARBLOST	ACK			STA	STO	ACK
Master Transmitter	1110	0	0	X	A master START was generated.	Load slave address + R/W into SMB0DAT.	0	0	X
	1100	0	0	0	A master data or address byte was transmitted; NACK received.	Set STA to restart transfer.	1	0	X
						Abort transfer.	0	1	X
		0	0	1	A master data or address byte was transmitted; ACK received.	Load next data byte into SMB0DAT.	0	0	X
						End transfer with STOP.	0	1	X
						End transfer with STOP and start another transfer.	1	1	X
						Send repeated START.	1	0	X
						Switch to Master Receiver Mode (clear SI without writing new data to SMB0DAT).	0	0	X
Master Receiver	1000	1	0	X	A master data byte was received; ACK requested.	Acknowledge received byte; Read SMB0DAT.	0	0	1
						Send NACK to indicate last byte, and send STOP.	0	1	0
						Send NACK to indicate last byte, and send STOP followed by START.	1	1	0
						Send ACK followed by repeated START.	1	0	1
						Send NACK to indicate last byte, and send repeated START.	1	0	0
						Send ACK and switch to Master Transmitter Mode (write to SMB0DAT before clearing SI).	0	0	1
						Send NACK and switch to Master Transmitter Mode (write to SMB0DAT before clearing SI).	0	0	0

MODE	VALUES READ				CURRENT SMBUS STATE	TYPICAL RESPONSE OPTIONS	VALUES WRITTEN		
	STATUS VECTOR	ACKRQ	ARBLOST	ACK			STA	STO	ACK
Slave Transmitter	0100	0	0	0	A slave byte was transmitted; NACK received.	No action required (expecting STOP condition).	0	0	X
		0	0	1	A slave byte was transmitted; ACK received.	Load SMB0DAT with next data byte to transmit.	0	0	X
		0	1	X	A Slave byte was transmitted; error detected.	No action required (expecting Master to end transfer).	0	0	X
	0101	0	X	X	A STOP was detected while an addressed Slave Transmitter.	No action required (transfer complete).	0	0	X
Slave Receiver	0010	1	0	X	A slave address was received; ACK requested.	Acknowledge received address.	0	0	1
						Do not acknowledge received address.	0	0	0
		1	1	X	Lost arbitration as master; slave address received; ACK requested.	Acknowledge received address.	0	0	1
						Do not acknowledge received address.	0	0	0
						Reschedule failed transfer; do not acknowledge received address.	1	0	0
	0010	0	1	X	Lost arbitration while attempting a repeated START.	Abort failed transfer.	0	0	X
						Reschedule failed transfer.	1	0	X
	0001	1	1	X	Lost arbitration while attempting a STOP.	No action required (transfer complete/aborted).	0	0	0
		0	0	X	A STOP was detected while an addressed slave receiver.	No action required (transfer complete).	0	0	X
		0	1	X	Lost arbitration due to a detected STOP.	Abort transfer.	0	0	X
						Reschedule failed transfer.	1	0	X
	0000	1	0	X	A slave byte was received; ACK requested.	Acknowledge received byte; Read SMB0DAT.	0	0	1
						Do not acknowledge received byte.	0	0	0
		1	1	X	Lost arbitration while transmitting a data byte as master.	Abort failed transfer.	0	0	0
						Reschedule failed transfer.	1	0	0

Table 31 SMBUS Status Decoding

15 ADC

A 10bits SAR ADC is integrated in BK51. Total 8 channels can be selected used for ADC transfer. The ADC supports continue mode and single transfer mode, and the sample rate can be 1kHz to 29kHz. In single transfer mode, it will generate interrupt every time after tansform. The input of ADC is share with P3 general I/O port.

In single transfer mode, the time used to convert is very little.

Convert time < 40us(Idle → Convert Done),

Convert time < 50us(Power Down→ Convert Done)

Low power: current=0.2mA at 16k sample rate

Mode	Chann	Intr_ En	Chann _en	Ready	Setti ng	Rate	Data[9:8]	Data[7:0]
0x96(ADC_CTL)					0x95(ADC_DATH)			0x94(ADC_ DATL)
7:6	5:3	2	1	0	7	6:2	1:0	7:0
Ctrl_Reg							**Data_High**	**Data_Low**

Table 32 ADC 的 SFR

The ADC is controlled by there SFR control的SFR as showed in above figure：

Data[9:0] ---- the result for ADC

Rate[4:0]----sample rate(1K~29K)。

Setting --- control the delay time until start to sample the analog input when the mode set to 1. When seting=1,the time is twice as setting=0.

Ready ---- the status of ADC. 0: transfer is done now. It will trigger interrupt if enabled. Ready=1, transfer is processing.

Chann_en--- =1,enable ADC function on P3 port. The p3 port is general I/O if clear it.

Intr_en---- interrupt enable or not when transfer is done.

Chann--- channel select.

Mode[1:0]----ADC mode

Sleep mode（mode==00）： ADC is power down now

Single mode（mode==01）：The system will enter sleep mode when transfer is done and waiting MCU to read the result. You should write mode=1 again for another transfer.

Controlled by software（mode==10）：In this mode, interrupt will be triggered after transfer and wait MCU to read. The interrupt will be cleared after MCU reading, then the transfer will start again.

Continue mode（mode==11）：the ADC will work at the sample rate set by register.

Sample rate:

F_sample = (Rate[4:0]+1)*1000。The highest sample rate is 29k。

For example ,if Rate=7，**F_sample** =8K。

Note 1: In continue mode, the sample rate is fixed in spite of read or not by MCU. In software mode, ADC will enter waiting state until the result is read by MCU

The interrupt of ADC can not be R/W by software; it will be set after tranform and be cleared after the result stored in DATA_HIGH is read out.

Note 2: The eight ADC can be used as voltage monitor. When used as this:

Voltage=data[9:0]/448 (the saturate voltage is 2.3v.)

16 LBD

The LBD circuit is used to monitor power supply.

SFR	7-2	1	0
0xD8	reserved (dont change)	LBD_thd[4]	reserved (dont change)

Table 33 LBD register 1

SFR	7 - 4	3-0
0xD9	reserved (dont change)	LBD_thd[3:0]

Table 34 LBD register 2

SFR	7	6	5-0
0xDA	reserved (dont change)	LBD power on/off	reserved (dont change)

Table 35 LBD register 3

LBD_thd	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
voltage	0.75	0.83	0.89	0.96	1.03	1.1	1.17	1.24	1.31	1.38	1.45	1.52	1.59	1.66	1.73	1.8

LBD_thd	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
voltage	1.87	1.94	2.01	2.08	2.15	2.22	2.29	2.36	2.43	2.5	2.57	2.64	2.71	2.78	2.85	2.92

Table 36 LBD votage table

The recommend detect process is described as follow:

1. Config the LBD_THD register correctly according to **Table 36**
2. Power up LBD circuit (write 1 into bit 6 of adress 0xDA.)
3. Wait about 50us
4. Clear register EICON.4
5. Read register EICON.4

   If EICON.4=1 it means that that POWER FAIL happened. (LBD voltage is less than the LBD_THD votage now)
6. Power down LBD circuit. (write 0 into bit 6 of adress 0xDA.)

17 PWM

The BK51 includes a two channel Pulse-Width Modulation (PWM) module. They are controlled by the next SFR.

PWM0C0	PWM0_PERIOD [7:0]							
PWM0C1	PWM0_DUTY [5:0]						PWM0_ PERIOD [9:8]	
PWM0C2	PWM0_ EN	PWM0_PRESCAL[2:0]			PWM0_DUTY [9:6]			
PWM1C0	PWM1_PERIOD [7:0]							
PWM1C1	PWM1_DUTY [5:0]						PWM1_ PERIOD [9:8]	
PWM1C2	PWM1_ EN	PWM1_PRESCAL[2:0]			PWM1_DUTY [9:6]			
PWMICTL	pwm1_ inte	pwm0_ inte	pwm1_ mod	pwm0_ mod			pwm1_ intf	pwm0_ intf

Table 37 PWM Module

PWM0C0：The lower 8bit of PWM0 period
PWM0_PERIOD [9:8]：The upper 2bit of PWM0 period
PWM0_DUTY [5:0]：The lower 6bit of PWM0 duty
PWM0_DUTY [9:6]：The upper 4bit of PWM0 duty
PWM0_EN：PWM0 enable bit
PWM0_PRESCAL[2:0]：PWM0 clock prescale number
PWM0 clk=(MCU clk)/(PWM0_PRESCAL＋1)
PWM1C0：The lower 8bit of PWM1 period
PWM1_PERIOD [9:8]：The upper 2bit of PWM1 period
PWM1_DUTY [5:0]：The lower 6bit of PWM1 duty
PWM1_DUTY [9:6]：The upper 4bit of PWM1 duty
PWM1_EN：PWM0 enable bit
PWM1_PRESCAL[2:0]：PWM1 clock prescale number
PWM1 clk=(MCU clk)/(PWM1_PRESCAL＋1)
PWM0_INTF: PWM0 interrupt bit
PWM1_INTF: PWM1 interrupt bit
PWM0_mode: PWM0 mode select bit
When PWM0_IE=1 and PWM0_mode=1, PWM0 trigger interrupt at the end of every period. When PWM0_IE=1 and PWM0_mode=0, PWM0 trigger interrupt at the end of every duty.
PWM1_mode: be same as PWM0_mode
PWM0_IE：Mask PWM0 interrupt or not
PWM1_IE：Mask PWM1 interrupt or not

NOTE: The counter number counts from zero to PWM_PERIOD, so the PWM period will be PWM_PERIOD+1. It is same to the PWM_DUTY.

18 Random Number Generator

There are two random number generators in BK2433. One is pseudo random number generator and the other is a true random number generator (RNG), which uses thermal noise to produce a non-deterministic bit stream. The bits are then queued into an 8-bit register for parallel readout. You can use either of them as you selected.

The RNG is interfaced through two registers; RNG_CTL and RNGDAT. RNG_CTL contains two control bits and a status bit. RNGDAT contains the random data.

RNG_CTL	7	6	5	4-0
0xEC	RNG_pwd (RW)		RNG_rdy (R) Used by true RNG only	/

RNG_pwd: write 0 to power up the true RNG module, 1 to power off the true RNG and use the pseudo RNG module.

RNG_rdy: read only register, when RNGDAT is ready RNG_rdy is set to1 by hardware, when software read RNGDAT, it's clear by hardware. Used by true RNG only.

RNG_DAT	7-0
0xEB	Data/seeds

RNG_DAT: RNG_rdy=1 to indicate the true random number (RNG_DAT) is ready for read.

When use pseudo random generator, RNG_DAT is always ready in the register, you can read the result at any time.

Also, you can change the seed by writing to SFR 0xEB. A different seed can be stored in the MTP in advance to get a different random number with different chip. For software, the seed should be read out from MTP and write to the SFR when power on.

When use the true random generator, it will take about 50us to prepare the first random number after you power up the module. After that, it will take about 2us to generate a new 8-bits random number.

19 MDU Multiply Divide Unit

The MDU – Multiplication Division Unit, is an on-chip arithmetic co-processor which enables the MCU to perform additional extended arithmetic operations like 32-bit division, 16-bit multiplication, shift and, normalize operations.

MDU support unsigned integer only. The MDU is handled by seven registers, which are memory mapped as Special Function Registers. The arithmetic unit allows concurrent operations to be performed independent of the MCU's activity.

Operands and results are stored in MD0.. MD5 registers. The module is controlled by the MDCTL register. Any calculation of the MDU overwrites its operands.

The MDU does not allow reentrant code and cannot be used in multiple threads of the main and interrupt routines at the same time.

Address	SFR
0xF1	MD0
0xF2	MD1
0xF3	MD1
0xF4	MD1
0xF5	MD1
0xF6	MD1
0xF7	MDCTL

Table 38 MDU SFR

MDCTL（R/W by software）MDCTL meaning is different when reading and writing

reading:

MDCON0	7	6	5	4-0
function	mdef	mdov	done	NORM_shift_number

Table 39 MDU Register (Read)

mdef : MDU Error flag MDEF. Indicates an improperly performed operation (when one of the arithmetic operations has been restarted or interrupted by a new operation)

mdov : MDU Overflow flag MDOV. Overflow occurrence in the MDU operation.

Done : Calculate is done or not. 1: operate is done now; 0: busy

NORM_shift_number: left shift number stored in it when NORM is done.

writing:

MDCON1	7	6	5	4	3	2	1	0
function	opcode			SC				

Table 40 MDU Register(Write)

SC: Shift counter.
opcode: operate code which is show as next table.

opcode	meaning
1	**32 bit / 16 bit divide**
2	**16bit * 16bit multiply**
3	**16 bit /16 bit divide**
4	**Left shift**
5	**Right shift**
6	**normalizing**
others	**idle**

Table 41 MDUo perationTable

Operate data

OPERATION	32bit / 16bit		16bit / 16bit		16bit x 16bit		shift or normalize	
DATA	MD0(LSB) MD1 MD2 MD3(MSB)	dividend	MD0(LSB) MD1(MSB)	dividend	MD0(LSB) MD1(MSB)	dividend	MD0(LSB) MD1 MD2 MD3(MSB)	Op-code
DATA	MD4(LSB) MD5(MSB)	divisor	MD4(LSB) MD5(MSB)	divisor	MD4(LSB) MD5(MSB)	divisor		

Table 42 Operation Data

Reading result

OPERATION	32bit / 16bit		16bit / 16bit		16bit x 16bit		shift or normalize	
DATA	MD0(LSB) MD1 MD2 MD3(MSB)	quotient	MD0(LSB) MD1(MSB)	quotient	MD0(LSB) MD1 MD2 MD3(MSB)	product	MD0(LSB) MD1 MD2 MD3(MSB)	result
DATA	MD4(LSB) MD5(MSB)	remainder	MD4(LSB) MD5(MSB)	remainder				

Table 43 Operate Result

Normalizing

When set opcode = 6, normalizing sart to run. All leading zeroes of 32-bit integer variable stored in the MD0.. MD3 registers are removed by shift left operations. The whole operation is completed when the MSB (Most Significant Bit) of MD3 register contains a ’ 1’. After normalizing, bits NORM_shift_number contain the number of shift left operations that were done.

Example:

Run code：

```
Mov  MD3 , #00001101b;
Mov  MD2 , #00000001b;
Mov  MD1 , #00000011b;
Mov  MD0 , #00000111b;
mov MDCTL #11000000b  ;  //start normalizing
```

after 6 clock period，we can read,

```
NORM_shift_number =4;
MD3 , #11010000b;   // left shift four bit until the MSB of MD3is 1
MD2 , #00010000b;   // left shift four bit
MD1 , #00110000b;   // left shift four bit
MD0 , #01110000b; //   left shift four bit。
```

Shift operation

n shift operation, 32-bit integer variable stored in the MD0... MD3 registers (the latter contains the most significant byte) is shifted left or right by a specified number of bits. The opcode defines the shift direction and the shift count. During shift operation, zeroes come into the left end of MD3 for shifting right or they come in the right end of the MD0 for shifting left.

Mdef error flag

The mdef error flag indicates an improperly performed operation (when one of the arithmetic operations is restarted or interrupted by a new operation).

The error flag is set when:

* If you write to MD0.. MD5 and/or opcode during phase two of MDU operation (restart or calculations interrupting).

* If any of the MDx registers are read during phase two of MDU operation when the error flag mechanism is enabled. In this case, the error flag is set but the calculation is not interrupted.

Mdef will be set when error happened and be cleared when new operation started.

Mdov MDU－overflow flag

This bit is set by hardware and cleared by software.

The mdov overflow flag is set when one of the following conditions occurs:

* Division by zero

* Multiplication with a result greater than 0000 FFFFh
* start of normalizing if the most significant bit of MD3 is set ("md3.7" = '1').

Note: any new operation will clear this bit.

Executing calculation

During executing operation, the MDU works on its own in parallel with the MCU.

operation	Number of clock cycles
32bit / 16bit	9 clock cycles (max and min)
16bit / 16bit	9 clock cycles (max and min)
mutiplication	9 clock cycles (max and min)
shift	7 clock cycles (max and min)
normalize	7 clock cycles (max and min)

Table 44 MDU operations execution times

Note： The clock cycle is CPU clock cycle

20 USB1.1

The USB module in BK2433 provides a full speed USB function interface that meets the 1.1 and 2.0 specification. USB module has 8 endpoints and the depth and start address of every endpoint FIFO can be configured. The FIFO can locate any position of the 2K EXRAM. It supports control, interrupt, bulk, synchronous transfer mode; also it supports multiple-buffer operation controlled by software for using the USB bandwidth sufficiently.

Note: It is assumed the reader is familiar with or has access to the supporting documents USB1.1.

20.1 Clock

USB clock is 48MHz which is generated by PLL integrated in the chip. USB module can enter into idle mode for saving power consumption by setting the USB_PWR_CN.1 (0x0841) SFR. In this state, the register of USB can be read or write, but the USB engine is halted and cannot respond any external operation.

20.2 USB Register Access

The register of USB located from 0x0808 to 0x0850 of external RAM. The access method is same to external RAM, use MOVX command.

USB register included interrupt register, configure register, power management register and address register.

20.3 ENDPOINT Configuration

The USB module should be configured before using USB to commutate. The configure item includes endpoint address in EXRAM, the depth of FIFO, how many endpoints are used, and the direction, mode, enable of every endpoints.

Next is the description of these register. All the register can read or write by software.

EP_ADDR_MSB [0x0840]
CFG_EP0_1 [0x0810], CFG_EP0_0 [0x0811] (endpoint 0 configure register**)**
CFG_EP1_1 [0x0812], CFG_EP1_0 [0x0813] (endpoint 1 configure register**)**
CFG_EP2_1 [0x0814], CFG_EP2_0 [0x0815] (endpoint 2 configure register**)**
CFG_EP3_1 [0x0816], CFG_EP3_0 [0x0817] (endpoint 3 configure register**)**
CFG_EP4_1 [0x0818], CFG_EP4_0 [0x0819] (endpoint 4 configure register**)**
CFG_EP5_1 [0x081a], CFG_EP5_0 [0x081b] (endpoint 5 configure register**)**
CFG_EP6_1 [0x081c], CFG_EP6_0 [0x081d] (endpoint 6 configure register**)**
CFG_EP7_1 [0x081e], CFG_EP7_0 [0x081f] (endpoint 7 configure register**)**

Note: endpoint 0 is the control port. It occupies 64k bytes xram space the size and mode of it cannot be configured.

Next is the detail description of the register：

EP_ADDR_MSB（the MSB address bit of endpoints）：

EP_ADDR_MSB	7	6	5	4	3	2	1	0

0x0840	-	-	-	-	-	-	-	-

Table 45 USB MSB endpoint address

7: the MSB of endpoint 7 address. It decides the port address locates above 1K space or below it.

6: the MSB of endpoint 6 address. It decides the port address locates above 1K space or below it.

5: the MSB of endpoint 5 address. It decides the port address locates above 1K space or below it.

4: the MSB of endpoint 4 address. It decides the port address locates above 1K space or below it.

3: the MSB of endpoint 3 address. It decides the port address locates above 1K space or below it.

2: the MSB of endpoint 2 address. It decides the port address locates above 1K space or below it.

1: the MSB of endpoint 1 address. It decides the port address locates above 1K space or below it.

0: the MSB of endpoint 0 address. It decides the port address locates above 1K space or below it.

CFG_EP0_1 (the configure register 1 of endpoint 0)：

CFG_EP0_1	7	6	5	4	3	2	1	0
0x0810	dir	ep0_en	/				addr[9:8]	

Table 46 Configure Register 1 of Endpoint 0

7. Dir, port direction

1: IN (BK2433 send out data);

0: OUT (the PC send out data)。

6. ep0_en，endpoint 0 enable

When ep_rdy[0] =0 and ep0_en=0, usb no respond to external now.

1-0. addr[9:8] ： The higher 2 bits ([9:8]) address of endpoint0. The low 8 bits address is stored in CFG_EP0_0.

Note: The dir bit of CFG_EP0_1 is set or cleared by software except that it is cleared by hardware when SETUP token coming. The direction is forced to OUT to access 8 bytes setup request in this condition. The setup request has the highest priority.

CFG_EPn_1 (endpoint n configure register)：(n=1 - 7)

CFG_ EPn_1	7	6	5	4	3	2	1	0
	Dir	Mode		Size			Addr[9:8]	

Table 47 Endpoint n Configure Register

7. Dir:

1: IN

0: OUT

6-5. Mode:
0 -- Control Transfer
1 -- Bulk Transfer
2 -- ISO Transfer
3 -- Interrupt Transfer

4-2. Size：
0— endpoint not available
1— 16 bytes buffer size
2—32 bytes buffer size
3—64 bytes buffer size
4—128 bytes buffer size
5—256 bytes buffer size
6—512 bytes buffer size
7—endpoint not available

1-0. Addr[9:8] ： The higher 2 bits ([9:8]) address of endpoint n.

CFG_EPn_0 (configure register 0 of endpoint n)：(n=0 - 7)

CFG_EPn_0	7	6	5	4	3	2	1	0
	addr[7:0]							

Table 48 Endpoint nConfigure Register 0

The lower 8 bits address of endpoint n.

20.4 Interrupt

External interrupt 4 is assigned to USB. Int4 will be triggered if any enabled interrupt bit in USBINT0 or USBINT1 is set to 1. Software should query the register to find out the relevant interrupt source. Also, software should clear the interrupt bit by set it to 1 after dealing with the interrupt.

USBINT0 interrupt register

USB INT0	7	6	5	4	3	2	1	0
0x080a	ctl_rec	ctl_send	rx_rdy	tx_rdy	usb_rst	usb_sus	usb_res	usb_sof

Table 49 USBINT0 Interrupt Register

7. ctl_rec: data received on control port (endpoint 0)
6. ctl_send: data send on control port (endpoint 0)
5. rx_rdy： data received on endpoint 1-7
4. tx_rdy ： data send on endpoint 1-7
3. usb_reset ： USB Reset interrupt
2. usb_sus： USB suspend interrupt

1. usb_res：USB Resume interrupt。
0. usb_sof： USB Start Of Frame interrupt

When ctl_rec, rx_rdy or tx_rdy triggered，need to query EP_STATUS register for detail information.

EP_STATUS (set by hardware and cleared by software)

EP_STATUS	7	6	5	4	3	2	1	0
0x080e	EP7	EP6	EP5	EP4	EP3	EP2	EP1	sudat

Table 50 EP_STATUS Register

7. EP7： indicate rx_rdy or tx_rdy is triggered by endpoint 7
6. EP6： indicate rx_rdy or tx_rdy is triggered by endpoint 6
5. EP5：indicate rx_rdy or tx_rdy is triggered by endpoint 5
4. EP4：indicate rx_rdy or tx_rdy is triggered by endpoint 4
3. EP3：indicate rx_rdy or tx_rdy is triggered by endpoint 3
2. EP2：indicate rx_rdy or tx_rdy is triggered by endpoint 2
1. EP1：indicate rx_rdy or tx_rdy is triggered by endpoint 1
0. Sudat：indicate that 8 bytes set up package arrived

USBINT1 interrupt register，set by hardware and cleared by software（write 1 to clear it）.

USB INT1	7	6	5	4	3	2	1	0
0x080b	bad_token	crc16_err	overtime	pid_err	/	/	/	/

Table 51 USBINT1 Interrupt Register

7. bad_token: unsupported token received
6. crc16_err : the package received crc16 check error
5. overtime ： timeout interrupt(no data received after OUT token or no ACK received after IN token)
4. pid_err ： endpoint1-7 transfer PID error interrupt

USB_EN0, USB_EN1 interrupt enable register (only can be read or write by software)

USB_EN0	7	6	5	4	3	2	1	0
0x080c	ctl_rec_en	ctl_send_en	rx_rdy_en	tx_rdy_en	usb_rst_en	usb_sus_en	usb_res_en	usb_sof_en

Table 52 USB_EN0 Interrupt Enable Register

7. ctl_rec_en : data received on control endpoint 0 interrupt enable bit
6. ctl_send_en : data sent on control endpoint 0 interrupt enable bit
5. rx_rdy_en： data received on endpoint 1-7 interrupt enable bit
4. tx_rdy_en： data sent on endpoint 1-7 interrupt enable bit
3. usb_reset_en: USB Reset interrupt enable bit
2. usb_sus_en： USB suspend interrupt enable bit
1. usb_res_en： USB Resume interrupt enable bit
0. usb_sof_en： USB Start Of Frame interrupt enable bit

USB_EN1	7	6	5	4	3	2	1	0
0x080d	bad_token_en	crc16_err_en	overtime_en	pid_err_en	/	/	/	/

Table 53 USB_EN1Interrupt Enable Register

7. bad_token_en: unsupported token received interrupt enable bit
6. crc16_err_en : the package received crc16 check error interrupt enable bit
5. overtime_en： timeout interrupt enable bit
4. pid_err_en ： endpoint1-7 transfer PID error interrupt enable bit

20.5 FIFO

It has been descripted that how to configure the register above. The next will depict how to use their register and how to operate them.

20.5.1 FIFO SFR register

(1) EP_RDY： (endpoint ready register)

EP_RDY	7	6	5	4	3	2	1	0
0x0821	ep7_rdy	ep6_rdy	ep5_rdy	ep4_rdy	ep3_rdy	ep2_rdy	ep1_rdy	ep0_rdy

Table 54 FIFO EP_RDY Register

Epn_rdy (n=1-7): endpoint n is ready for transferring USB data now.\
Cleared by hardware and set by software.

Note: Ep0_rdy is not same with Epn_rdy. It will be forced to 1 by hardware when setup token coming to receive 8 bytes setup request. (setup has the highest priority for USB protocol)

When Epn_rdy=0, device will send back NACK pakage for PC's IN/OUT request to indicate not ready now.

(2) FIFO capacity counters

If one endpoint has been configured as IN direction, software need write the length number into the FIFO capacity register to tell USB the package length need send.

When one port configured as OUT direction, software can read out the package length from the counter register once one package received successfully. (the unit is byte)

Every endpoint use 2 bytes register, so total 16 registers are occupied which are descripted as follow:

CNTn : the lower 8 bits FIFO counter register of endpoint n

CNTn	7	6	5	4	3	2	1	0
	counter [7:0]							

Table 55 FIFO lower 8 bits counter register

CNTn_HBIT: the upper 2 bits FIFO counter register of endpoint n

CNTn_HBIT	7	6	5	4	3	2	1	0
	/						counter [9:8]	

Table 56 FIFO upper 2 bits counter register

All the 16 registers address:

CNT0 [0x0823]
CNT0_HBIT [0x082b]
CNT1 [0x0824]
CNT1_HBIT [0x082c]
CNT2 [0x0825]
CNT2_HBIT [0x082d]
CNT3 [0x0826]
CNT3_HBIT [0x082e]
CNT4 [0x0827]
CNT4_HBIT [0x082f]
CNT5 [0x0828]
CNT5_HBIT [0x0830]
CNT6 [0x0829]
CNT6_HBIT [0x0831]
CNT7 [0x082a]
CNT7_HBIT [0x0832]

(3) EP_HALT(endpoint suspend register)

EP_HALT	7	6	5	4	3	2	1	0
0x0820	ep7_halt	ep6_halt	ep5_halt	ep4_halt	ep3_halt	ep2_halt	ep1_halt	ep0_halt

Table 57 FIFO EP_HALT Register

epn_halt: the suspend flag of endpoint n. 1 indicates the endpoint has been suspended and this endpoint is not available now. This endpoint will send back STALL when IN/OUT token received to indicate it is not available now. (epn_halt can only be

read/written by software except ep0_halt)

ep0_halt can be cleared by hardware. According to USB protocol, ep0_halt is cleared by haredware when setup token received to avoid that device can not receive control information.

20.5.2 FIFO Access

The access to FIFO is very simple for BK2433. Software can read or write the 2K EXRAM directly with MOVX instruction and without any register interface or control logic.

When using C language, you only need to initialize a start address for one endpoint FIFO which should be consistent with the address configured in endpoint regiser.

For example: the address of endpoint 1

addr[10:0]={**EP_ADDR_MSB.1, CFG_EP1_1[1:0], CFG_EP1_0** }

=

$$\mathbf{EP\_ADDR\_MSB.1}\times 2^{10}+\mathbf{CFG\_EP1\_1.1}\times 2^{9}+\mathbf{CFG\_EP1\_1.0}\times 2^{8}+\mathbf{CFG\_EP1\_0}$$

This 11 bits address can cover all the 2K EXRAM space from 0 to 0x7FF.

20.5.3 FIFO Operation

The above describe how to access the EXRAM by MCU, and, the USB part need access the EXRAM also. It will be explained next.

Accessing EXRAM by USB is implemented by DMA controller, and it is transparent to software.

According protocol, the host send out SETUP, IN and OUT token to request device transfer. The device would start to transfer data after software inform device the relevant endpoint is “ready”. The DMA controller will write the received data into the FIFO assigned in the EXRAM (out endpoint), or read out the data that needed send to the host from FIFO (IN endpoint).

What time is ready? From software view, there are two cases:

1. The software had written the data needed send to host into FIFO. It is ready to send now.(IN)
2. The software had read out the data received from host from FIFO. It is ready to receive now.(OUT)

When it is ready, Software can set the corresponding EP_RDY to indicate it is ready now, and then USB will start to work automatically.

20.6 Device Address

The 7 bits function address is stored in FADDR register. The address is set by host through SET_ADDRESS command. The software should write the 7 bits address into FADDR after received this command. The address will act immediately after received SET_ADDRESS command. USB only can accept the data or token send to this address.

Device address(R/W by software only)

FUNCT_	7	6	5	4	3	2	1	0

ADDR								
0x0822	/	function_addr [6:0]						

Table 58 device address register

20.7 Frame number register

FRAM_NO_0： Frame number lower 8 bits （write by hardware, read by software only）

FRAM_NO_0	7	6	5	4	3	2	1	0
0x0808	Frame number [7:0]							

Table 59 FRAM_NO_0 lower 8 bits register

FRAM_NO_1: Frame number upper 3 bits，（write by hardware, read by software only）

FRAM_NO_1	7	6	5	4	3	2	1	0
0x0809	/					Frame number [10:8]		

Table 60 FRAM_NO_0 upper 3 bits register

20.8 USB power management

USB_PWR_CN ： USB power control register

USB_PWR_CN	7	6	5	4	3	2	1	0
0x0841	pu_en	DN	DP	/		usb_rst	usb_sus	remote_wakeup

Table 61 USB power control register

Pu_en: PULL UP enable， D+（dp）pull up enable in chip. When it is disabled, device disconnect with outside circuit.

DN: indicate D+ logic level（can used to debug）.（read only）

DP : indicate D- logic level（can used to debug）.（read only）

Usb_rst: USB module will reset when write 1 into it. (Exclude control register)

USB_sus： USB module will enter low-power mode when write 1 into it. The USB protocol engineer is stopped and no response to outside. It is used as suspend state usually in USB protocol.（can R/W by software）

remote_wakeup：according USB protocol, the device with remote wakeup function can send wake up signal to host.（R/W）When it is set to 1, USB force D+ and D- into K state, and release it when clear it.

20.9 Endpoint Buffer

For a transfer without buffer, it is described as follow: (IN direction)

1. MCU write the first package into the endpoint buffer and set relevant EP_RDY.
2. Wait transfer command from host, and send out interrupt when transfer is done.
3. MCU responds to interrupt and enter into relevant interrupt application. Then write the next package into the buffer and set EP_RDY.
4. Wait transfer command from host, and recurrence as described before.

The USB bandwidth utilize efficiency is the main disadvantage for this transfer mode. The host should wait when mcu wrote data into FIFO, and MCU should wait when USB sent data out.

For this, multi-buffer mechanism is applied in BK2433. MCU can write next package into FIFO when the current pakage is sending. So, when transfer command coming, the data can be sent immediately.

For example, 2-buffer is implemented as follows:

Config EP1 as IN endpoint, the capacity of EP1 is 64byte.

1. Configure the start FIFO address of EP1 as 0x500 and depth is 0x40.
2. Write the first pakage into 0x0500-0x0540, and then set EP_RDY register to indicate the data is ready.
3. At once, write the next package into 0x0540-0x0580 and wait the send out interrupt coming.
4. When the first package transfer complete, configure the start address of EP1 as 0x540, and then set EP_RDY to indicate the data is ready.
5. When the send out interrupt come, back to step 1.

Like this, 3 4 5 …-buffer is also can be implemented.

21 DES/3DES Encryption Decryption Unit

The BK2433 has dedicated HW for data encryption or decryption according to the international encryption standard algorithm and support ECB mode and CBC mode.

DES control register:

DES_CTL	7	6	5	4	3	2	1	0
	DES_EN	DES_TDES	DEC_ENC	MAC	/	ADDR_RST	OUT_RST	DES_RST

Table 62 DES control register

DES_EN：DES/3DES module enable or not, R/W by software

DES_TDES：DES TDES select.

1: use KEY1 to calculate DES,

0: use KEY1 KEY2 KEY3 to calculate 3DES(TDES)

DEC_ENC：1: decryption 0: encryption

MAC：Encryption mode select

0: ECB mode. Encrypt the 64 bits input only

1: CBC mode. The result of the previous encryption operation is XOR'ed with incoming data.

ADDR_RST：When write 1 into it, reset the counter used by 64 bits input and output. The data bus is 8 bits, so you should enter 8 times for one 64 bits input or output.

OUT_RST：Reset DES_OUT when write 1 into it.

DES_RST：Reset all the DES/3DES module when write 1 into it.（include DES configure register）

KEY1,KEY2,KEY3 *all the register is 64 bits. It can be written by software only and cannot be read by software.*

For example: write 64 bits KEY1

KEY1=0x1020304050607000;

So software need continued write 8 times as follow：

```
KEY1 = 0x10;
KEY1 = 0x20;
KEY1 = 0x30;
KEY1 = 0x40;
KEY1 = 0x50;
KEY1 = 0x60;
KEY1 = 0x70;
KEY1 = 0x00;
```

After write 8 times, the KEY will be left-shift into KEY1.

KEY2,KEY3 is same as KEY1。

DES_IN: the 64 bits input data used to encrypt. The write method is same as KEY1, when DES_IN is written 8 times, the encryption will start.

It wills generate interrupt after calculation, the interrupt flag is located DES_INT.

DES_INT	7	6	5	4	3	2	1	0
								DES_INTF

Table 63 DES_INT Register

DES_IN: calculation complete interrupt. It indicates that software can read the result from DES_OUT register now.

Note: write 0 to clear **DES_INT**.

DES_OUT: the calculation result. Software should read 8 times continuously for the the result is 64 bits.

Note:

DES: encrypt DES_IN with KEY1 or decrypt DES_IN with KEY1.

3DES:

Encryption: encrypt DES_IN with KEY1 fisrt, then encrypt the first result with KEY2, then encrypt the second result with KEY3 to get the finally result.

Decryption: decrypt DES_IN with KEY3 fisrt, then decrypt the first result with KEY2, finally decrypt the second result with KEY1 to get the result

22 BK2433 RF transceiver

22.1 General Description

The RF part BK2423 is embedded in BK2433, and BK2423 is a high performance IP of Beken corporation.
BK2423 is a GFSK transceiver operating in the world wide ISM frequency band at 2400-2483.5 MHz. Burst mode transmission and up to 2Mbps air data rate make them suitable for applications requiring ultra low power consumption. The embedded packet processing engines enable their full operation with a very simple MCU as a radio system. Auto re-transmission and auto acknowledge give reliable link without any MCU interference.

BK2423 operates in TDD mode, either as a transmitter or as a receiver.

The RF channel frequency determines the center of the channel used by BK2423. The frequency is set by the RF_CH register in register bank 0 according to the following formula: F0= 2400 + RF_CH (MHz). The resolution of the RF channel frequency is 1MHz.

A transmitter and a receiver must be programmed with the same RF channel frequency to be able to communicate with each other.

The output power of BK2423 is set by the RF_PWR bits in the RF_SETUP register.

Demodulation is done with embedded data slicer and bit recovery logic. The air data rate can be programmed to 1Mbps or 2Mbps by RF_DR register. A transmitter and a receiver must be programmed with the same setting.

In the following chapters, all registers are in register bank 0 except with explicit claim.

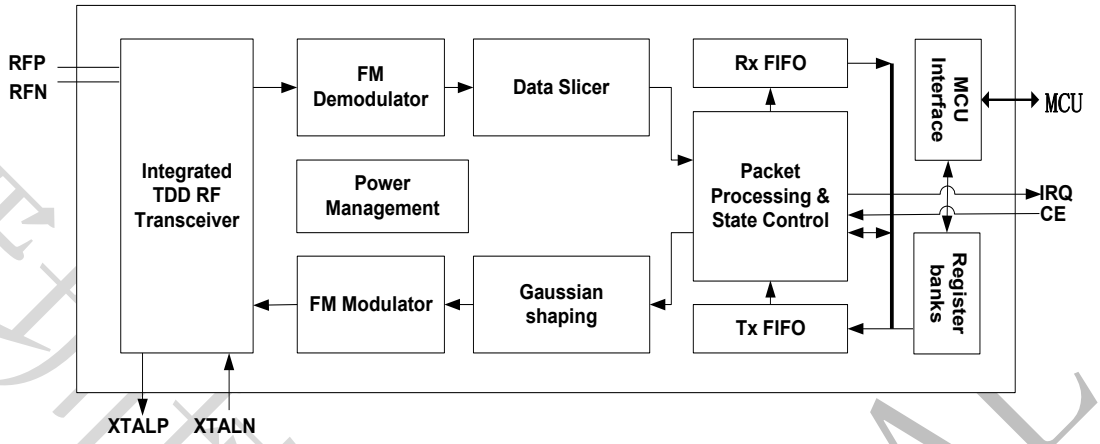


Figure 33 BK2423 Chip Block Diagram

22.2 Abbreviations

ACK	Acknowledgement
ARC	Auto Retransmission Count
ARD	Auto Retransmission Delay
CD	Carrier Detection
CE	Chip Enable
CRC	Cyclic Redundancy Check
CSN	Chip Select Not
DPL	Dynamic Payload Length
FIFO	First-In-First-Out
GFSK	Gaussian Frequency Shift Keying
GHz	Gigahertz
LNA	Low Noise Amplifier
IRQ	Interrupt Request
ISM	Industrial-Scientific-Medical
LSB	Least Significant Bit
MAX_RT	Maximum Retransmit
Mbps	Megabit per second
MCU	Microcontroller Unit
MHz	Megahertz
MISO	Master In Slave Out
MOSI	Master Out Slave In
MSB	Most Significant Bit
PA	Power Amplifier
PID	Packet Identity Bits
PLD	Payload
PRX	Primary RX
PTX	Primary TX
PWD_DWN	Power Down
PWD_UP	Power Up
RF_CH	Radio Frequency Channel
RSSI	Received Signal Strength Indicator
RX	Receive
RX_DR	Receive Data Ready
SCK	SPI Clock
SPI	Serial Peripheral Interface
TDD	Time Division Duplex
TX	Transmit
TX_DS	Transmit Data Sent
XTAL	Crystal

22.3 State Control

22.3.1 State Control Diagram

- Internal signal: POR,VDD
- SPI register: CE, PWR_UP, PRIM_RX, EN_AA, NO_ACK, ARC, ARD
- System information: Time out, ACK received, ARD elapsed, ARC_CNT, TX FIFO empty, ACK packet transmitted, Packet received

BK2423 has built-in state machines that control the state transition between different modes.

When auto acknowledge feature is disabled, state transition will be fully controlled by MCU.

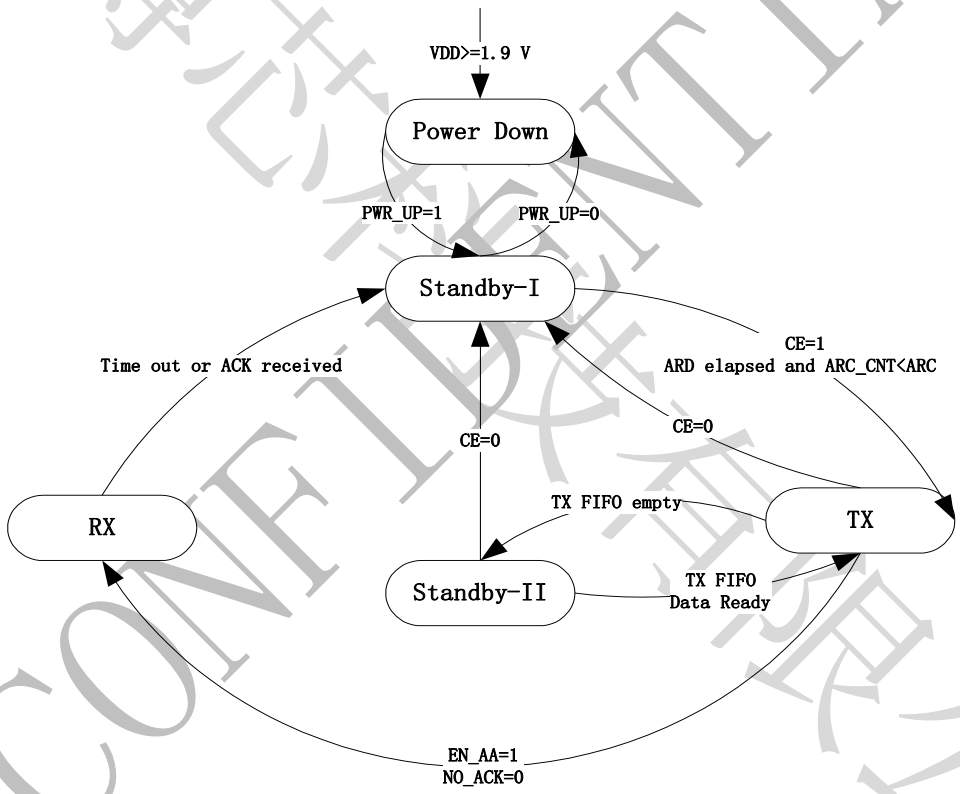


Figure 34 PTX (PRIM_RX=0) state control diagram

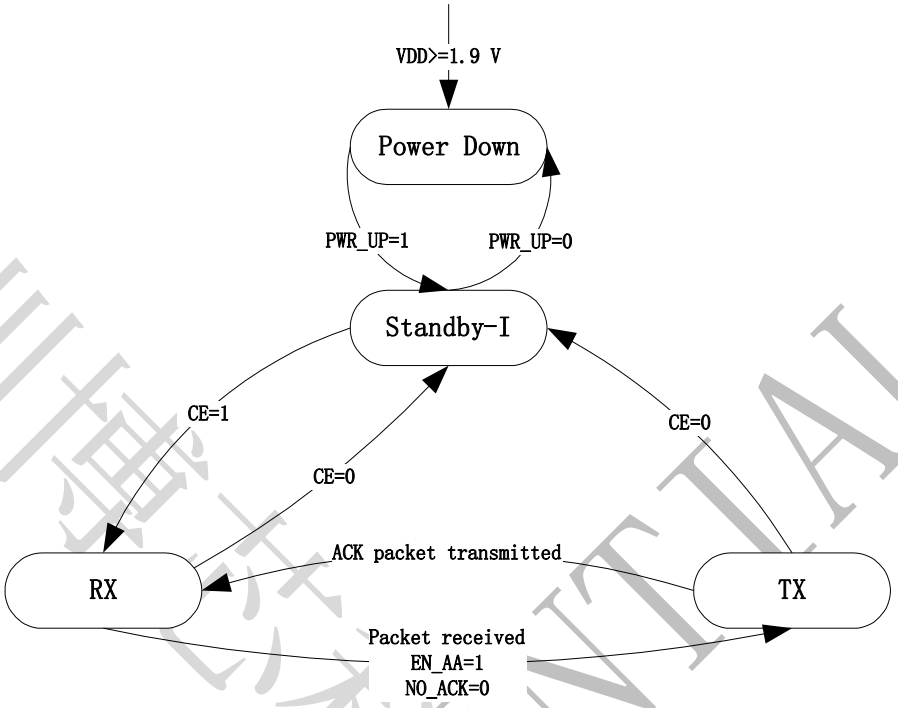


Figure 35 PRX (PRIM_RX=1) state control diagram

22.3.2 Power Down Mode

In power down mode BK2423 is in sleep mode with minimal current consumption. SPI interface is still active in this mode, and all register values are available by SPI. Power down mode is entered by setting the PWR_UP bit in the CONFIG register to low.

22.3.3 Standby-I Mode

By setting the PWR_UP bit in the CONFIG register to 1 and de-asserting CE to 0, the device enters standby-I mode. Standby-I mode is used to minimize average current consumption while maintaining short start-up time. In this mode, part of the crystal oscillator is active. This is also the mode which the BK2423 returns to from TX or RX mode when CE is set low.

22.3.4 Standby-II Mode

In standby-II mode more clock buffers are active than in standby-I mode and much more current is used. Standby-II occurs when CE is held high on a PTX device with empty TX FIFO. If a new packet is uploaded to the TX FIFO in this mode, the device will automatically enter TX mode and the packet is transmitted.

22.3.5 TX Mode

- PTX device (PRIM_RX=0)

The TX mode is an active mode where the PTX device transmits a packet. To enter this mode from power down mode, the PTX device must have the PWR_UP bit set high, PRIM_RX bit set low, a payload in the TX FIFO, and a high pulse on the CE for more than 10μs.
The PTX device stays in TX mode until it finishes transmitting the current packet. If CE = 0 it returns to standby-I mode. If CE = 1, the next action is determined by the status of the TX FIFO. If the TX FIFO is not empty the PTX device remains in TX mode, transmitting the next packet. If the TX FIFO is empty the PTX device goes into standby-II mode.

If the auto retransmit is enabled (EN_AA=1) and auto acknowledge is required (NO_ACK=0), the PTX device will enter TX mode from standby-I mode when ARD elapsed and number of retried is less than ARC.

- PRX device (PRIM_RX=1)

The PRX device will enter TX mode from RX mode only when EN_AA=1 and NO_ACK=0 in received packet to transmit acknowledge packet with pending payload in TX FIFO.

22.3.6 RX Mode

- PRX device (PRIM_RX=1)

The RX mode is an active mode where the BK2423 radio is configured to be a receiver. To enter this mode from standby-I mode, the PRX device must have the PWR_UP bit set high, PRIM_RX bit set high and the CE pin set high. Or PRX device can enter this mode from TX mode after transmitting an acknowledge packet when EN_AA=1 and NO_ACK=0 in received packet.

In this mode the receiver demodulates the signals from the RF channel, constantly presenting the demodulated data to the packet processing engine. The packet processing engine continuously searches for a valid packet. If a valid packet is found (by a matching address and a valid CRC) the payload of the packet is presented in a vacant slot in the RX FIFO. If the RX FIFO is full, the received packet is discarded.

The PRX device remains in RX mode until the MCU configures it to standby-I mode or power down mode.

In RX mode a carrier detection (CD) signal is available. The CD is set to high when a RF signal is detected inside the receiving frequency channel. The internal CD signal is filtered before presented to CD register. The RF signal must be present for at least 128 μs before the CD is set high.

- PTX device (PRIM_RX=0)

The PTX device will enter RX mode from TX mode only when EN_AA=1 and NO_ACK=0 to receive acknowledge packet.

22.4 Packet Processing

22.4.1 Packet Format

The packet format has a preamble, address, packet control, payload and CRC field.

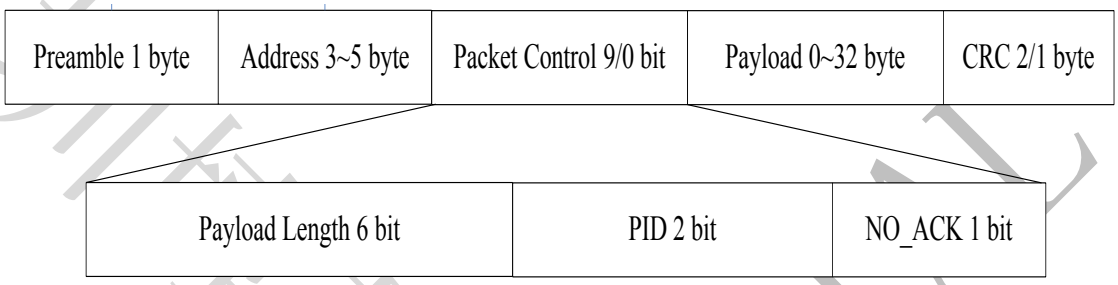


Figure 36 Packet Format

➢ Preamble

The preamble is a bit sequence used to detect 0 and 1 levels in the receiver. The preamble is one byte long and is either 01010101 or 10101010. If the first bit in the address is 1 the preamble is automatically set to 10101010 and if the first bit is 0 the preamble is automatically set to 01010101. This is done to ensure there are enough transitions in the preamble to stabilize the receiver.

➢ Address

This is the address for the receiver. An address ensures that the packet is detected by the target receiver. The address field can be configured to be 3, 4, or 5 bytes long by the AW register.

The PRX device can open up to six data pipes to support up to six PTX devices with unique addresses. All six PTX device addresses are searched simultaneously. In PRX side, the data pipes are enabled with the bits in the EN_RXADDR register. By default only data pipe 0 and 1 are enabled.

Each data pipe address is configured in the RX_ADDR_PX registers.

Each pipe can have up to 5 bytes configurable address. Data pipe 0 has a unique 5 byte address. Data pipes 1-5 share the 4 most significant address bytes. The LSB byte must be unique for all 6 pipes.

To ensure that the ACK packet from the PRX is transmitted to the correct PTX, the PRX takes the data pipe address where it received the packet and uses it as the TX address when transmitting the ACK packet.

On the PRX the RX_ADDR_Pn, defined as the pipe address, must be unique. On the PTX the TX_ADDR must be the same as the RX_ADDR_P0 on the PTX, and as the pipe address for the designated pipe on the PRX.

No other data pipe can receive data until

a complete packet is received by a data pipe that has detected its address. When multiple PTX devices are transmitting to a PRX, the ARD can be used to skew the auto retransmission so that they only block each other once.

➢ Packet Control

When Dynamic Payload Length function is enabled, the packet control field contains a 6 bit payload length field, a 2 bit PID (Packet Identity) field and, a 1 bit NO_ACK flag.

➢ Payload length

The payload length field is only used if the Dynamic Payload Length function is enabled.

➢ PID

The 2 bit PID field is used to detect whether the received packet is new or retransmitted. PID prevents the PRX device from presenting the same payload more than once to the MCU. The PID field is incremented at the TX side for each new packet received through the SPI. The PID and CRC fields are used by the PRX device to determine whether a packet is old or new. When several data packets are lost on the link, the PID fields may become equal to the last received PID. If a packet has the same PID as the previous packet, BK2423 compares the CRC sums from both packets. If the CRC sums are also equal, the last received packet is considered a copy of the previously received packet and discarded.

➢ NO_ACK

The NO_ACK flag is only used when the auto acknowledgement feature is used. Setting the flag high, tells the receiver that the packet is not to be auto acknowledged.

The PTX can set the NO_ACK flag bit in the Packet Control Field with the command: W_TX_PAYLOAD_NOACK.However, the function must first be enabled in the FEATURE register by setting the EN_DYN_ACK bit. When you use this option, the PTX goes directly to standby-I mode after transmitting the packet and the PRX does not transmit an ACK packet when it receives the packet.

➢ Payload

The payload is the user defined content of the packet. It can be 0 to 32 bytes wide, and it is transmitted on-air as it is uploaded (unmodified) to the device.

The BK2423 provides two alternatives for handling payload lengths, static and dynamic payload length. The static payload length of each of six data pipes can be individually set.

The default alternative is static payload length. With static payload length all packets between a transmitter and a receiver have the same length. Static payload length is set by the RX_PW_Px registers. The payload length on the transmitter side is set by the number of bytes clocked into the TX_FIFO and must equal the value in the RX_PW_Px register on the receiver side. Each pipe has its own payload length.

Dynamic Payload Length (DPL) is an alternative to static payload length. DPL enables the transmitter to send packets

with variable payload length to the receiver. This means for a system with different payload lengths it is not necessary to scale the packet length to the longest payload.

With DPL feature the BK2423 can decode the payload length of the received packet automatically instead of using the RX_PW_Px registers. The MCU can read the length of the received payload by using the command: R_RX_PL_WID.

In order to enable DPL the EN_DPL bit in the FEATURE register must be set. In RX mode the DYNPD register has to be set. A PTX that transmits to a PRX with DPL enabled must have the DPL_P0 bit in DYNPD set.

- CRC

The CRC is the error detection mechanism in the packet. The number of bytes in the CRC is set by the CRCO bit in the CONFIG register. It may be either 1 or 2 bytes and is calculated over the address, Packet Control Field, and Payload.

The polynomial for 1 byte CRC is $X^8 + X^2 + X + 1$. Initial value is 0xFF.
The polynomial for 2 byte CRC is $X^{16} + X^{12} + X^5 + 1$. Initial value is 0xFFFF.

No packet is accepted by receiver side if the CRC fails.

22.4.2 Packet Handling

BK2423 uses burst mode for payload transmission and receive.

The transmitter fetches payload from TX FIFO, automatically assembles it into packet and transmits the packet in a very short burst period with 1Mbps or 2Mbps air data rate.

After transmission, if the PTX packet has the NO_ACK flag set, BK2423 sets TX_DS and gives an active low interrupt IRQ to MCU. If the PTX is ACK packet, the PTX needs receive ACK from the PRX and then asserts the TX_DS IRQ.

The receiver automatically validates and disassembles received packet, if there is a valid packet within the new payload, it will write the payload into RX FIFO, set RX_DR and give an active low interrupt IRQ to MCU.

When auto acknowledge is enabled (EN_AA=1), the PTX device will automatically wait for acknowledge packet after transmission, and re-transmit original packet with the delay of ARD until an acknowledge packet is received or the number of re-transmission exceeds a threshold ARC. If the later one happens, BK2423 will set MAX_RT and give an active low interrupt IRQ to MCU. Two packet loss counters (ARC_CNT and PLOS_CNT) are incremented each time a packet is lost. The ARC_CNT counts the number of retransmissions for the current transaction. The PLOS_CNT counts the total number of retransmissions since the last channel change. ARC_CNT is reset by initiating a new transaction. PLOS_CNT is reset by writing to the RF_CH register. It is possible to use the information in the OBSERVE_TX register to make an overall assessment of

the channel quality.

The PTX device will retransmit if its RX FIFO is full but received ACK frame has payload.

As an alternative for PTX device to auto retransmit it is possible to manually set the BK2423 to retransmit a packet a number of times. This is done by the REUSE_TX_PL command.

When auto acknowledge is enabled, the PRX device will automatically check the NO_ACK field in received packet, and if NO_ACK=0, it will automatically send an acknowledge packet to PTX device. If EN_ACK_PAY is set, and the acknowledge packet can also include pending payload in TX FIFO.

22.5 Data and Control Interface

22.5.1 TX/RX FIFO

The data FIFOs are used to store payload that is to be transmitted (TX FIFO) or payload that is received and ready to be clocked out (RX FIFO). The FIFO is accessible in both PTX mode and PRX mode.

There are three levels 32 bytes FIFO for both TX and RX, supporting both acknowledge mode or no acknowledge mode with up to six pipes.

- TX three levels, 32 byte FIFO
- RX three levels, 32 byte FIFO

Both FIFOs have a controller and are accessible by using dedicated SPI commands. A TX FIFO in PRX can store payload for ACK packets to three different PTX devices. If the TX FIFO contains more than one payload to a pipe, payloads are handled using the first in first out principle. The TX FIFO in a PRX is blocked if all pending payloads are addressed to pipes where the link to the PTX is lost. In this case, the MCU can flush the TX FIFO by using the FLUSH_TX command.

The RX FIFO in PRX may contain payload from up to three different PTX devices.
.
A TX FIFO in PTX can have up to three payloads stored.

The TX FIFO can be written to by three commands, W_TX_PAYLOAD and W_TX_PAYLOAD_NO_ACK in PTX mode and W_ACK_PAYLOAD in PRX mode. All three commands give access to the TX_PLD register.

The RX FIFO can be read by the command R_RX_PAYLOAD in both PTX and PRX mode. This command gives access to the RX_PLD register.

The payload in TX FIFO in a PTX is NOT removed if the MAX_RT IRQ is asserted.

In the FIFO_STATUS register it is possible to read if the TX and RX FIFO are full or empty. The TX_REUSE bit is also available in the FIFO_STATUS register. TX_REUSE is set by the command REUSE_TX_PL, and is reset by the command: W_TX_PAYLOAD or FLUSH TX.

22.5.2 Interrupt

In BK2433-RF there is an active low interrupt (IRQ), which is activated when TX_DS IRQ, RX_DR IRQ or MAX_RT IRQ are set high by the state machine in the STATUS register. The IRQ resets when MCU writes '1' to the IRQ source bit in the STATUS register. The IRQ mask in the CONFIG register is used to select the IRQ sources that are allowed to assert the IRQ. By setting one of the MASK bits high, the corresponding IRQ source is disabled. By default all IRQ sources are enabled.

The 3 bit pipe information in the STATUS register is updated during the IRQ high to low transition. If the STATUS register is read during an IRQ high to low transition, the pipe information is unreliable.

22.6 RF Command

The RF commands are shown in the table:

Command name	Command word (binary)	# Data bytes	Operation
R_REGISTER	Read directly		
W_REGISTER	Write directly		
W_ANALOG_REG	Write through register0X8B8-0X8BC		
R_RX_PAYLOAD	8'b01000000	1 to 32 LSB byte first	Read RX-payload: 1 – 32 bytes. A read operation always starts at byte 0. Payload is deleted from FIFO after it is read. Used in RX mode.
W_TX_PAYLOAD	8'b01100000	1 to 32 LSB byte first	Write TX-payload: 1 – 32 bytes. A write operation always starts at byte 0 used in TX payload. This command used for ENABLE_ACK payload
FLUSH_TX	8'b10100000	0	Flush TX FIFO, used in TX mode
FLUSH_RX	8'b10000000	0	Flush RX FIFO, used in RX mode Should not be executed during transmission of acknowledge, that is, acknowledge package will not be completed.
REUSE_TX_PL	8'b00010000	0	Used for a PTX device Reuse last transmitted payload. Packets are repeatedly retransmitted as long as CE is high. TX payload reuse is active until W_TX_PAYLOAD or FLUSH TX is executed. TX payload reuse must not be activated or deactivated during package transmission

R_RX_PL_WID	Read register 0x8C4 directly		Read RX-payload width for the top R_RX_PAYLOAD in the RX FIFO.
W_ACK_PAYLOAD	8'b01101ppp	1 to 32 LSB byte first	Used in RX mode. Write Payload to be transmitted together with ACK packet on PIPE PPP. (PPP valid in the range from 000 to 101). Maximum three ACK packet payloads can be pending. Payloads with same PPP are handled using first in - first out principle. Write payload: 1– 32 bytes. A write operation always starts at byte 0.
W_TX_PAYLOAD_NO ACK	8'b01101000	1 to 32 LSB byte first	Used in TX mode. Disables AUTOACK on this specific packet.
NOP	8'b00000000	0	No Operation.

Table 64 RF command

22.7 Register Map

There are two register groups in BK2433 that is digital register and analog register.

22.7.1 Digital Register

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
0x0880	CONFIG				Configuration Register
	Reserved	7	0	R/W	Only '0' allowed
	MASK_RX_DR	6	0	R/W	Mask interrupt caused by RX_DR 1: Interrupt not reflected on the IRQ pin 0: Reflect RX_DR as active low interrupt on the IRQ pin
	MASK_TX_DS	5	0	R/W	Mask interrupt caused by TX_DS 1: Interrupt not reflected on the IRQ pin 0: Reflect TX_DS as active low interrupt on the IRQ pin
	MASK_MAX_RT	4	0	R/W	Mask interrupt caused by MAX_RT 1: Interrupt not reflected on the IRQ pin 0: Reflect MAX_RT as active low interrupt on the IRQ pin
	EN_CRC	3	1	R/W	Enable CRC. Forced high if one of the bits in the EN_AA is high
	CRCO	2	0	R/W	CRC encoding scheme '0' - 1 byte '1' - 2 bytes
	PWR_UP	1	0	R/W	1: POWER UP, 0:POWER DOWN
	PRIM_RX	0	0	R/W	RX/TX control, 1: PRX, 0: PTX
0x0881	EN_AA				Enable 'Auto Acknowledgment' Function (only used by RX part)
	Reserved	7:6	00	R/W	Only '00' allowed
	ENAA_P5	5	1	R/W	Enable auto acknowledgement data pipe 5
	ENAA_P4	4	1	R/W	Enable auto acknowledgement data pipe 4
	ENAA_P3	3	1	R/W	Enable auto acknowledgement data pipe 3
	ENAA_P2	2	1	R/W	Enable auto acknowledgement data pipe 2
	ENAA_P1	1	1	R/W	Enable auto acknowledgement data pipe 1
	ENAA_P0	0	1	R/W	Enable auto acknowledgement data pipe 0
0x0882	EN_RXADDR				Enabled RX Addresses

	Reserved	7:6	00	R/W	Only '00' allowed
	ERX_P5	5	0	R/W	Enable data pipe 5.
	ERX_P4	4	0	R/W	Enable data pipe 4.
	ERX_P3	3	0	R/W	Enable data pipe 3.
	ERX_P2	2	0	R/W	Enable data pipe 2.
	ERX_P1	1	1	R/W	Enable data pipe 1.
	ERX_P0	0	1	R/W	Enable data pipe 0.
0x0883	SETUP_AW				Setup of Address Widths (common for all data pipes)
	Reserved	7:2	000000	R/W	Only '000000' allowed
	AW	1:0	11	R/W	RX/TX Address field width '00' - Illegal '01' - 3 bytes '10' - 4 bytes '11' - 5 bytes LSB bytes are used if address width is below 5 bytes
0x0884	SETUP_RETR				Setup of Automatic Retransmission
	ARD	7:4	0000	R/W	Auto Retransmission Delay '0000' – Wait 250 us '0001' – Wait 500 us '0010' – Wait 750 us …….. '1111' – Wait 4000 us (Delay defined from end of transmission to start of next transmission)
	ARC	3:0	0011	R/W	Auto Retransmission Count '0000' –Re-Transmit disabled '0001' – Up to 1 Re-Transmission on fail of AA …… '1111' – Up to 15 Re-Transmission on fail of AA
0x0885	RF_CH				RF Channel
	Reserved	7	0	R/W	Only '0' allowed
	RF_CH	6:0	0000010	R/W	Sets the frequency channel
0 x0886	RF_SETUP				RF Setup Register
	Reserved	7	0	R/W	Reserved
		6	0	R/W	Reserved
	En_250k_rate	5	0	R/W	Set RF datarate to 250k
		4	0		Reserved,pls don't change it
	RF_DR	3	1	R/W	Air Data Rate ,decide by 0x886 bit {[5],[3]} '00' – 1Mbps '01' – 2Mbps '10' – 250kbps '11' – reserved
	RF_PWR[1:0]	2:1	11	R/W	Set RF output power in TX mode RF_PWR[1:0] '00' – -10 dBm '01' – -5 dBm '10' – 0 dBm

					'11' – 5 dBm
	LNA_HCURR	0	1	R/W	Setup LNA gain 0:Low gain(20dB down) 1:High gain
0 x0887 0 x0888 0 x0889 0 x088A 0 x088B	RX_ADDR_P0	39:0	0xE7E7E7E7E7	R/W	Receive address data pipe 0. 5 Bytes maximum length. Write the number of bytes defined by SETUP_AW) {X88B,X88A,X889,X888,X887}
0 x088C 0 x088D 0 x088E 0 x088F 0 x0890	RX_ADDR_P1	39:0	0xC2C2C2C2C2	R/W	Receive address data pipe 1. 5 Bytes maximum length. Write the number of bytes defined by SETUP_AW)
0 x0891	RX_ADDR_P2	7:0	0xC3	R/W	Receive address data pipe 2. Only LSB. MSB bytes is equal to RX_ADDR_P1[39:8]
0 x0892	RX_ADDR_P3	7:0	0xC4	R/W	Receive address data pipe 3. Only LSB. MSB bytes is equal to RX_ADDR_P1[39:8]
0 x0893	RX_ADDR_P4	7:0	0xC5	R/W	Receive address data pipe 4. Only LSB. MSB bytes is equal to RX_ADDR_P1[39:8]
0 x0894	RX_ADDR_P5	7:0	0xC6	R/W	Receive address data pipe 5. Only LSB. MSB bytes is equal to RX_ADDR_P1[39:8]
0 x0895 0 x0896 0 x0897 0 x0898 0 x0899	TX_ADDR	39:0	0xE7E7E7E7E7	R/W	Transmit address. Used for a PTX device only. (LSB byte is written first) Set RX_ADDR_P0 equal to this address to handle automatic acknowledge if this is a PTX device
0 x089A	RX_PW_P0				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P0	5:0	000000	R/W	Number of bytes in RX payload in data pipe 0 (1 to 32 bytes). 0: not used 1 = 1 byte … 32 = 32 bytes
0 x089B	RX_PW_P1				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P1	5:0	000000	R/W	Number of bytes in RX payload in data pipe 1 (1 to 32 bytes). 0: not used 1 = 1 byte … 32 = 32 bytes
0 x089C	RX_PW_P2				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P2	5:0	000000	R/W	Number of bytes in RX payload in data pipe 2 (1 to 32 bytes). 0: not used 1 = 1 byte … 32 = 32 bytes

0 x089D	RX_PW_P3				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P3	5:0	000000	R/W	Number of bytes in RX payload in data pipe 3 (1 to 32 bytes). 0: not used 1 = 1 byte … 32 = 32 bytes
0 x089E	RX_PW_P4				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P4	5:0	000000	R/W	Number of bytes in RX payload in data pipe 4 (1 to 32 bytes). 0: not used 1 = 1 byte … 32 = 32 bytes
0 x089F	RX_PW_P5				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P5	5:0	000000	R/W	Number of bytes in RX payload in data pipe 5 (1 to 32 bytes). 0: not used 1 = 1 byte … 32 = 32 bytes
0 x08A0	DYNPD				Enable dynamic payload length
	Reserved	7:6	0	R/W	Only ‘00’ allowed
	DPL_P5	5	0	R/W	Enable dynamic payload length data pipe 5. (Requires EN_DPL and ENAA_P5)
	DPL_P4	4	0	R/W	Enable dynamic payload length data pipe 4. (Requires EN_DPL and ENAA_P4)
	DPL_P3	3	0	R/W	Enable dynamic payload length data pipe 3. (Requires EN_DPL and ENAA_P3)
	DPL_P2	2	0	R/W	Enable dynamic payload length data pipe 2. (Requires EN_DPL and ENAA_P2)
	DPL_P1	1	0	R/W	Enable dynamic payload length data pipe 1. (Requires EN_DPL and ENAA_P1)
	DPL_P0	0	0	R/W	Enable dynamic payload length data pipe 0. (Requires EN_DPL and ENAA_P0)
0 x08A1	FEATURE			R/W	Feature Register
	Reserved	7:3	0	R/W	Only ‘00000’ allowed
	EN_DPL	2	0	R/W	Enables Dynamic Payload Length
	EN_ACK_PAY	1	0	R/W	Enables Payload with ACK
	EN_DYN_ACK	0	0	R/W	Enables the W_TX_PAYLOAD_NOACK command

0 x08A5 0 x08A4 0 x08A3 0 x08A2	(cfg0c0--3)	31:0	0		Please initialize with 0x00731200
0 x08A9 0 x08A8 0 x08A7 0 x08A6	NEW_FEATURE (cfg0d0--3)	31:0	0		Please initialize with 0x0080B436
0 x08B4 0 x08B3 0 x08B2 0 x08B1 0 x08B0 0 x08AF 0 x08AE 0 x08AD 0 x08AC 0 x08AB 0 x08AA	RAMP (2402table_0 --2401table_A)	87:0	NA	W	Ramp curve Please write with 0xFFFFFEF7CF208104082041
0 x08B5	BK2423_ce				
		7:1			reserved
		0			ce
0 x08B6	BK2423_cmd				8'b10000000 : Flush RX 8'b10100000 : Flush TX 8'b00010000 : Reusle TX PL 8'b01000000 : Read RX Payload 8'b01100000 : Write TX Payload 8'b01101ppp : W_ACK_PAYLOAD 8'b01101000 : W_TX_PAYLAOD_NOACK 8'b00000000 : NOP
0 x08B7	BK2423_FIFO			R/W	TX MODE: TX data payload register 1 - 32 bytes. RX MODE: RX data payload register 1 - 32 bytes.
0 x08B8	BK2423_sdata_0				Analog register0[7:0]
0 x08B9	BK2423_sdata_1				Analog register1[15:8]
0 x08BA	BK2423_sdata_2				Analog register2[23:16]
0 x08BB	BK2423_sdata_3				Analog register3[31:24]
0 x08BC	BK2423_sctrl				Write address of Analog rgister (only can be writen)
0 x08C0	BK2423_status				Status, read only
	RX_DR	6	0	R/W	Data Ready RX FIFO interrupt Asserted when new data arrives RX FIFO Write 1 to clear bit.
	TX_DS	5	0	R/W	Data Sent TX FIFO interrupt Asserted when packet transmitted on TX. If AUTO_ACK is activated, this bit is set high only when ACK is received. Write 1 to clear bit.
	MAX_RT	4	0	R/W	Maximum number of TX retransmits interrupt Write 1 to clear bit. If MAX_RT is asserted it must be cleared to enable further communication.

	RX_P_NO	3:1	111	R	Data pipe number for the payload available for reading from RX_FIFO 000-101: Data Pipe Number 110: Not used 111: RX FIFO Empty
	TX_FULL	0	0	R	TX FIFO full flag. 1: TX FIFO full 0: Available locations in TX FIFO
0 x08C1	BK2423_observetx				Status, read only
	PLOS_CNT	7:4	0000	R	Count lost packets. The counter is overflow protected to 15, and discontinues at max until reset. The counter is reset by writing to RF_CH.
	ARC_CNT	3:0	0000	R	Count retransmitted packets. The counter is reset when transmission of a new packet starts.
0 x08C2	BK2423_cdstatus				Status, read only
	Reserved	7:1	000000	R	
	CD	0	0	R	Carrier Detect
0 x08C3	BK2423_fifostatus				Status, read only
	Reserved	7	0	R/W	Only '0' allowed
	TX_REUSE	6	0	R	Reuse last transmitted data packet if set high. The packet is repeatedly retransmitted as long as CE is high. TX_REUSE is set by the SPI command REUSE_TX_PL, and is reset by the SPI command W_TX_PAYLOAD or FLUSH TX
	TX_FULL	5	0	R	TX FIFO full flag 1: TX FIFO full; 0: Available locations in TX FIFO
	TX_EMPTY	4	1	R	TX FIFO empty flag. 1: TX FIFO empty 0: Data in TX FIFO
	Reserved	3:2	00	R/W	Only '00' allowed
	RX_FULL	1	0	R	RX FIFO full flag 1: RX FIFO full 0: Available locations in RX FIFO
	RX_EMPTY	0	1	R	RX FIFO empty flag 1: RX FIFO empty 0: Data in RX FIFO
0 x08 C4	BK2423_rpl_width				Status, read only
					The width of the payload
0X8C5	BK2423_mbist_st				Status, read only
	reseved			R	5'b0
	Done	2		R	test_done
	Pass	1		R	test_pass
	Fail	0		R	test_fail
0X8C6~0X8C7	Chip_id			R	
				R	Chip_id[7:0]

				R	Chip_id[15:8]
0X8C8~0X8CB	BK2423_bit_cnt			R	Status, read only Total number of bits received register
0X8CC~0XCF	BK2423_err_cnt			R	Status, read only Error counter register
Note: Don't write reserved registers and registers at other addresses in register bank 0					

Table 65 Digital Register

Note:

1. ARD-auto retransmission delay. If the ACK payload is more than 15 byte in 2Mbps mode the ARD must be 500μS or more, if the ACK payload is more than 5byte in 1Mbps mode the ARD must be 500μS or more. In 250kbps mode (even when the payload is not in ACK) the ARD must be 500μS or more.
2. The RX_DR IRQ is asserted by a new packet arrival event. The procedure for handling this interrupt should be: 1) read payload from FIFO, 2) clear RX_DR IRQ, 3) read FIFO_STATUS to check if thereare more payloads available in RX FIFO, 4) if there are more data in RX FIFO, repeat from step 1).
3. Register 0x881 EN_AA only used for RX part now. For TX part, the command W_TX_PAYLOAD used for auto-ack payload, the command W_TX_PAYLOAD_NOACK used for disable-ack payload.

22.7.2 Analog Register

The analog registers can be written through writing 0X8B8 to 0X8BC.
Analog register data [31:0] = {0X8BB, 0X8Ba, 0X8B9, 0X8B8}
Analog register address [7:0] = {0X8BC}
Writing corresponding data and address into these serial registers can update the analog register value.

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
00		31:0	0	W	Must write with 0x404B01E2
01		31:0	0	W	Must write with 0xC04B0000
02		31:0	0	W	Must write with 0xD0FC8C02
03		31:0	0x 03001200	W	Must write with 0x99003941
04		31:0	0	W	Must write with 0xD99E860B(High Power) For single carrier mode:0xC19E8621
		20	1	W	RF output power in TX mode: 0:Low power(-30dB down) 1:High power
05		31:0	0	W	Must write with 0x24067FA6(Disable RSSI)
	RSSI_TH	29:26		W	RSSI Threshold for CD detect 0: -97 dBm, 2 dB/step, 15: -67 dBm
	RSSI_EN	18	0	W	RSSI measurement: 0:Enable 1:Disable
06		31:0	0	W	Reserved
07		31:0	0	W	Reserved
08					Reserved
09			0		Reserved
0A			0		Reserved
0B			0		Reserved
Note: Don't write reserved registers and no definition registers in register bank 1					

Table 66 Register Bank 1

22.8 Electrical Specifications

RF part

Name	Parameter (Condition)	Min	Typical	Max	Unit	Comment
	Operating Condition					
VDD	Voltage	1.9	3.0	3.6	V	
TEMP	Temperature	-40	+27	+85	ºC	
	Digital input Pin					
VIH	High level	0.7VDD		5.25	V	
VIL	Low level	VSS		0.3VDD	V	
	Digital output Pin					
VOH	High level (IOH=-0.25mA)	VDD- 0.3		VDD	V	
VOL	Low level(IOL=0.25mA)	0		0.3	V	
	Normal condition					
IVDD	Power Down current		-		uA	
IVDD	Standby-I current			50	uA	
IVDD	Standby-II current			400	uA	
	Normal RF condition					
FOP	Operating frequency	2400		2527	MHz	
FXTAL	Crystal frequency		16		MHz	
RFSK	Air data rate	0.25	1	2	Mbps	
	Transmitter					
PRF	Output power	-40	0	5	dBm	
PBW	Modulation 20 dB bandwidth(2Mbps)		2.5		MHz	
PBW	Modulation 20 dB bandwidth (1Mbps)		1.3		MHz	
PRF1	Out of band emission 2 MHz		-20		dBm	
PRF2	Out of band emission 4 MHz		-40		dBm	
IVDD	Current at -40 dBm output power		11		mA	
IVDD	Current at -30 dBm output power		11		mA	
IVDD	Current at -25 dBm output power		12		mA	
IVDD	Current at -10 dBm output power		13		mA	
IVDD	Current at -5 dBm output power		15		mA	
IVDD	Current at 0 dBm output power		17		mA	
IVDD	Current at 5 dBm output power		23		mA	
	Receiver					
IVDD	Current (2Mbps)		18		mA	
IVDD	Current (1Mbps)		17		mA	
Max Input	1 E-3 BER		10		dBm	
RXSENS	1 E-3 BER sensitivity (2Mbps)		-89		dBm	
RXSENS	1 E-3 BER sensitivity (1Mbps)		-92		dBm	
RXSENS	1 E-3 BER sensitivity (250Kbps)		-98		dBm	
C/ICO	Co-channel C/I (2Mbps)		4		dB	
C/I1ST	ACS C/I 2MHz (2Mbps)		-5		dB	
C/I2ND	ACS C/I 4MHz (2Mbps)		-20		dB	
C/I3RD	ACS C/I 6MHz (2Mbps)		-25		dB	
C/ICO	Co-channel C/I (1Mbps)		4		dB	
C/I1ST	ACS C/I 1MHz (1Mbps)		4		dB	
C/I2ND	ACS C/I 2MHz (1Mbps)		-18		dB	
C/I3RD	ACS C/I 3MHz (1Mbps)		-19		dB	

Table 67 RF Electrical Specification

MCU part

Name	Parameter (Condition)	Min	Typical	Max	Unit	Comment
	Core functions					
	Deep sleep mode		0.5		uA	
	Sleep mode (RCOSC 32k)		4		uA	
	Idle mode at 16M		1		mA	
	Idle mode at 8M		0.8		mA	
	Idle mode at 4M		0.45		mA	
	Idle mode at XOSC32k(16M running)		0.15		mA	
	Active mode (16M)		4.9		mA	
	Active mode (8M)		3.9		mA	
	Active mode (4M)		3.3			
	Peripheral					
	MTP write		TBD		mA	
	MTP write		TBD		mA	
	MTP write		TBD		mA	
	ADC (8k byte rates)		350		uA	
	ADC SINAD (fin=1khz, fs=8khz)		55		DB	
	RNG				mA	
	LBD (always on)		250		uA	
	USB		2		mA	
	GPIO					
	Drive ability		4	8	mA	

Table 68 MCU Electrical Specification

23 Typical Application Schematic

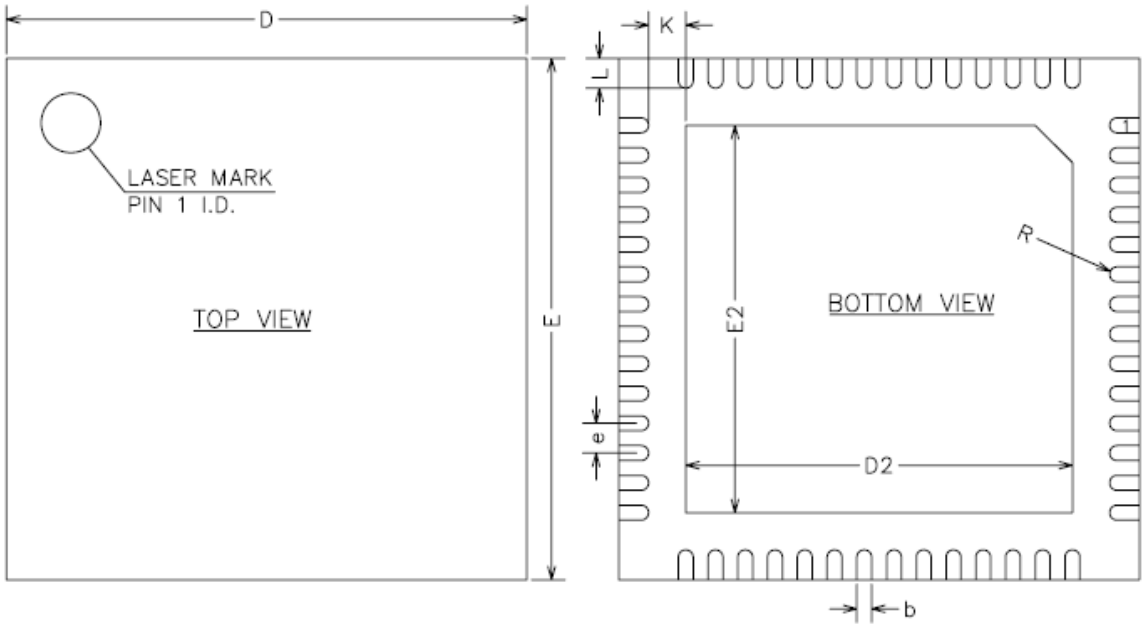
Please refer to the separate documents for detail.

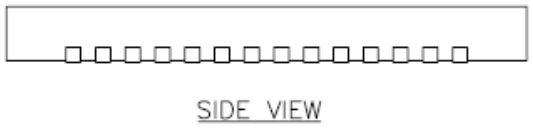
24 Package Information

BK2433 have three type package sizes: BK2433KB(QFN56 7x7mm), BK2433MB(QFN32 5x5mm),and BK2423QB(QFN24 4x4mm).

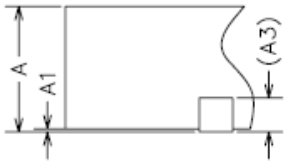
24.1 BK2433KB(QFN56 7x7mm) package

Package marking BK2433 KYYWWXX					
K	Y	Y	W	W	XX
Internal Code	Year number		Week number		Internal Code





SIDE VIEW



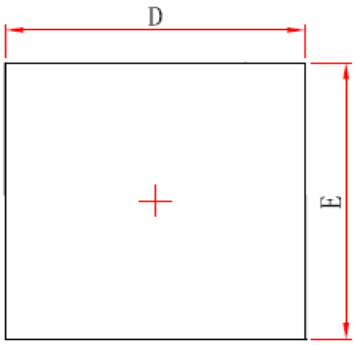
COMMON DIMENSIONS
(UNITS OF MEASURE=MILLIMETER)

SYMBOL	MIN	NOM	MAX
A	0.70	0.75	0.80
A1	0	0.02	0.05
A3		0.20REF	
b	0.15	0.20	0.25
D	6.90	7.00	7.10
E	6.90	7.00	7.10
D2	5.05	5.20	5.35
E2	5.05	5.20	5.35
e	0.30	0.40	0.50
K	0.20	–	–
L	0.35	0.40	0.45
R	0.09	–	–

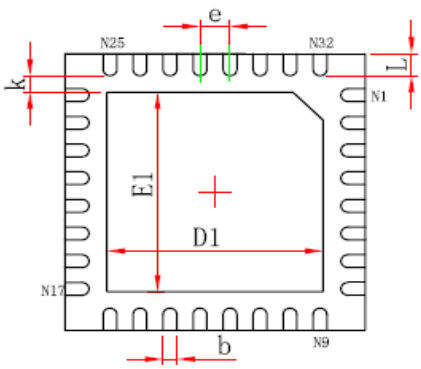
Figure 37 QFN56 7x7mm Pin package diagram

24.2 BK2433MB(QFN32 5x5mm) package

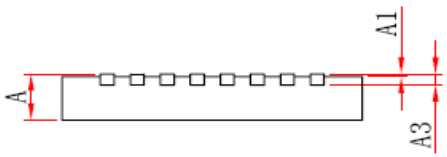
Package marking BK2433 MYWWXX				
M	Y	W	W	XX
Internal Code	Year number	Week number		Internal Code



Top View



Bottom View



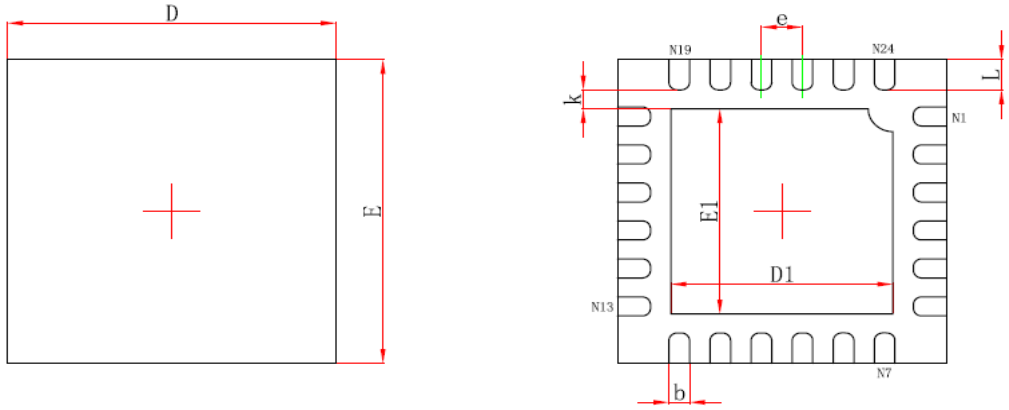
Side View

Symbol	Dimensions In Millimeters		Dimensions In Inches	
	Min.	Max.	Min.	Max.
A	0.700/0.800	0.800/0.900	0.028/0.031	0.031/0.035
A1	0.000	0.050	0.000	0.002
A3	0.203REF.		0.008REF.	
D	4.924	5.076	0.194	0.200
E	4.924	5.076	0.194	0.200
D1	3.300	3.500	0.130	0.138
E1	3.300	3.500	0.130	0.138
k	0.200MIN.		0.008MIN.	
b	0.180	0.300	0.007	0.012
e	0.500TYP.		0.020TYP.	
L	0.324	0.476	0.013	0.019

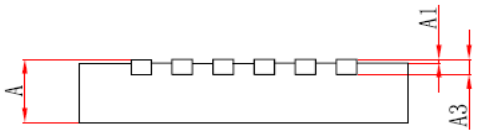
Figure 38 QFN32 5x5mm Pin package diagram

24.3 BK2423QB(QFN24 4x4mm) package

Package marking BK2433 DYWWXX				
D	Y	W	W	XX
Internal Code	Year number	Week number		Internal Code



Top View **Bottom View**



Symbol	Dimensions In Millimeters		Dimensions In Inches	
	Min.	Max.	Min.	Max.
A	0.700/0.800	0.800/0.900	0.028/0.031	0.031/0.035
A1	0.000	0.050	0.000	0.002
A3	0.203REF.		0.008REF.	
D	3.900	4.100	0.154	0.161
E	3.900	4.100	0.154	0.161
D1	2.600	2.800	0.102	0.110
E1	2.600	2.800	0.102	0.110
k	0.200MIN.		0.008MIN.	
b	0.180	0.300	0.007	0.012
e	0.500TYP.		0.020TYP.	
L	0.300	0.500	0.012	0.020

Figure 39 QFN24 4x4mm Pin package diagram

25 Order Information

Part number	Package	Packing	MPQ (ea)
BK2433KB	QFN56	Tape Reel	3K
BK2433MB	QFN32	Tape Reel	3K
BK2423QB	QFN24	Tape Reel	3K

Table 69 BK2433 order information

Remark:
MPQ: Minimum Package Quantity

26 Solder Reflow Profile

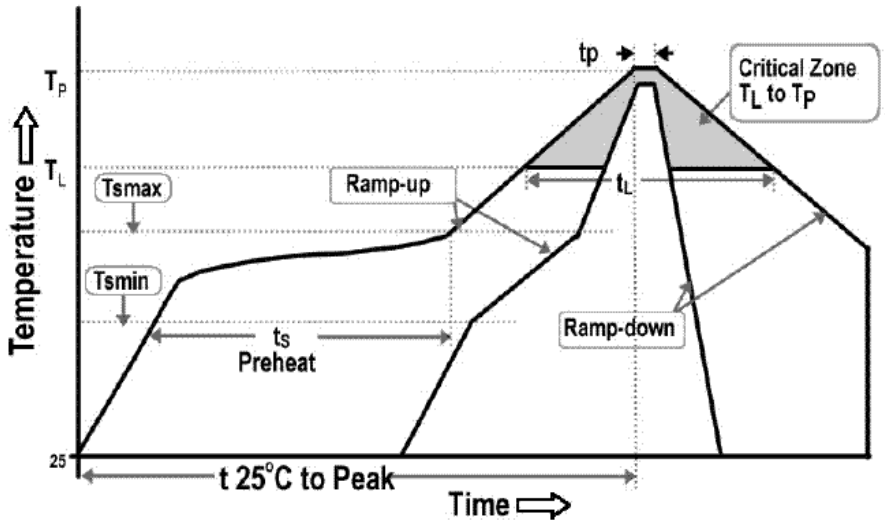


Figure 40 Classification Reflow Profile

Profile Feature		Specification
Average Ramp-Up Rate (tsmax to tp)		3°C/second max.
Pre_heat	Temperature Min (Tsmin)	150°C
	Temperature Max (Tsmax)	200°C
	Time (ts)	60-180 seconds
Time Maintained above	Temperature (TL)	217°C
	Time (tL)	60-150 seconds
Peak/Classification Temperature (Tp)		260°C
Time within 5°C of Actual PeakTemperature (tp)		20-40 seconds
Ramp-Down Rate 6		6°C/second max.
Time 25°C to Peak Temperature 8		8 minutes max.

RoHS Compliant

The product does not contain lead, mercury, cadmium, hexavalent chromium, PBB&PBDE content in accordance with directive 2002/95/EC(RoHS).

ESD Sensitivity

Integrated circuits are ESD sensitive and can be damaged by static electricity. Proper ESD Techniques should be used when handling these devices.

27 Contact Information

Beken Corporation Technical Support Center

Shanghai office
Suite 3A,1278 Keyuan Road, Zhangjiang High-Tech Park,
Pudong New District, Shanghai, P.R. China
Phone: 86-21-51086811,60871276
Fax: 86-21-60871277
Postal Code: 201203
Email: info@bekencorp.com
Website: www.bekencorp.com

Shenzhen office
Room 718,Shenzhen High-Tech Industrial Estate,
Nanshan, Shenzhen, P.R. China
Phone: 86-755-2655 1063
Postal Code: 518057