九齊科技股份有限公司
Nyquest Technology Co., Ltd.

USER MANUAL

NY9UP08A

Learning Remote Controller with 13 I/O

Version 1.1

Mar. 28, 2019

Revision History

Version	Date	Description	Modified Page
1.0	2019/02/27	Formal Release.	-
1.1	2019/03/28	Adjust set timer0 8M code from 0xC to 0x3	20

Table of Contents

Chapter 1. Introduction

1.1 General Description

The NY9UP08A is a powerful 4-bit MCU with remote controller. It has 13 I/O ports and supports T-type key matrix. One large sink current IR port can fulfill transmitting and receiving function without any bipolar transistor. The RISC MCU architecture is very easy to use, and various applications can be easily implemented. There are 51 instructions, and most of them are executed in single cycle. Furthermore, it provides the HALT mode (sleep mode) to extend battery life.

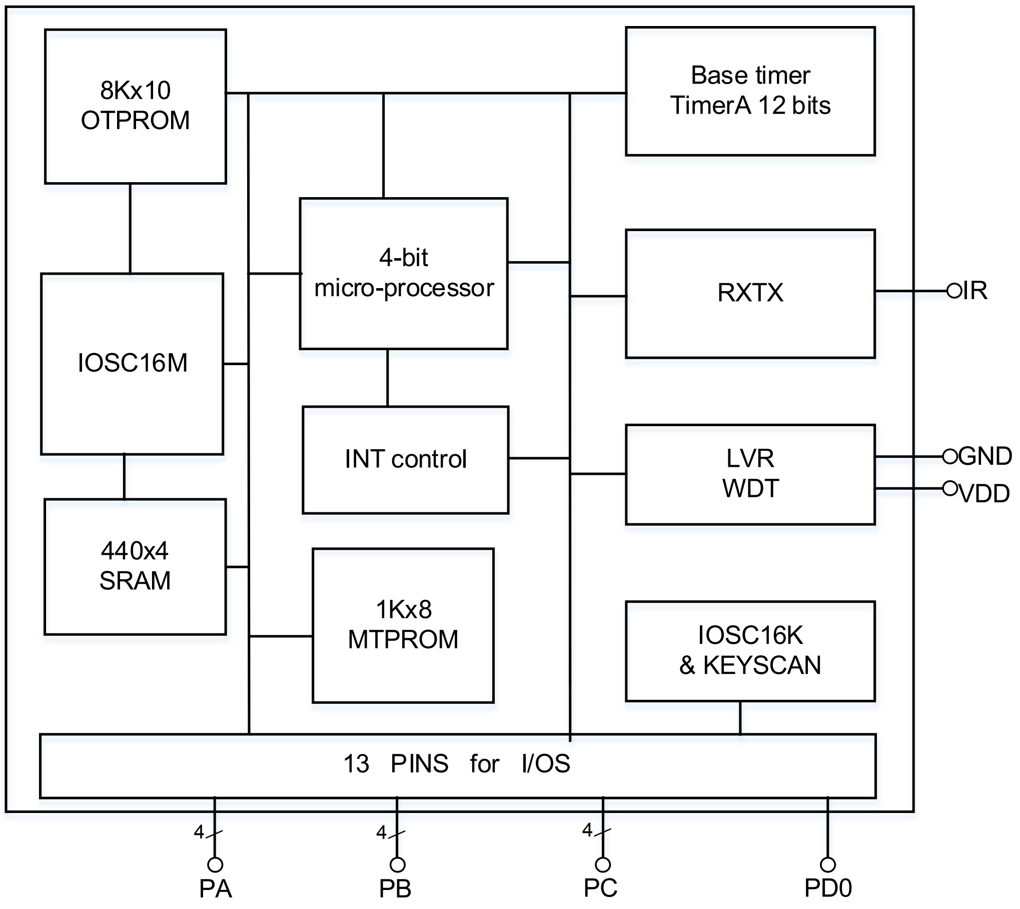
1.2 Features

- Operating voltage range: 2.0V to 3.6V.
- 4-bit RISC type micro-controller with 51 instructions.
- 8Kx10-bit OTP ROM
- 1Kx8-bit MTP ROM
- 440x4-bit RAM, indirect RAM addressing mode is supported.
- 2MHz instruction frequency.
- HALT mode to save power, standby current <1uA @3V.
- Precisely embedded oscillator with build-in resistor, +/- 1.5% deviation in 2.0V~3.6V and -20°C~+70°C.
- Low voltage reset (LVR=1.7V) and watch-dog reset are all supported to protect the system.
- 2.2V LVD flag for low battery detection.
- 2 entrances for interrupt operation with an independent stack, multiple interrupt sources.
- 13 flexible I/Os.
  - 13 I/Os of PAx, PBx, PCx and PD0 with function: bi-direction I/O with pull-high.
- M-Type, T-type or mixed type key wakeup supported.
- Infrared port provides RX application, and large current IR carrier output for TX.
- 12-bit readable Timer0 with selectable timer clock source for IR TX carrier frequency and RX learning counter.

1.3 Product List

IC Type	OTP ROM (bits)	MTP ROM (bits)	RAM (bits)	I/O	T-scan	LVD / LVR	12-bit Timer0	IR TX	IR RX
NY9UP08A	8Kx10	1Kx8	440x4	13	v	v	v	1	1

1.4 Block Diagram



1.5 Pad Description

Pad	ATTR.	Description
VDD	Power	Positive power.
GND	Power	Negative power.
IR	I/O	Infrared port (TX & RX).
PA0/VPP	I/O	Bit 0 for Port A, or positive high power for programming.
PA1/Mode	I/O	Bit 1 for Port A, or select programming mode.
PA2/SCL	I/O	Bit 2 for Port A, or serial clock input at programming mode.
PA3/SDA	I/O	Bit 3 for Port A, or serial data input at programming mode.
PB0~3	I/O	PA0~3, PB0~3, PC0~3, PD0: 13 bi-direction I/Os with pull-high. Port can be set as normal I/O or key scan I/O, and key scan I/O can send key scan signal under halt mode. And PA3 and PD0 can be set as no waking up function IO through option.
PC0~3		
PD0		

1.6 Electrical Characteristics

The following lists the electrical characteristics of the NY9UP08A OTP chip.

1.6.1 Absolute Maximum Rating

Symbol	Parameter	Rated Value	Unit
VDD - GND	Supply voltage	-0.3 ~ +4.0	V
V_{IN}	Input voltage	GND-0.3V ~ VDD+0.3	V
T_{OP}	Operating Temperature	-20 ~ +70	°C
T_{ST}	Storage Temperature	-40 ~ +85	°C

1.6.2 DC Characteristics *(VDD=3.0V, T_A=25ºC, unless otherwise specified)*

Symbol	Parameter		Min.	Typ.	Max.	Unit	Condition
VDD	Operating voltage		2.0	3	3.6	V	2MHz
I_{SB}	Supply current	Halt mode			1	uA	Sleep, no load
I_{Scan}		Scan mode			2	uA	T-type key scan
I_{OP}		Operating mode		2		mA	2MHz, no load
I_{IL}	Input current (Internal 125KΩ pull-high)			24		uA	V_{IL} = 0V
I_{OL}	Output low current		10	18		mA	V_{OL} = 1.0V
V_{IL}	input low level			0.5*VDD		V	VDD = 3.0V
V_{IH}	input high level			0.7*VDD		V	VDD = 3.0V
I_{IR}	IR sink current (Normal)			450		mA	V_{IR} = 1.5V
R_{TXPH}	IR Pull high Resistor			37/3.7		KΩ	V_{IR} = 0V
D_{CAP}	IR capture distance			15		cm	VDD = 3.0V, related to IRLED
∆F/F	Internal OSC frequency deviation		-1.5		1.5	%	VDD: 2.0V ~ 3.6V, Temp: -20°C ~ +70°C

Chapter 2. Hardware Architecture

2.1 Overview

2.1.1 Function Block Diagram

The NY9UP08A belongs to the LRC family of the NYQUEST devices. Block diagram of the device is shown below.

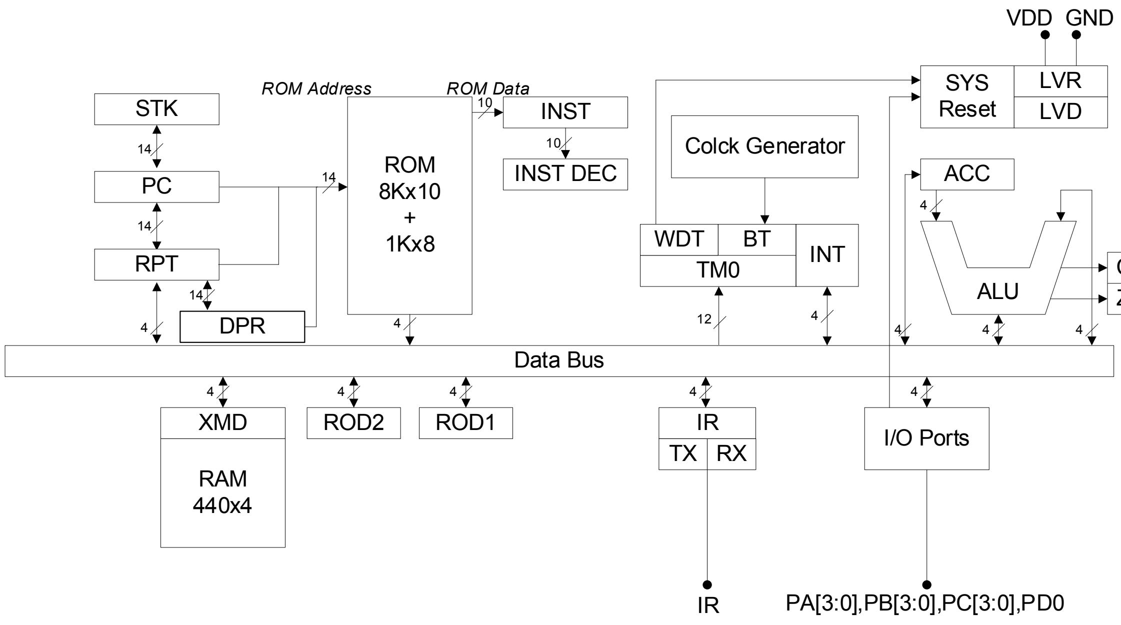


Figure 2-1 NY9UP08A Function Block Diagram

2.1.2 Arithmetic Logic Unit

The NY9UP08A provides a 4-bit arithmetic logic unit with a 4-bit accumulator to perform logic, unsigned arithmetic, data transfer and conditional branch operation. There are two flags (carry and zero) to indicate the result of the operation. One or two operands will be the data sources of the ALU operation. The operands can be ACC, RAM, register, or literal constant data.

2.1.3 ALU Related Status Flag

Besides CLRC and SETC commands directly assign the value of the carry flag, C is influenced by the arithmetic result. C means carry and also means the complement of borrow. If the addition operation larger than 0xF, C=1, and C=0 if the result ≤15. If the subtraction operation smaller than 0, C=0, and C=1 if the result ≥0.

Symbol	Flag	Description
C	Carry flag	C=1 if a carry-out occurs after an addition operation.
		C=0 if a borrow-in occurs after a subtraction operation.
Z	Zero flag	Z=1 if the result of an ALU operation is zero.

2.1.4 Address Pointer

The NY9UP08A micro-controller contains a program counter (PC), an interrupt dedicated stack (STK), a multi-function register pointer (RPT), and a data pointer (DPR). The length of each address pointer is 14-bit. Users have to keep in mind that the initial value of all the pointers is unknown, except the PC.

2.1.5 Program Counter (PC)

As a program instruction is executed, the PC will contain the address of the next program instruction to be executed. The PC starts from the reset vector (address 0x0000) after the system reset, and its value is increased by one every instruction cycle unless changed by an interrupt or a branch instruction. The interrupt vector for Timer0 & BT is at address 0x0010. The interrupt vector for RX falling & rising is at address 0x0018.

2.1.6 Stack (STK)

One level hardware push/pop stack dedicated to the interrupt is available. When an interrupt occurs, the system pushes the PC to the STK automatically. When the program returns to the main program from the interrupt routine by IRET instruction, the system pops the STK back to the PC.

2.1.7 Multi-function Register Pointer (RPT)

As implied in the name, RPT are multi-function registers. Users have to operate RPT in coordination with instructions below.

Inst./Event	Function
CALL	Pushes PC+2 to RPT
RJMP	Move RPT to PC
RBPC	Reads back PC+1 to RPT
LDPR	Load RPT to DPR
RBPR	Read DPR to RPT
INCR	RPT[8:0]+1

2.1.8 Data Pointer Register(DPR)

As implied in the name, DPR with 14-bit width is necessary for reading ROM data. When LDPR is executed, the system loads RPT to DPR. Besides, User can read DPR back by RBPR instruction. And DPR are special registers without address allocation. When RD or RDI instruction is used, DPR value represents the ROM address and the read back ROM data is placed on ROD2[1:0], ROD1, ACC, ACC is 4 LSB of ROM data. After RD and RDI operation, the DPR with RDI will add one, but the RD will keep the present value.

Inst./Event	Function
RD	Data Table Read to {ROD2[1:0], ROD1, ACC}
RDI	Data Table Read to {ROD2[1:0], ROD1, ACC}, and DPR = DPR+1
LDPR	Load RPT to DPR
RBPR	Read DPR to RPT

2.2 Memory Organization

The NY9UP08A has 8K words OTP ROM, 1K bytes MTP ROM, 440 nibbles of RAM and some dedicated system control register. The registers are divided into normal system registers and 7 nibbles of Multi-function registers. The detail is shown in Figure 2-3 .

2.2.1 ROM

A program data single ROM is provided and its structure is shown below. The reserved region contains system information and can’t be utilized by users.

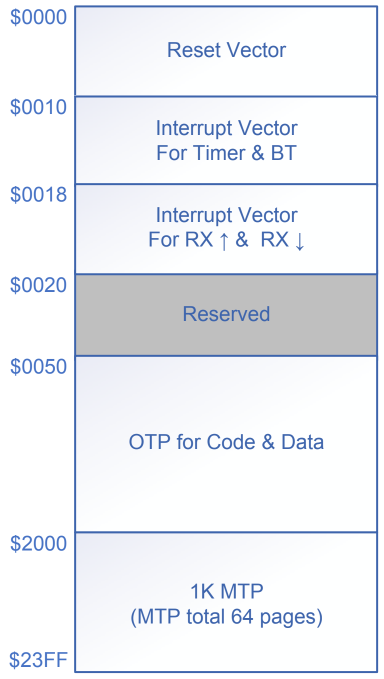

$0000
Reset Vector
$0010
Interrupt Vector For Timer & BT
$0018
Interrupt Vector For RX ↑ & RX ↓
$0020
Reserved
$0050
OTP for Code & Data
$2000
1K MTP (MTP total 64 pages)
$23FF

Figure 2-2 NY9UP08A ROM Map

2.2.2 RAM

NY9UP08A provide 440 nibbles RAM space. the address for RAM is 0x08~0x1BF. The first space from 0x08 to 0x3F is function RAM space, it only needs one-word instruction operation. And the second space from 0x40 to 0x1BF is data RAM space; the related operation is two-word instruction.

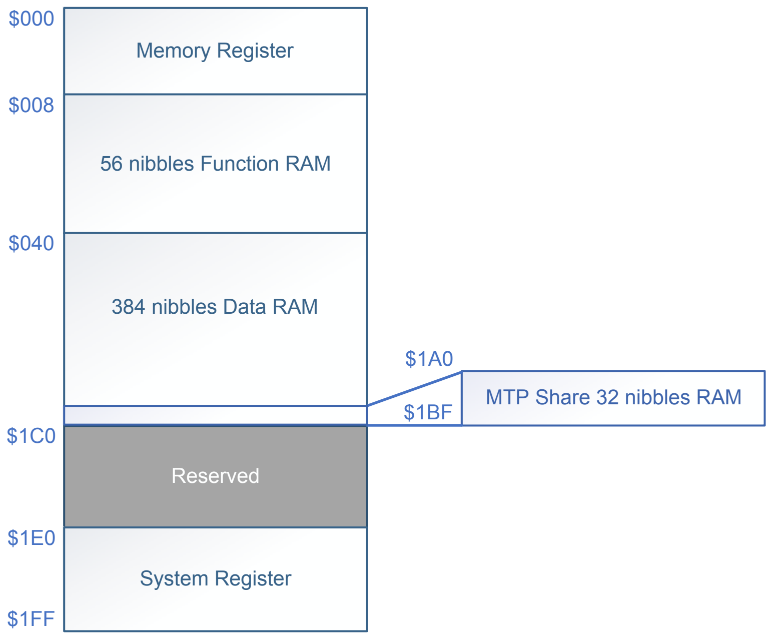


Figure 2-3 NY9UP08A RAM Map

In addition to the immediate addressing mode, the indexed addressing mode is also supported. The address of the indexed RAM should be stored into RPT2, RPT1 and RPT0 first, and users can read from or write in the XMD0 memory register to realize the indexed RAM access.

And the RAM space from $1A0 to $1BF is MTP page data store place which need to dump to MTP when page programming.

2.2.3 Memory Register

The 7 memory registers share the address with RAM. The page number of RAM has no relationship with the memory register address.

Address	Name	Bit	Description
0	RPT0	[3:0]	Multi-function register pointer bit [3:0]
1	RPT1	[3:0]	Multi-function register pointer bit [7:4]
2	RPT2	[3:0]	Multi-function register pointer bit [11:8]
3	RPT3	[1:0]	Multi-function register pointer bit [13:12]
4	-	-	Reserved
5	ROD1	[3:0]	ROM data bit [7:4] access register
6	ROD2	[1:0]	ROM data bit [9:8] access register
7	XMD0	[3:0]	Indexed RAM data access register

2.2.4 System Register

The system registers of NY9UP08A is part of memory space. The memory operation instruction is compatible with register operation.

Address	Name	Bit	Description
0x1E0	MTPG_L	[3:0]	MTP page low 4-bit
0x1E1	MTPG_H	[1:0]	MTP page high 2-bit
0x1E2	MTPC	[3:0]	MTP control register
0x1E3	MTPPT	[3:0]	MTP write protect register
0x1E4	PA	[3:0]	PA I/O port control register
0x1E5	PB	[3:0]	PB I/O port control register
0x1E6	PC	[3:0]	PC I/O port control register
0x1E7	PD	[0]	PD I/O port control register
0x1E8	TM0CS	[1:0]	Timer0 clock source selection
0x1E9	LVD	[1:0]	LVD/POR flag register
0x1F0	TM0_CapL	[3:0]	Low 4-bit of capture counter for falling edge
0x1F1	TM0_CapM	[3:0]	Middle 4-bit of capture counter for falling edge
0x1F2	TM0_CapH	[3:0]	High 4-bit of capture counter for falling edge
0x1F3	TM0_L	[3:0]	Low 4-bit of Timer0 register
0x1F4	TM0_M	[3:0]	Middle 4-bit of Timer0 register
0x1F5	TM0_H	[3:0]	High 4-bit of Timer0 register
0x1F6	TM0_DutL	[3:0]	Duty low nibble Low 4-bits of capture counter for rising edge
0x1F7	TM0_DutH	[3:0]	Duty high nibble Middle 4-bit of capture counter for rising edge
0x1F8	TM0_CapRH	[3:0]	High 4-bits of capture counter for rising edge
0x1F9	TXOPT	[3:0]	TX control register
0x1FA	RXOPT	[3:0]	RX control register
0x1FB	BTF	[3:0]	Base timer control register
0x1FC	IRCTRL	[3:0]	IR control register
0x1FD	ONOFF	[3:0]	IR and Timer0 configurations
0x1FE	INTF0	[3:0]	Interrupt flag
0x1FF	INT0	[3:0]	Interrupt control register

2.3 System Reset

2.3.1 System Power-On & Power-Down

After power-on, the power-on reset initialization will automatically be set out. The system takes 16ms to leave from the reset initialization procedure and enters the normal operation and the program counter (PC) will start at the reset vector to execute the desired program.

2.3.2 Low Voltage Reset & Detection (LVR & LVD)

When the system enters the normal operation, the power voltage must be kept in an effective working voltage range. If the power voltage is lower than the effective working voltage range, the system will work improperly.

To prevent the system crash, NY9UP08A supplies Low Voltage Reset (LVR) detectors. Once the LVR detector detects a harmful low voltage supply, it will cause a low voltage reset. The so-called "low voltage reset" point of the NY9UP08A IC is about 1.7V. The Low Voltage Detection (LVD) is an advanced detection of supply voltage. The LVD flag is read only and will be set to "0" if the supply voltage is lower than V_{LVD}; otherwise, set to "1".

2.3.3 Watch-Dog Timer Reset (WDTR)

To recover from program malfunction, the NY9UP08A IC supports an embedded watch-dog timer reset. The WDTR function always works with the program executing. Users have to clear the WDT periodically to prevent from timing up with a reset generation. Typically, the minimum time-up period of the WDT is about 0.45s. The WDT can be cleared by instruction CWDT1 next to CWDT0 only.

2.4 I/O Ports

There are 13 I/O ports, designated as PAx, PBx, PCx, PD0, and x=0~3. All ports are IO (bi-direction) with pull-high, and users can set I/O functions by I/O registers. And when the chip is running, the status can be changed by I/O register control.

The table below shows the relation between them.

Category	PX register write 1	PX register write 0
PAx PBx PCx PD0 (x=0~3)	input with pull-high	output low

The pull-high resistor of all the I/O ports is about 125KΩ @3V for key matrix function usually.

PA3 and PD0 can be controlled by option to enable or disable wakeup function.

2.5 Infrared Transmitter/Receiver

The NY9UP08A provides an independent pin (IR) for infrared transmit/receiver block, which is used to send or receive infrared signal. For the function of transmitter, users can set a variety of IR carrier frequency by the given clock source (TM0CS), 8-bit duty value(TM0_DutH, TM_DutL) and 12-bit IR Timer0 value(TM0_H, TM0_M, TM0_L). As for the detailed calculation of IR carrier frequency and applications, refer to section 3.8, IR control register. For the receiver, the external BJT is no longer needed. Users can utilize the RX falling edge flag (RXFF) or RX rising edge flag (RXRF) interrupt mechanism through register INT0/INTF0 or read received data directly through IRCTRL[3] to apply for the desired program. The Figure 3-4 shows the usage of IRCTRL[3].

2.6 Interrupt Generator

There is one hardware interrupt and it has 4 different sources in NY9UP08A. The interrupt event can be a fixed interval of the system base timer (BT), the Timer0 overflow flag (TOF), RXFF, RXRF. The TOF arises as Timer0 overflows. The RXFF arises as a falling edge of RX signal detected. The RXRF arises as a rising edge of RX signal detected. There is a system base timer in the NY9UP08A IC. The NY9UP08A provide 4 fixed intervals from the system base timer for interrupt source: 0.064ms, 0.128ms, 0.256ms and 16.384ms.

As an interrupt occurs, NY9UP08A stores the accumulator (ACC), carry flag (C), zero flag (Z) and RPT automatically. Then move PC to STK, backup RPT. If the interrupt source isTimer0 or base timer, jump to the interrupt vector (0x0010). If the interrupt source is RX falling or RX rising edge, jump to the interrupt vector (0x0018). An interrupt routine finishes with an IRET instruction. The IC draws the ACC, C, Z and RPT back, and moves STK to PC back to jump back the main program.

The interrupt event of RXFF and RXRF will be automatically cleared after entering the interrupt routine, but the BT and TOF have to be cleared by users.

Chapter 3. System Control Register

3.1 Introduction of System Control Register

TM0_CapL, TM0_CapM, TM0_CapH, TM0_L, TM0_M, TM0_H, TM0_DutL, TM0_DutH, TM0_CapRH and TM0CS are Timer0 control related registers. The MTPG_L, MTPG_H, MTPC and MTPPT are MTP control related registers. The IRCTRL, TXOPT and RXOPT are IR control related registers. The PA PB PC and PD are I/O ports registers. The LVD register is used to monitor IC power with 2.2V, the output flag goes high while power is higher. The ONOFF register is to turn on block function such as Timer0 and IR. INT0 register is used to enable interrupt entrance for BT, TOF, RXFF and RXRF. INTF0 register are reading flag originated from those interrupt sources and written 0 to reset its flag individually. The combination of RPT0~3 are multi-function register pointer. The C and Z are arithmetic associated flags. The XMD0 is RAM access registers. The ROD1 and ROD2 registers are used to read the ROM data.

3.1.1 System Control Register Address Map

Addr	Name	Initial	R/W	Bit	Data	Description
0x1E0	MTPG_L	0000	R/W	[3:0]	0/1	MTP page low 4-bit
0x1E1	MTPG_H	00	R/W	[1:0]	0/1	MTP page high 2-bit
		00	R	[3:2]	00	Reserved
0x1E2	MTPC	0	R/W	[0]	0/1	Page program and erase finish flag
		1	R/W	[1]	0/1	MTP read disable / enable
		0	R	[2]	0/1	Page program finish / Page programming (MTP busy)
		0	W		1	Write 1 to enable page program
		0	R	[3]	0/1	Erase finish / Erasing (MTP busy)
		0	W		1	Write 1 to enable chip erase
0x1E3	MTPPT	0000	W	[3:0]	0/1	Write 0xA to enable MTP operation once
0x1E4	PA	xxxx	R	[3:0]	0/1	Read port A input pad data
		1111	W	[3:0]	0/1	Write to port A data register
0x1E5	PB	xxxx	R	[3:0]	0/1	Read port B input pad data
		1111	W	[3:0]	0/1	Write to port B data register
0x1E6	PC	xxxx	R	[3:0]	0/1	Read port C input pad data
		1111	W	[3:0]	0/1	Write to port C data register
0x1E7	PD	x	R	[0]	0/1	Read port D input pad data
		1	W	[0]	0/1	Write to port D data register
		000	R	[3:1]	000	Reserved
0x1E8	TM0CS	11	R/W	[1:0]	00	Timer0 clock source : 1M Hz
					01	Timer0 clock source : 2M Hz
					10	Timer0 clock source : 4M Hz
					11	Timer0 clock source : 8M Hz (Default)
		00	R	[3:2]	00	Reserved
0x1E9	LVD	x	R	[0]	0/1	LVD flag : VDD > 2.2V, Flag to high
		x	R/W	[1]	0/1	POR flag : Power on reset flag, write 0 only (clear flag)
		00	R	[3:2]	00	Reserved

Addr	Name	Initial	R/W	Bit	Data	Description
0x1F0	TM0_CapL	xxxx	R	[3:0]	0/1	Read Timer0 capture counter data of RX falling edge
0x1F1	TM0_CapM	xxxx	R	[3:0]	0/1	
0x1F2	TM0_CapH	xxxx	R	[3:0]	0/1	
0x1F3	TM0_L	1111	R/W	[3:0]	0/1	R: Read Timer0 counter W: Load Timer0 value to Timer0 initial data (R/W TM0_H to latch from counter / to initial data)
0x1F4	TM0_M	1111	R/W	[3:0]	0/1	
0x1F5	TM0_H	1111	R/W	[3:0]	0/1	
0x1F6	TM0_DutL	0000	R/W	[3:0]	0/1	R: Read Timer0 low 8-bit capture counter data of RX rising edge W: Load duty of carrier (Write TM0_DutH to latch)
0x1F7	TM0_DutH	0000	R/W	[3:0]	0/1	
0x1F8	TM0_CapRH	0000	R	[3:0]	0/1	Read Timer0 high 4-bit capture counter data of RX rising edge
0x1F9	TXOPT	1	R/W	[0]	0/1	TX sink current select: Large current / Normal current
		0	R	[1]	0	Reserved
		1	R/W	[2]	0/1	TX pull high select when no sinking With pull-high / Without pull-high
		1	R/W	[3]	0/1	TX pull high resistor value select: 3.7K / 37K
0x1FA	RXOPT	1111	R/W	[1:0]	00	RX sensitivity 10cm
					01	RX sensitivity 5cm
					10	RX sensitivity 20cm
					11	RX sensitivity 15cm(Default)
			R/W	[2]	0/1	RX filter select 0.5us filter / No filter
			R	[3]	0	Reserved
0x1FB	BTF	xxxx	R	[0]	0/1	Bit toggles in every 0.032 ms
				[1]	0/1	Bit toggles in every 0.064 ms
				[2]	0/1	Bit toggles in every 0.128 ms
				[3]	0/1	Bit toggles in every 8.192 ms
		0000	W	[3:0]	0000	Set base timer interrupt source = 0.064 ms
					0001	Set base timer interrupt source = 0.128 ms
					0010	Set base timer interrupt source = 0.256 ms
					0011	Set base timer interrupt source = 16.384 ms
					Others	Reserved
0x1FC	IRCTRL	11	R/W	[0]	0/1	IR TX data enable / disable
				[1]	0/1	IR carrier disable / enable
		0	R	[2]	0	Reserved
		x	R	[3]	0/1	Read status from IR pad
0x1FD	ONOFF	0000	R/W	[0]	0/1	Timer0 disable / enable
				[1]	0/1	IR TX enable / IR RX enable
				[2]	0/1	Timer0 counter capture mode select RX falling edge only / RX falling & rising edge both TM0_Cap / TM0_Cap & TM0_CapRH + TM0_Dut
				[3]	0/1	Adjust carrier No adjust / Adjust carrier frame to complete duty
0x1FE	INTF0	xxxx	R/W	[0]	0/1	BT interrupt flag, write 0 to clear
				[1]	0/1	Timer0 interrupt flag, write 0 to clear

Addr	Name	Initial	R/W	Bit	Data	Description
				[2]	0/1	RX↓ interrupt flag, write 0 to clear
				[3]	0/1	RX↑ interrupt flag, write 0 to clear
0x1FF	INT0	0000	R/W	[0]	0/1	Disable / Enable BT interrupt
				[1]	0/1	Disable / Enable Timer0 interrupt
				[2]	0/1	Disable / Enable RX↓ interrupt
				[3]	0/1	Disable / Enable RX↑ interrupt

3.1.2 Memory Register Address Map

Addr	Name	R/W	Bit	Description	Initial	Wake-up
0	RPT0	R/W	[3:0]	Multi-function register pointer [3:0]	0	U
1	RPT1	R/W	[3:0]	Multi-function register pointer [7:4]	0	U
2	RPT2	R/W	[3:0]	Multi-function register pointer [11:8]	X	U
3	RPT3	R/W	[1:0]	Multi-function register pointer [13:12]	X	U
		R	[3:2]	Reserved	0	0
4	-	-	[3:0]	Reserved	-	-
5	ROD1	R/W	[3:0]	ROM[7:4] data access register	X	U
6	ROD2	R/W	[1:0]	ROM[9:8] data access register	X	U
		R	[3:2]	Reserved	0	0
7	XMD0	R/W	[3:0]	Indexed RAM data access register	X	X

3.1.3 Register without Memory Allocation Map

Name	R/W	Bit	Description	Initial	Wake-up
C	-	1	Arithmetic carry flag	0	U
Z	-	1	Arithmetic zero flag	0	U
DPR	R/W	14	Data pointer register	X	U

R : Can be read from the register

U : Unchanged (the same as before wake-up)

W : Can be written to the register

X : Unknown

3.2 RPT

The RPT of NY9UP08A is 14-bit, and the functions of RPT are listed in the section 2.1.7. Besides the instructions related to the XMD0 only access bit [8:0] of the RPT, others access all available bits.

The CALL instruction pushes the PC to the RPT and jump to the subroutine address of the operand `a'. When the subroutine is finished, use RJMP to come back to the main program.

3.3 ROD

The NY9UP08A provides the RD/RDI instruction to read the ROM data out. When RD/RDI is executed, the system takes the DPR as ROM address, and the ROM data is loaded to ROD2[1:0], ROD1, and ACC. Bit [9:8] of the ROM data is loaded to ROD2, bit [7:4] to ROD1, and bit [3:0] to ACC. Using RD/RDI to read the data of the reserved ROM area out is unacceptable. The RD means the DPR value keeps unchanged after RD, and RDI means the DPR adds 1.

3.4 XMD

Users access XMD0 taking the RPT[8:0] = {RPT2[0], RPT1, RPT0} for NY9UP08A as the RAM address. Users have to watch out that the NY9UP08A does not support using XMD0 to access memory registers, so the RPT[8:0] can't be 0x000~0x007 when accessing XMD. And RPT[8:0] adds 1 automatically, when the instruction INCR is executed.

3.5 I/O Ports Register

Each I/O port has its corresponding register PX. Reading from the register reveals the pad status, and writing to it means writing to the I/O register.

The four registers of PA, PB, PC and PD can configure the corresponding port status. The detail can check the section 2.4.

The register of a bi-direction port is used for switch the I/O between input and output. When the register PX=1, it is an input with pull-high. When PX=0, it is an output and output low level. The register PX value of an output port simply means the output data. Users have to note that reading from an output port also getting the pad potential level, not the register value.

The pull-high resistor of all the I/O ports is only the strong pull-high, which is about 125KΩ @3V for key matrix function usually.

Example 3-1 Setting PA, PC

```
MVLA        0xC        ; PA0、PA1 output low, PA2、PA3 input  pull-high
MVAM        PA
MVLA        0x3        ; PC0、PC1 input  pull-high, PC2、PC3 output low
MVAM        PC
```

3.6 LVD Register

The LVD register is used to recognize the status of supply voltage. The LVD[0] is read only and points out that the voltage is higher/lower than optioned voltage 2.2V. The data is 1 if higher; otherwise 0. The LVD[1] is POR Flag and set to 1 by any reset sources but LVR. Even the voltage is lower than 1.7V to cause LVR, the POR flag won't be forced to high unless the power disappears. The flag can be cleared by setting 0 to its register.

Category	Bit	Description
LVD	[0]	LVD flag
	[1]	POR flag

3.7 INT

3.7.1 System Base Timer Polling

Reading the 4-bit data of BTF acquires the value of the BT counter. The NY9UP08A provides 4 different base timer intervals for polling: 0.064ms, 0.128ms, 0.256ms and 16.384ms, shown as Figure 3-1. The value of time means the period, so polling a data toggle means half time of the interval. And the base timer restarts to count from 0, when CBT instruction is used.

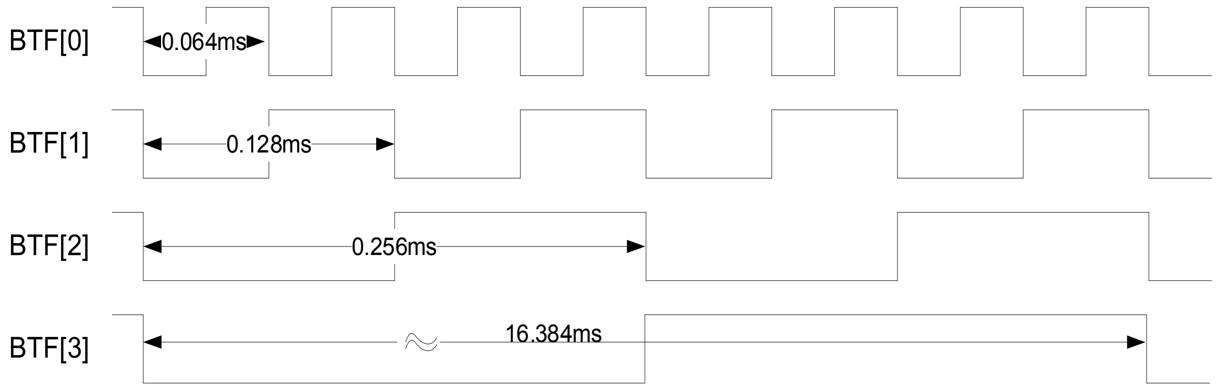


Figure 3-1 INT timing diagram

3.7.2 Interrupt Source

As mentioned in the section 2.6, the only one interrupt has 7 interrupt sources including 4 different BT intervals, the TOF, the RXFF and the RXRF. The INT0 register is interrupt enable/disable control register. The INTF0 is the interrupt flag, and it can be cleaned by writing 0 to related bit. The BT flag is base timer interrupt source select by writing, and timer status can be read.

3.7.3 Flag Clear

Using instruction CWDT0 and CWDT1 to clear the WDT.

Writing 0x0 to the INTF0[0] clears the Base Timer flag.

Writing 0x0 to the INTF0[1] clears the TOF.

Writing 0x0 to the INTF0[2] clears the RXFF.

Writing 0x0 to the INTF0[3] clears the RXRF.

3.8 IR Control Register

3.8.1 TM0CS

The TM0CS register indicates the Timer0 clock source of IR carrier frequency. The different clock sources combined with a variety of value of the 12-bit Timer0 registers will cause different IR carrier frequency.

TM0CS		Timer0 clock source
[1]	[0]	
0	0	1M Hz
0	1	2M Hz
1	0	4M Hz
1	1	8M Hz(Default)

3.8.2 TM0_x / TM0_Dutx

The Timer0 includes a 12-bit initial data latch and a 12-bit downward counter. Users can set initial timer value by writing TM0_L, TM0_M, TM0_H, and latch will be updated by writing TM0_H. The counter value can be reloaded from the initial data latch by executing CLRTM0. As the counter underflow, counter is automatically reloaded from the initial data latch. When reading TM0_x, users have to read TM0_H first and system will save LSB 8-bit data of timer counter value into TM0_M / TM0_L at same time.
With the different Timer0 initial value, the IR carrier frequency will be affected correspondingly; TM0_Dutx decides the duty of IR carrier. In addition, the timer will be stopped counting if the timer is turned off (ONOFF[0] equals to 0).
The TM0_Dutx register are used to store 8-bit duty value of Timer0, Writing TM0_DutL first and update after write TM0_DutH for setting duty value.

In order to generate IR carrier, user need to select Timer0 source at first, then set the 12-bit value of Timer0 and 8-bit value of duty. Then set IRCTRL[1:0]=2'b10 to enable IR TX and carrier. At last need to turn on Timer0. At this time, the timer starts to countdown from the set value, if counter meets the duty value, the IR pin start to sink current driving IRLED transmitting. When Timer downward counter reach 0, IR pin cancel sinking current status, until next timer value matching the value of duty, and IR pin start to sink again. The carrier will generate automatically through IR pin with specific duty.

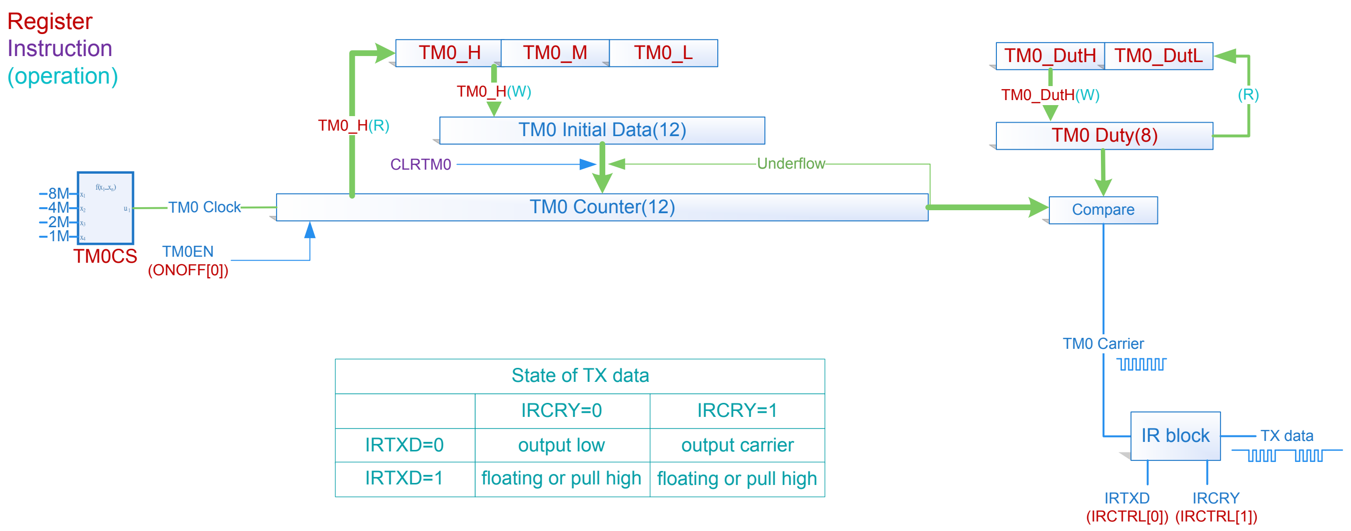
The equation of calculating the timer value to generate specific frequency carrier is shown below.

TM=(Ftcs/Fca)-1

TM: Timer0 value in decimal

Ftcs: Frequency of Timer0 clock

Fca: Frequency of carrier



State of TX data		
	IRCRY=0	IRCRY=1
IRTXD=0	output low	output carrier
IRTXD=1	floating or pull high	floating or pull high

Figure 3-2 Timer0 diagram for TX

3.8.3 TM0_Capx / TM0_CapRH

The TM0_Capx and TM0_CapRH registers are used to store current Timer0 counter value, when the RX rising or falling edge flag (INTF0[3:2]) equals to high, User can catch IR carrier frequency and duty from IR pin by this function when IR RX enable(ONOFF[1]=1). The TM0_CapL, TM0_CapM, anf TM0_CapH registers capture the counter vlaue at RX falling edge; The TM0_CapRH, TM0_DutH, anf TM0_DutL registers capture the counter vlaue at RX rising edge. Users can sellect RX falling edge capture only or RX falling & rising edge capture both by setting the ONOFF[2] register.

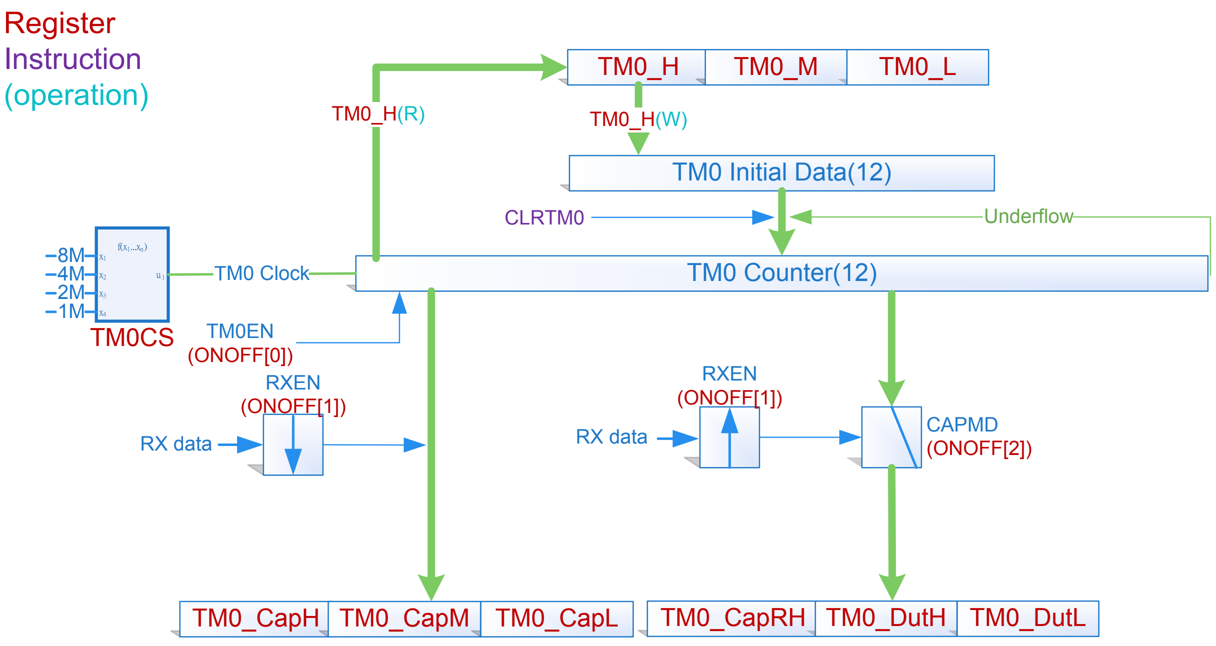


Figure 3-3 Timer0 diagram for RX

Example 3-2 Generate 38KHz (about 50% duty) carrier wave from IR

```
L_TST_IR:
          MVLA      0x3           ; Set Timer0 source = 8M
          MVAM      TM0CS
          MVLA      0x2           ; Setting Timer0 Data low 4-bit
          MVAM      TM0_L
          MVLA      0xD           ; Setting Timer0 Data middle 4-bit
          MVAM      TM0_M
          MVLA      0x0
          MVAM      TM0_H         ; Setting Timer0 Data high 4-bit
          MVLA      0x9           ; Setting Timer0 duty low 4-bit
          MVAM      TM0_DutL
          MVLA      0x6           ; Setting Timer0 duty high 4-bit
          MVAM      TM0_DutH
          CLRTM0                  ; Initial Timer0 counter
          MVLA      0x1           ; TX Pull high resistor = 3.7K
          MVAM      TXOPT
          MVLA      0x2           ; Enable TX data and IR carrier
          MVAM      IRCTRL
          MVLA      0x1           ; Enable Timer0 and TX
          MVAM      ONOFF
```

Example 3-3 Capture carrier wave from IR

```
L_TST_IR_RX_ FREQ:
          MVLA      0x3        ; Set Timer0 source = 8M
          MVAM      TM0CS
          MVLA      0x2        ; RX sensitivity select = 2uA[20cm], enable 0.5us RX filter
          MVAM      RXOPT
          MVLA      0x2        ; Enable Timer0 and RX, use falling edge capture only
          MVAM      ONOFF
          MVLA      0x1        ; Disable TX data and IR carrier
          MVAM      IRCTRL
          MVLA      0x4        ; Enable RX falling edge interrupt
          MVAM      INT0
L_TST_IR_RX_ FREQ_CAL:
          CWDT0
          CWDT1
          SETC                 ; Calculated frequency and dump to IO
          MVLA      0xF
          SUBA      0x42
          MVAM      PC
          MVLA      0xF
          SUBA      0x41
          MVAM      PB
          MVLA      0xF
          SUBA      0x40
          MVAM      PA
          JMP       L_TST_IR_RX_ FREQ_CAL
V_SYS_IntSrvRX:                ; RX interrupt
          CLRTM0               ; Initial Timer0 counter
L_Freq_Capture:
          MVMA      INTF0      ; Check next RX falling
          ANDL      b'0100'
          SZEZ
          JMP       L_Freq_Capture
          MVMA      TM0_CapH
          MVAM      0x40
          MVMA          TM0_CapM
          MVAM          0x41
          MVMA          TM0_CapL
          MVAM          0x42
          IRET
```

Example 3-4 Timer overflow

```
L_TST_INT_TOF:
          MVLA          0xF                 ; Setting Timer0 Data low 4-bit
          MVAM          TM0_L
          MVLA          0xF                 ; Setting Timer0 Data middle 4-bit
          MVAM          TM0_M
          MVLA          0xF                 ; Setting Timer0 Data high 4-bit
          MVAM          TM0_H
          CLRTM0
          MVLA          0x1                 ; Enable Timer0
          MVAM          ONOFF
          MVLA          0x2                 ; Set INT on, detect Timer0 overflow in interrupt process
          MVAM          INT0
```

3.8.4 IRCTRL and ONOFF

The IRCTRL register includes the control of IR TX data enable / disable, the IR carrier disable / enable and read status of TX or RX data from IR PAD.

The ONOFF register includes the Timer0 enable/disable, TX / RX enable and Timer0 counter capture mode selection.The ONOFF[3] register can adjust the duty in the carrier frame to be complete since the TX data low period is not always divisible by the carrier frequency.

Category	Bit	Description
IRCTRL	[0]	IR TX data enable / disable
	[1]	IR carrier disable / enable
	[3]	Read status from IR pad
ONOFF	[0]	Timer0 disable / enable
	[1]	IR TX enable / IR RX enable
	[2]	Timer0 counter capture mode select RX falling edge only / RX falling & rising edge both (TM0_Cap / TM0_Cap & TM0_CapRH + TM0_Dut)
	[3]	Adjust carrier No adjust / Adjust carrier frame to complete duty

3.8.5 TXOPT and RXOPT

The TXOPT register controls the TX output status. The TX sink current scale is about 600mA for large current and 450mA for normal. Users can also define the IR PAD with or without pull-high resistor when no sinking as TX enable, and the pull-high resistor can be set as 3.7KΩ or 37KΩ.

The RXOPT register controls the RX sensitivity and RX filter for noise immunity.

Category	Bit	Description
TXOPT	[0]	TX sink current select: Large current / Normal current
	[2]	TX pull high select when no sinking With pull-high / Without pull-high
	[3]	TX pull high resistor value select: 3.7K / 37K
RXOPT	[1:0]	00: RX sensitivity 10cm 01: RX sensitivity 5cm 10: RX sensitivity 20cm 11: RX sensitivity 15cm(Default)
	[2]	RX filter select 0.5us filter / No filter

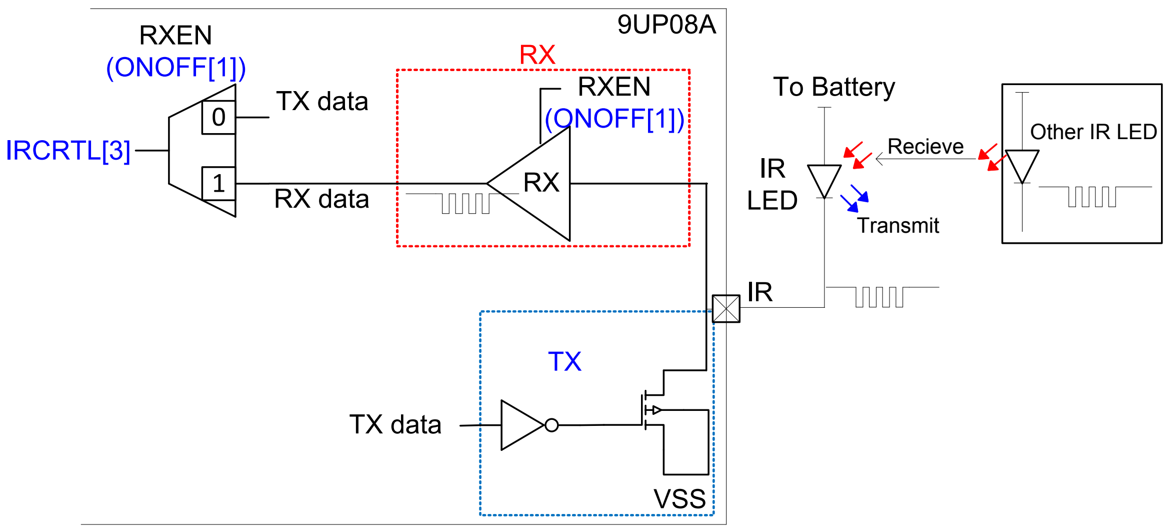


Figure 3-4 TX/RX control & signal phase diagram

Example 3-5 Drive TX with no carrier

```
        MVLA        0x5         ; TX without pull high
        MVAM        TXOPT
        MVLA        0x1         ; Enable TX data and disable IR carrier
        MVAM        IRCTRL
```

3.9 MTP Control Register

3.9.1 MTPC & MTPPT

The MTPC register includes the MTP control of CERS, PGEN, RDEN and PGFG.

Category	Bit	Description
MTPC	[0]	PGFG : MTP page program and erase finish flag From 0 to 1, Page program or erase is finished, can be cleaned by write 0.
	[1]	RDEN: MTP read enable control. 0: MTP read disabled--> save power 1: MTP read enabled--> need to delay 10us to read first data

Category	Bit	Description
	[2]	PGEN: MTP Page program enable control, Write 1 to enable page program
	[3]	CERS: MTP Chip erase Write 1 to enable chip erase

The CERS is the control bit of MTP chip erase. When the CERS is enable, the MTP begin to erase whole MTP storage space. The time is about 9ms, when erase is finished, and the PGFG will go high at the time. And each byte of MTP storage space changes to 0xFF.

The PGEN is the control bit of MTP page program. It is necessary to set MTPG_L & MTPG_H and set MTP page data to SRAM storage space ($1A0~$1BF) at first, before setting the PGEN to 1. The chip starts the page erase at first, and then dump the SRAM data ($1A0~$1BF) to the page 16 byte data. Next wait the PGFG changing from 0 to 1, the page program is finished.

The RDEN is MTP read control bit. When the RDEN enable, it need to delay 10us to read first MTP data by ROD method, and ROD2 is always read as 0x3. The MTP address is from $2000~$23FF. It will save whole chip power, when disable MTP read function.

The PGFG is the flag of MTP page program or chip erase. It needs to clean PGFG before enable CERS or PGEN.

The MTPPT is the MTP write & erase protect register. Before set the CERS & PGEN register, always write 0xA to MTPPT to release the MTP protect. After writing 0xA to MTPPT, the MTP will release protect in 8 instruction time for erasing or programming.

Note : For saving power, users have to disable MTP read function before entering halt mode.

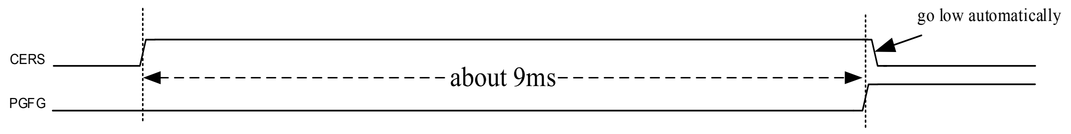


Figure 3-5 MTP chip erase function diagram

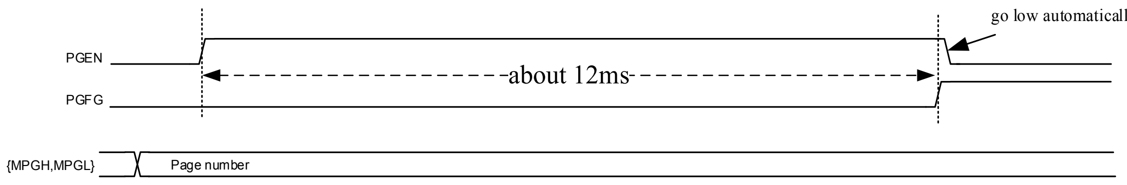


Figure 3-6 MTP page program function diagram

Example 3-6 Read MTP data

```
L_MTP_Read:
        MVLA        0x2
        MVAM        RPT3
        MVLA        0x0
        MVAM        RPT2
        MVLA        0x0
        MVAM        RPT1
        MVLA        0x0
        MVAM        RPT0
        LDPR                         ; Load RPT to DPR
        RD                           ; Data Table Read to [ROD2, ROD1, ACC]
```

3.9.2 MTPG_L & MTPG_H

The MTPG_L & MTPG_H register are MTP page number setting during page programming. Users must set page number before page program enable.

Category	Bit	Description
MTPG_L	[3:0]	MTP Page low 4-bit
MTPG_H	[1:0]	MTP Page high 2-bit

Example 3-7 MTP Chip erase

```
L_MTP_Erase:
        MVLA        0xA
        MVAM        MTPPT                ; Enable MTP operation
        MVLA        0x8
        MVAM        MTPC                 ; Start MTP erase
L_MTP_Erase_Loop:
        CWDT0
        CWDT1
        NOP
        MVMA        MTPC
        ANDL        b'0001'
        SZEZ
        JMP         L_MTP_Erase_Loop     ; Wait MTP erase
```

Example 3-8 MTP page program

```
L_MTP_PageProgram:
        MVLA        0x2                  ; Setting MTP page 0x22
        MVAM        MTPG_H
        MVAM        MTPG_L
        MVLA        0xA
        MVAM        MTPPT                ; Enable MTP operation
        MVLA        0x4
        MVAM        MTPC                 ; Start page program
L_MTP_ PageProgram _Chk:
        CWDT0
        CWDT1
        MVMA        MTPC
        ANDL        b'0001'
        SZEZ
        JMP         L_MTP_ PageProgram _Chk   ; Wait MTP program
```

3.10 Power Saving Mode

The relationship between power saving mode, reset & normal mode is shown below.

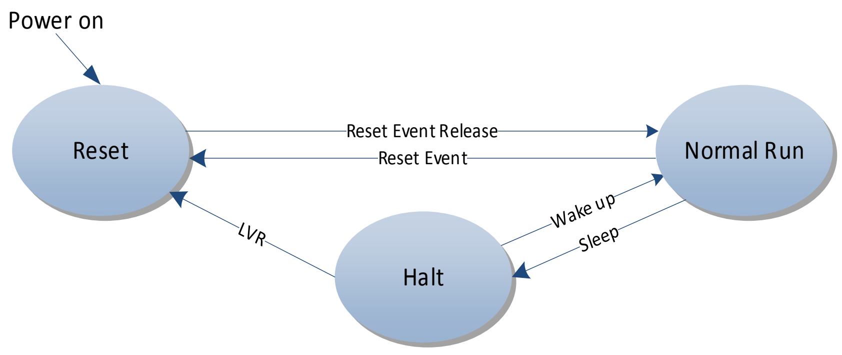


Figure 3-7 Power Saving Mode Flow Diagram

3.10.1 Halt Mode

The system enters the halt mode if the HALT command executed. The halt mode is also known as the sleep mode. As implied by the name, the IC falls asleep and the system clock is completely turned off, so all the IC functions are halted and it minimizes the power consumption.

The only way to wake-up the sleeping system is an input port wake-up. The IC keeps monitoring the input pads during the halt mode. If the input status of any input pad changes to low, the system will be woken-up. Then the succeeding instructions after the HALT instruction will be executed after the wake-up stable time (about 60us). So before executing the HALT instruction, users have to keep in mind that the input port status is high.

If the IC is waked-up from the halt mode by the occurrence of LVR, it goes into the reset procedure.

Example 3-9 Halt Mode operation

```
L_HALT_Loop:
            CWDT0
            CWDT1                                           ; Clear watch dog
            MVMA        PA                                  ; Confirm PA is high status
            SANL        0xF
             JMP        L_PB_CHK
             JMP        L_HALT_Proc
L_PB_CHK:
            MVMA        PB                                  ; Confirm PB is high status
            SANL        0xF
            JMP         L_PC_CHK
            JMP         L_HALT_Proc
L_PC_CHK:
            MVMA        PC                                  ; Confirm PC is high status
            SANL        0xF
            JMP         L_PD_CHK
            JMP         L_HALT_Proc
L_PD_CHK:
            MVMA        PD                                  ; Confirm PD0 is high status
            SANL        0x1
            JMP         L_HALT_Loop
L_HALT_Proc:
            NOP
            NOP
            NOP
            HALT                                           ; Enter Halt mode
            NOP                                            ; If input status I/O change to low, wake up
            NOP
            NOP
            JMP         L_HALT_Loop
```

3.10.2 T-type Scan Mode

In T-type scanning application, each port (PA~PD) can be selected as scan key independently by option PXx (X=A~C, x=0~3) & PD0. It works as input with pull-high resistor and output fixed frequency low pulse in halt mode. Any of the keys touch would cause system wake-up.

Meanwhile, the frequency of key scan can be adjusted by option codes, such as about 15.625Hz, 31.25Hz, 62.5Hz and 125Hz.

Chapter 4. Instruction Set

4.1 Instruction Classified Table

Item	Inst.	Op1	Op2	Operation	Inst. Length	Exec. Cycle	Oper. Flag	Flag Affected
				Arithmetic Instructions				
1	INCM	6m		{C,M} = M +1	1/2	1/2		C, Z
2	DECM	6m		{C,M} = M - 1	1/2	1/2		C, Z
3	ADDA	6m		{C,A} = A + M + C	1/2	1/2	C	C, Z
4	XORA	6m		A = A ^ M	1/2	1/2		Z
5	ANDA	6m		A = A & M	1/2	1/2		Z
6	ORA	6m		A = A | M	1/2	1/2		Z
7	SUBA	6m		{C,A} = A – M - (B)	1/2	1/2	C	C, Z
8	MVAM	6m		Move A to M	1/2	1/2		
9	MVMA	6m		Move M to A	1/2	1/2		Z
10	RRM	6m		Right Rotate M with C	1/2	1/2	C	C,Z
11	RLM	6m		Left Rotate M with C	1/2	1/2	C	C,Z
12	SUBL	4L		{C,A} = A – L - (B)	1	1	C	C, Z
13	ADDL	4L		{C,A} = A + L + C	1	1	C	C, Z
14	XORL	4L		A = A ^ L	1	1		Z
15	ANDL	4L		A = A & L	1	1		Z
16	ORL	4L		A = A | L	1	1		Z
17	MVLA	4L		Move L to A	1	1		
18	RRA			Right Rotate A	1	1		
19	RLA			Left Rotate A	1	1		
20	RRC			Right Rotate A with C	1	1	C	C, Z
21	RLC			Left Rotate A with C	1	1	C	C, Z
22	CLRC			Clear Carry flag	1	1		C
23	SETC			Set Carry flag	1	1		C
24	INCA			A = A + 1	1	1		C, Z
25	DECA			A = A - 1	1	1		C, Z
				Conditional Instructions				
26	SAGT	4L		Skip when A > L	1	1/2/3		
27	SALT	4L		Skip when A < L	1	1/2/3		
28	SANL	4L		Skip if A != L	1	1/2/3		
29	SCEZ			Skip if C == 0	1	1/2/3	C	
30	SCNZ			Skip if C != 0	1	1/2/3	C	
31	SZEZ			Skip if Z == 0	1	1/2/3	Z	
32	SZNZ			Skip if Z != 0	1	1/2/3	Z	
33	SBEZ	2b		Skip when A[b] = 0	1	1/2/3		
				Other Instructions				
34	NOP			No operation	1	1		
35	CWDT0			Clear WDT Step1	1	1		

Item	Inst.	Op1	Op2	Operation	Inst. Length	Exec. Cycle	Oper. Flag	Flag Affected
36	CWDT1			Clear WDT Step2	1	1		
37	HALT			Enter HALT mode	1	1		
38	LDPR			Load RPT to DPR	1	2		
39	RBPR			Read DPR to RPT	1	2		
40	RD			Data Table Read to [ROD2, ROD1, ACC]	1	3		
41	RDI			Data Table Read to [ROD2, ROD1, ACC]， DPR = DPR + 1	1	3		
42	JMP	13a		Jump to Address	2	2		
43	CALL	13a		Jump to Address, and Move PC+2 to RPT	2	2		
44	IRET			Return from interrupt	2	2		
45	RJMP			Move RPT to PC	1	2		
46	RBPC			Move PC+1 to RPT	1	2		
47	SMKI			Set Mask Interrupt	1	1		
48	CMKI			Clear Mask Interrupt	1	1		
49	INCR			[RPT2[0], RPT1, RPT0]+1	1	3		
50	CLRTM0			Timer0 reload latch value	1	1		
51	CBT			Clear Base timer	1	1		

A: 4-bit Accumulator data

B: 1-bit borrow flag data, shared with carry flag,

B=~C

C: 1-bit carry flag data

M: 4-bit RAM or register data

L: 4-bit immediately literal data

Z: 1-bit zero flag data

RPT: Multi-function register pointer

DPR: Data pointer register

RODx: ROM data access register

PC: Program counter address pointer

STK: Interrupt dedicated stack address pointer

a: ROM address

b: bit address

m: RAM or memory register address

4.2 Instruction Descriptions

4.2.1 Arithmetic Instructions

INCM m

Function: Add 1 to M of address m, and save the result back to M.

Operation: M ← M + 1

Operand: 0x0≤m0≤0x3F
0x40≤m1≤0x1FF

Words: 1 (m0), 2 (m1)

Cycles: 1 (m0), 2 (m1)

Operative Flags: None

Flags Affected: C, Z

Example: INCM m0
Before Instruction
M0=0x0
After Instruction
M0=0x1, C=0, Z=0

DECM m

Function: Subtract 1 from M of address m, and save the result back to M.

Operation: M ← M - 1

Operand: 0x0≤m0≤0x3F
0x40≤m1≤0x1FF

Words: 1 (m0), 2 (m1)

Cycles: 1 (m0), 2 (m1)

Operative Flags: None

Flags Affected: C, Z

Example: DECM m0
Before Instruction
M0=0x0
After Instruction
M0=0xF, C=0, Z=0

ADDA m

Function: Add M to A with C and the result is saved back to A.

Operation: { C, A } ← A + M + C

Operand: 0x0≤m0≤0x3F
0x40≤m1≤0x1FF

Words: 1 (m0), 2 (m1)

Cycles: 1 (m0), 2 (m1)

Operative Flags: C

Flags Affected: C, Z

Example :ADDA m0
Before Instruction
A=0x7, M0=0xA, C=0
After Instruction
A=0x1, M0=0xA, C=1, Z=0

XORA m

Function: Exclusive OR A with M of address m, and the result is saved back to A.

Operation: A ← A ^ M

Operand: 0x0≤m0≤0x3F
0x40≤m1≤0x1FF

Words: 1 (m0), 2 (m1)

Cycles: 1 (m0), 2 (m1)

Operative Flags:None

Flags Affected: Z

Example: XORA m0
Before Instruction
A=0x3, M0=0xB
After Instruction
A=0x8, M0=0xB, Z=0

ANDA m

Function: AND A with M of address m, and save the result back to A.

Operation: A ← A & M

Operand: 0x0≤m0≤0x3F
0x40≤m1≤0x1FF

Words: 1 (m0), 2 (m1)

Cycles: 1 (m0), 2 (m1)

Operative Flags: None

Flags Affected: Z

Example: ANDA m0

Before Instruction

A=0x7, M0=0xA

After Instruction

A=0x2, M0=0xA, Z=0

SUBA m

Function : Subtract M of address m from A with B, i.e. The (B) quantity effectively implements a borrow capability for multi-precision subtractions.

Operation : { C, A } = A - m - B

Operand : 0x0≤m0≤0x3F
0x40≤m1≤0x1FF

Words: 1 (m0), 2 (m1)

Cycles: 1 (m0), 2 (m1)

Operative Flags: C

Flags Affected: C, Z

Example : SUBA m0

Before Instruction

A=0xA, M0=0x2, C=1

After Instruction

A=0x8, M0=0x2, Z=0, C=1

ORA m

Function: Inclusive OR A with M of address m, and save the result back to M.

Operation: A ← A | M

Operand: 0x0≤m0≤0x3F
0x40≤m1≤0x1FF

Words: 1 (m0), 2 (m1)

Cycles: 1 (m0), 2 (m1)

Operative Flags: None

Flags Affected: Z

Example: ORA m0

Before Instruction

A=0x3, M0=0x8

After Instruction

A=0xB, M0=0x8, Z=0

MVAM m

Function: Move A to M of address m.

Operation: M ← A

Operand: 0x0≤m0≤0x3F
0x40≤m1≤0x1FF

Words: 1 (m0), 2 (m1)

Cycles: 1 (m0), 2 (m1)

Operative Flags: None

Flags Affected: None

Example: MVAM m0

Before Instruction

A=0x8

After Instruction

M0=0x8

MVMA m

Function: Move M of address m to A.

Operation: A ← M

Operand: 0x0≤m0≤0x3F

0x40≤m1≤0x1FF

Words: 1 (m0), 2 (m1)

Cycles: 1 (m0), 2 (m1)

Operative Flags: None

Flags Affected: Z

Example: MVMA m0

Before Instruction

M0=0x8

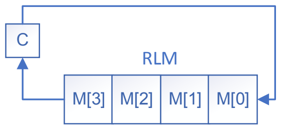
After Instruction

A=0x8

RLM m

Function: Left Rotate M with C.

Operation: { C, M[3], M[2], M[1], M[0] } → { M[3], M[2], M[1], M[0], C }



Operand: 0x0≤m0≤0x3F

0x40≤m1≤0x1FF

Words: 1 (m0), 2 (m1)

Cycles: 1 (m0), 2 (m1)

Operative Flags: C

Flags Affected: C, Z

Example: RLM m0

Before Instruction

m0=0xE, C=1

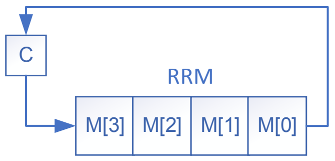
After Instruction

m0=0xD, C=1, Z=0

RRM m

Function: Right Rotate M with C.

Operation: { C, M[3], M[2], M[1], M[0] } → { M[0], C, M[3], M[2], M[1] }



Operand: 0x0≤m0≤0x3F

0x40≤m1≤0x1FF

Words: 1 (m0), 2 (m1)

Cycles: 1 (m0), 2 (m1)

Operative Flags: C

Flags Affected: C, Z

Example: RRM m0

Before Instruction

m0=0x3, C=1

After Instruction

m0=0x9, C=1, Z=0

SUBL L

Function : Subtract L from A with B, i.e. The (B) quantity effectively implements a borrow capability for multi-precision subtractions.

Operation : { C, A } = A - L - B

Operand: 0x0≤L≤0xF

Words : 1

Cycles : 1

Operative Flags: C

Flags Affected: C, Z

Example : SUBL 0x2

Before Instruction

A=0xA, L=0x2, C=1

After Instruction

A=0x8, Z=0, C=1

ADDL L

Function: Add L and C to A.

Operation: A ← A + L+ C

Operand: 0x0≤L≤0xF

Words: 1

Cycles: 1

Operative Flags: C

Flags Affected: C, Z

Example: ADDL 0x9

Before Instruction

A=0xB, C=1

After Instruction

A=0x5, C=1, Z=0

ANDL L

Function: AND A with L.

Operation: A ← A & L

Operand: 0x0≤L≤0xF

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: ANDL 0xA

Before Instruction

A=0x0

After Instruction

A=0x0, Z=1

XORL L

Function: Exclusive OR A with L.

Operation: A ← A ^ L

Operand: 0x0≤L≤0xF

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: XORL 0xA

Before Instruction

A=0x9

After Instruction

A=0x3, Z=0

ORL L

Function: Inclusive OR A with L.

Operation: A ← A | L

Operand: 0x0≤L≤0xF

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: ORL 0xA

Before Instruction

A=0x9

After Instruction

A=0xB, Z=0

MVLA L

Function: Move L to A.

Operation: A ← L

Operand: 0x0≤L≤0xF

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MVLA 0x5

Before Instruction

A=0x8

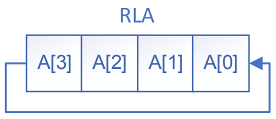
After Instruction

A=0x5

RLA

Function : Left rotate A.

Operation: { A[3], A[2], A[1], A[0] }→ { A[2], A[1], A[0], A[3] }



Operand: None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example :RLA

Before Instruction

A =0x3

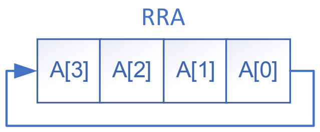
After Instruction

A =0x6

RRA

Function : Right rotate A.

Operation: { A[3], A[2], A[1], A[0] } → { A[0], A[3], A[2], A[1] }



Operand: None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example :RRA

Before Instruction

A =0xE

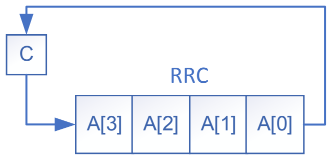
After Instruction

A =0x7

RRC

Function : Right rotate A with C.

Operation: { C, A[3], A[2], A[1], A[0] } → { A[0], C, A[3], A[2], A[1] }



Operand: None

Words : 1

Cycles : 1

Operative Flags: C

Flags Affected: C, Z

Example :RRC

Before Instruction

A =0x3 ,C=1

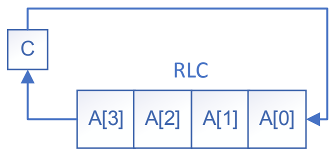
After Instruction

A =0x9 ,C=1 ,Z=0

RLC

Function : Left rotate A with C.

Operation: { C, A[3], A[2], A[1], A[0]} → { A[3], A[2], A[1], A[0], C }



Operand: None

Words : 1

Cycles : 1

Operative Flags: C

Flags Affected: C, Z

Example :RLC

Before Instruction

A =0xE ,C=0

After Instruction

A =0xC ,C=1 ,Z=0

SETC

Function: Set C to 1.

Operation: C ← 1

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C

Example: SETC

Before Instruction

C=0

After Instruction

C=1

CLRC

Function: Clear C to 0.

Operation: C ← 0

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C

Example: CLRC

Before Instruction

C=1

After Instruction

C=0

INCA

Function: Add 1 to A.

Operation: A ← A + 1

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C, Z

Example: INCA

Before Instruction

A=0xF

After Instruction

A=0x0, C=1, Z=1

DECA

Function: Subtract 1 from A.

Operation: A ← A - 1

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C, Z

Example: DECA

Before Instruction

A=0x1

After Instruction

A=0x0, C=1, Z=1

4.2.2 Conditional Instructions

SAGT L

Function: Skip the next instruction if A greater than L.

Operation: Skip next if A > L

Operand: 0x0≤L≤0xF

Words: 1

Cycles: 1, (2, 3)

Operative Flags: None

Flags Affected: None

Example: SAGT 0x8

Inst1

Inst2

After Instruction

If A=(or <) 0x8, \`Inst1’ is executed.

If A>0x8, \`Inst1’ is discarded, and \`Inst2’ is executed.

SALT L

Function: Skip the next instruction if A greater than L.

Operation: Skip next if A < L

Operand: 0x0≤L≤0xF

Words: 1

Cycles: 1, (2, 3)

Operative Flags: None

Flags Affected: None

Example: SALT 0x8

Inst1

Inst2

After Instruction

If A=(or >) 0x8, \`Inst1’ is executed.

If A<0x8, \`Inst1’ is discarded, and \`Inst2’ is executed.

SANL L

Function: Skip the next instruction if A not equal L.

Operation: Skip next if A != L

Operand: 0x0≤L≤0xF

Words: 1

Cycles: 1, (2, 3)

Operative Flags: None

Flags Affected: None

Example: SANL 0x8

Inst1

Inst2

After Instruction

If A≠0x8, `Inst1' is discarded, and `Inst2' is executed

If A=0x8, `Inst1' is executed.

SCNZ

Function: Skip the next instruction if C equal to 1.

Operation: Skip next if C = 1

Operand: None

Words: 1

Cycles: 1, (2, 3)

Operative Flags: C

Flags Affected: None

Example: SCNZ

Inst1

Inst2

After Instruction

If C≠0x1, `Inst1' is executed.

If C=0x1, `Inst1' is discarded, and `Inst2' is executed.

SCEZ

Function: Skip the next instruction if C equal to 0.

Operation: Skip next if C=0

Operand: None

Words: 1

Cycles: 1, (2, 3)

Operative Flags: C

Flags Affected: None

Example: SCEZ

CALL a1

CALL a2

After Instruction

If C≠0 `CALL a1' is executed

If C=0 `CALL a1' is discarded, and `CALL a2' is executed

SZEZ

Function: Skip the next instruction if Z equal to 0.

Operation: Skip next if Z=0

Operand: None

Words: 1

Cycles: 1, (2, 3)

Operative Flags: Z

Flags Affected: None

Example: SZEZ

CALL a1

CALL a2

After Instruction

If Z≠0 `CALL a1' is executed

If Z=0 `CALL a1' is discarded, and `CALL a2' is executed

SZNZ

Function: Skip the next instruction if Z equal to 1

Operation: Skip next if Z = 1

Operand: None

Words: 1

Cycles: 1, (2, 3)

Operative Flags: Z

Flags Affected: None

Example: SZNZ

Inst1

Inst2

After Instruction

If Z≠0x1, `Inst1’ is executed.

If Z=0x1, `Inst1’ is discarded, and `Inst2’ is executed.

SBEZ b

Function: Skip the next instruction if A[b] is not set.

Operation: Skip next if A[b]=0.

Operand: 0x0≤b≤0x3

Words: 1

Cycles: 1, (2, 3)

Example: SBEZ 0x3

Inst1

Inst2

After Instruction

If A[3]=1, `Inst1’ is executed.

If A[3]=0, `Inst1’ is discarded, and `Inst2’ is executed.

4.2.3 Other Instructions

NOP

Function: No operation.

Operation: None

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: NOP

After Instruction

No operation for 1 cycle.

CWDT0

Function: Clear Watch Dog Timer Step1.

Operation: Step1 for clear Watch Dog Timer

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CWDT0

Before Instruction

WDT counter = ???

After Instruction

WDT counter = ???

CWDT1

Function: Clear Watch Dog Timer Step2.

Operation: Watch dog counter ← 0x0

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CWDT0

CWDT1

Before Instruction

WDT counter = ???

After Instruction

WDT counter = 0x0

Note : The CWDT0/CWDT1 instructions have to be executed step by step, othwise the watch dog timer won't be clear.

LDPR

Function: Load RPT to DPR.

Operation: DPR ← RPT

Operand: None

Words: 1

Cycles: 2

Operative Flags: None

Flags Affected: None

Example: LDPR

Before Instruction

RPT=0x0321

After Instruction

DPR=0x0321

HALT

Function: Enter the halt (sleep) mode.

Operation: Stop system clock

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: HALT

After Instruction

The system enters the halt mode and the system clock is halted.

RBPR

Function: Load DPR to RPT.

Operation: RPT ← DPR

Operand: None

Words: 1

Cycles: 2

Operative Flags: None

Flags Affected: None

Example: RBPR

Before Instruction

DPR=0x0321

After Instruction

RPT=0x0321

RD

Function: Read ROM data out to A and ROD using the DPR as address (data pointer).

Operation: A ← ROM data [3:0]

ROD1 ← ROM data [7:4]

ROD2 ← ROM data [9:8]

Operand: None

Words: 1

Cycles: 3

Operative Flags: None

Flags Affected: None

Example: RD

After Instruction

A=ROM[3:0] @ DPR

ROD1=ROM[7:4] @ DPR

ROD2=ROM[9:8] @ DPR

RDI

Function: Read ROM data out to A and ROD using the DPR as address (data pointer).

Operation: A ← ROM data [3:0]

ROD1 ← ROM data [7:4]

ROD2 ← ROM data [9:8]

DPR← DPR + 1

Operand: None

Words: 1

Cycles: 3

Operative Flags: None

Flags Affected: None

Example: RDI

After Instruction

A=ROM[3:0] @ DPR

ROD1=ROM[7:4] @ DPR

ROD2=ROM[9:8] @ DPR

DPR = DPR +1

JMP a

Function: Unconditionally jump by a direct address a.

Operation: PC ← a

Operand: 0x0≤a≤0x1FFF

Words: 2

Cycles: 2

Operative Flags: None

Flags Affected: None

Example: JMP a1

Before Instruction

PC=a0

After Instruction

PC=a1

CALL a

Function: Call subroutine by a direct address a, and save next address to RPT.

Operation: RPT ← PC+2

PC ← a

Operand: 0x0≤a≤0x1FFF

Words: 2

Cycles: 2

Operative Flags: None

Flags Affected: None

Example: CALL a1

Before Instruction

PC=a0

After Instruction

PC=a1, RPT=a0+2

IRET

Function: Return from the interrupt sub-routine.

Operation: PC ← STK

Operand: None

Words: 2

Cycles: 2

Operative Flags: None

Flags Affected: None

Example: IRET

Before Instruction

PC=a0

After Instruction

PC=STK

ACC, PG, C, and Z are restored to the values that are backed up when entering the ISR.

RBPC

Function: Read address in PC to RPT.

Operation: RPT ← PC+1

Operand: None

Words: 1

Cycles: 2

Operative Flags: None

Flags Affected: None

Example: RBPC

Before Instruction

PC=0x1234

After Instruction

RPT=0x1235

RJMP

Function: Load RPT to PC. Unconditionally jump by the indirect address RPT. The address should be loaded into RPT first.

Operation: PC ← RPT

Operand: None

Words: 1

Cycles: 2

Operative Flags: None

Flags Affected: None

Example: RJMP

Before Instruction

RPT=0x0321

After Instruction

PC=0x0321

SMKI

Function: Disable interrupt entrance

Operation: Mask all interrupt

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: SMKI

Before Instruction

Interrupt entrance is enable

After Instruction

Interrupt entrance is disable

CMKI

Function: Enable interrupt entrance

Operation: Non-mask interrupt

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CMKI

Before Instruction

Interrupt entrance is disable

After Instruction

Interrupt entrance is enabled

CLRTM0

Function: Reload Timer0 initial latch data to Timer0 counter.

Operation:Timer0 counter reload

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CLRTM0

Before Instruction

Timer0 counter=0xAFF

Timer0 initial data latch=0x321

After Instruction

Timer0 counter=0x321

INCR

Function: Increase RPT by 1.

Operation: { RPT2[0], RPT1, RPT0 } ← { RPT2[0], RPT1, RPT0 } +1

Operand: None

Words: 1

Cycles: 3

Operative Flags: None

Flags Affected: None

Example: INCR

Before Instruction

RPT=0x0134

After Instruction

RPT=0x0135

CBT

Function: Clear whole base timer counter to zero.

Operation: BT ← 0

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CBT

Before Instruction

BT=?

After Instruction

BT=0