九齊科技股份有限公司

Nyquest Technology Co., Ltd.

USER MANUAL

NY9T Series

4-bit MCU with LED Controller and Capacitive Touch of Low Stand-by Current

Version 1.3

Aug. 31, 2016

Revision History

Version#	***Date***	***Description***	***Modified Page***
1.0	2015/05/29	Formally release.	-
1.1	2015/08/20	1. Modify PWM-IO description. 2. Update DC characteristics.	7, 17 9
1.2	2016/02/24	1. Modify NY9T001 I/O function. 2. Modify NY9T008 I/O function. 3. Modify Interrupt description. 4. Modify the recommended R_{ADJ} value.	18 20 26 40
1.3	2016/08/31	1. Add CDC, PWM Frame Rate and Hysteresis description. 2. Modify R_{ADJ} description and add notice of PCB layout for R_{ADJ}. 3. Add CDC parameter to DC Characteristics. 4. Add CDC pin description and the IC body with this function. 5. Add 4KHz and 66KHz of LED PWM Frame Rate. 6. Add Enlarge Scan Time. 7. Modify R_{ADJ} description and recommended R_{ADJ} value. 8. Add Touch-Key Hysteresis Level.	7 8 9 17, 18, 20 28, 29 34, 35 42 43

Table of Contents

Chapter 1. Introduction

1.1 General Description

The NY9T series IC is a powerful 4-bit micro-controller based LED control processor with capacitive touch sensing function. The RISC MCU architecture is very easy to program and control, and various applications can be easily implemented. There are total 45 instructions, and most of them are executed in one cycle.

Through the accurate 400kHz internal oscillation of +/- 1% tolerance, an external resister is not required. Furthermore, a RC16K mode is designed to save power consumption and a HALT is to minimize the power.

There are maximum 8 channels of PWM-IO LED output. PWM-IO output can provide 4 kinds of analog output current cooperating with 256-level software digital current control to drive different brightness LED. The optional constant current is provided to protect LED, and it's not necessary to add the external current-limit resistor. The interactive software-developing tool of "*Q-Visio*" is user-friendly and quick for LED pattern programming.

The Touch-Key function replaces the mechanical switch or button. Multiple Touch-Keys is from 1 touch key up to 16 touch keys. A built-in LDO regulator for touch sensor can provide a stable capacitive sensing design for touch application. The auto distinguishing methodology supports the diversity key pattern in a thickness range of non-conductive dielectric material, and time sharing Touch-Key scan also reduces the power consumption. An innovative software tool "*Q-Touch*" is provided to develop the different touch applications.

1.2 Features

- Wide operating voltage range: 2.0V to 6.0V.
- 4-bit RISC type micro-controller with 45 instructions.
- Maximum 16Kx10-bit ROM, program and LED data share the same ROM space.
- Maximum 96x4-bit RAM, divided into 2 pages.
- 400KHz instruction frequency.
- Precisely 400KHz oscillator with built-in resistor Rosc for operating mode. (+/- 1% tolerance)
- RC16K mode to save the power consumption. (+/- 30% tolerance)
- HALT mode to minimize power, less than 1uA standby current.
- Very low standby current for Touch-Key scan to reduce the power consumption.
- Low Voltage Reset (1.8V), Watch-Dog Reset and I/O Port reset are all supported to protect the system.
- Maximum 24 flexible I/Os with optional function: General purpose I/O, PWM-IO output or Touch-Key input.
- General purpose I/O's direction is controlled by registers. For the input option, users can select one of three kinds of option: input without pull-low resistor, input with weak pull-low resistor or input with strong pull-low resistor. For the output Port, users can select the CMOS output or Open Drain output with normal sink and drive output.

- Maximum 8-channel PWM-IO output of PE0~3 and PF0~3 shared with I/Os. Each output has other 3 kinds of output option: large sink current, constant sink current (CSC) or drive current.
- PE0~3 and PF0~3 output current option: (PE0~3 and PF0~3)
  a. Large sink current: 4 kinds of current, 100%, 83%, 50%, 33%.
  b. Constant sink current: 4 kinds of current, 100%, 83%, 50%, 33%.
  c. Drive current: only 100% current and constant drive current (CDC) only for PE1 and PE2.
- Programmable PWM-IO has maximum 256 levels LED brightness with software control. The LED frame rate can be selected from 128Hz, 4KHz and 66KHz.
- Maximum 16 Touch-Keys of PA0~3, PB0~3, PC0~3 and PD0~3 shared with I/Os.
- Touch-Key supports simultaneous multi-key or single-key touch function configured by option.
- Hardware Auto-Calibration function of Touch-Key is provided for touch accuracy to prevent the environment change.
- Touch-Key wake-up can be any key or specified key (Key1/PA0) configured by option.
- 2 kinds of scan modes for Touch-Key: normal mode and slow mode, selected by software programming. At normal mode, sequential scan can shorten the touch detecting time but more power consumed. Oppositely, at slow mode, time sharing scan can reduce power consumption but longer wake-up time.
- 4 kinds of debounce time for Touch-Key press/release detection configured by option in order to keep away from the noise interference.
- 2 kinds of hysteresis level: Small Hysteresis Level and Large Hysteresis Level for all Touch-Keys configured by option.
- 2 kinds of sensitivity mode: Preset mode and Custom mode for all Touch-Keys configured by option.
- 8 kinds of sensitivity levels can be selected by software programming or external R_{ADJ} resistor.
- Built-in LDO regulator (2.1V) for a stable power that avoids the abnormal sensitivity and false detection.

1.3 Product List

IC Type	***I/O***	***ROM (bits)***	***RAM (bits)***	***Touch-Key***	***PWM-IO***	***LED Brightness Level***
NY9T001A	4	4K x 10	48 x 4	1	3	256
NY9T004A	8	8K x 10	48 x 4	4	4	256
NY9T008A	16	12K x 10	96 x 4	8	8	256
NY9T016A	24	16K x 10	96 x 4	16	8	256

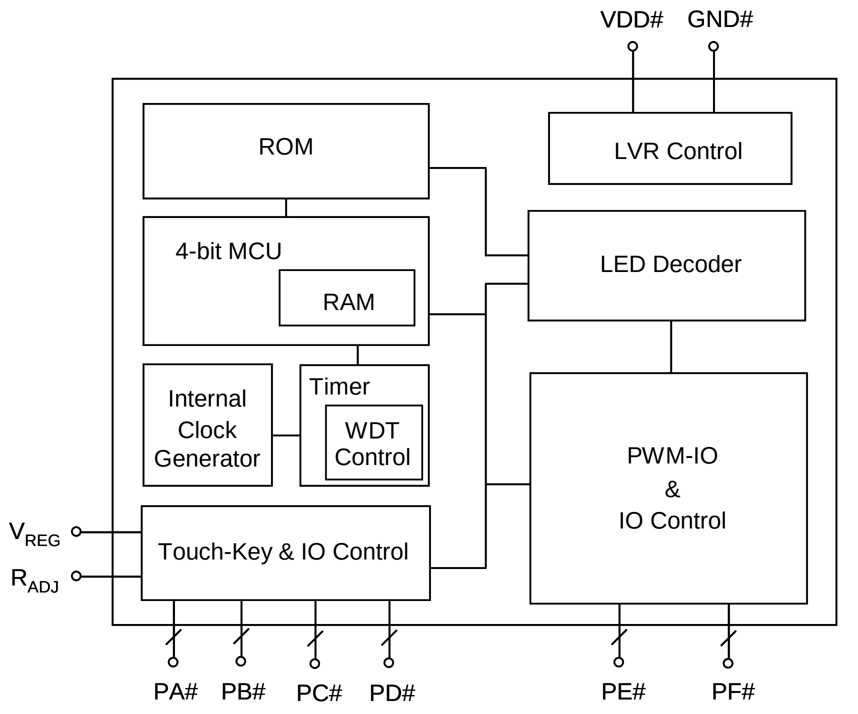
* NY9T001A – Touch-Key: *PA0,* PWM-IO: *PE0~PE2.*

* NY9T004A – Touch-Key: *PA0~PA3*, PWM-IO: *PE0~PE3.*

* NY9T008A – Touch-Key: PA0~PA3, PB0~PB3, PWM-IO: *PE0~PE3, PF0~PF3.*

* NY9T016A – Touch-Key: *PA0~PA3, PB0~PB3, PC0~PC3, PD0~PD3,* PWM-IO: *PE0~PE3, PF0~PF3.*

1.4 Block Diagram



1.5 Pad Description

Pad	***ATTR.***	***Description***
VDD#	Power	Positive power
GND#	Power	Negative power
V_{REG}	power	Regulator input. Connect a 1000pF (102) cap to GND or keep floating.
PA0~3	I/O	Bit 0~3 for Port A
PB0~3	I/O	Bit 0~3 for Port B
PC0~3	I/O	Bit 0~3 for Port C
PD0~3	I/O	Bit 0~3 for Port D
PE0~3	I/O	Bit 0~3 for Port E
PF0~3	I/O	Bit 0~3 for Port F
R_{ADJ}	I	Connect an external resistor to adjust the sensitivity, or keep floating.

* PA0~PA3, PB0~PB3, PC0~PC3 and PD0~PD3 shared with Touch-Key1~16.

* PE0~PE3 and PF0~PF3 shared with PWM-IO.

* PE0 is shared with reset pin.

* PCB designer should notice that the length of R_{ADJ} trace from IC to resistor must be within 0.5cm, and also avoids any parallel trace, or parallel GND nearby or at different layer.

1.6 Electrical Characteristics

The following lists the electrical characteristics of the NY9T EV chip. All the product's properties must refer to each part's datasheet.

1.6.1 Absolute Maximum Rating

Symbol	***Parameter***	***Rated Value***	***Unit***
VDD - VSS	Supply voltage	-0.5 ~ +6.5	V
Vin	Input voltage	VSS–0.3V ~ VDD+0.3	V
Top	Operating Temperature	0 ~ +70	°C
Tst	Storage Temperature	-25 ~ +85	°C

1.6.2 DC Characteristics

Symbol	***Parameter***		***V_{DD}***	***Min.***	***Typ.***	***Max.***	***Unit***	***Condition***
V_{DD}	Operating voltage			2.0	3	6.0	V	400KHz
I_{SB}	Supply Current	Halt mode	3		0.1	0.5	uA	Sleep, no load LVR disable
			4.5		0.1	0.5		
I_{OP}		RC16K mode	3		0.7		uA	16KHz, no load
			4.5		2.2			
		Standby mode	3		1.6		uA	Key1 (PA0) wakeup, 16 samples per frame
			4.5		3.3			
		Operating mode	3		350		uA	400KHz, no load
			4.5		500			
I_{IH}	Input current (Internal pull-low)	Weak (1M ohms)	3		3		uA	$V_{IH}=V_{DD}$
			4.5		8			
		Strong (100k ohms)	3		30			
			4.5		80			
I_{OH}	Output drive current		3		-10		mA	V_{OH}=2.0V
			4.5		-16			V_{OH}=3.5V
	Output constant drive current		3		-3.4		mA	V_{OH}=2.0V
			4.5		-3.8			V_{OH}=3.5V
I_{OL}	Output normal sink current		3		20		mA	V_{OL}=1.0V
			4.5		30			
	Output large sink current (100%)		3		50			
			4.5		70			
	Output constant sink current (100%)		3		19			
			4.5		20			
∆F/F	Frequency deviation by voltage drop (400kHz)		3		-0.5		%	$\frac{Fosc(3.0v)-Fosc(2.4v)}{Fosc(3v)}$
			4.5		1.0			$\frac{Fosc(4.5v)-Fosc(3.0v)}{Fosc(4.5v)}$
∆F/F	Frequency lot deviation (400kHz)		3	-1		1	%	$\frac{Fmax(3.0v)-Fmin(3.0v)}{Fmax(3.0v)}$
F_{OSC}	Oscillation Frequency		-	370	400	430	KHz	V_{DD}=2.0~6.0V

1.6.3 Touch-Key Scan Current *(Sample_Time=1ms, Calibration_Period=4s, T_A=25ºC, unless otherwise specified)*

Body (Wakeup Key)	Touch Keys	Samples of Frame	VDD	Touch-Key Scan Current			
				Normal Mode	PA0 Wakeup	Any-Key Wakeup	
					Slow Mode	Slow Mode	Slow-Green Mode
NY9T001A	1-key	8	3.0V	4.1 uA	1.6 uA		
			4.5V	6.3 uA	3.3 uA		
		16	3.0V	2.4 uA	1.2 uA		
			4.5V	4.3 uA	2.8 uA		
NY9T004A	2-key	8	3.0V	10.1 uA	1.6 uA	2.8 uA	
			4.5V	12.7 uA	3.3 uA	4.8 uA	
		16	3.0V	5.5 uA	1.2 uA	2.0 uA	
			4.5V	7.5 uA	2.8 uA	3.4 uA	
	4-key	8	3.0V	13.0 uA	1.6 uA	3.7 uA	
			4.5V	17.0 uA	3.3 uA	6.0 uA	
		16	3.0V	7.0 uA	1.2 uA	2.3 uA	
			4.5V	9.6 uA	2.8 uA	4.1 uA	
NY9T008A	6-key	8	3.0V	21.5 uA	1.8 uA	6.0 uA	4.1 uA
			4.5V	27.0 uA	3.5 uA	8.8 uA	6.4 uA
		16	3.0V	11.5 uA	1.4 uA	3.5 uA	2.7 uA
			4.5V	15.1 uA	3.0 uA	5.7 uA	4.6 uA
	8-key	8	3.0V	24.8 uA	1.8 uA	6.9 uA	4.1 uA
			4.5V	31.6 uA	3.5 uA	9.8 uA	6.4 uA
		16	3.0V	13.0 uA	1.4 uA	4.1 uA	2.7 uA
			4.5V	17.2 uA	3.0 uA	6.2 uA	4.6 uA
NY9T016A	10-key		3.0V				
			4.5V				
		16	3.0V	20.0 uA	1.6 uA	5.8 uA	2.9 uA
			4.5V	25.0 uA	3.3 uA	8.5 uA	4.9 uA
	12-key		3.0V				
			4.5V				
		16	3.0V	21.7 uA	1.6 uA	6.3 uA	2.9 uA
			4.5V	27.1 uA	3.3 uA	9.0 uA	4.9 uA
	14-key		3.0V				
			4.5V				
		16	3.0V	23.3 uA	1.6 uA	6.7 uA	2.9 uA
			4.5V	29.3 uA	3.3 uA	9.5 uA	4.9 uA
	16-key		3.0V				
			4.5V				
		16	3.0V	24.6 uA	1.6 uA	7.1 uA	2.9 uA
			4.5V	31.8 uA	3.3 uA	11.8 uA	4.9 uA

Chapter 2. Hardware Architecture

2.1 Hardware Summary Table

Name	***Function***	***Address***
PC	Program counter	
LPR	LED pointer register	
RPT	Multi-function register pointer	M[0x0~0x3]
RAM	96 nibbles RAM	
ROM	Program & data ROM	
INT	Interrupt function register	M[0x6~0x7]
PWM	PWM-IO LED output	
INST	Instruction registers	
INST DEC	Instruction decoder	
LED	LED function register	M[0x8]
LED DEC	LED decoder	
TOUCH	Touch-Key function register	M[0x9~0xF]
Clock Generator	Ring oscillator clock generator	
WDT	Watch-dog timer and reset generator	
ROD1	ROM[7:4] data access register	M[0x4]
ROD2	ROM[9:8] data access register	M[0x5]
SYS Reset	System reset generator	
POR	Power reset generator	
LVR	Low voltage reset generator	
ACC	4-bit accumulator	
ALU	4-bit arithmetic logic unit	
C	Carry flag for arithmetic	
Z	Zero flag for arithmetic	
I/O Ports	I/O Port register	T[0x0~0xB]

M[ ] : Memory register and the hex number 0x? between the brackets is its address.

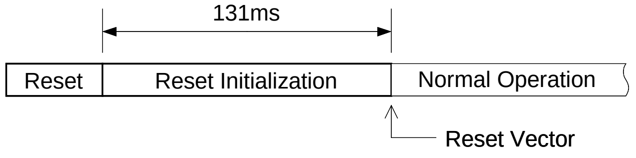
T[ ] : System register and the hex number 0x? between the brackets means its address.

2.2 Clock Generator

The clock generator is a built-in Ring oscillator, and users can only select the internal resistor (INT-R). The INT-R oscillator accuracy is up to ±1%. The system clock is 400KHz for instructions running.

In normal operating mode, the system clock is 400kHz. User can implement sorts of application in this mode. On the other hand, users can select RC16K mode or Halt mode to save power consumption.

2.3 System Reset



Reset Initialization Procedure

2.3.1 Power-On Reset (POR)

After Power-on, the power-on reset initialization will automatically be set out. After the system leaves the reset initialization procedure, it enters the normal operation and the program counter starts at the reset vector. The power on stable time is about 131ms.

2.3.2 Low Voltage Reset (LVR)

When the system enters the normal operation, the power supply voltage must be kept in an effective working voltage range. When the power supply voltage is lower than the effective working voltage range, the system can’t work properly. To prevent the system crash, a low voltage detector is built in NY9T. When the detector detects a harmful low voltage supply, it will cause a low voltage reset. The so-called “low voltage” point is about 1.8V. The NY9T provides mask option to decide LVR global enable or disable. The other mask option decide enable or disable at HALT mode.

2.3.3 Watch-Dog Timer Reset (WDTR)

To recover from program malfunction, the NY9T 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 56ms. Users can use CWDT instruction to clear WDT.

It should be noted that WDT must be cleared carefully. Theoretically, it can't be placed at any loop other than main loop. One thing should be kept in mind, clear the WDT only when you make sure the execution of the MCU is still under the control of the program.

2.3.4 IO Port External Reset

The PE0 pin can be optioned as a reset pin. A reset pin should always be pulled-down in normal operation, whether to use the built-in internal pull-low resister option or to use the external one on PCB with the floating input option. When the reset pin rises to the power level, it generates an external reset.

2.4 Address Pointer

The NY9T micro-controller contains a program counter (PC) and LED pointer (LPR). The length of each address pointer is 14-bit maximum, varies by the product parts. Not like the PC, the initial value of the LPR is unknown.

2.4.1 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 0x000000) after the system reset, and its value is increased by one every instruction cycle unless changed by a branch instruction.

Inst./Event	***Function***
JMP	Changes the LSB 14-bit of PC, and the reminder MSB bits keep their value.
CALL	The same functions as JMP. In addition, it pushes PC+2 to RPT.
LDPC	Loads RPT to PC, so users can execute a long jump or jump by table.
RBPC	Reads back PC to RPT.

2.4.2 LED Pointer Register (LPR)

The NY9T is also an 8-channel LED processor, a LED pointer is necessary for playing LED. When PLAY is executed, the system loads RPT to LPR. So users have to move the start address of the LED to RPT first. Besides, users can read LPR back by RBLP instruction.

2.5 Arithmetic Logic Unit (ALU)

The NY9T provides a 4-bit arithmetic logic unit with a 4-bit accumulator to perform logic, unsigned arithmetic, data transfer and conditional branch operation. We have 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.

2.5.1 ALU Instruction Summary

2.5.1.1 Logic Instruction

Instruction	***Function***	***Flag Influenced***
XORM m	A ← M[m] ⊕ A	Z
ANDM m	A ← M[m] & A	Z
ORM m	A ← M[m] | A	Z
XORL L	A ← A ⊕ L	Z
ANDL L	A ← A & L	Z
ORL L	A ← A | L	Z
ROLA	C ← A[3]; A ← {A[2:0], C}	C, Z
RORA	C ← A[0]; A ← {C, A[3:1]}	C, Z

2.5.1.2 Arithmetic Instruction

Instruction	***Function***	***Flag Influenced***
INCM m	A ← M[m] + 1	C, Z
DECM m	A ← M[m] - 1	C, Z
ADDM m	A ← A + M[m] + C	C, Z
ADDL L	A ← A + L + C	C, Z
INCA	A ← A + 1	C, Z
DECA	A ← A - 1	C, Z

2.5.1.3 Data Transfer Instruction

Instruction	***Function***	***Flag Influenced***
MVAM	M[m] ← A	
MVMA	A ← M[m]	Z
MVRM	M[m] ← R[r]	
MVMR	R[r] ← M[m]	
MVLR	R[r] ← L	
MVLA	A ← L	
MVAT	T[t] ← A	
MVTA	A ← T[t]	
RSTC	C ← 0	C
SETC	C ← 1	C

The width of the memory register address ‘r’ of MVRM, MVMR, and MVLR command is 2-bit and the MSB of the memory register is forced to be 0. So users can only use the three commands to handle RPT0~3. The width of the RAM or memory register address ‘m’ of MVRM, and MVMR command is 4-bit and the MSB 2-bit of the address is forced to be 0x3. Users can only use the two instructions to handle RAM or memory register of address 0x10~0x3F, but the RAM page is still working. The width of the PORT register address ‘t’ of MVAT, and MVTA command is 4-bit. Users can only use the two instructions to handle PORT register of address 0x0~0xB, but the RAM page is still working.

2.5.1.4 Conditional Branch Instruction

Instruction	***Function***	***Flag Influenced***
CPAL	Skip if A = L	
CNAL	Skip if A != L	
CPAB	Skip if (A & L) = 0	
CPCZ	Skip if C = 0	
CPZZ	Skip if Z = 0	
CNCZ	Skip if C != 0	C
CNZZ	Skip if Z != 0	C

A conditional branch instruction compares two data and skips next instruction if they are equal. The skip operation is making an instruction NOP, not jump across it.

⊕ : Exclusive OR bitwise logical operation

& : AND bitwise logical operation

| : OR bitwise logical operation

A : 4-bit Accumulator data

C : 1-bit carry flag data

L : 4-bit immediately literal data

M[m] : 4-bit RAM or memory register data at memory address m

R[r] : 4-bit memory register data at register address r

Z : 1-bit zero flag data

2.5.1.5 Interrupt Instruction

Instruction	***Function***	***Flag Influenced***
IRET	Return from interrupt subroutine	C, Z

2.5.2 ALU Related Status Flag

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.

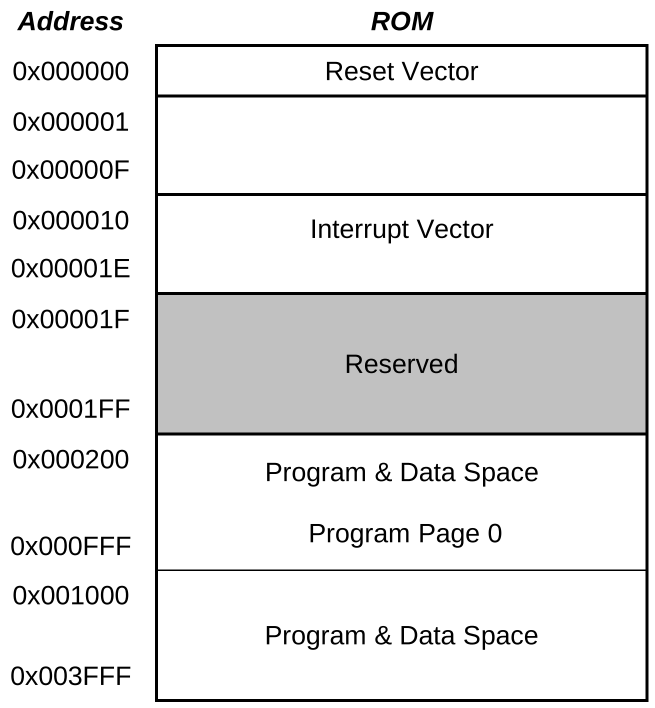
Besides RSTC and SETC commands directly assign the value of the carry flag, C is influenced by the arithmetic result. C means carry and also means the complement of borrow. If the addition operation larger than 0xF, C=1, and C=0 if the result ≦15. If the subtraction operation smaller than 0, C=0, and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

2.6 Memory Organization

There are maximum 16K words ROM, 96 nibbles of RAM and 14 nibbles of dedicated system control register. The RAM page register (PG) is without address allocation, and they can only be accessed by the special instruction MPG.

2.6.1 ROM

A large program/data/LED single ROM is provided and its structure is shown below. The reserved region contains system information and can't be utilized by users. The program page is limited by the unconditional branch instruction: JMP and CALL. Because it can only handle 14-bit length address of ROM, the program page size is 16K words.



2.6.2 RAM

Each page of RAM contains 48 nibbles. The NY9T provides maximum 96 nibbles of 2 pages. The page number (PG) register of RAM defined by the MPG instruction, and its initial value is 0. The address for RAM is 0x10~0x3F.

2.6.3 System Register

The system registers are located at the first 16 nibbles of the memory space, and the addresses of them are not affected by the memory page register (PG). By the memory mapping, user can perform any register addressing operations on them.

Address	***Name***	***R/W***	***Description***
0	RPT0	R/W	Register Pointer [3:0]
1	RPT1	R/W	Register Pointer [7:4]
2	RPT2	R/W	Register Pointer [11:8]
3	RPT3	R/W	Register Pointer [14:12]
4	ROD1	R/W	ROM data bit [7:4] access register
5	ROD2	R/W	ROM data bit [9:8] access register
6	INTE	R/W	Interrupt enable register
7	INTF	R/W	Interrupt flag register
8	LEDCTL	R/W	LED control register
9	SENS	R/W	Sensitivity selection register
A	TPCTL	R/W	Touch-Key control register
B	IPCTL	R/W	IP control register
C	KEYD0	R	Touch-Key1~4 status register
D	KEYD1	R	Touch-Key 5~8 status register
E	KEYD2	R	Touch-Key 9~12 status register
F	KEYD3	R	Touch-Key 13~16 status register

2.7 IO Ports

There are 24 I/O Ports at most, designated as PAx through PFx, and x=0~3. All the I/O Ports can be configured as input or output by registers, i.e. normal I/O. For the input Port, we provide an internal pull-low register option for convenience. For the output port, users can select the normal sink current output or drive current output. When the PE0~3 or PF0~3 pins are optioned as normal I/O, the output sink current can be optioned as normal sink current, large sink current (33%, 50%, 83%, 100%) or constant sink current (33%, 50%, 83%, 100%). The PE1 and PE2 pins can be optioned as constant drive current (CDC) output by body (refer to the following table).

The PA0~3, PB0~3, PC0~3 or PD0~3 pins can be optioned as a Touch-Key individually. The PE0~3 or PF0~3 pins can be optioned as PWM-IO output pin individually, and each output pin can be optioned as large sink current (33%, 50%, 83%, 100%), constant sink current (33%, 50%, 83%, 100%) or drive current output. The PE1 and PE2 pins can be optioned as constant drive current (CDC) output by body (refer to the following table). The PE0 pin can be optioned as an external reset pin, which can possess a pull-low resister.

The pull-low resister of all the I/O Ports has two kinds of option: weak and strong. The weak one is about 1MΩ @3V for normal application and the strong one is about 100kΩ @3V usually for key matrix function. Each I/O can be configured as the weak or strong pull-low resister individually.

The following table shows the I/O Port options for MaskROM products.

NY9T001A with 4 I/Os

I/O	***Option***		***Description***
PA0	Touch-Key input		Enable
PE0~PE2	Normal I/O	Input pull-low resister	Floating
			Weak
			Strong
		Output type	CMOS
			Open Drain
		Output Drive current	Constant Drive (PE1, PE2)
		Output sink current	Normal Sink
			Large Sink 33%
			Large Sink 50%
			Large Sink 83%
			Large Sink 100%
			Constant Sink 33%
			Constant Sink 50%
			Constant Sink 83%
			Constant Sink 100%
	PWM-IO output		Disable
			Large Sink 33%
			Large Sink 50%
			Large Sink 83%
			Large Sink 100%
			Constant Sink 33%
			Constant Sink 50%
			Constant Sink 83%
			Constant Sink 100%
			Drive
			Constant Drive (PE1, PE2)
	Reset input *(PE0 only)*		Disable
			Enable

NY9T004A with 8 I/Os

I/O	*Option*		*Description*
PA0~PA3	Normal I/O	Input pull-low resister	Floating
			Weak
			Strong
		Output type	CMOS
			Open Drain
	Touch-Key input		Disable
			Enable
PE0~PE3	Normal I/O	Input pull-low resister	Floating
			Weak
			Strong
		Output type	CMOS
			Open Drain
		Output sink current	Normal Sink
			Large Sink 33%
			Large Sink 50%
			Large Sink 83%
			Large Sink 100%
			Constant Sink 33%
			Constant Sink 50%
			Constant Sink 83%
			Constant Sink 100%
	PWM-IO output		Disable
			Large Sink 33%
			Large Sink 50%
			Large Sink 83%
			Large Sink 100%
			Constant Sink 33%
			Constant Sink 50%
			Constant Sink 83%
			Constant Sink 100%
			Drive
	Reset input *(PE0 only)*		Disable
			Enable

NY9T008A with 16 I/Os

I/O	*Option*		*Description*
PA0~PA3 PB0~PB3	Normal I/O	Input pull-low resister	Floating
			Weak
			Strong
		Output type	CMOS
			Open Drain
	Touch-Key input		Disable
			Enable
PE0~PE3 PF0~PF3	Normal I/O	Input pull-low resister	Floating
			Weak
			Strong
		Output type	CMOS
			Open Drain
		Output drive current	Constant Drive (PE2)
		Output sink current	Normal Sink
			Large Sink 33%
			Large Sink 50%
			Large Sink 83%
			Large Sink 100%
			Constant Sink 33%
			Constant Sink 50%
			Constant Sink 83%
			Constant Sink 100%
	PWM-IO output		Disable
			Large Sink 33%
			Large Sink 50%
			Large Sink 83%
			Large Sink 100%
			Constant Sink 33%
			Constant Sink 50%
			Constant Sink 83%
			Constant Sink 100%
			Drive
			Constant Drive (PE2)
	Reset input *(PE0 only)*		Disable
			Enable

NY9T016A with 24 I/Os

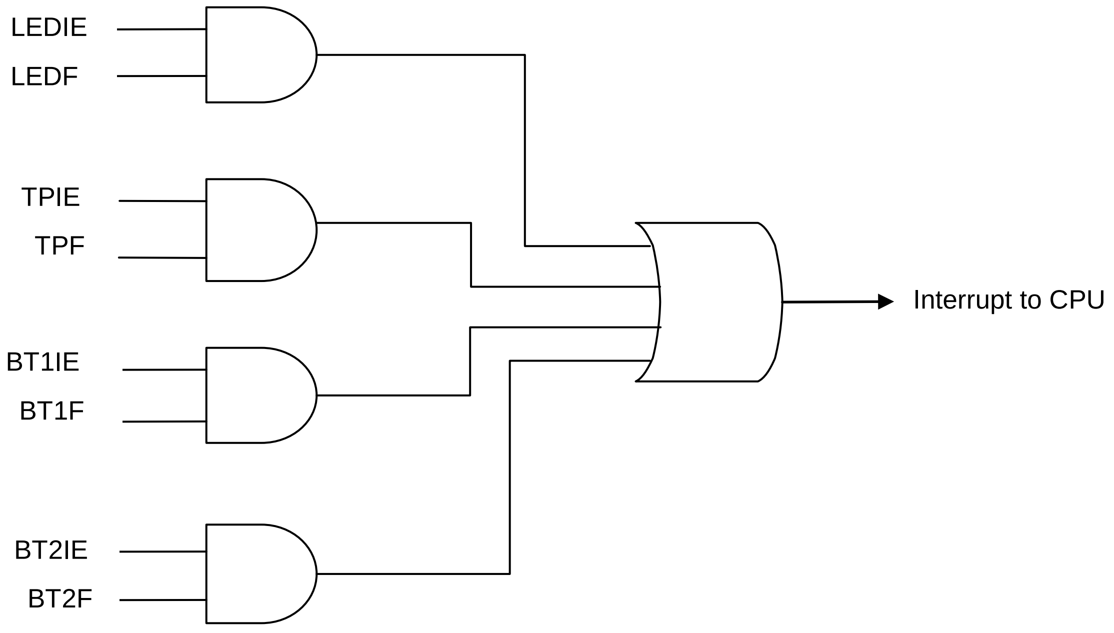
I/O	***Option***		***Description***
PA0~PA3 PB0~PB3 PC0~PC3 PD0~PD3	Normal I/O	Input pull-low resister	Floating
			Weak
			Strong
		Output type	CMOS
			Open Drain
	Touch-Key input		Disable
			Enable
PE0~PE3 PF0~PF3	Normal I/O	Input pull-low resister	Floating
			Weak
			Strong
		Output type	CMOS
			Open Drain
		Output sink current	Normal Sink
			Large Sink 33%
			Large Sink 50%
			Large Sink 83%
			Large Sink 100%
			Constant Sink 33%
			Constant Sink 50%
			Constant Sink 83%
			Constant Sink 100%
	PWM-IO output		Disable
			Large Sink 33%
			Large Sink 50%
			Large Sink 83%
			Large Sink 100%
			Constant Sink 33%
			Constant Sink 50%
			Constant Sink 83%
			Constant Sink 100%
			Drive
	Reset input *(PE0 only)*		Disable
			Enable

2.8 Interrupt Generator

The NY9T has 4 sources of interrupt. Two of the interrupt events are fixed intervals, which are derived from system base timer (BT). There is a system base timer, and it functions as long as the IC isn't in the halt mode. The NY9T provides 2 kinds of fixed intervals from the system base timer for interrupt source: 4ms/1ms and 4s/0.5s. When using interrupts, user must select interrupt source first and then turn on. However, the time interval from BT is enabled to first occurrence of interrupt may be not as accurate as specified due to NY9T characteristic.

The peripheral interrupt have both LED and Touch-Key function. When the corresponding interrupt enable bit is set, and the corresponding interrupt flag bit is set, the interrupt will vector immediately.

As an interrupt occurs, NY9T stores the accumulator (ACC), carry flag (C), zero flag (Z) and RAM page (PG) automatically. PG is controlled by the command MPG. Then NY9T move PC+2 to STK, and jump to the interrupt vector (0x000010). An interrupt routine finishes with an IRET instruction. The IC draws back ACC, C, Z and PG back, and moves STK to PC back to jump back the main program.



Chapter 3. System Control

3.1 Introduction

The LEDCTL is LED control related register. The Px and PxIO (x = A ~ F) are I/O Ports registers. TPCTL, SENS, KEYD0 ~ KEYD3 are Touch-Key control related registers. INTIE and INTF are the interrupt control related registers. The combination of RPT0~3 are multi-function register pointer. The C and Z are arithmetic associated flags. The PG is RAM access register. The ROD1 and ROD2 registers together with ACC are used to read the ROM data.

3.1.1 System Register Address Map

Addr	*Name*	*R/W*	*Bit*	*Data*	*Description*	*Initial*	*Wake-up*
0	RPT0	R/W	[3:0]	0/1	Register Pointer[3:0]	X	U
1	RPT1	R/W	[3:0]	0/1	Register Pointer[7:4]	X	U
2	RPT2	R/W	[3:0]	0/1	Register Pointer[11:8]	X	U
3	RPT3	R/W	[2:0]	0/1	Register Pointer[14:12]	X	U
4	ROD1	R/W	[3:0]	0/1	Register ROM Data[7:4]	X	U
5	ROD2	R/W	[1:0]	0/1	Register ROM Data[9:8]	X	U
6	INTIE	R/W	[3:0]	0/1	Interrupt enable register	X	U
7	INTF	R/W	[3:0]	0/1	Interrupt flag register	0	U
8	LEDCTL	R	[3:0]	0/1	LED control register	X	U
9	SENS	R/W	[3:0]	0/1	Touch sensitivity select register	0	U
A	TPCTL	R/W	[3:0]	0/1	Touch-Key control register	0	U
B	IPCTL	R/W	[3:0]	0/1	ENLARGE/RC16K_EN/ROV/SMALL	0	U
C	KEYD0	R	[3:0]	0/1	Key4~Key1 status	0	U
D	KEYD1	R	[3:0]	0/1	Key8~Key5 status	0	U
E	KEYD2	R	[3:0]	0/1	Key12~Key9 status	0	U
F	KEYD3	R	[3:0]	0/1	Key16~Key13 status	0	U
0	PA	R	[3:0]	0/1	PAIO=1, Read Port A input pad data	X	X
					PAIO=0, Read Port A output register	X	X
		W	[3:0]	0/1	Write to Port A output register	X	U
1	PAIO	R/W	[3:0]	0/1	Port A direction = Output / Input	Input	U
2	PB	R	[3:0]	0/1	PBIO=1, Read Port B input pad data	X	X
					PBIO=0, Read Port B output register	X	X
		W	[3:0]	0/1	Write to Port B output register	X	U
3	PBIO	R/W	[3:0]	0/1	Port B direction = Output / Input	Input	U
4	PC	R	[3:0]	0/1	PCIO=1, Read Port C input pad data	X	X
					PCIO=0, Read Port C output register	X	X
		W	[3:0]	0/1	Write to Port C output register	X	U
5	PCIO	R/W	[3:0]	0/1	Port C direction = Output / Input	Input	U

Addr	Name	R/W	Bit	Data	Description	Initial	Wake-up
6	PD	R	[3:0]	0/1	PDIO=1, Read Port D input pad data	X	X
					PDIO=0, Read Port D output register	X	X
		W	[3:0]	0/1	Write to Port D output register	X	U
7	PDIO	R/W	[3:0]	0/1	Port D direction = Output / Input	Input	U
8	PE	R	[3:0]	0/1	PEIO=1, Read Port E input pad data	X	X
					PEIO=0, Read Port E output register	X	X
		W	[3:0]	0/1	Write to Port E output register	X	U
9	PEIO	R/W	[3:0]	0/1	Port E direction = Output / Input (when Port E option as normal I/O)	Input	U
		R/W	[3:0]	0/1	LED channel output enable/disable (when Port E option as PWM-IO output)	disable	U
A	PF	R	[3:0]	0/1	PFIO=1, Read Port F input pad data	X	X
					PFIO=0, Read Port F output register	X	X
		W	[3:0]	0/1	Write to Port F output register	X	U
B	PFIO	R/W	[3:0]	0/1	Port F direction = Output / Input (when Port F option as normal I/O)	Input	U
		R/W	[3:0]	0/1	LED channel output enable/disable (when Port F option as PWM-IO output)	disable	U

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

X : Unknown

- : The flags R/W property explained in the related section 2.5

3.2 RPT

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

The RPT of NY9T is 14-bit long at most. Whether the bits of RPT are redundant or useful, users have to initialize all RPT(RPT[14:0]) to "0". The RPT will be frequently accessed because of its multi-functionality, so NY9T provides 3 instructions to accelerate the access of RPT0~3: MVRM, MVMR and MVLR.

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 LDPC to go back to the main program.

Inst./Event	Function
CALL	Pushes PC+2 to RPT.
LDPC	Loads RPT to PC.
RBPC	Reads back PC to RPT.
PLAY	Loads RPT to LPR.
RBLP	Reads back LPR to RPT.
RBRO	Use RPT as address to read ROM data.

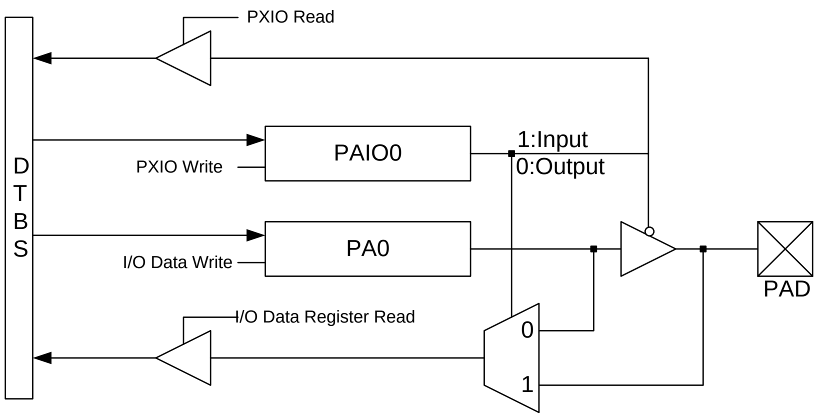
3.3 ROD

The NY9T provides the RBRO instruction to read the ROM data out. When RBRO is executed, the system takes the RPT as ROM address, and the ROM data is loaded to ROD2, 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 RBRO to read the data of the reserved ROM area out is unacceptable, and results in a reset. The RBRO instruction has a 1-bit operand 'n'. 0 means the RPT value keeps unchanged after RBRO, and 1 means the RPT adds 1.

3.4 I/O Ports Register

Before using an I/O Port, configure its direction register first. The execution result of the read / write operation perform on the data register depends on the direction register. When an I/O Port is configured as an input Port (PxIO = 1), the data read from the data register is the pad status. Otherwise, it reads out the data stored in the data register.

The data register of an input Port is only used for wake-up sequence. Because the difference between the pad and the register leads to a wake-up from the halt mode, users have to read the pad status and save back to the data register before entering the halt mode.



IO Port A0 Block Diagram

Contrastingly, the content of the data register is the pad status output by this output Port.

If a PWM-IO output is required, option an I/O Port as a PWM-IO Port first. If a Touch-Key is required, option an I/O Port as Touch-Key first. When I/O Port is assign to PWM-IO or Touch-Key, the I/O direction is not controlled by the direction register.

If users want to use PWM-IO output with constant sink current (CSC), the related I/O option must be set as constant sink current.

3.5 Interrupt

The interrupt flags are contained in the special function registers INTF. The corresponding interrupt enable bits are contained in special function registers INTIE. When an interrupt is responded to, the return address is pushed onto the stack and the PC is loaded with 10h.

Once in the interrupt service routine the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bit(s) must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

Control register	***Name***	***Mode***	***Bit***	***Function description***
INTIE	LEDIE	R/W	[0]	LED interrupt enable/disable
	TPIE	R/W	[1]	Touch-Key interrupt enable/disable
	BT2IE	R/W	[2]	BT2 interrupt enable/disable
	BT1IE	R/W	[3]	BT1 interrupt enable/disable
INTF	LEDF	R/W	[0]	LED interrupt flag
	TPF	R/W	[1]	Touch-Key interrupt flag
	BT2F	R/W	[2]	BT2 interrupt flag
	BT1F	R/W	[3]	BT1 interrupt flag

3.5.1 LED Interrupt

For LED interrupt flag bit (LEDF) is set, that means LED finish one section. If the LED corresponded interrupt enable bit (LEDIE) is set, IC vectors to interrupt immediately. LEDF can't write to "1" by software, it's set as "1" by hardware and cleared by software write to "0".

3.5.2 Touch-Key Interrupt

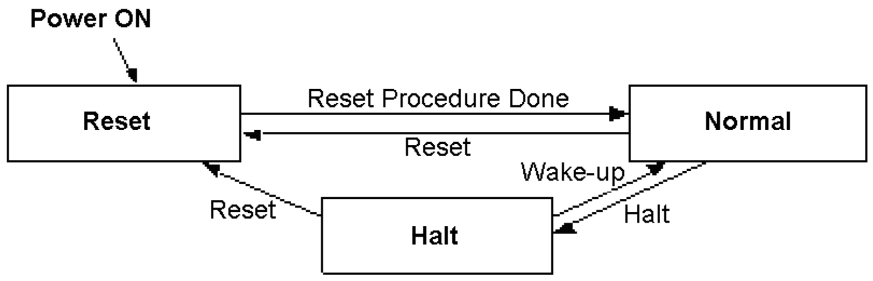
If there is any status change for Touch-Key, the Touch-Key interrupt flag (TPF) bit is set. If the Touch-Key corresponded interrupt enable bit (TPIE) is set, IC vectors to interrupt immediately. TPF can't be written to "1" by software, it can be set as "1" by hardware and cleared by software write to "0".

3.5.3 Base Timer (BT)

The NY9T provides 2-kinds of base time interrupt timing as 4ms/1ms, 4s/0.5s. The corresponded interrupt enable bits are BT1IE/BT2IE, and the corresponded interrupt flag bits are BT1F/BT2F. BT1F/BT2F can't write to "1" by software, it's set as "1" by hardware and cleared by software write to "0".

Options	***Selected items***
BT1S	Base time 1 is 4ms/1ms
BT2S	Base time 2 is 4s/0.5s

3.6 Power Saving Mode



Power Saving Mode Flow Chart

3.6.1 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 differs from the corresponding Port register, the system will be waked-up. Then the succeeding instructions after the HALT instruction will be executed after the wake-up stable time (about 64us). So before executing the HALT instruction, users have to keep in mind to store the current input Port statuses into Port registers.

If the IC is waked-up from the halt mode by a reset Port, it goes into the reset procedure.

3.6.2 RC16K Mode

The NY9T has built-in one RC16K oscillator for base time and Touch-Key function. The tolerance of RC16K frequency is up to ±30%. The RC16K_EN bit of IPCTL register must be set as “1” before executing RC16K base time or Touch-Key function. The working current of RC16K is very low (~2uA), and it’s suitable for low power timer application if users don’t care about the +/-30% frequency deviation. Normally, for the applications of interrupt wake-up by RC16K base time at halt mode, it’s not necessary to clear RC16K_EN as “0”. Besides, users can clear RC16K_EN bit to enter halt mode for power saving.

Control register	***Name***	***Mode***	***Bit***	***Function description***
IPCTL	RC16K_EN	R/W	[2]	Built-in RC16K enable/disable

Chapter 4. LED Control

4.1 LED Control

4.1.1 LED Play

The NY9T provides a hardware LED control and driver function. Users can play a LED section by setting the extension value to FOSC[1:0] and LED channel output register OEN[7:0] first, then executing the PLAY command. The playback will be continued until a section ends or users pause or stop it. When a section ends, the LEDF is set to "1" automatically, and users can observe the LEDIF to be aware of the LED playing state. STOP automatically pulls the LEDF to "0" and stops the LED data to the initial value (0x000). Write a "1" to the LEDCTL register PAUSE/RESUME bit to pause playing LED, and that keeps the last LED data unchanged. Write a "0" to the LEDCTL register PAUSE/RESUME bit to resume the LED playing. RBLP is a useful instruction to check the playing address of the LED data.

Step	***Process***	***Instruction***
1	Set LED start address in RPT3, RPT2, RPT1, RPT0	
2	Set extension value FOSC[1:0] and LED channel output register OEN[7:0]	
3	Start to play	PLAY
4	Pause: Write 1 to PAUSE/RESUME	
5	Resume: Write 0 to PAUSE/RESUME	
6	Stop playing	STOP

The LED output is an 8-bit hardware PWM. The NY9T provides 3 kinds of options of constant sink current (CSC), large sink current or drive current for each PWM-IO pin individually. The large sink current and constant sink current can be optioned as different current output of 100%, 83%, 50% and 33%, but the drive current and constant drive current (CDC) is only 100%.

The NY9T has maximum 8-channel PWM-IO output, optioned by mask. LED channel output control register decides which channel will be played.

4.1.2 LED Section

The LED start address is loaded to LPR when executing the PLAY command. The NY9T supports 8-bit PWM-IO data that means 256 levels resolution of LED brightness. The LED frame rate can be selected from 128Hz, 4KHz and 66KHz. Frame rate 128Hz and 4KHz can support up to 256 levels of brightness, however 66KHz only supports 32 levels resolutions. Basically, the LED section is encoded and combined by different LED samples, and any LED sample period can be one of 1/128, 1/64, 1/16 and 1/4 second.

LED sample period = (1 / frame rate) * n,   n=1, 2, 8, 32.

- The LED frame rate can select from 128Hz, 4KHz and 66KHz. Different IC Body provides different LED frame rate option, and there are some restrictions in 4KHz and 66KHz.

- For 4KHz, only PE0, PE1 and PE2 can be set as PWM-IO.
- For 66KHz, only supports 32 levels resolutions, and only one I/O can be set as PWM-IO.

The following table shows the PWM frame rate of each NY9T Body.

IC Type	***PWM Frame Rate***
NY9T001A	4KHz, 66KHz
NY9T004A	128Hz
NY9T008A	128Hz, 4KHz
NY9T016A	128Hz, 4KHz

4.2 LED Control Register

The LED control register LEDCTL contains the FOSC of LED sample extension control, the PAUSE/RESUME and the BUSY flag of playing LED.

Control Register	***Name***	***Mode***	***Bit***	***Function description***
LEDCTL	FOSC[1:0]	R/W	[1:0]	LED sample extension x 8 LED sample extension x 4 LED sample extension x 2 LED sample extension x 1
	PAUSE/RESUME	R/W	[2]	Pause/Resume LED
	BUSY	R	[3]	Busy

4.2.1 LED Step

The LED step combines LED section with the extension that is used to save the ROM space. The FOSC controls the extension rate of LED sample, and it can extend the LED playing period without ROM usage increase. The FOSC value can only be changed at STOP mode or LEDF flag equal to “1” (end of section).

LED step = LED section * Extension

FOSC[1:0]	***Extension***
00	8
01	4
10	2
11	1

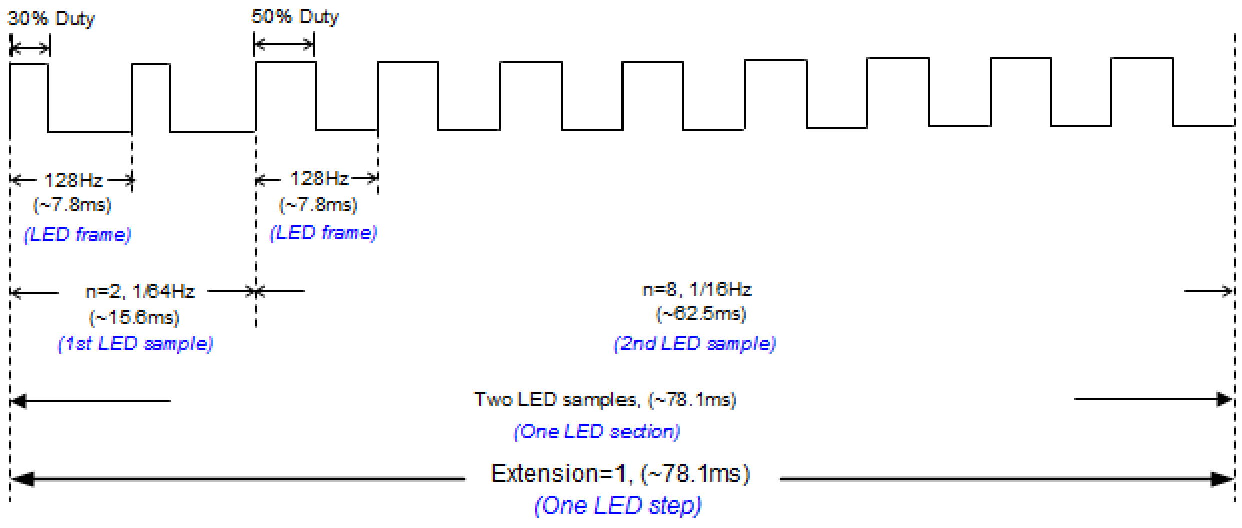
Example:

1st sample: brightness level: *30%*. If n=2 → LED sample period = (1/128Hz) * n = 1/64Hz.

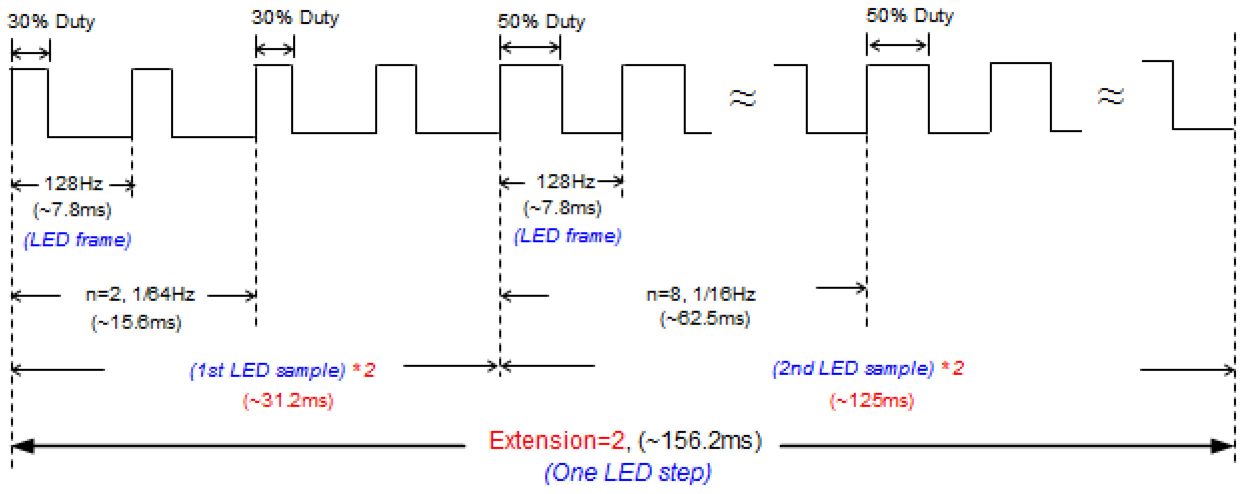
2nd sample: brightness level: 5*0%*. If n=8 → LED sample period = (1/128Hz) * n = 1/16Hz.

One LED section can be combined by two different LED samples.

If extension is set as 1, the LED step is shown below.



If extension is set as 2, <u>each LED sample will be multiply by 2</u>, and the LED step is shown below.



4.2.2 Pause/Resume

The NY9T supports another special PAUSE/RESUME mode for LED playing. Write “1” to the PAUSE/RESUME bit to enter PAUSE mode that keeps the last LED data unchanged. If users want to resume LED paying continuously, write “0” to the PAUSE/RESUME bit.

4.2.3 Busy Condition

When the PLAY or RESUME command is executed, the BUSY bit will be set as “1”. When the STOP command is executed, the BUSY bit will be set as “0” and stop playing LED immediately. But if the PAUSE command is executed, the BUSY bit will be not cleared as “0” immediately and still kept as “1” until the end of the LED frame period. In the other word, the PAUSE action only happens in the end of each LED frame. After BUSY bit is set as “0” by PAUSE command, LED data will be kept in the PWM value of last frame and the LED frame will be output continuously. Users can set OEN bit as “0” to stop the LED frame output before IC enter halt mode. To keep LED playing normally, IC doesn't enter halt mode until BUSY=0.

4.3 LED Channel Output Enable Register

The NY9T provides maximum 8 channel PWM-IO output. When Port E (Port F) is optioned as PWM-IO output, the PEIO (PFIO) register acts as LED channel output enable register. Therefore, each channel has its own enable/disable control register. If the OEN bit set as "1", the corresponding channel output will be enabled. Otherwise, the channel output will be disabled. The value of OEN[7:0] can only be changed at STOP mode or LEDF flag equal to "1" (end of section).

Control register	***Name***	***Mode***	***Bit***	***Function description***
PEIO	OEN[3:0]	R/W	[3:0]	LED channel[3:0] output enable/disable (When Port E option as PWM-IO output)
PFIO	OEN[7:4]	R/W	[3:0]	LED channel[7:4] output enable/disable (When Port F option as PWM-IO output)

Chapter 5. Touch-Key Control

5.1 Touch-Key Control

The NY9T provides a capacitive sensing Touch-Key function. With built-in LDO for touch sensor, it can reduce the false detection which is induced by the various power supply voltage. The hardware calibration function is also built-in to adjust the environment changing to fit the sensitivity. Besides, it accepts the multi-key entry function or only single-key entry function optioned by mask. And a slow mode of Touch-Key that re-arranges the Touch-Key scan mode can reduce the power consummation. From the Touch-Key data register, users can distinguish which Touch-Key is pressed on/off by software polling or interrupt routine.

5.1.1 Touch-Key Control Register

Control register	***Name***	***Mode***	***Bit***	***Function description***
TPCTL	TPEN	R/W	[0]	Touch-Key enable/disable
	TP_SLOW	R/W	[1]	Touch-Key slow mode
	CALIB1	R/W	[2]	Auto-Judge calibration mode
	CALIB2	R/W	[3]	Enforce calibration mode

The TPEN bit of TPCTL register can enable/disable Touch-Key function. To enable the Touch-Key function, the bit has to be set as “1”. On the contrary, to disable the Touch-Key function, the bit has to be set as “0”.

Touch-Key function enables the slow mode by setting the TP_SLOW bit of TPCTL register as “1”. When IC enters the slow mode, the Touch-Key scan rate will be slower than normal mode for saving power. The Touch-Key scan mode will be described in details after at section 5.3.

Users must calibrate the environment to acquire the background parameter. NY9T provides two kinds of calibration modes that are Auto-Judge calibration and Enforce calibration by setting the corresponding bit of CALIB1 and CALIB2 to “1”. The details will be described after at section 5.4.

5.1.2 Sensitivity Control Register

The NY9T provides 2 ways of Custom mode and Preset mode for users to set up the sensitivity threshold. At any mode, there are eight levels sensitivity to control the touch sensitivity, which can be selected by software. The details will be described after at section 5.5.3.

Control register	***Mode***	***Bit***	***Function description***
SENS	R/W	[2:0]	Sensitivity selection

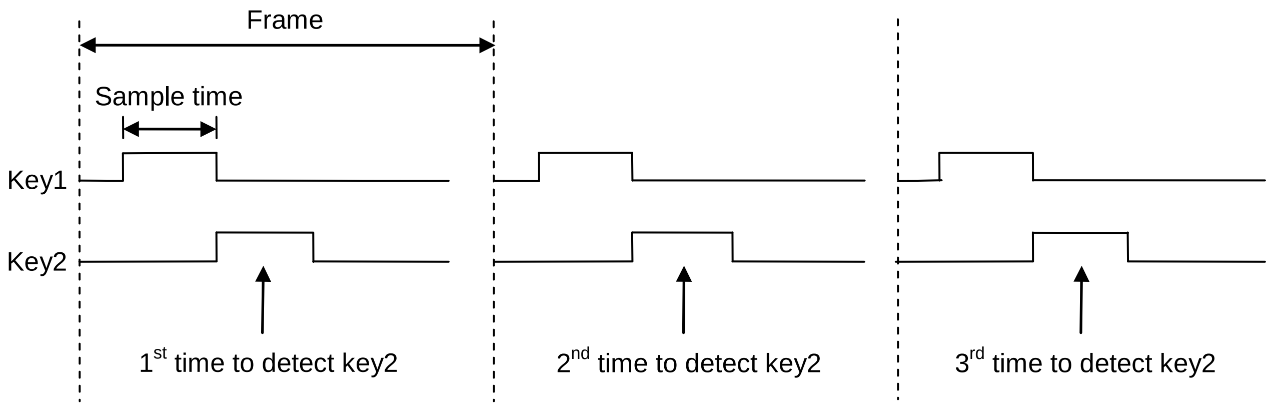
5.1.3 Touch-Key Data Register

When NY9T is executing Touch-Key function, the KEYD0~KEYD3 data register will store the key status.

Control register	***Mode***	***Bit***	***Function description***
KEYD0	R	[3:0]	Key1~Key4 key status
KEYD1	R	[3:0]	Key5~Key8 key status
KEYD2	R	[3:0]	Key9~Key12 key status
KEYD3	R	[3:0]	Key13~Key16 key status

5.2 Touch-Key Scan Time

The touch hardware can not sense all Touch-Keys at the same time, and it only senses a Touch-Key once every time normally. Therefore, a timing of Touch-Key scan is arranged to detect all Touch-Keys. Those are some parameters to accomplish the detecting time.



5.2.1 Sample Time

The sample time is the duration to scan one Touch-Key, and either 1ms or 2ms sample time is provided for Touch-Keys. The TIMES option represents how much time to scan one Touch-Key. Longer sample time will get more touch accuracy but longer touch detecting time.

Options	***Selected items***
TIMES	1ms/2ms

5.2.2 Frame Rate

The frame rate is the period to complete the scanning for all Touch-Keys, and it can be composed of 8 or 16 samples. That FM32MS option decides how many samples to finish one frame. The frame rate is TIMES*FM32MS. More samples in one frame will consume less power, but longer touch detecting time.

Options	***Selected items***
FM32MS	8/16 samples in one frame

Note: For NY9T016A, the FM32MS is fixed as 16 samples.

5.2.3 Debounce Time

The debounce time defines how many frames to detect the identical Touch-Key pressed/released continuously in order to recognize the specified Touch-Key On/Off. The DETS option provides four kinds of debounce time. More debounce time will get more touch accuracy but longer touch detecting time. For high noise environment such as home appliance application, more debounce time is suggested.

Options	***Selected items***
DETS	"Touch-Key On" needs 1 frame, "Touch-Key Off" need 1 frame
	"Touch-Key On" needs 2 frames, "Touch-Key Off" need 1 frame.
	"Touch-Key On" needs 4 frames, "Touch-Key Off" need 2 frames.
	"Touch-Key On" needs 6 frames, "Touch-Key Off" need 3 frames.

5.2.4 Detect Time

Base on the Touch-Key sensing time arrangement, the maximum scan time to detect one key touched is

TIMES*FM32MS*n + Wake-up time,   n=1/2/4/6 depend on the "Touch-Key On" frame of DETS.

Wake-up time is zero at normal mode, 3*TIMES*FM32MS at slow mode or 4*TIMES*FM32MS at slow-green mode. The detailed description for Wake-up is at section 5.3.4.

5.2.5 Enlarge Scan Time

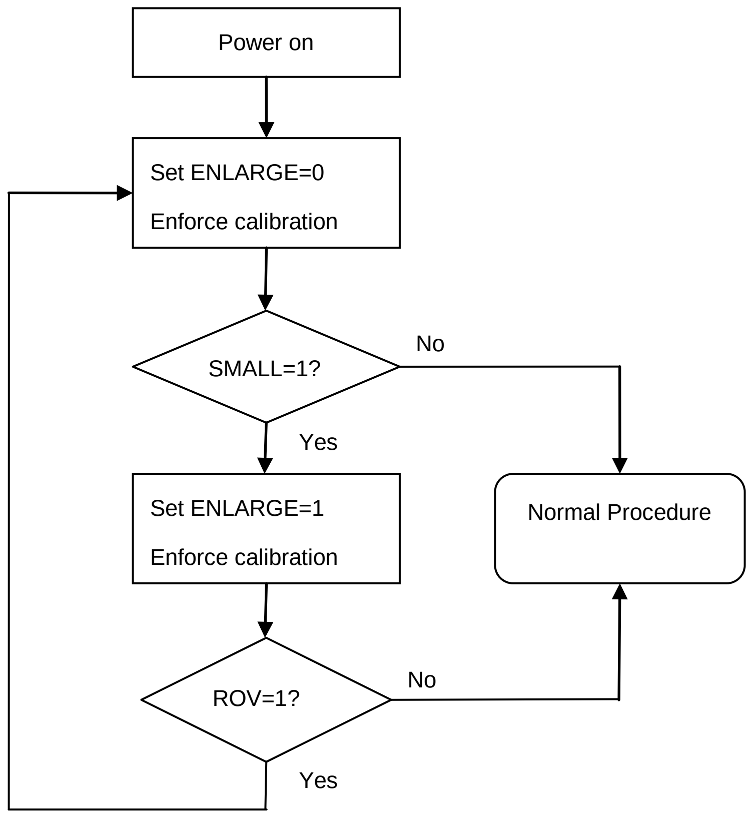
If touch key sensitivity threshold do not distinguish enough, user can enlarge scan time to gain better resolution and the detect time will extend too. Every touch keys can be set as Enlarge Scan Time or not through *Q-Touch*. Different IC type has different enlarge multiple.

IC Type	***Enlarge***
NY9T001A	X2, X4
NY9T004A	*N/A*
NY9T008A	X2
NY9T016A	*N/A*

Enlarge Scan Time can also be used to determine whether to open the function of Time Scan extension through IPCTL data register.

Control register	***Name***	***Mode***	***Bit***	***Function description***
IPCTL	SMALL	R	[0]	Reference count too small
	ROV	R	[1]	Reference count overflow
	ENLARGE	R/W	[3]	Enlarge resolution

Enlarge Scan Time software flow chart:



5.3 Touch-Key Scan Mode

In Touch-Key scan mode, there are normal mode and slow mode configured by software. Users can enable the slow mode by setting the TP_SLOW bit of TPCTL register as “1”. When IC enters the slow mode, the interval of each Touch-Key scan will become 4 times of the normal mode in order to save power. And when any Touch-Key is pressed on successfully, IC will leave the slow mode for normal mode and the TP_SLOW will be set as “0” automatically.

NY9T supports maximum 16 Touch-Keys, which shares with Port A to Port D optioned by mask. When users configure 8 Touch-Keys above, another slow-green mode could be optioned instead of slow mode. That is, if GREEN option is enabled, the slow mode will become slow-green mode. At this moment, if the TP_SLOW register is set as “1”, it will enter slow-green mode.

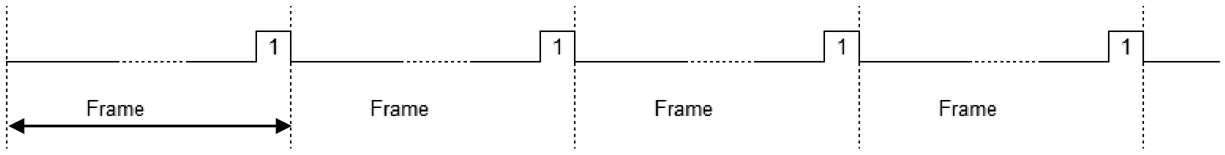
Options	***Selected item***
GREEN	Slow-green mode enable/disable

Basically, it’s a trade-off between the operating current and touch sensitivity for different scan modes. At normal mode, every Touch-Key is scanned once a frame, and it will get more sensitive but more power consumption. At slow mode, every Touch-Key is scanned once 4 frames, and it will get less power consumption but less sensitive. At slow-green mode, every 2 or 4 Touch-Keys is simultaneously scanned once 4 frames. Users can set different scan mode by software for the balance between power consumption and sensitivity.

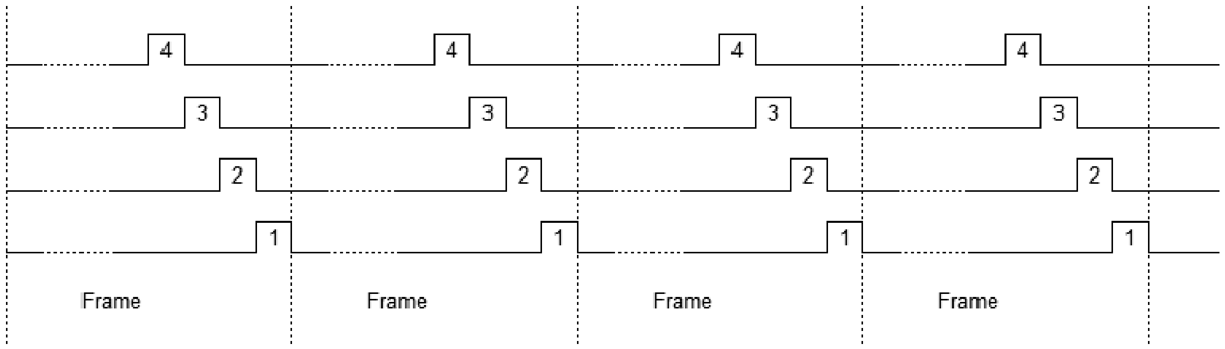
5.3.1 Normal Mode

NY9T samples one Touch-Key once every frame periodically. The timing is shown below.

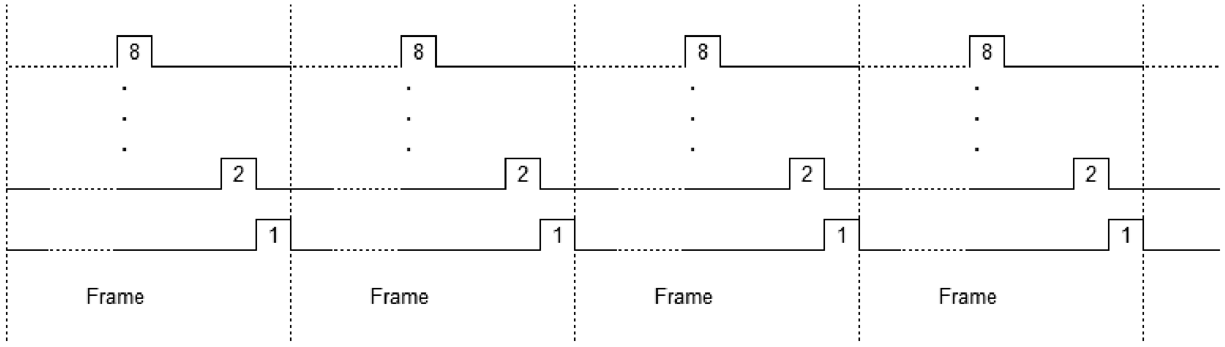
(i) 1 Touch-Key



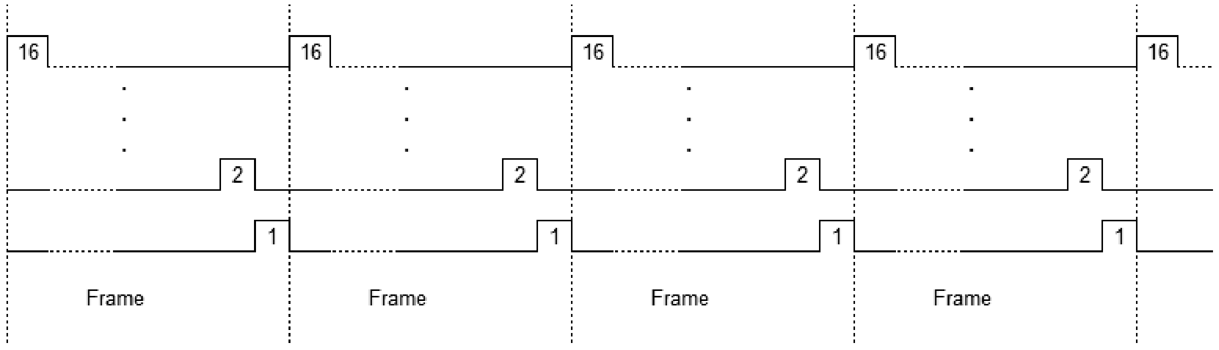
(ii) 4 Touch-Keys



(iii) 8 Touch-Keys



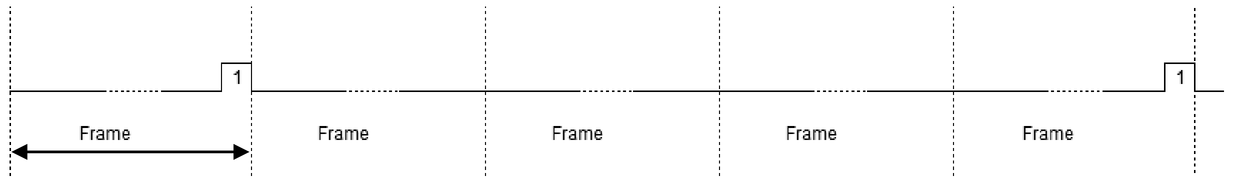
(iv) 16 Touch-Keys



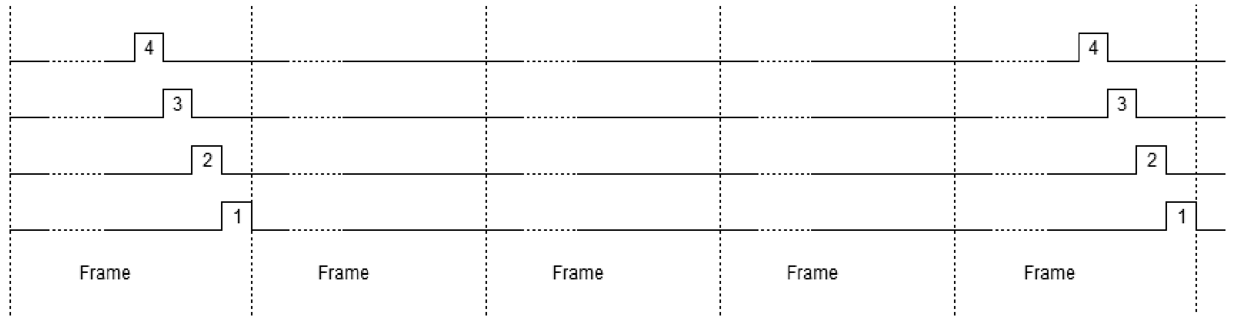
5.3.2 Slow Mode

NY9T samples one Touch-Key once every 4 frames periodically. The timing is shown below.

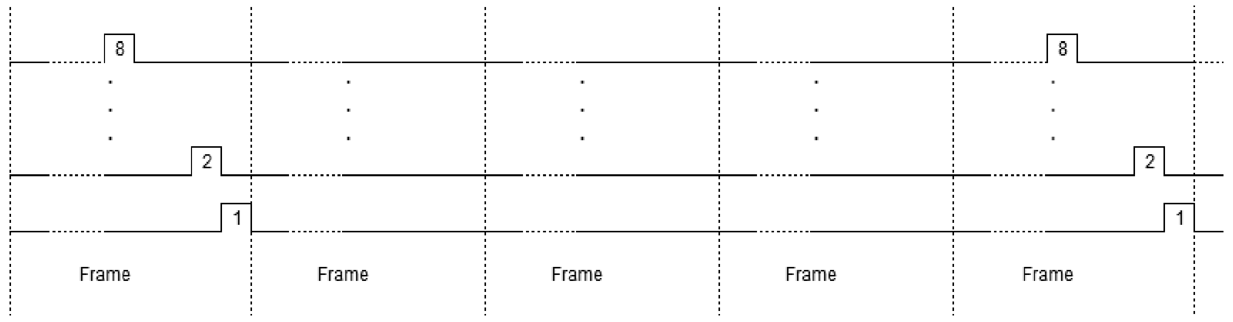
(i) 1 Touch-Key



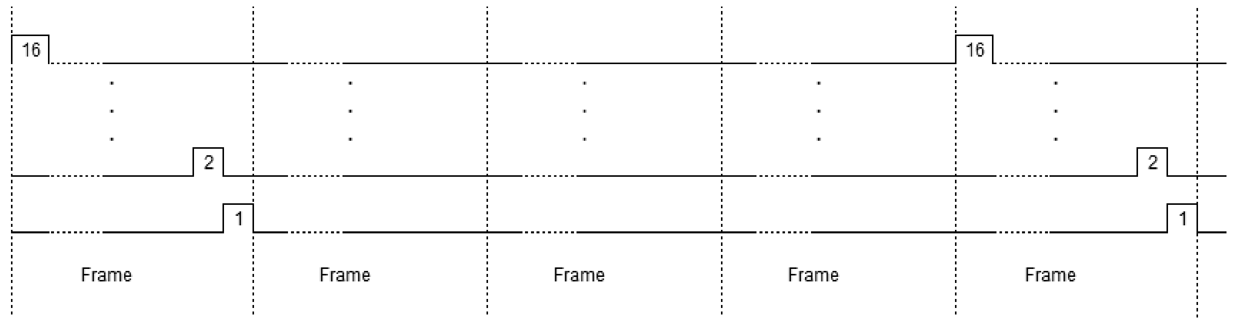
(ii) 4 Touch-Keys



(iii) 8 Touch-Keys



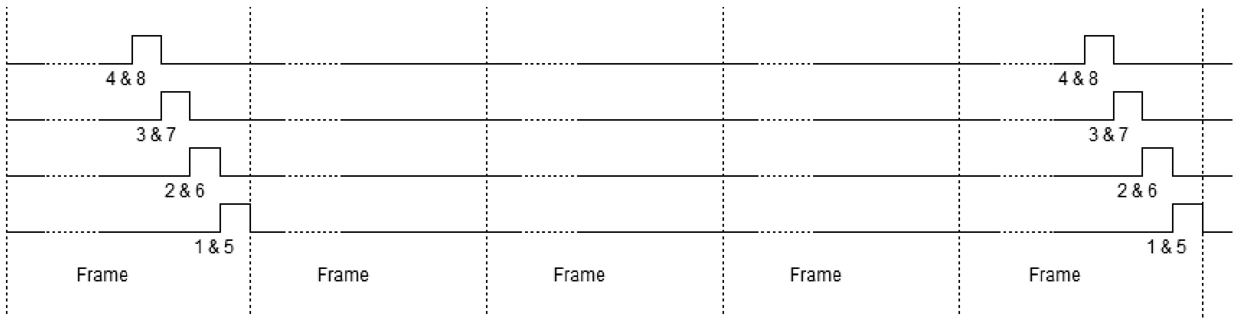
(iv) 16 Touch-Keys



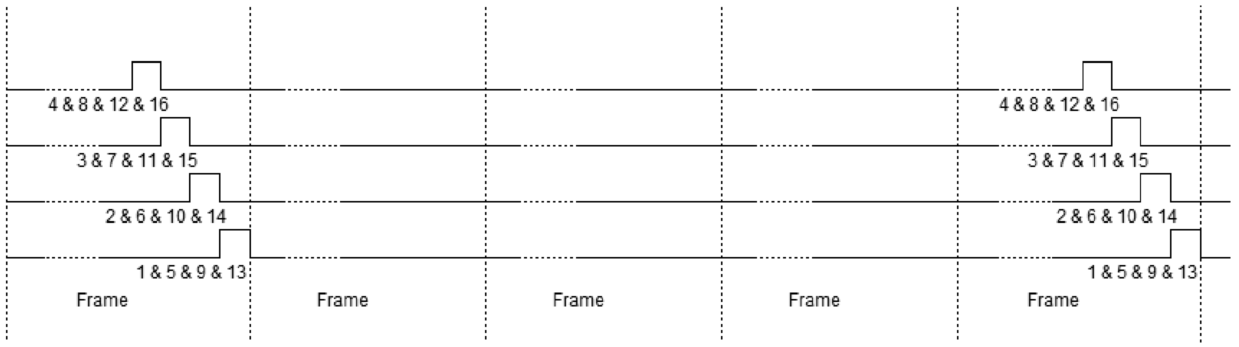
5.3.3 Slow-Green Mode

Only above 8 Touch-Keys can be optioned as slow-green mode. The timing is shown below.

(i) 8 Touch-Keys: NY9T samples 2 Touch-Keys once in same time every 4 frames periodically.



(ii) 16 Touch-Keys: NY9T samples 4 Touch-Keys once in same time every 4 frames periodically.



5.3.4 Wake-Up

When NY9T enters slow mode (or slow-green mode), if any Touch-Key is detected as pressed, IC will go back to normal mode automatically. This operation is called wake-up. Therefore, if users want to calculate the maximum “Touch-Key On” time while Touch-Key scanning, the wake-up time must be included.

Once the Touch-Key is pressed successfully, the TP_SLOW bit will be cleared automatically, and the scan mode will be kept at normal mode immediately. But if touch is not successful, the scan mode is still at slow mode (or slow-green) after the finger touch is released.

The wake-up key can be optioned as Any Key or Key1 (PA0) by mask. When Key1 (PA0) is optioned, the other Touch-Keys can’t wake-up the system, and the power consumption is minimized.

Options	***Selection item***
WAKEUPS	Any-Key / Key1 (PA0)

5.3.5 Single/Multi-Key Touch

The NY9T supports single-key touch or multi-key simultaneous touch by hardware option. The single-key function is for detecting one key pressed and the other keys have no effect, and it is similar with input-irretrigger function and simple to use. The multi-key function enables multi keys active at the same time, and users can design complicated touch function by software programming as they want. The SINGLE option decides the Touch-Key function is single or not.

Options	***Selected item***
SINGLE	Single-key / Multi-key

5.4 Calibration Mode

When IC power is on, it is necessary to calibrate the environment background parameter first. Without calibration, the Touch-Key may not function correctly. The calibration is executed by hardware automatically when write “1” to CALIBx bit, and the calibration time needs 8 frames period. There are 2 calibration modes of the auto-judge calibration and the Enforce calibration selected by software for different conditions.

5.4.1 Auto-Judge Calibration

At the Auto-Judge calibration mode, if any key is pressed in the process of calibration, IC won’t update the environment background parameter after IC finishes this calibration. Except to judge the key-pressed, it is able to detect the environment noise and is effective to eliminate the noise interference. That is why it’s called Auto-Judge calibration. It’s useful to monitor the environment changing at any time to maintain sensitivity of the Touch-Key. This Auto-Judge calibration is suggested to process once when not any Touch-Key is pressed in 4 seconds.

Write “1” to CALIB1 bit of TPCTL register to do the Auto-Judge calibration. When finishing calibration, the CALIB1 bit will be set as “0” automatically.

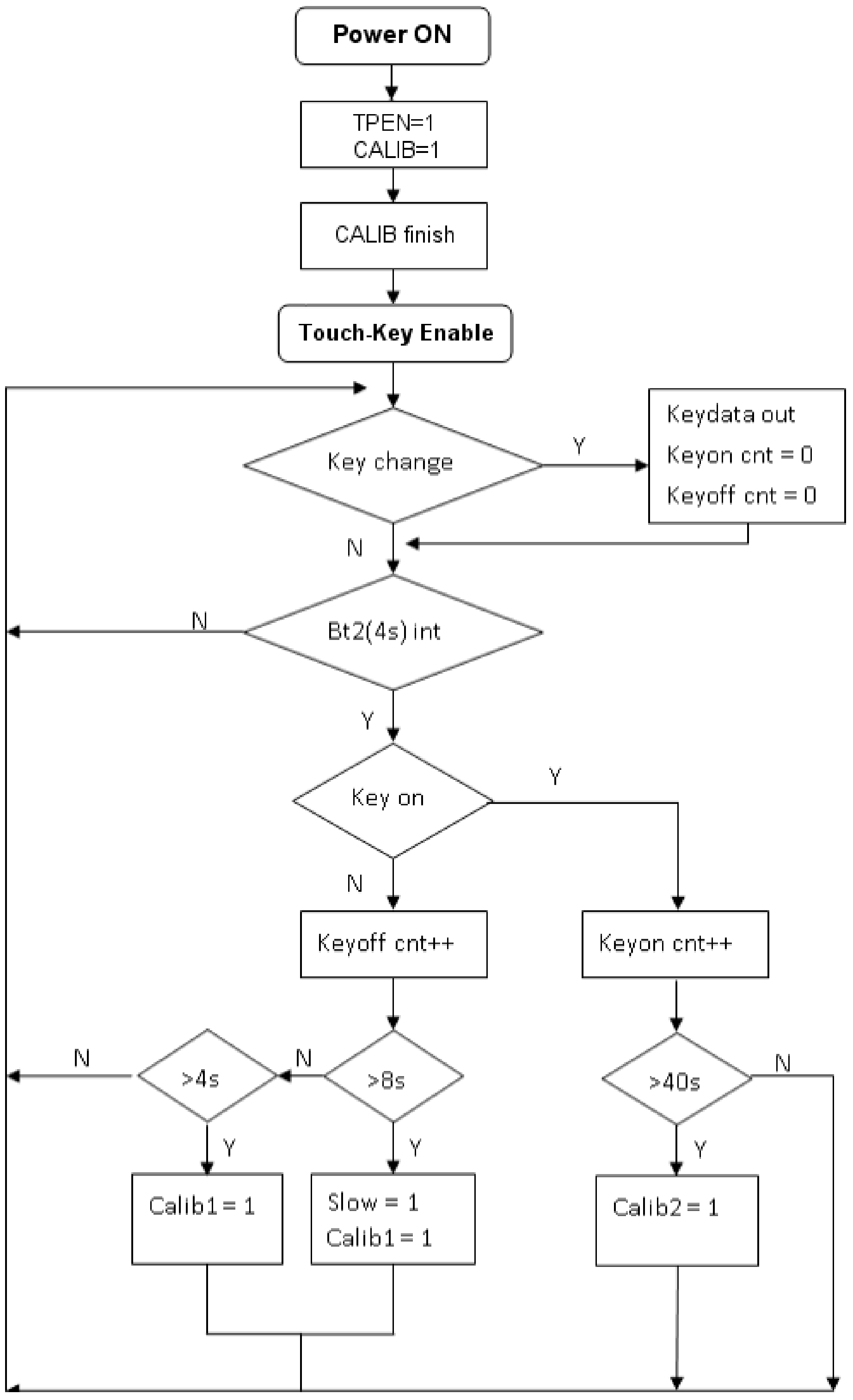
5.4.2 Enforce Calibration

At the Enforce calibration mode, whatever key is pressed or not in the process of calibration, IC will still update the environment background parameter after IC finishes this calibration. If Touch-Key is covered with objects for a long time, the Auto-Judge calibration can‘t update the environment background parameter and the Touch-Key will be detected as key-pressed status. It will cause IC working at operating mode constantly and power exhausted. At this condition, this Enforce calibration can calibrate key-pressed status as normal background, and IC can enter slow mode to avoid power exhaustion. If the covered objects are removed, the function will recover by Auto-Judge calibration. The Enforce calibration is suggested to process once when any Touch-Key has been pressed over 120 seconds.

Write “1” to CALIB2 bit of TPCTL register to do Enforce calibration. When finishing calibration, the CALIB2 bit will be set as “0” automatically.

Note: When IC works at the period of calibration, it will read the CALIB2 and CALIB1 bit of TPCTL register both high, no matter what kind of calibration setting.

5.4.3 Software Example



5.5 Touch Sensitivity

The touch background environment consists of the thickness of non-conductive dielectric material, the electrode pad and the connecting line between IC and electrode pad. In order to fit various types of application, NY9T provides two kinds of sensitivity mode by mask options to judge the background environment. One is Preset mode, the other is Custom mode. An innovative software tool *Q-Touch* can help users to measurement the background environment, judge the sensitivity mode and adjust the touch threshold without any external capacitor. The PAL option decides this sensitivity mode.

Options	***Selected item***
PAL	Preset/Custom

5.5.1 Preset Mode

Usually the Preset mode can be used in most touch applications. It is useful for Touch-Keys with different length of PCB connecting lines or connecting wires between IC and electrode pad. For example, in a poster with 50cm*50cm area, the many Touch-Keys are distributed in a large area with the connecting lines of different length. Through the *Q-Touch* software tool, the Preset mode can automatically set up the sensitivity for all Touch-Keys, and users can set the sensitivity threshold to one value for all Touch-Keys. If the thickness of dielectric material is not changed, the sensitivity threshold doesn’t need to be set up again even the other factors of effecting sensitivity are different.

5.5.2 Custom Mode

If any one Touch-Key thickness of dielectric material is different from other Touch-Key, the Custom mode will be suggested instead of Preset mode. For example, in a plush toy, there are 2 Touch-Keys, one’s thickness is 2mm and another is 5mm. Through the *Q-Touch* software tool, the Custom mode can individually set up the sensitivity for each Touch-Key, and users can adjust the sensitivity threshold one by one for each Touch-Key. When the electrode size, the connecting line of PCB or the thickness of dielectric material is changed, the sensitivity threshold for all Touch-Keys must be set up again.

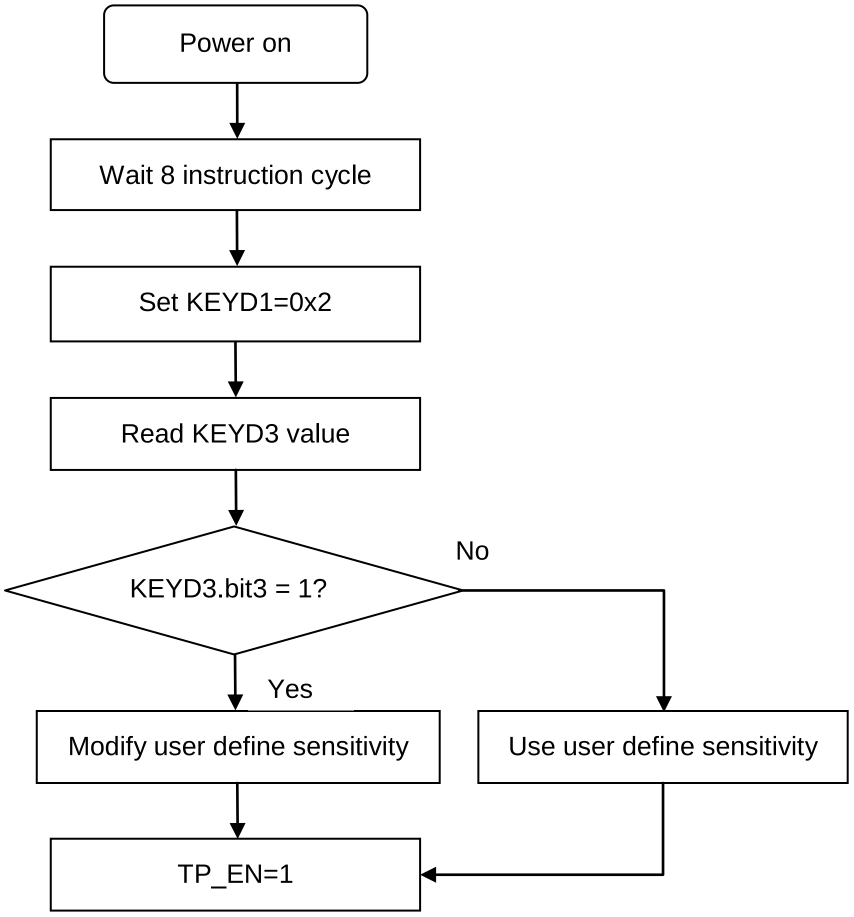
5.5.3 Sensitivity Level

8 levels of sensitivity can be set by software after users finish the sensitivity threshold measurement. The most to less sensitive is level-0 to level-7. The SENS register store the sensitivity level as following.

SENS	***Sensitivity level***	***Sensitivity***
000	0	Most sensitive
001	1	↑
010	2	
011	3	
100	4	Typical
101	5	
110	6	↓
111	7	Less sensitive

5.5.4 External Adjust

Except software sensitivity control, an external hardware sensitivity control is reserved. Through R_{ADJ} pad connecting a resistor to GND, users can adjust the sensitivity for different background environment. Otherwise, PCB designer should notice that the length of R_{ADJ} trace from IC to resistor must be within 0.5cm, and also avoids any parallel trace, or parallel GND nearby or at different layer. After IC power on and wait 8 instruction cycle, user can follow the steps below to get SENS value to modify sensitivity level.



Recommended R_{ADJ}	***Mapping Sensitivity Level***
750KΩ *(+/-1% tolerance)*	0 (000)
360KΩ *(+/-1% tolerance)*	1 (001)
180KΩ *(+/-1% tolerance)*	2 (010)
120KΩ *(+/-1% tolerance)*	3 (011)
82KΩ *(+/-1% tolerance)*	4 (100)
51KΩ *(+/-1% tolerance)*	5 (101)
33KΩ *(+/-1% tolerance)*	6 (110)
16KΩ *(+/-1% tolerance)*	7 (111)

5.5.5 Touch-Key Anti-Interference

According to PCB layout, if the electrode pad of Touch-Keys are very close, the neighboring Touch-Key will influence each other by the parasitic capacitance. The Touch-Key Anti-Interference function can reduce the coupling effect to improve the noise immunity. But the total capacitance of Touch-Key will be increased, and it will reduce the sensitivity a little. Thus, if the neighboring Touch-Key will interfere with other Touch-Key, users can enable Touch-Key Anti-Interference option. If not, it is suggested to disable Touch-Key Anti-Interference option. The TP_NOSCAN option decides the Touch-Key Anti-Interference enabled or disabled.

Options	***Selected item***
TP_NOSCAN	Touch-Key Anti-Interference

5.5.6 Touch-Key Hysteresis Level

The hysteresis function is to avoid noise interference when the edge of trigger pad is touched by hand caused continuous triggers. There are 2 kinds of hysteresis level, Hysteresis Level Small (shift 1 level) and Hysteresis Level Large (shift 1/2). When any Touch-Key is pressed, the sensitivity level auto decrease 1 level or divided the sensitivity threshold value by 2, until all Touch-Key is released.

Options	***Selected item***
Touch-Key Hysteresis	Small/Large/Disable

Note: This function is not supported by NY9T004A.

Chapter 6. Instruction Set

6.1 Instruction Classified Table

Item	*Inst.*	*Op1*	*Op2*	*Operation*	*Inst. Length*	*Exec. Cycle*	*Oper. Flag*	*Flag Affected*
Arithmetic Instructions								
1	INCM	6m		A = M +1	1	1		C, Z
2	DECM	6m		A = M - 1	1	1		C, Z
3	ADDM	6m		A = A + M + C	1	1	C	C, Z
4	XORM	6m		A = A ^ M	1	1		Z
5	ANDM	6m		A = A & M	1	1		Z
6	ORM	6m		A = A | M	1	1		Z
7	MVTA	6m		Move PortT to A	1	1		
8	MVAT	6m		Move A to PortT	1	1		Z
9	MVAM	6m		Move A to M	1	1		
10	MVMA	6m		Move M to A	1	1		Z
11	MVRM	4m	2r	Move R to M	1	1		
12	MVMR	4m	2r	Move M to R	1	1		
13	MVLR	4L	2r	Move L to R	1	1		
14	ADDL	4L		A = A + L + C	1	1	C	C, Z
15	XORL	4L		A = A ^ L	1	1		Z
16	ANDL	4L		A = A & L	1	1		Z
17	ORL	4L		A = A | L	1	1		Z
18	MVLA	4L		Move L to A	1	1		
19	RORA			C = A[0]; A = {C, A[3:1]}	1	1		C, Z
20	ROLA			C = A[3]; A = {A[2:0], C}	1	1		C, Z
21	RSTC			Reset C = 0	1	1		C
22	SETC			Set C = 1	1	1		C
23	INCA			A = A + 1	1	1		C, Z
24	DECA			A = A - 1	1	1		C, Z
Conditional Instructions								
25	CNAM	6m		Skip if A ≠ M	1	1		
26	CPAL	4L		Skip if A = L	1	1		
27	CNAL	4L		Skip if A ≠ L	1	1		
28	CPAB	4L		Skip if (A & L) = 0	1	1		
29	CPCZ			Skip if C = 0	1	1	C	
30	CNCZ			Skip if C ≠ 0	1	1	C	
31	CPZZ			Skip if Z = 0	1	1	Z	
32	CNZZ			Skip if Z ≠ 0	1	1	Z	
LED Instructions								
33	RBLP			Move LPR to RPT	2	3		
34	STOP			STOP playing LED immediately	1	1		
35	PLAY			Move RPT to LPR	1	3		

Item	***Inst.***	***Op1***	***Op2***	***Operation***	***Inst. Length***	***Exec. Cycle***	***Oper. Flag***	***Flag Affected***
Other Instructions								
36	MPG	1p		Set RAM page	1	1		
37	RBRO	1n		Move ROM[RPT] data to ROD and ACC	1	3		
38	JMP	14a		Jump to Address	2	2		
39	CALL	14a		Jump to Address, and Move PC to RPT	2	2		
40	LDPC			Move RPT to PC	1	2		
41	RBPC			Move PC to RPT	1	2		
42	IRET			Return from interrupt routine	2	2		
43	HALT			Enter HALT mode	1	1		
44	CWDT			Clear WDT	1	1		
45	NOP			No operation	1	1		

A : 4-bit Accumulator data

C : 1-bit carry flag data

M : 4-bit RAM or memory register data

R : 4-bit memory register data

L : 4-bit immediately literal data

Z : 1-bit zero flag data

RPT : Multi-function register data

LPR : LED address pointer of CH

ROM : 10-bit ROM data

ROD : ROM data access register data

PC : Program counter address pointer

a : ROM address

m : RAM or memory register address

n : data address Plus/Not plus 1

p : RAM page

r : Memory register address

6.2 Instruction Descriptions

6.2.1 Arithmetic Instructions

INCM m

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

Operation: A ← M + 1

Operand: 0x0≦m≦0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C, Z

Example: INCM m0

Before Instruction

M0=0x0

After Instruction

A=0x1, C=0, Z=0

DECM m

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

Operation: A ← M - 1

Operand: 0x0≦m≦0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C, Z

Example: DECM m0

Before Instruction

M0=0x0

After Instruction

A=0xF, C=0, Z=0

ADDM m

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

Operation: A ← A + M + C

Operand: 0x0≦m≦0x3F

Words: 1

Cycles: 1

Operative Flags: C

Flags Affected: C, Z

Example: ADDM m0

Before Instruction

A=0x7, M0=0xA, C=0

After Instruction

A=0x1, M0=0xA, C=1, Z=0

XORM m

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

Operation: A ← A ^ M

Operand: 0x0≦m≦3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: XORM m0

Before Instruction

A=0x3, M0=0xB

After Instruction

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

ANDM m

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

Operation: A ← A & M

Operand: 0x0 ≦ m ≦ 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: ANDM m0

Before Instruction

A=0x7, M0=0xA

After Instruction

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

ORM m

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

Operation: A ← A | M

Operand: 0x0 ≦ m ≦ 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: ORM m0

Before Instruction

A=0x3, M0=0x8

After Instruction

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

MVTA t

Function: Move Port of address t to A.

Operation: A ← PortT

Operand: 0x0 ≦ m ≦ 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: MVTA t0

Before Instruction

T0=0x8

After Instruction

A=0x8

MVAT t

Function: Move A to PortT of address t.

Operation: PortT ← A

Operand: 0x0 ≦ m ≦ 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MVAT t0

Before Instruction

A=0x8

After Instruction

T0=0x8

MVAM m

Function: Move A to M of address m

Operation: M ← A

Operand: 0x0 ≤ m ≤ 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

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 ≤ m ≤ 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: MVMA m0

Before Instruction

M0=0x8

After Instruction

A=0x8

MVRM m, r

Function: Move R to M. Which R address MSB is 0 and M address MSB 2-bit is 0x3.

Operation: M ← R

Operand: 0x0 ≤ m ≤ 0xF

0x0 ≤ r ≤ 0x3

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MVRM m0, 0x0

Before Instruction

PG=0, RPT0=0x8

After Instruction

M30=0x8

MVMR m, r

Function: Move M to R. Which R address MSB is 0 and M address MSB 2-bit is 0x3.

Operation: R ← M

Operand: 0x0 ≤ m ≤ 0xF

0x0 ≤ r ≤ 0x3

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MVMR m1, 0x1

Before Instruction

PG=0, M31=0x8

After Instruction

RPT1=0x8

MVLR L, r

Function: Move the immediate constant value to R. Which R address MSB is 0.

Operation: R ← L

Operand: 0x0 ≤ L ≤ 0xF

0x0 ≤ r ≤ 0x3

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MVLR 0x7, 0x3

Before Instruction

RPT3=0x0

After Instruction

RPT3=0x7

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

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

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

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 0x7

Before Instruction

A=0x9

After Instruction

A=0x7

RORA

Function: Right Rotate A to C and A.

Operation: C ← A[0]; A ← {C, A[3:1]}

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C

Example: RORA

Before Instruction

A=0x5, C=0

After Instruction

A=0x2, C=1

RSTC

Function: Clear C to 0.

Operation: C ← 0

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C

Example: RSTC

Before Instruction

C=1

After Instruction

C=0

ROLA

Function: Left Rotate A to C and A.

Operation: C ← A[3]; A ← {A[2:0], C}

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C

Example: ROLA

Before Instruction

A=0x5, C=1

After Instruction

A=0xB, C=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

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

6.2.2 Conditional Instructions

CNAM m

Function: Skip the next instruction if A not equals M of address m.

Operation: Skip next if A≠M

Operand: 0x0≤m≤0x3F

Words: 1

Cycles: 1, (2, 3)

Operative Flags: None

Flags Affected: None

Example: CNAM m0

Inst1

Inst2

After Instruction

If A=M0, 'Inst1' is executed.

If A≠M0, 'Inst1' is discarded, and 'Inst2' is executed.

CNAL L

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

Operation: Skip next if A≠L

Operand: 0x0≤L≤0xF

Words: 1

Cycles: 1, (2, 3)

Operative Flags: None

Flags Affected: None

Example: CNAL 0x4

CALL a1

CALL a2

After Instruction

If A=0x4 'CALL a1' is executed

If A≠0x4 'CALL a1' is discarded, and 'CALL a2' is executed

CPAL L

Function: Skip the next instruction if A equals L.

Operation: Skip next if A=L

Operand: 0x0≤L≤0xF

Words: 1

Cycles: 1, (2, 3)

Operative Flags: None

Flags Affected: None

Example: CPAL 0x4

CALL a1

CALL a2

After Instruction

If A≠0x4 'CALL a1' is executed

If A=0x4 'CALL a1' is discarded, and 'CALL a2' is executed

CPAB L

Function: Skip the next instruction if (A and L) not equals Zero.

Operation: Skip next if (A & L)≠0

Operand: 0x0≤L≤0xF

Words: 1

Cycles: 1, (2, 3)

Operative Flags: None

Flags Affected: None

Example: CPAB 0x4

CALL a1

CALL a2

After Instruction

If (A & L)≠0 'CALL a1' is executed

If (A & L) =0 'CALL a1' is discarded, and 'CALL a2' is executed

CPCZ

Function: Skip the next instruction if C equals zero.

Operation: Skip next if C=0

Operand: None

Words: 1

Cycles: 1, (2, 3)

Operative Flags: C

Flags Affected: None

Example: CPCZ

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

CPZZ

Function: Skip the next instruction if Z equals zero.

Operation: Skip next if Z=0

Operand: None

Words: 1

Cycles: 1, (2, 3)

Operative Flags: Z

Flags Affected: None

Example: CPZZ

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

CNCZ

Function: Skip the next instruction if C not equals zero.

Operation: Skip next if C≠0

Operand: None

Words: 1

Cycles: 1, (2, 3)

Operative Flags: C

Flags Affected: None

Example: CNCZ

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

CNZZ

Function: Skip the next instruction if Z not equals zero.

Operation: Skip next if Z≠0

Operand: None

Words: 1

Cycles: 1, (2, 3)

Operative Flags: Z

Flags Affected: None

Example: CNZZ

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

6.2.3 LED Instructions

RBLP

Function: Read address in LPR to RPT.

Operation: RPT ← LPR

Operand: None

Words: 1

Cycles: 3

Operative Flags: None

Flags Affected: None

Example: RBLP

Before Instruction

LPR=0xABCDE

After Instruction

RPT=0xABCDE

PLAY

Function: Play a new LED section. The LED data address should be loaded into RPT first.

Operation: LPR ← RPT

Operand: None

Words: 1

Cycles: 3

Operative Flags: None

Flags Affected: None

Example: PLAY

Before Instruction

RPT=0x345

After Instruction

LPR=0x345, BUSY=1

STOP

Function: Stop LED playing immediately.

Operation: Stop playing

BUSY ← 0

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: STOP

Before Instruction

BUSY=1

After Instruction

BUSY=0,

6.2.4 Other Instructions

MPG p

Function: Change the RAM page to page p.

Operation: RAM PAGE ← p

Operand: p∈ [0,1]

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MPG 1

MVAM 0x12

Before Instruction

A=0xC

After Instruction

M62=0xC

JMP a

Function: Unconditionally jump by a direct address a.

Operation: PC ← a

Operand: 0x0≤a≤0x3FFF

Words: 2

Cycles: 2

Operative Flags: None

Flags Affected: None

Example: JMP a1

Before Instruction

PC=a0

After Instruction

PC=a1

Note: PC[19:14] will not be changed

RBRO n

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

Operation: A ← ROM data [3:0]

ROD1 ← ROM data [7:4]

ROD2 ← ROM data [9:8]

RPT ← RPT + n

Operand: 0≤n≤1

Words: 1

Cycles: 3

Operative Flags: None

Flags Affected: None

Example: RBRO 1

After Instruction

A=ROM[3:0] @ RPT

ROD1=ROM[7:4] @ RPT

ROD2=ROM[9:8] @ RPT

RPT=RPT+1

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≤0x3FFF

Words: 2

Cycles: 2

Operative Flags: None

Flags Affected: None

Example: CALL a1

Before Instruction

PC=a0

After Instruction

PC=a1, RPT=a0+2

Note: PC[19:14] will not be changed.

LDPC

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: LDPC

Before Instruction

RPT=0x54321

After Instruction

PC=0x54321

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=0x01234

After Instruction

RPT=0x01235

IRET

Function: Return from interrupt routine.

Operation: PC ← STK

Operand: None

Words: 2

Cycles: 2

Operative Flags: None

Flags Affected: None

Example: IRET

After Instruction

PC ← STK and

ACC, SRAM Page, Zero and Carry bit will be restored to those values backup at the moment when entering the interrupt routine.

HALT

Function: Enter the halt (sleep) mode.

Operation: Stop system clock

Operand: None

Words: 1

Cycles: 2

Operative Flags: None

Flags Affected: None

Example: HALT

After Instruction

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

CWDT

Function: Clear Watch-Dog Timer

Operation: Clear WDT

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CWDT

After Instruction

Recount WDT reset timer to 56ms.

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.