九齊科技股份有限公司

Nyquest Technology Co., Ltd.

User Manual

NY5+ Series

Single-Chip 4-bit MCU with 8~32 I/O and 4-Ch Speech/MIDI

Version 1.3

Aug. 24, 2022

Revision History

Version#	*Date*	*Description*	*Modified Page*
1.0	2021/05/31	Formal release.	-
1.1	2021/07/26	Correct content and typos.	7, 10, 16, 33
1.2	2022/05/27	Add NY5Q026A, NY5Q046A, NY5Q080A, and NY5Q160A	7, 8
1.3	2022/08/24	Modify the frequency of IR Carrier	21

Table of Contents

Chapter 1. Introduction 6
1.1 General Description 6
1.2 Features 6
1.3 OTP Product List 7
1.4 Block Diagram 7
1.5 Pad Description 8
1.6 Electrical Characteristics 8
1.6.1 Absolute Maximum Rating 8
1.6.2 DC Characteristics (Preliminary) 9

Chapter 2. Hardware Architecture 10
2.1 Overview 10
2.1.1 Function Block Diagram 10
2.1.2 Hardware Summary Table 10
2.2 Clock Generator 11
2.3 System Reset 11
2.3.1 Power-On Reset (POR) 11
2.3.2 Low Voltage Reset (LVR) 12
2.3.3 Watch-Dog Timer Reset (WDTR) 12
2.3.4 IO Port External Reset 12
2.4 Address Pointer 12
2.4.1 Program Counter (PC) 12
2.4.2 Stack (STK) 13
2.4.3 Multi-function Register Pointer (RPT) 13
2.4.4 Voice Pointer (VPR) 13
2.5 Arithmetic Logic Unit (ALU) 14
2.5.1 ALU Instruction Summary 14
2.5.2 ALU Related Status Flag 15
2.6 Memory Organization 16
2.6.1 ROM 16
2.6.2 RAM 17
2.6.3 Memory Register 17
2.6.4 System Register 18
2.6.5 Register without Memory Allocation 19
2.7 IO Ports 19
2.7.1 Pull-High Input Mode 20
2.7.2 Floating Input Mode 21

2.7.3 Output Mode 21
2.8 Infrared Transmitter 21
2.9 PWM-IO Generator 22
2.10 Interrupt Generator 22
2.11 Audio Synthesizer Structure 22
2.11.1 Audio Output 22
2.11.2 Volume Control 23
2.12 Low Voltage Detector (LVD) 23

Chapter 3. System Control 24
3.1 Introduction 24
3.1.1 System Register Address Map 24
3.1.2 Memory Register Address Map 26
3.1.3 Register without Memory Allocation Map 27
3.2 RPT 27
3.3 ROD 28
3.4 RAM Control Register 28
3.4.1 XMD 28
3.5 I/O Ports Register 28
3.6 INT 29
3.6.1 System Base Timer Polling 29
3.6.2 Interrupt Source 29
3.6.3 TOF source 29
3.6.4 Flag Clear 29
3.7 Audio Control Register 30
3.7.1 VOL 30
3.7.2 PFLG 30
3.7.3 MIX 30
3.7.4 AUD 30
3.7.5 CH# 31
3.7.6 TCSx 31
3.7.8 TMx 31
3.7.9 ENVx 31
3.8 Power Saving Mode 32
3.8.1 Halt Mode 32
3.8.2 Slow Mode 32
3.8.3 RC Slow Mode 33

Chapter 4. Audio Playback 34
4.1 Introduction 34
4.2 Speech Playback 34

4.2.1 Speech Playing Procedure *34*
4.2.2 Speech Data *34*
4.2.3 Quick-IO *34*
4.3 MIDI Control 35
4.3.1 MIDI Playing Procedure *35*

Chapter 5. Instruction Set 36
5.1 Instruction Table 36
5.2 Instruction Descriptions 38
5.2.1 Arithmetic Instructions *38*
5.2.2 Conditional Instructions *43*
5.2.3 Audio Instructions *44*
5.2.4 Other Instructions *46*

Chapter 1. Introduction

1.1 General Description

The NY5+ series IC is a powerful 4-bit micro-controller based sound processor. There are 4 channels that are configured as speech or MIDI, and all of them can be auto-played back simultaneously. By using the high fidelity ADPCM speech synthesis algorithm, it can produce outstanding quality voices. Wide range sampling rate up to 44.1kHz and different volume level are supported. It is also equipped two kinds of audio outputs with fine resolution, including a current D/A converter and a PWM direct-drive. The RISC MCU architecture is very easy to program and control, various applications can be easily implemented. There are 43 instructions, and most of them are executed in single cycle. Furthermore, in addition to the HALT mode (sleep mode), it offers the SLOW mode to minimize power dissipation.

1.2 Features

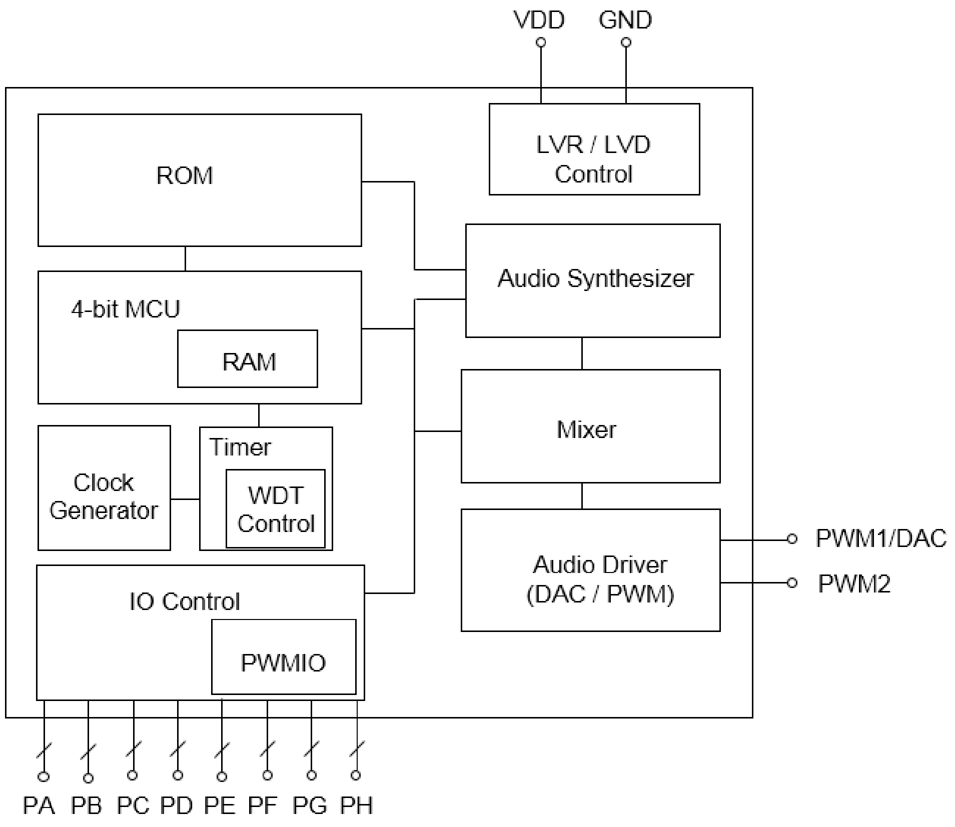
- Wide operating voltage range: 2.0V to 5.5V.
- 4-bit RISC type micro-controller with 43 instructions.
- 4Mx10-bit ROM maximum, program and voice data share the same ROM space.
- 248x4-bit RAM maximum, indirect RAM addressing mode is supported.
- 1MHz or 2MHz instruction frequency (2MHz is required for over 2-ch speech or MIDI).
- SLOW mode to operate at low power consumption.
- HALT mode to save power, less than 1uA@3V standby current.
- Built-in RC oscillation is accurate with +/- 0.5% frequency deviation.
- Low voltage reset (LVR=1.8V), watch-dog reset and I/O port reset are all supported to protect the system.
- Single interrupt entrance with an independent stack, multiple interrupt sources.
- Up to 32 flexible Bi-direction I/Os. Each I/O direction is independently controlled by individual register bit.
- Shared pins to provide IR carrier and external reset feature. (Mask option)
- Selection of IR carrier frequency and data high/low IR output is supported.
- Multiple groups of 4-ch PWM-IO share a group of time, each channel has 8-bit resolution
- NY5+ series are all 4 channels and can play simultaneously; each channel can be arbitrarily assigned as speech or MIDI channel based on the product spec.
- New high fidelity ADPCM speech synthesis algorithm.
- New high fidelity mixed ADPCM or PCM speech synthesis algorithm and ADSR with 256-step envelope for MIDI synthesis.
- High quality 12-bit D/A converter or 12-bit PWM driver.
- Support 3 levels Normal, Large, Ultra PWM current output.
- 16-level digital volume control for synthetic Speech/MIDI.
- Quick-IO control supported.
- Low Voltage Detector (LVD) is built-in for monitoring the status of power and protect malfunction if unstable power is given.

1.3 OTP Product List

IC Type	Time* (sec)	ROM (bits)	RAM (bits)	I/O	PWM-IO	Channel	DAC
NY5Q020A	18.3	48K x 10	248 x 4	8		4	Y
NY5Q026A	25	64K x 10	248 x 4	4		4	Y
NY5Q040A	38.3	96K x 10	248 x 4	8		4	Y
NY5Q046A	45	112K x 10	248 x 4	12	8	4	Y
NY5Q060A	58.3	144K x 10	248 x 4	16	8	4	Y
NY5Q080A	78.3	192K x 10	248 x 4	12	8	4	Y
NY5Q092A	91.6	224K x10	248 x 4	16	8	4	Y
NY5Q160A	158.3	384K x 10	248 x 4	12	8	4	Y
NY5Q172A	171.6	416K x 10	248 x 4	16	8	4	Y
NY5Q342A	345	832K x 10	248 x 4	20	8	4	Y

* The voice duration is calculated at 6 KHz by 4-bit ADPCM algorithm.

1.4 Block Diagram



1.5 Pad Description

Pin	ATTR.	Description
VDD#	Power	Positive power
GND#	Power	Negative power
PWM1/DAC	O	PWM1 output or DAC output
PWM2	O	PWM2 output
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
PG0~3	I/O	Bit 0~3 for Port G
PH0~3	I/O	Bit 0~3 for Port H

* (4 I/O) NY5Q026A : PB0, PA1, PA2, PA3

* (8 I/O) NY5Q020A、NY5Q040A : PA0~3, PB0~3

* (12 I/O) NY5Q046A、NY5Q080A、NY5Q160A : PA0~3, PB0~3, PD0~3

* (16 I/O) NY5Q060A、NY5Q092A、NY5Q172A : PA0~3, PB0~3, PC0~3, PD0~3

* (20 I/O) NY5Q342A : PA0~3, PB0~3, PC0~3, PD0~3, PE0~3

1.6 Electrical Characteristics

The following lists the electrical characteristics of the NY5+ 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 ~ +6.0	V
Vin	Input voltage	VSS–0.3V ~ VDD+0.3	V
Top	Operating Temperature	0 ~ +70	°C
Tst	Storage Temperature	-25 ~ +85	°C

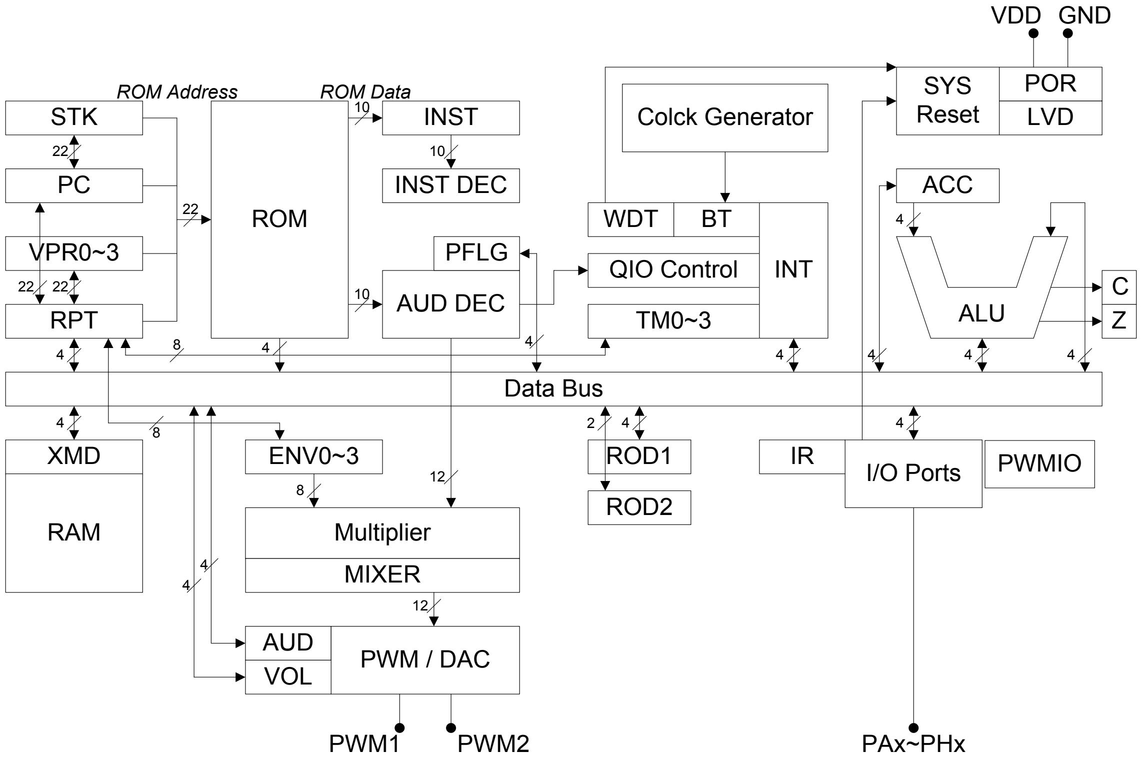
1.6.2 DC Characteristics *(Preliminary)*

Symbol	Parameter		V_{DD}	Min.	Typ.	Max.	Unit	Condition
V_{DD}	Operating voltage			2.0	3	5.5	V	1MHz & 2MHz
I_{sb}	Supply current	Halt mode	3			2	uA	Sleep, no load with on board Flash
			4.5			3		
I_{sl}		Slow mode	3		35		uA	1ms interrupt, no load
			4.5		40			
I_{op}		Operating mode	3		1.0		mA	1MHz, no loading
			4.5		1.1			
			3		1.2		mA	2MHz, no loading
			4.5		1.3			
I_{il}	Input current (Internal pull-high)	Weak (1.2M ohms)	3		2.5		uA	V_{il}=0v
			4.5		7.4			
		Strong (100K ohms)	3		30		uA	
			4.5		75			
I_{oh}	Output high current		3		-7		mA	V_{oh}=2.0V
			4.5		-11			V_{oh}=3.5V
I_{ol}	Output low current (Normal current)		3		10		mA	V_{OL}=1.0V
			4.5		16			V_{OL}=1.0V
	Output low current (Large current)		3		20		mA	V_{OL}=1.0V
			4.5		30			V_{OL}=1.0V
I_{PWM}	PWM output current (Normal)		3		60		mA	Load=8 ohms
			4.5		100			
	PWM output current (Large)		3		70		mA	Load=8 ohms
			4.5		117			
	PWM output current (Ultra)		3		80		mA	Load=8 ohms
			4.5		134			
I_{DAC}	DAC output current		3		1.4		mA	Half scale
			4.5		1.6			
ΔF/F	Frequency deviation by voltage drop		3		1.0		%	$\frac{Fosc(3.0v)-Fosc(2.4v)}{Fosc(3v)}$
			4.5		-0.5			$\frac{Fosc(4.5v)-Fosc(3.0v)}{Fosc(4.5v)}$
ΔF/F	Frequency lot deviation		3	-0.5		0.5	%	$\frac{Fmax(3.0v)-Fmin(3.0v)}{Fmax(3.0v)}$
Fosc	Oscillation Frequency		-		1		MHz	V_{DD}=2.0~5.5V
				1.99	2	2.01		

Chapter 2. Hardware Architecture

2.1 Overview

2.1.1 Function Block Diagram



2.1.2 Hardware Summary Table

Name	Function	Address
STK	1-level interrupt dedicated stack	
PC	Program counter	
VPR0~3	Voice pointer of channel 0~3	
RPT	Multi-function register pointer	M[0x0~0x4, 0x7]
XMD	Indexed RAM data access register	M[0x5]
RAM	248 nibbles RAM	
ROM	Program & data ROM	
CHNO	Active channel select	
ENV0~3	8-bit Envelope of channel 0~3	
DECMD	PCM / ADPCM control register	M[0x1CD]
Multiplier	Hardware multiplier for MIDI	
MIXER	Channels audio data mixer	
AUD	Audio output control register	M[0x1CC]
VOL	Volume control register	M[0x1C8]
PWM / DAC	PWM and D/A converter audio output	
INST	Instruction registers	

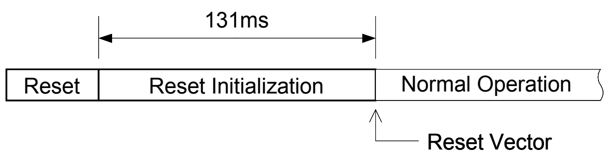
Name	Function	Address
INST DEC	Instruction decoder	
PFLG	Play flag register	M[0x1C9]
AUD DEC	Audio decoder	
Clock Generator	Ring oscillator clock generator	
WDT	Watch-dog timer and reset generator	
BT	System base timer	
QIO Control	Quick-IO control code generator	
TM0~3	Sample rate timer of channel 0~3	
INT	Interrupt generator	M[0x1CB]
ROD1	ROM[7:4] data access register	M[0x6]
ROD2	ROM[9:8] data access register	M[0x7]
SYS Reset	System reset generator	
POR	Power reset generator	
LVD	Low voltage detector and reset generator	
ACC	4-bit accumulator	
ALU	4-bit arithmetic logic unit	
C	Carry flag for arithmetic	
Z	Zero flag for arithmetic	
IR	Infrared transmit block	
I/O Ports	I/O port register	M[0x1E0~0x1EF]

M[ ] : Memory or System register and the hex number 0x? Between the brackets means its address.

2.2 Clock Generator

The clock generator is a Ring oscillator, and users can only select the internal resistor oscillation (INT-R). The INT-R oscillator accuracy is up to ± 0.5%.

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. POR set a POR flag(0x1CB[2]) to high for system low voltage management. It can be cleared by user.

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, we have a low voltage detector in the NY5+ IC. When the detector detects a harmful low voltage supply, it will cause a low voltage reset. The so-called "low voltage" point of the NY5+ IC is about 1.8v.

2.3.3 Watch-Dog Timer Reset (WDTR)

To recover from program malfunction, the NY5+ IC supports an embedded watch-dog timer reset. The WDTR function always works with the program executing. Users have clear the WDT periodically to prevent from timing up with a reset generation. Typically, the minimum time-up period of the WDT is about 15ms. Users can move a 0x5 value to the 0x1CA BTF system register to clear WDT.

2.3.4 IO Port External Reset

The PX0 (PA0, PB0, PC0, PD0, PE0, PF0, PG0 and PH0) I/O port of the NY5+ can be optioned as a reset pin. A reset pin should always be pulled-high in normal operation, whether users use the built-in internal pull-high resister option or use an external one on PCB with the floating reset option. When the reset pin falls to the ground level, it generates an external reset.

2.4 Address Pointer

The NY5+ micro-controller contains a program counter (PC), an interrupt dedicated stack (STK), a multi-function register pointer (RPT) and 4 voice pointers (VPR0~3) for channel 0~3. The length of each address pointer is 22-bit maximum, depends on the product parts. Users have to keep in mind that the initial value of all the pointers is unknown, except the PC.

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 an interrupt or a branch instruction. The interrupt vector for TOF/QIO is at address 0x000010. The interrupt vector for BT is at address 0x000018.

Inst./Event	Function
JMP	Changes the LSB 16-bit of PC, and the reminder MSB bits keep their value.
CALL	Pushes PC+2 to RPT.
RJMP	Loads RPT to PC, so users can execute a long jump.
RBPC	Reads back PC+1 to RPT.
Interrupt	Pushes PC to STK automatically.

Inst./Event	Function
IRET	Pops STK back to PC. Returns to the main program from the interrupt routine.

2.4.2 Stack (STK)

One level hardware push/pop stack dedicated to the interrupt is available. When an interrupt takes apart, 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.

2.4.3 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	Loads RPT to PC.
RBPC	Reads back PC+1 to RPT.
LDTM	Loads RPT[7:0] to TM0~3.
LDEN	Loads RPT[7:0] to ENV0~3.
RBEN	Reads back ENV0~3 to RPT[7:0].
PLAY	Loads RPT to VPR0~3.
RBVPR	Reads back VPR0~3 to RPT.
RBRO	Use RPT as address to read ROM data.
XMD	Use {RPT3, RPT2} as address to access indexed RAM data.

2.4.4 Voice Pointer (VPR)

Because NY5+ is a 4-channel sound processor, 4 voice pointers are necessary for playing speech or MIDI of each channel. When PLAY is executed, the system loads RPT to VPR of the channel which assigned by the CH# register. So users have to move the start address of the speech or MIDI data to RPT first. Besides, users can read VPR back by RBVPR instruction, because RBVPR moves VPR of the channel which assigned by the CH# register to RPT.

2.5 Arithmetic Logic Unit (ALU)

The NY5+ series 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
XORA m	A ← M[m] ⊕ A	Z
ANDA m	A ← M[m] & A	Z
ORA m	A ← M[m] | A	Z
XORL L	A ← A ⊕ L	Z
ANDL L	A ← A & L	Z
ORL L	A ← A | L	Z

2.5.1.2 Arithmetic Instruction

Instruction	Function	Flag Influenced
INCM m	M[m] ← M[m] + 1	C, Z
DECM m	M[m] ← M[m] - 1	C, Z
ADDA 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]	
MVLA	A ← L	
RSTC	C ← 0	C
SETC	C ← 1	C

The width of the memory register address `r’ of MVRM and MVMR command is 2-bit, and the MSB of the memory register is forced to be 1. 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 0x30~0x3F, but the RAM page is still working.

2.5.1.4 Conditional Branch Instruction

Instruction	Function	Flag Influenced
SANL	Skip if A != L	
CPAB	Skip if (A & L)=0	
SCEZ(CPCZ)	Skip if C = 0	
SZEZ(CPZZ)	Skip if Z = 0	

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

2.5.2 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.

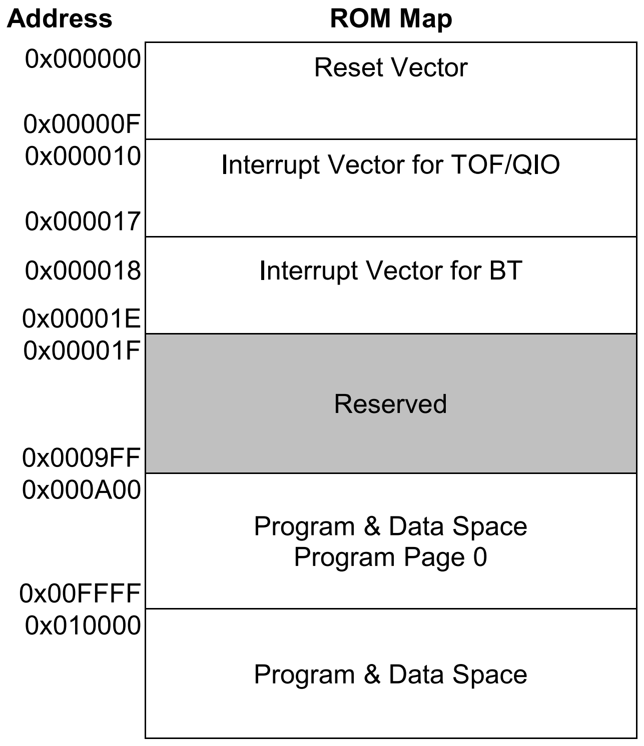
Besides RSTC and SETC commands directly assign the value of the carry flag, C is influenced by the arithmetic result. C means carry and also means the complement of borrow. If the addition operation is larger than 0xF, C=1, and C=0 if the result ≦15. If the subtraction operation is smaller than 0, C=0,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

2.6 Memory Organization

There are maximum 4M words ROM, 248 nibbles of RAM and some dedicated system control register. The registers are divided into normal system registers and 8 nibbles of Multi-function registers.

2.6.1 ROM

A large program/data/voice 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 16-bit length address of ROM, the program page size is 64K words.



2.6.2 RAM

NY5+ provide 248 nibbles RAM space. The address for RAM is 0x008~0x0FF. The first space from 0x08 to 0x3F is function RAM space, it only needs one-word instruction operation. And the second space from 0x040 to 0x0FF is data RAM space; the related operation is two-word instruction.

In addition to the immediate addressing mode, the indexed addressing mode is also supported. The page and address of the indexed RAM should be stored into RPT3 and RPT2 first, and users can read from or write in the XMD memory register to realize the indexed RAM access.

Address	RAM map
0x000 0x007	Memory Register
0x008 0x03F	56 nibble Function RAM
0x040 0x0FF	192 nibble Function RAM
0x100 0x1C7	Reserved
0x1C8 0x1CF	System registers
0x1D0 0x1DF	Reserved
0x1E0 0x1EF	System registers
0x1F0 0x1F3	Reserved
0x1F4 0x1FF	System registers

2.6.3 Memory Register

8 nibbles of memory register share the address with RAM. The page number of RAM has no relationship with the memory register address.

Address	Name	Description
0	RPT0	Multi-function register pointer bit [3:0]
1	RPT1	Multi-function register pointer bit [7:4]
2	RPT2	Multi-function register pointer bit [11:8]
3	RPT3	Multi-function register pointer bit [15:12]

Address	Name	Description
4	RPT4	Multi-function register pointer bit [19:16]
5	XMD	Indexed RAM data access register
6	ROD1	ROM data bit [7:4] access register
7	RPT5	Multi-function register pointer bit [21:20]
	ROD2	ROM data bit [9:8] access register

2.6.4 System Register

The NY5+ series provides only two instructions to access the system registers.

Address	Name	Description
1C8	VOL	16-level Volume control register
1C9	PFLG	Play flag register
1CA	BTF	Interrupt flag control register
1CB	BTC	Interrupt control register
1CC	AUD	Audio output control register
1CD	CHMD	Channel control register
1CE	RBDA	MSB 4 bits of the data after Mixer
1CF	LVD	Low Voltage Detect control register

Address	Name	Description
1E0	PA	PA control register
1E1	PAIO	PAIO control register
1E2	PB	PB control register
1E3	PBIO	PBIO control register
1E4	PC	PC control register
1E5	PCIO	PCIO control register
1E6	PD	PD control register
1E7	PDIO	PDIO control register
1E8	PE	PE control register
1E9	PEIO	PEIO control register
1EA	PF	PF control register
1EB	PFIO	PFIO control register
1EC	PG	PG control register
1ED	PGIO	PGIO control register
1EE	PH	PH control register
1EF	PHIO	PHIO control register

Address	Name	Description
1F4	PCHNO	PWM-IO CH# control register
1F5	Frame Rate	PWM-IO Frame Rate control register
1F6	Step	PWM-IO Step control register

Address	Name	Description
1F7	EN	PWM-IO Enable control register
1F8	B0_DutyL	Bit-0 Low nibbles Duty control register
1F9	B0_DutyH	Bit-0 High nibbles Duty control register
1FA	B1_DutyL	Bit-1 Low nibbles Duty control register
1FB	B1_DutyH	Bit-1 High nibbles Duty control register
1FC	B2_DutyL	Bit-2 Low nibbles Duty control register
1FD	B2_DutyH	Bit-2 High nibbles Duty control register
1FE	B3_DutyL	Bit-3 Low nibbles Duty control register
1FF	B3_DutyH	Bit-3 High nibbles Duty control register

2.6.5 Register without Memory Allocation

Name	Description	Instruction
CH#	2-bit channel number register of working channel	CHNO
TCS	2-bit timer clock source register of CH#	CHTCS
TM	8-bit sample rate timer of CH#	LDTM
ENV	8-bit envelope of CH#	LDEN
		RBEN

2.7 IO Ports

There are at most 32 I/O pins, designated as PAx through PHx, and x=0~3. All the I/O pins are bi-directional. An individual and independent register bit can determine the direction of each I/O pin. These register bits are PAIO ($1E1), PBIO ($1E3), PCIO ($1E5), PDIO ($1E7), PEIO ($1E9), PFIO ($1EB), PGIO ($1ED) and PHIO ($1EF).

Using as input pin of each I/O, there are 3 kinds of mask option. Users can select input with pull-high resistor, input without pull-high resistor, or input with register-controlled pull-high resistor (high-to-low wakeup only). If users want to enable/disable pull-high resistor by register during program execution, only high-to-low level change on this pin can wakeup NY5+. On the other hand, if the pull-high resistor is fixed by option, either high-to-low or low-to-high level change on this pin can wakeup NY5+. Users can refer Chapter 3.14 I/O Ports Register for details.

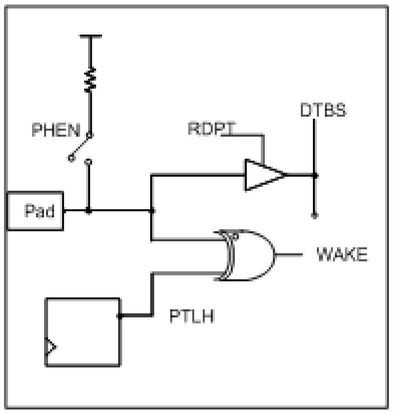
The pull-high resistor of all the I/O pins has two kinds of option: weak and strong. The weak one is about 1.2MΩ@3V for normal application and the strong one is about 100KΩ@3V usually for key matrix function. When users decide this option, the same strength of pull-high resistor will be applied to all I/O pin.

Using as output pin of each I/O, there are 3 kinds of mask option. Users can select output with normal drive current and normal sink current or normal drive current and large sink current.

The PX0 port means the PA0, PB0, PC0, PD0, PE0, PF0, PG0 or PH0 port can also be optioned as an external reset pin or an infrared (IR) output pin. A reset port can possess a pull-high resister or not, and an IR port can be initial low or high and also large sink current or not.

Category	Option	Description
PA0 PC0 PE0 PG0	Config as IR output	choose 1 of 3
	Config as RESET input	
	Config as normal IO	
	Config Pull-high	Disable / Enable
	Config large sink current	Disable / Enable
	Config PX register	wakeup status / pull-high/floating
PA1~3 PC1~3 PE1~3 PG1~3	Config Pull-high	Disable / Enable
	Config large sink current	Disable / Enable
	Config PX register	wakeup status / pull-high/floating
PB0 PD0 PF0 PH0	Config as IR output	choose 1 of 4
	Config as RESET input	
	Config as PWM-IO output	
	Config as normal IO	
	Config Pull-high	Disable / Enable
	Config large sink current	Disable / Enable
	Config PX register	wakeup status / pull-high/floating
PB1~3 PD1~3 PF1~3 PH1~3	Config Pull-high	Disable / Enable
	Config large sink current	Disable / Enable
	Config PX register	wakeup status / pull-high/floating
All I/O	I/O port pull-high resister	Weak/ Strong

2.7.1 Pull-High Input Mode



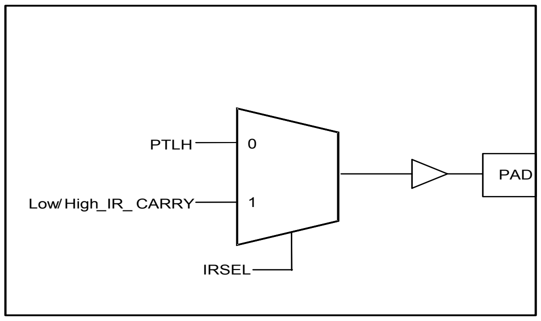
Pull-high Input Mode Configuration

Pad status of PA~PH, which are set as input mode, can be read in by MVMA. If the pads are not connected, an internal pull-high resistor will be optioned to pull the pad toward supply voltage. All I/O pins set as input mode can be used to wake-up the system, and the wake-up procedure will be launched if the comparison between PTLH and pad status is unmatched. Therefore, users have to store the current pad status into PTLH before entering Halt or Slow mode. The system will be waked-up when pad voltage change is detected.

2.7.2 Floating Input Mode

It is similar to the pull-high input mode except the internal pull-high resistor is not connected. User should apply external pull-high resistor or pull-low resistor for high-resistance switch applications.

2.7.3 Output Mode



Output Mode Configuration

User can select output mode to supply both normal drive current and normal/large sink current by setting related mask options. But drive current of NY5+ is always weaker than normal sink current, about half the scale.

2.8 Infrared Transmitter

The NY5+ series provides an infrared transmit block which is used to send infrared signal. Users can option a PX[0] (PA0, PB0, PC0, PD0, PE0, PF0, PG0, or PH0) IO as an IR output. Users can option both the IR carrier frequency and IR Low/High carrier. The IR Low/High carrier means that if users option the IR Low carrier, the IR output port sends infrared signal when the IO port register value is low, and vice versa.

Category	Option	Description
IR	IR frequency	38.46~55.56 KHz
	IR low/high carrier	Low
		High

2.9 PWM-IO Generator

The NY5+ has 4 sets of PWM-IO groups - PB/ PD/ PF/ PH, each with four PWM outputs (PB0~PB3, PD0~PD3, PF0~PF3 and PH0~PH3). Each four PWM outputs share a PWM 8-bit timer, every PWM output has its own duty and output port.

Four PWM outputs share a timer

Programmable divider of timer clock

8-bit counter for each timer

8-bit PWM duty

2.10 Interrupt Generator

There is one hardware interrupt and it has 3 different sources in NY5+. The interrupt event can be a fixed interval of the system base timer (BT), the timer overflow flag (TOF), or the quick-IO flag (QIOF). The TOF can be selected as one of the sample rate timer overflow by the register INT, and the QIOF arises as a QIO control code of any channel coming up. There is a system base timer in the NY5+ IC, which functions as long as the IC isn't in the halt mode. We provide 4 fixed intervals from the system base timer for interrupt source: 0.128, 0.256, 0.512 and 1.024ms.

As an interrupt occurs, NY5+ stores the accumulator (ACC), carry flag (C), zero flag (Z) and RAM page (PG) automatically. Then move PC to STK, and jump to the interrupt vector (0x000010 or 0x000018). An interrupt routine finishes with an IRET instruction. The IC draws the ACC, C, Z and PG back, and moves STK to PC back to jump back the main program.

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

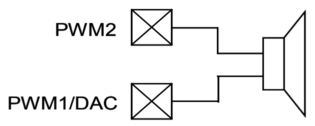
2.11 Audio Synthesizer Structure

There are 4-ch speech or MIDI audio output, and all modes are auto-played back by hardware. Different channel mode possesses different hardware structure. It provides a hardware mixer to mix the channel data. Two audio output stages: DAC and PWM are supported. Please noticed that 2MHz is required for over 2-ch speech or MIDI.

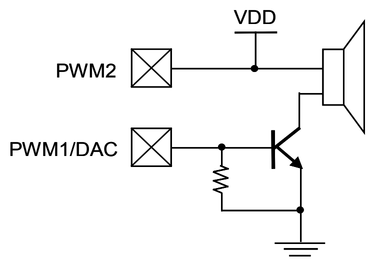
2.11.1 Audio Output

By set the AUD register, PWM or DAC can be easily chosen as the audio output stage. Besides, it provides a pad detecting mechanism. The pad detecting mechanism detects the PWM2 pad during the reset initialization period, and sets the initial value of the audio output register as PWM if the PWM2 connection is floating, or sets the initial value of the audio output register as DAC if the PWM2 connection is high. In conclusion, connect the speaker to PWM1 and PWM2 only if using PWM, otherwise connect PWM2 to VDD if using DAC. Since the mechanism sets only the initial value of AUD, don't change the value of the AUD register if the pad detecting mechanism is adopted.

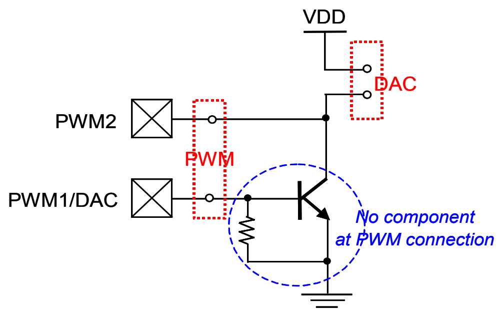
PWM2 Pad	Audio Output Initialization
Speaker (Floating)	PWM
VDD	DAC



PWM Output Connection



DAC Output Connection



PWM/DAC Connection Together

When using the PWM output, we provide an option of normal PWM current, large PWM current or ultra PWM current for different customer demand. The ultra PWM current consumes more current and makes sound louder.

2.11.2 Volume Control

Both PWM and DAC supports 16 steps hardware volume control by the VOL register, 0x0~0xF.

2.12 Low Voltage Detector (LVD)

There is one hardware voltage detector in NY5+. It offers four levels for various application, 2.0v, 2.2v, 2.4v, 2.8v, 3.0v, 3.3v and 3.6v controlled by register $LVD. The voltage detection function has to be enabled first, then select specific level for application, the flag will go to high while VDD is lower than selected level. User can check power status by setting different level and monitoring the flag.

Since VDD voltage is fixed on ICE system, so the LVD result is always same. Also for NY5+_FDB (both 4Mb & 16Mb), the lowest 3 levels 2.0v, 2.2v, and 2.4v cannot be detected due to Flash memory voltage limitation on PCB. Please verify LVD function on OTP chips.

Chapter 3. System Control

3.1 Introduction

The VOL, PFLG, AUD, CH#, TCS0~3, TM0~3 and ENV0~3 are audio control related registers. The PA~H are I/O ports registers. INT register is used to control or access the system base timer (BT) the interrupt (INT), timer overflow flag (TOF), quick-IO flag (QIOF) and watch dog timer (WDT). The combination of RPT0~5 are multi-function register pointer. The C and Z are arithmetic associated flags. The PG and XMD are RAM access registers. The ROD1 and ROD2 registers are used to read the ROM data.

3.1.1 System Register Address Map

Addr	Name	R/W	Bit	Data	Description	Initial	Wake-up
1C8	VOL	R/W	[3:0]		16- level volume control	0x4	U
1C9	PFLG	R	[0]	0/1	Play Flag of Channel 0	0x0	U
		R	[1]	0/1	Play Flag of Channel 1		
		R	[2]	0/1	Play Flag of Channel 2		
		R	[3]	0/1	Play Flag of Channel 3		
1CA	BTF	R	[0]	0/1	System base timer 0.128ms	X	X
			[1]	0/1	System base timer 0.256ms	X	X
			[2]	0/1	System base timer 0.512ms	X	X
			[3]	0/1	System base timer 1.024ms	X	X
		W	[3:0]	0101	Clear WDT	Clear	U
		W	[3:0]	1000	TOF = Timer 0 overflow	Timer 0	U
		W	[3:0]	1001	TOF = Timer 1 overflow		
		W	[3:0]	1010	TOF = Timer 2 overflow		
		W	[3:0]	1011	TOF = Timer 3 overflow		
1CB	BTC	R/W	[1:0]	00	Interrupt ~= 0.125 ms	0.125 ms	U
				01	Interrupt ~= 0.25 ms		
				10	Interrupt ~= 0.5 ms		
				11	Interrupt ~= 1.0 ms		
		R/W	[2]	0/1	POR Flag		
		R/W	[3]	0/1	BT interrupt Disable / Enable		
1CC	AUD	R/W	[0]	0/1	QIO flag, write 0 to clear flag		
		R/W	[1]	0/1	TOF flag, write 0 to clear flag		
		R/W	[2]	0/1	Audio Output Disable / Enable		
		R/W	[3]	0/1	Audio output = Current DAC / PWM output		
1CD	CHMD	R/W	[0]	0/1	QIO interrupt Disable / Enable		
		R/W	[1]	0/1	TOF interrupt Disable / Enable		
		R/W	[2]	0/1	Reserved		
		R/W	[3]	0/1	Tail Disable / Enable		
1CE	RBDA	R	[3:0]	0/1	MSB 4 bits of the data after Mixer		
1CF	LVD	R/W	[2:0]	000	Voltage Detect function disable/enable	U	U

Addr	Name	R/W	Bit	Data	Description	Initial	Wake-up
				001	Check Power < 2.0v => Flag = High		
				010	Check Power < 2.2v => Flag = High		
				011	Check Power < 2.4v => Flag = High		
				100	Check Power < 2.8v => Flag = High		
				101	Check Power < 3.0v => Flag = High		
				110	Check Power < 3.3v => Flag = High		
				111	Check Power < 3.6v => Flag = High		
		R	[3]	0/1	LVD Flag, disable to Low		
1D0~ 1DF		R/W	[3:0]		Reserved		
1E0	PA	R	[3:0]	0/1	PxIO = 1: Read input pad data		
					PxIO = 0: Read output register		
		W	[3:0]	0/1	PxIO = 1: Wake-up status (Option Disable) PxIO = 1: Floating / Pull-high (Option Enable)		
					PxIO = 0: Write to port A output register		
1E1	PAIO	R/W	[3:0]	0/1	Port X direction = Output / Input		
1E2	PB	R	[3:0]	0/1	PxIO = 1: Read input pad data		
					PxIO = 0: Read output register		
		W	[3:0]	0/1	PxIO = 1: Wake-up status (Option Disable) PxIO = 1: Floating / Pull-high (Option Enable)		
					PxIO = 0: Write to port A output register		
1E3	PBIO	R/W	[3:0]	0/1	Port X direction = Output / Input		
1E4	PC	R	[3:0]	0/1	PxIO = 1: Read input pad data		
					PxIO = 0: Read output register		
		W	[3:0]	0/1	PxIO = 1: Wake-up status (Option Disable) PxIO = 1: Floating / Pull-high (Option Enable)		
					PxIO = 0: Write to port A output register		
1E5	PCIO	R/W	[3:0]	0/1	Port X direction = Output / Input		
1E6	PD	R	[3:0]	0/1	PxIO = 1: Read input pad data		
					PxIO = 0: Read output register		
		W	[3:0]	0/1	PxIO = 1: Wake-up status (Option Disable) PxIO = 1: Floating / Pull-high (Option Enable)		
					PxIO = 0: Write to port A output register		
1E7	PDIO	R/W	[3:0]	0/1	Port X direction = Output / Input		
1E8	PE	R	[3:0]	0/1	PxIO = 1: Read input pad data		
					PxIO = 0: Read output register		
		W	[3:0]	0/1	PxIO = 1: Wake-up status (Option Disable) PxIO = 1: Floating / Pull-high (Option Enable)		
					PxIO = 0: Write to port A output register		
1E9	PEIO	R/W	[3:0]	0/1	Port X direction = Output / Input		
1EA	PF	R	[3:0]	0/1	PxIO = 1: Read input pad data		
					PxIO = 0: Read output register		
		W	[3:0]	0/1	PxIO = 1: Wake-up status (Option Disable) PxIO = 1: Floating / Pull-high (Option Enable)		
					PxIO = 0: Write to port A output register		
1EB	PFIO	R/W	[3:0]	0/1	Port X direction = Output / Input		
1EC	PG	R	[3:0]	0/1	PxIO = 1: Read input pad data		

Addr	Name	R/W	Bit	Data	Description	Initial	Wake-up
					PxIO = 0: Read output register		
		W	[3:0]	0/1	PxIO = 1: Wake-up status (Option Disable) PxIO = 1: Floating / Pull-high (Option Enable)		
					PxIO = 0: Write to port A output register		
1ED	PGIO	R/W	[3:0]	0/1	Port X direction = Output / Input		
1EE	PH	R	[3:0]	0/1	PxIO = 1: Read input pad data		
					PxIO = 0: Read output register		
		W	[3:0]	0/1	PxIO = 1: Wake-up status (Option Disable) PxIO = 1: Floating / Pull-high (Option Enable)		
					PxIO = 0: Write to port A output register		
1EF	PHIO	R/W	[3:0]	0/1	Port X direction = Output / Input		
1F0 ~ 1F3		R/W	[3:0]		Reserved		
1F4	PWM-IO CHNO	R/W	[1:0]		PCH# = 0, 1, 2. 3		
			[3:2]		Reserved		
1F5	PWM-IO FRate	R/W	[1:0]	00	Frame Rate = 128us / 16us(2M)		
				01	Frame Rate = 512us / 64us (500K)		
				10	Frame Rate = 2ms / 256us(125K)		
				11	Frame Rate = 16ms / 2ms(15.625K)		
			[3:2]		Reserved		
1F6	PWM-IO Px_Step	R/W	[0]	0/1	Px[0] 256step(8-b) / 32step(5-b)		
		R/W	[1]	0/1	Px[1] 256step(8-b) / 32step(5-b)		
		R/W	[2]	0/1	Px[2] 256step(8-b) / 32step(5-b)		
		R/W	[3]	0/1	Px[3] 256step(8-b) / 32step(5-b)		
1F7	PWM-IO Px_Enable	R/W	[0]	0/1	Px[0] Dis/En		
		R/W	[1]	0/1	Px[1] Dis/En		
		R/W	[2]	0/1	Px[2] Dis/En		
		R/W	[3]	0/1	Px[3] Dis/En		
1F8	Px[0]_Duty	R/W	[3:0]		Px[0]_Duty[3:0]		
1F9		R/W	[3:0]		Px[0]_Duty[7:4]		
1FA	Px[1]_Duty	R/W	[3:0]		Px[1]_Duty[3:0]		
1FB		R/W	[3:0]		Px[1]_Duty[7:4]		
1FC	Px[2]_Duty	R/W	[3:0]		Px[2]_Duty[3:0]		
1FD		R/W	[3:0]		Px[2]_Duty[7:4]		
1FE	Px[3]_Duty	R/W	[3:0]		Px[3]_Duty[3:0]		
1FF		R/W	[3:0]		Px[3]_Duty[7:4]		

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]	X	U
1	RPT1	R/W	[3:0]	Multi-function register pointer [7:4]	X	U
2	RPT2	R/W	[3:0]	Multi-function register pointer [11:8]	X	U
3	RPT3	R/W	[3:0]	Multi-function register pointer [15:12]	X	U
4	RPT4	R/W	[3:0]	Multi-function register pointer [19:16]	X	U

Addr	Name	R/W	Bit	Description	Initial	Wake-up
5	XMD	R/W	[3:0]	Indexed RAM data access register	X	X
6	ROD1	R/W	[3:0]	ROM[7:4] data access register	X	U
7	ROD2	R/W	[1:0]	ROM[9:8] data access register	X	U
	RPT5		[3:2]	Multi-function register pointer [21:20]	X	U

3.1.3 Register without Memory Allocation Map

Name	R/W	Bit	Description	Initial	Wake-up
C	-	1	Arithmetic carry flag	0x0	U
Z	-	1	Arithmetic zero flag	0x0	U
CH#	W	2	Active channel number	0x0	U
TCS0	W	3	Timer clock source selection of TM0	0x3	U
TCS1	W	3	Timer clock source selection of TM1	0x3	U
TCS2	W	3	Timer clock source selection of TM2	0x3	U
TCS3	W	3	Timer clock source selection of TM3	0x3	U
TM0	W	8	Sample rate timer of channel 0	0x7F	U
TM1	W	8	Sample rate timer of channel 1	0x0	U
TM2	W	8	Sample rate timer of channel 2	0x0	U
TM3	W	8	Sample rate timer of channel 3	0x0	U
ENV0	R/W	8	Envelope of channel 0	X	U
ENV1	R/W	8	Envelope of channel 1	X	U
ENV2	R/W	8	Envelope of channel 2	X	U
ENV3	R/W	8	Envelope of channel 3	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

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

3.2 RPT

The RPT of NY5Q040A and NY5Q060A is 18-bit long, and the NY5Q172A's RPT is 19-bit except the NY5Q346A's is 20-bit. The redundant bits of RPT (PPT[21] of NY5Q346A , RPT[21:20] of NY5Q172A , and RPT[21:18] of other NY5QxxxA Body) are un-writable and reveal 0 if users read them. Users have to watch out that the RPT5 is 2-bit and its allocation is [3:2]. The functions of RPT are listed in the section 2.4.3. Whether the bits of RPT are redundant or useful, user have to initial all RPT (RPT[21:0]) to "0".

Besides the instructions related to the TM and the ENV only access bit [7:0] of the RPT, The XMD only access bit [15:8] of the RPT, others access all available bits. The RPT will be frequently accessed because of its multi-functionality, so the NY5+ series provides 2 instructions to accelerate the access of RPT0~3: MVRM and MVMR.

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 NY5+ series 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 RAM Control Register

3.4.1 XMD

As mentioned in section 2.6.2, users access XMD taking the RPT3[3:2] as the RAM page and the {RPT3[1:0], RPT2} as the address. Users have to watch out that the NY5+ series does not support using XMD to access memory registers, so the {RPT3, RPT2} can’t be 0x0~0x7 when accessing XMD.

3.5 I/O Ports Register

As PA, PB, PC, PD, PE, PF, PG and PH are bi-directional I/O ports, System register PAIO, PBIO, PCIO, PDIO, PEIO, PFIO, PGIO and PHIO are used to determine the direction of each I/O pin. Writing 1 to any bit of register PXIO (X=A~H), the corresponding I/O pin is configured as input pin. Writing 0 to any bit of PXIO (X=A~H), the corresponding I/O pin is configured as output pin. For I/O pin used as input pin, reading system register PX (X=A~H) will obtain current state on I/O pin.

For I/O pin used as input pin, there is a mask option to define whether pull-high resistor of corresponding I/O pin can be enabled or disabled during program execution. When this mask option is disabled, either high-to-low or low-to-high level change on this pin can wake up NY5+ from Halt mode or Slow mode. Therefore, user has to read the I/O pin state before entering Halt mode or Slow mode and write back to register PX[y] (X=A~H, y=0~3). When 1 is written to register PX[y], high-to-low level change on this pin will wake up NY5+. When 0 is written to register PX[y], low-to-high level change on this pin will wake up NY5+. However, user can enable or disable pull-high resistor during program execution when wake-up mask option is enable. When this mask option is enabled, only high-to-low level change on this pin can wake up NY5+ from Halt mode or Slow mode. On the other hand, the pull-high resistor can be disabled or enabled again during program execution by writing 1 or 0 to register PX[y]. When 1 is written to PX[y], the pull-high resistor is enabled and when 0 is written to PX[y], the pull-high resistor is disabled.

For I/O pin used as output pin, writing value to register PX is to write this value to output register of this I/O pin.

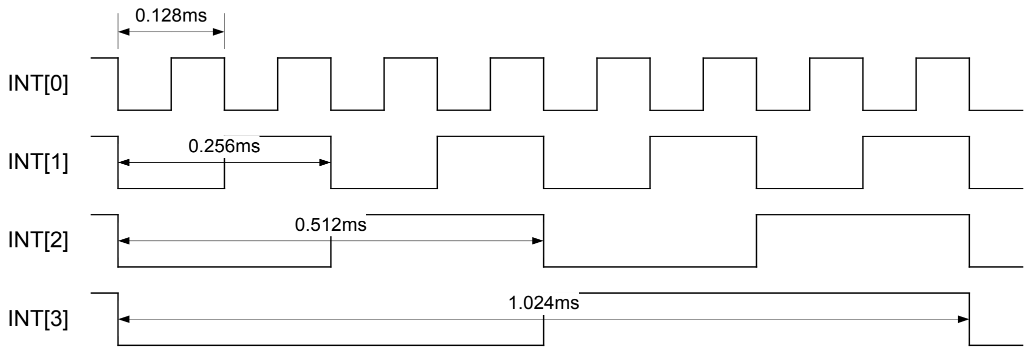
The register value of an output pin simply means the output data. If the pin is an IR output, it outputs the IR carrier frequency when the register is 0 and the IR low/high carrier option is low; it outputs 1 when the register is 1 and the IR low/high carrier option is low. An IR port output 0 when the register is 0 and the IR low/high carrier option is high; it outputs the IR carrier frequency when the register is 1 and the IR low/high carrier option is high.

3.6 INT

The reading source and the writing destination of the system register of address 0x2 are different.

3.6.1 System Base Timer Polling

Reading the 4-bit data of BTF acquires the value of the BT counter. The NY5+ series provides 4 different base timer intervals for polling: 0.128ms, 0.256ms, 0.512ms and 1.024ms. The value of time means the period, so polling a data toggle means half time of the interval.



INT timing figure

3.6.2 Interrupt Source

As mentioned in the section 2.10, the only one interrupt has 6 interrupt sources including 4 different BT intervals, the TOF and the QIOF.

Writing 0x0 to 0x3 to BTC[1:0] selects the BT interrupt source. Writing 0x1/0x0 to BTC[3] turns on/off the interrupt generator. Writing 0x1/0x0 to CHMD[1] turns on/off the TOF interrupt generator. Writing 0x1/0x0 to CHMD[0] turns on/off the QIOF interrupt generator.

Remember to set the source of interrupt before turning the interrupt on. If users want to change the interrupt source, turn off the interrupt first, set the source, and then turn it on.

3.6.3 TOF source

The TOF arises when the TM of the chosen channel overflows. Writing 0x8 to 0xB to BTF selects the TOF source. Always clear the TOF after setting the TOF source. Chosen channel is redundant leads the TOF never arising.

3.6.4 Flag Clear

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

Writing 0x0 to the AUD[0] clears the QIOF.

Writing 0x5 to the BTF clears the WDT.

3.7 Audio Control Register

3.7.1 VOL

The VOL register specifies the digital volume control of Mixer. The VOL has 16 steps. 0x0 means the mute mode and 0xF is the loudest level. Recommended VOL value associated with total active channels are listed in the table below.

3.7.2 PFLG

The PFLG register contains the play flag of each channel. PFLG=1 means the channel is in the act of playing, and PFLG=0 means the channel is stopped. PLAY command sets the PFLG to 1, and STOP command sets the PFLG to 0.

3.7.3 MIX

The NY5+ series can play 4 channels speech or MIDI. The mixer mixes these four channels.

3.7.3.1 Audio Data MSB

Reading the 4-bit data of RBDA acquires the value of the MSB 4-bit audio data (16 levels). This information helps users to get the amplitude of the playing sound. 0x0 means the smallest and 0xF means the largest level of the output audio data.

3.7.4 AUD

The reading source and the writing destination of the system register of address 0x1CC are different.

3.7.4.1 Audio Related Flag

Reading from bit [0] shows the QIOF value. Quick-IO control code is a special code buried in the encoded speech data. As anyone of the 4 channels got a QIO code, the QIOF arises. Whatever users take it as the interrupt source or a polling object, remember to clear it.

Reading from bit [1] shows the TOF value. The TOF represent the timer selected by the TOF source register has been overflowed. Whatever users take it as the interrupt source or a polling object, remember to clear it.

Reading from bit [3] shows the audio output selected. The flag tells us we are using PWM or DAC now. If using DAC, it is 0, and it is 1 if using PWM. The initial value shows (~AUD2) means the pad detecting mechanism. If AUD2=1, then the flag is 0, so DAC is chosen. If AUD2 connected to a speaker which means floating, and AUD2=0 because of the built-in weak pull-low resistance, then the flag is 0, so PWM is chosen.

3.7.4.2 Audio Output Control Register

Writing 0x0 to the AUD[3] selects DAC, and writing 0x1 to the AUD[3] selects PWM. The initial value shows (AUD2) also means the pad detecting mechanism.

Writing 0x0 to the AUD[2] turns the selected audio output off, and writing 0x1 to the AUD[2] turns it on. Turning off the audio output can save the power consumption.

3.7.5 CH#

The CH# register indicates the active channel which affects the CHTCS, LDTM, RBVPR, LDEN, RBEN, STOP and PLAY instruction operation. So the access of the TCSx, TMx and ENVx register has relations with the CH#.CHNO loads the ACC to the CH#, and RBCH backup the CH# to ACC.

3.7.6 TCSx

The TCSx registers indicate the TM clock source of the `x' channel. Different channel mode has different frequency of the TCS. The value of the TCS affects MIDI octave. CHTCS loads the ACC to the TCSx of the channel which the CH# indicates, and the TCSx can't be read.

TCSx	TMx Source
0x0	8M
0x1	4M
0x2	2M
0x3	1M

3.7.8 TMx

The TMx registers include a set 8-bit timer reload value latch and a set 8-bit downward counter of the `x' channel. We load the latch and counter by LDTM, then the counter counts down until to 0, and then the counter is automatically reloaded from the latch. When the counter counts to 0, the audio engine plays the audio data by hardware. Setting the TM to 0 stops the counter. The TM value affects the sample rate of a speech or the note pitch of MIDI.

Loading 0 to the TM stops the timer counting and pauses the audio engine of the CH#.

LDTM loads the RPT[7:0] to the TM of the channel which the CH# indicates, and the TM can't be read.

$TM = (F_{TCS} / F_{SR}) – 1$

TM : TM value in decimal

F_{TCS} : Frequency of the timer clock source

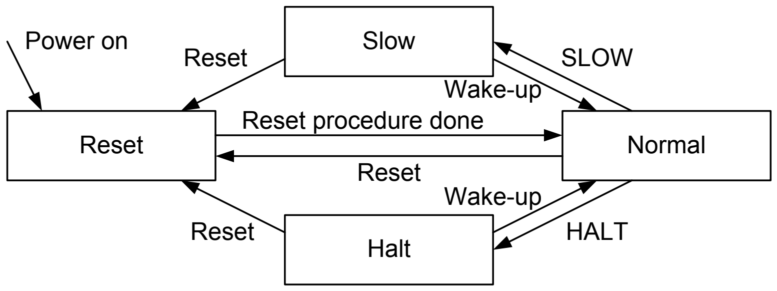
F_{SR} : Frequency of the sample rate

3.7.9 ENVx

The 8-bit ENVx registers control the envelope of the `x' channel. Different envelope variations of a tone or MIDI timbre can display varied expressions.

LDEN loads the RPT[7:0] to the ENV of the channel which the CH# indicates, and RBEN reads the ENV of the channel which the CH# indicates to the RPT[7:0].

3.8 Power Saving Mode



Power Saving Mode Flow Chart

3.8.1 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 48us). 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.

3.8.2 Slow Mode

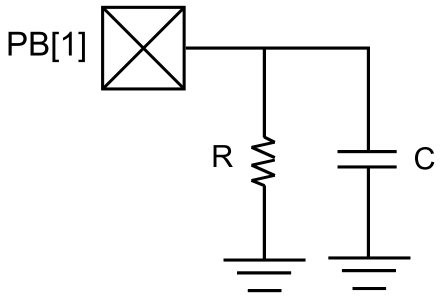
The system enters the slow mode if the SLOW command executed. The system clock in the slow mode slows down about 32 times slower than in the normal mode. The difference between the halt mode and the slow is only the system clock. So the IC can be waked-up from the slow mode by the interrupt in addition to the input port. Since the sample rate timer and the audio engine are suspended during the slow mode, interrupt from TOF and QIOF in the slow mode can't operate, of course can't wake-up the system.

The input wake-up manner is the same as the halt mode. So before executing the SLOW 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 slow mode by a reset port, it goes into the reset procedure. After the IC is waked-up by the input port or an INT of BT, the succeeding instructions after the SLOW instruction will be executed after the wake-up stable time (128us maximum).

Remember to turn off the audio output before entering to the slow mode.

3.8.3 RC Slow Mode

Another power saving mode is RC Slow mode, using external components with PB[1] and HALT commands, an application example is shown below.



There are three differences between using RC slow mode and slow mode. The first is that the user can adjust the value of R and C to determine the wake-up interval, instead of using the fixed BT interrupt. The second is that under the same interval time, the value of R can be increased, which saves more power during work. Third, the interval time of slow mode between different ICs will be slightly different and cannot be corrected. In RC slow mode, all products can get closer results by adjusting the value of RC. And it should be noted that the RC slow mode requires a resistor and a capacitor for charging and discharging, and the charging and discharging time will vary slightly with the voltage.

Chapter 4. Audio Playback

4.1 Introduction

The NY5+ series can play 4 channels speech or MIDI. The mixer mixes these four channels and output the result audio data through DAC or PWM. VOL controls the output volume level.

4.2 Speech Playback

4.2.1 Speech Playing Procedure

Play a speech by setting the sample rate to TM and the speech data starting address to VPR, and then executing the PLAY command. The playing will continue to play until the speech ends or users pause or stop it. Stop a speech by STOP or pause it by setting the TM to 0. When a speech ends, the PFLG of the playing channel falls to 0 automatically, and users can observe the PFLG to be aware of the channel state. STOP automatically pulls the PFLG to 0 and stops the speech data to the middle value (0x800). Setting the TM to 0 pauses the speech, but keeps the speech data unchanged.

RBVPR is a useful instruction to check the playing address of the speech data.

Step	Process	Instruction
1	Set CH#	CHNO
2	Set TCS 8M/4M/2M/1M	CHTCS
3	Set TM in RPT1, RPT0	
4	Load TM from RPT	LDTM
5	Set VPR in RPT	
6	Start to play	PLAY
7	Stop playing	STOP

4.2.2 Speech Data

The NY5+ series supports 12-bit PCM and encoded ADPCM speech data. Of course, the PCM speech has higher quality and occupies more ROM space than the ADPCM one. Use the encode software provided by the Nyquest to generate the PCM or ADPCM speech data. One important thing to take care when users put the speech data to the ROM is the starting address of the data.

4.2.3 Quick-IO

The NY5+ series products support the quick-IO control. The QIO is a way to bury a control code into the speech data to handle the I/O ports. Use the Quick-IO software provided by Nyquest to utilize the function. As mentioned before, when playing a QIO buried data, the QIOF arises as a QIO code occurs. So users could do anything they want by managing the QIO control table.

4.3 MIDI Control

4.3.1 MIDI Playing Procedure

The NY5+ series has three kinds of mode to synthesize MIDI melody. Play the starting address of the timbre and it will be automatically played back unless users stop it by STOP. PLAY pulls the PFLG to 1 and loads the stating address in the RPT to the VPR. STOP pushes the PFLG to 0, stops the playing MIDI to the middle value (0x800) immediately.

RBVPR reads the playing address of the MIDI data.

Step	Process	Instruction
1	Set CH#	CHNO
2	Set TCS to alter the octave	CHTCS
3	Set Tail Loop Enable	
4	Set TM in RPT1, RPT0	
5	Change TM to alter the pitch	LDTM
6	Set the timbre data as VPR in RPT	
7	Start to play the timbre	PLAY
8	Set ENV in RPT1. RPT0	
9	Change ENV to alter the intensity	LDEN
10	Stop playing after the playing timbre	
11	Stop playing immediately	STOP

Chapter 5. Instruction Set

5.1 Instruction Table

Inst.	OP1	OP2	Operation	Flag
Data Transfer, ROM size=1, Cycle=1				
MVAM	6m		Move A to M	
MVMA	6m		Move M to A	Z
MVRM	4m	2r	Move R to M	
MVMR	4m	2r	Move M to R	
MVLA	4L		Move L to A	
Arithmetic, ROM size=1, Cycle=1				
INCM	6m		M = M +1	C, Z
DECM	6m		M = M - 1	C, Z
ADDA	6m		A = A + M + C	C, Z
ADDL	4L		A = A + L + C	C, Z
XORA	6m		A = A ^ M	Z
ANDA	6m		A = A & M	Z
ORA	6m		A = A | M	Z
XORL	4L		A = A ^ L	Z
ANDL	4L		A = A & L	Z
ORL	4L		A = A | L	Z
INCA			A = A + 1	C, Z
DECA			A = A - 1	C, Z
CLRC			Reset C = 0	C
SETC			Set C = 1	C
Conditional, ROM size=1, Cycle=1 or 1+N				
SANL			Skip if A != L	
CPAB			Skip if (A & L) == 0	
SCEZ			Skip if C = 0	
SZEZ			Skip if Z = 0	

Inst.	Operation	ROM Size	Cycle
Branch & Pointer			
JMP	Jump to Address[16]	2	2
CALL	Jump to Address[16], and Move PC+2 to RPT	2	2
RBRO	Move ROM[RPT] data to ROD and ACC	1	3
Audio			
CHNO	Move A[1:0] to CH#	1	1
RBCH	Move CH# to A[1:0]	1	1
CHTCS	Move A[2:0] to TCS[CH#]	1	1
LDTM	Move RPT[7:0] to TM[CH#]	1	1
RBVPR	Move VPR[CH#] to RPT	1	3
LDEN	Move RPT[7:0] to ENV[CH#]	1	1
RBEN	Move ENV[CH#] to RPT[7:0]	1	1
STOP	STOP playing CH# immediately	1	1
PLAY	Move RPT to VPR[CH#]	1	3
Other			
SMKI	Mask interrupt	1	1
CMKI	Non-Mask interrupt	1	1
IRET	Move STK to PC	2	2
RJMP	Move RPT to PC	1	2
RBPC	Move PC+1 to RPT	1	2
HALT	Enter HALT mode	1	1
SLOW	Enter SLOW mode	1	1
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
CH# : 2-bit channel number register
TCS : 2-bit channel clock source selection register of CH#

RPT : Multi-function register data
TM : 8-bit timer data of each channel
VPR : Voice address pointer of CH#
ENV : 8-bit envelope data of CH#
ROM : 10-bit ROM data
ROD : ROM data access register data
PC : Program counter address pointer
STK : Interrupt dedicated stack address pointer

r : Memory register address

m : RAM or Memory/ System register address,
ROM size=2 and cycle=2 if m=8

5.2 Instruction Descriptions

5.2.1 Arithmetic Instructions

<u>INCM m</u>

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

Operation: M ← 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

M0=0x1, C=0, Z=0

<u>DECM m</u>

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

Operation: M ← 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

M0=0xF, C=0, Z=0

<u>ADDA m</u>

Function: Add M of address m and C to A, 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: ADDA m0

Before Instruction

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

After Instruction

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

<u>XORA m</u>

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: 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 M.

Operation: A ← A & M

Operand: 0x0 ≤ m ≤ 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: ANDA m0

Before Instruction

A=0x7, M0=0xA

After Instruction

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

ORA m

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

Operation: A ← A | M

Operand: 0x0 ≤ m ≤ 0x3F

Words: 1

Cycles: 1

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 ≤ m ≤ 0x3F

Words: 1

Cycles: 1

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 ≤ 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

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

After Instruction

A=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 AL.

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

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

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

5.2.2 Conditional Instructions

SANL 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: SANL 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 & L 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 = 0x0 `CALL a1' is executed

If A&L≠0x0 `CALL a1' is discarded, and `CALL a2' is executed

SZEZ(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: 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

SCEZ(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: 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

5.2.3 Audio Instructions

CHNO

Function: Move A[1:0] to CH#.

Operation: CH# ← A[1:0]

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CHNO

Before Instruction

A=0x3

After Instruction

CH#=0x3

RBCH

Function: Move CH# to A[1:0].

Operation: A[1:0] ← CH#

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CHNO

Before Instruction

CH#=0x3

After Instruction

A=0x3

CHTCS

Function: Move A[1:0] to TCS of CH#.

Operation: TCS ← A[1:0]

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CHTCS

Before Instruction

CH#=2, A=0x1

After Instruction

TCS2=0x1

LDTM

Function: Load RPT[7:0] value to TM of CH#.

Operation: TM ← RPT[7:0]

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MVLR 0x5, 0x1

MVLR 0x3, 0x0

LDTM

Before Instruction

CH#=1

After Instruction

RPT0=0x3, RPT1=0x5, TM1=0x53

RBVPR

Function: Read address in VPR of CH# to RPT.

Operation: RPT ← VPR

Operand: None

Words: 2

Cycles: 3

Operative Flags: None

Flags Affected: None

Example: RBVPR

Before Instruction

CH#=0, VPR0=0xABCDE

After Instruction

RPT=0xABCDE

RBEN

Function: Load ENV of CH# value to RPT[7:0].

Operation: RPT[7:0] ← ENV

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: RBEN

Before Instruction

CH=0x1, ENV1=0x48.

After Instruction

RPT[7:0]=0x48

LDEN

Function: Load RPT[7:0] value to ENV of CH#.

Operation: ENV ← RPT[7:0]

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MVLA 0x6

```
MVAM RPT1
MVLA 0x7
MVAM RPT0
LDEN
```

Before Instruction

CH#=2.

After Instruction

ENV2=0x67

STOP

Function: Stop all audio playing of CH# immediately,

and force the audio data of CH# to 0x200.

Operation: Stop playing

PFLG[CH#] ← 0

Audio data of CH# ← 0x200

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: STOP

Before Instruction

CH=0x1, MD1≠0x0

After Instruction

CH1 stopped, PFLG[1]=0,

Audio data of CH1 = 0x200

PLAY

Function: Play an audio in CH#. The voice data

address should be loaded into RPT first.

Operation: VPR[CH#] ← RPT

PFLG[CH#] ← 1

Operand: None

Words: 1

Cycles: 1 or 3

Operative Flags: None

Flags Affected: None

Example: PLAY

Before Instruction

RPT=0x12345

After Instruction

VPR[CH#]=0x12345, PFLG[CH#]=1

5.2.4 Other Instructions

SMKI

Function :Enable interrupt entrance

Operation:

Operand :None

Words :1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :SMKI

Before Instruction

Interrupt entrance is disable

After Instruction

Interrupt entrance is enable

CMKI

Function : Disable interrupt entrance

Operation:

Operand : None

Words :1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :CMKI

Before Instruction

Interrupt entrance is enable

After Instruction

Interrupt entrance is disable

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[21:16] will not be changed.

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[21:16] will not be changed

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.

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=0x54321

After Instruction

PC=0x54321

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.

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

SLOW

Function: Enter the slow mode.

Operation: Slow down the system clock

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: SLOW

After Instruction

The system enters the slow mode and the system clock slows down.

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.