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

USER MANUAL

NY5 Series

4-Channel Speech/Midi MCU with 8~24 I/O

Version 2.1

Jan. 28, 2021

Revision History

Version	*Date*	*Description*	*Modified Page*
1.0	2012/10/31	1. NY5A/5B/5C system clock can be 1MHz or 2MHz. 2. NY5A is 4-channel speech/MIDI synthesizer with 224 nibbles RAM. 3. Operating voltage of NY5A/5B/5C can be as low as 2.0V at 2MHz.	-
1.1	2013/11/29	1. Modify RPT descriptions.	24
1.5	2014/11/21	1. Change IC body to “C” version of NY5C158x, NY5C185x, NY5C305x and NY5C345x.	7
1.6	2016/09/22	1. Change IC body to “C” version of NY5C112x, NY5C132x, NY5C225x and NY5C265x.	7
1.7	2019/05/22	NY5P(J) does not support Volume Control in DAC mode.	7, 21, 27
1.8	2019/11/26	Add NY5A018C/25C, NY5B035C/45C IC bodies.	7, 8
1.9	2020/03/20	Add NY5A(C), NY5B(C) and remove corresponding (B) version.	7
2.0	2020/07/13	Add NY5A075C, NY5A085C, and NY5B046C. Remove NY5B045C	7, 11
2.1	2021/01/28	Remove NY5C450B, NY5C520B, NY5C640B and NY5C720B.	-

Table of Contents

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 48 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

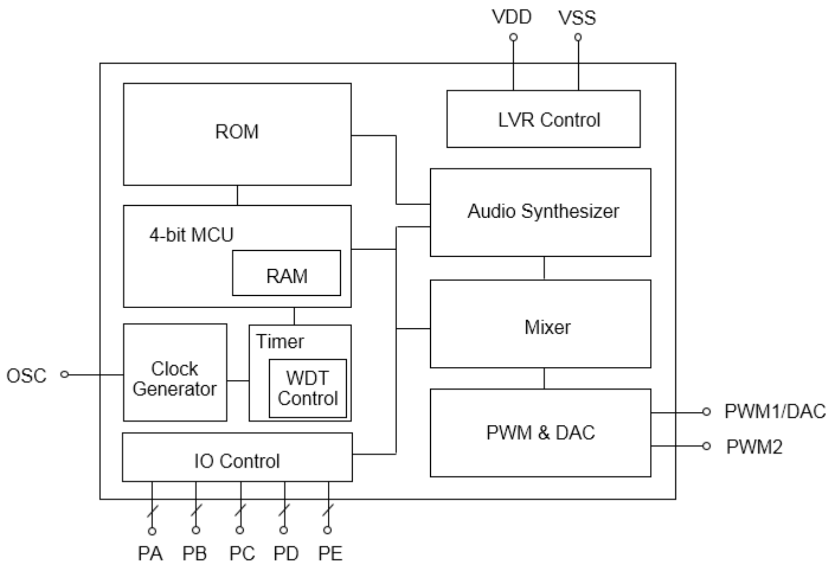
- Wide operating voltage range: 2.0V to 5.5V.
- 4-bit RISC type micro-controller with 48 instructions.
- 1.5Mx10-bit ROM maximum, program and voice data share the same ROM space.
- 224x4-bit RAM maximum, indirect RAM addressing mode is supported.
- 1MHz or 2MHz instruction frequency.
- SLOW mode to operate at low power consumption.
- HALT mode to save power, less than 1uA@3V standby current.
- Precisely embedded oscillator with build-in resistor (+/- 1%). External resistor to adjust system frequency is optional.
- Low voltage reset (LVR=1.8V), watch-dog reset and I/O port reset are all supported to protect the system.
- One interrupt entrance with an independent stack, multiple interrupt sources.
- 20 flexible I/Os maximum with optional function: input, output, large current output, IO, floating-type reset, pull-high reset, IR carrier output and large current IR carrier output.
- Support Open-Drain (OD) bi-direction IO.
- Infrared output: optional IR carrier frequency and optional data high/low IR output supported.
- NY5A/5B/5C 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.
- 256 points instrument waveform provides outstanding MIDI quality for MIDI.
- 256 steps envelope control for tone and MIDI.
- High quality direct-drive 9-bit PWM and 10-bit D/A converter audio output.
- Support large PWM current output.
- 16 steps volume control for audio output. (NY5PxxxJ does not support volume control in DAC Mode)
- Quick-IO control supported.
- Mute mode speech algorithm to save ROM size.

1.3 Product List

IC Type	Time* (sec)	ROM (bits)	RAM (bits)	I/O	Channel	PWM	D/A
NY5A003C	3.3	12K x 10	224 x 4	8	4	1	1
NY5A005C	5.0	16K x 10	224 x 4	8	4	1	1
NY5A008C	8.3	24K x 10	224 x 4	8	4	1	1
NY5A011C	11.7	32K x 10	224 x 4	8	4	1	1
NY5A018C	18.3	48K x 10	224 x 4	8	4	1	1
NY5A025C	25.0	64K x 10	224 x 4	8	4	1	1
NY5A035C	35.0	88K x 10	224 x 4	8	4	1	1
NY5A045C	45.0	112K x 10	224 x 4	8	4	1	1
NY5A055C	55.0	136K x 10	224 x 4	8	4	1	1
NY5A065C	65.0	160K x 10	224 x 4	8	4	1	1
NY5A075C	75.0	184K x 10	224 x 4	8	4	1	1
NY5A085C	85.0	208K x 10	224 x 4	8	4	1	1
NY5B005C	5.0	16K x 10	224 x 4	15	4	1	1
NY5B008C	8.3	24K x 10	224 x 4	15	4	1	1
NY5B011C	11.7	32K x 10	224 x 4	15	4	1	1
NY5B018C	18.3	48K x 10	224 x 4	15	4	1	1
NY5B025C	25.0	64K x 10	224 x 4	15	4	1	1
NY5B035C	35.0	88K x 10	224 x 4	15	4	1	1
NY5B046C	45.0	112K x 10	224 x 4	15	4	1	1
NY5B055C	55.0	136K x 10	224 x 4	15	4	1	1
NY5B065C	65.0	160K x 10	224 x 4	15	4	1	1
NY5B075C	75.0	184K x 10	224 x 4	15	4	1	1
NY5B085C	85.0	208K x 10	224 x 4	15	4	1	1
NY5B112C	111.7	272K x 10	224 x 4	15	4	1	1
NY5B132C	131.7	320K x 10	224 x 4	15	4	1	1
NY5B158C	158.3	384K x 10	224 x 4	15	4	1	1
NY5B185C	185.0	448K x 10	224 x 4	15	4	1	1
NY5C112C	111.7	272K x 10	224 x 4	20	4	1	1
NY5C132C	131.7	320K x 10	224 x 4	20	4	1	1
NY5C158C	158.3	384K x 10	224 x 4	20	4	1	1
NY5C185C	185.0	448K x 10	224 x 4	20	4	1	1
NY5C225C	225.0	544K x 10	224 x 4	20	4	1	1
NY5C265C	265.0	640K x 10	224 x 4	20	4	1	1
NY5C305C	305.0	736K x 10	224 x 4	20	4	1	1
NY5C345C	345.0	832K x 10	224 x 4	20	4	1	1

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

1.4 Block Diagram



1.5 Pad Description

Pin	ATTR.	Description
VDD#	Power	Positive power
GND#	Power	Negative power
OSC	I	External resistor for oscillator input
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

* NY5A: PA0~PB3 (OSC pad is shared with PB3)

* NY5B: PA0~PD2 (OSC pad is shared with PD2)

* NY5C112x ~ NY5C345x : PA0~PE3 (OSC pad is shared with PE3)

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
V_{DD} - V_{SS}	Supply voltage	-0.5 ~ +6.0	V
Vin	Input voltage	V_{SS}–0.3V ~ V_{DD}+0.3	V
Top	Operating Temperature	0 ~ +70	°C
Tst	Storage Temperature	-25 ~ +85	°C

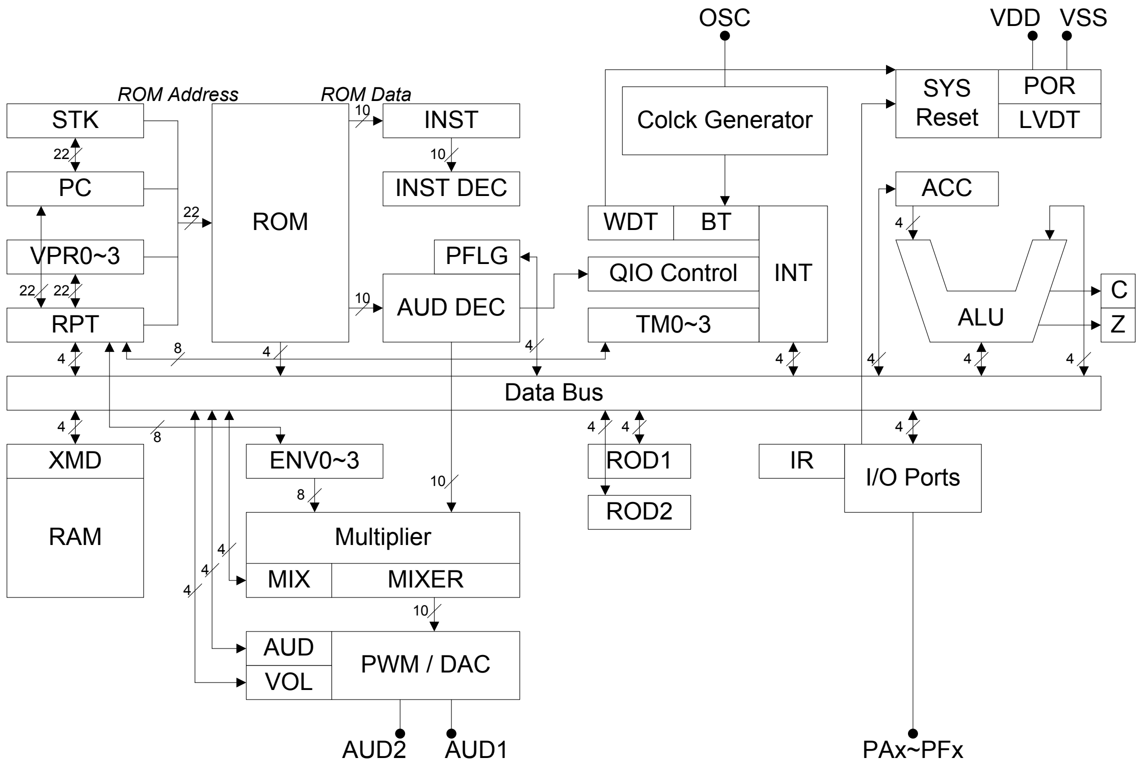
1.6.2 DC Characteristics

Symbol	Parameter		VDD	Min.	Typ.	Max.	Unit	Condition
V_{DD}	Operating voltage			2.0	3	5.5	V	1MHz & 2MHz
I_{sb}	Supply current	Halt mode	3			1	uA	Sleep, no load
			4.5			1		
I_{sl}		Slow mode	3		150		uA	1ms interrupt, no load
			4.5		350			
I_{op}		Operating mode	3		1.2		mA	1MHz, no loading
			4.5		3			
			3		1.5		mA	2MHz, no loading
			4.5		3.5			
I_{il}	Input current (Internal pull-high)	Weak (850k ohms)	3		-3.5		uA	V_{il}=0v
			4.5		-10			
		Strong (480k ohms)	3		-7		uA	
			4.5		-20			
I_{oh}	Output high current		3		-10		mA	V_{oh}=1.0V
			4.5		-22			V_{oh}=2.2V
I_{ol}	Output low current (Normal current)		3		10		mA	V_{ol}=2.0V
			4.5		20			V_{ol}=2.5V
	Output low current (Large current)		3		20		mA	V_{ol}=2.0V
			4.5		40			V_{ol}=2.5V
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			
I_{DAC}	DAC output current		3	-0.36 ~ -4.20			mA	Half scale
			4.5	-0.47 ~ -4.85				
$\Delta F/F$	Frequency deviation by voltage drop		3		1.0		%	$\frac{Fosc(3.0v)-Fosc(2.4v)}{Fosc(3v)}$
			4.5		-0.5			$\frac{Fosc(4.5v)-Fosc(3.0v)}{Fosc(4.5v)}$
$\Delta F/F$	Frequency lot deviation		3	-1		1	%	$\frac{Fmax(3.0v)-Fmin(3.0v)}{Fmax(3.0v)}$
Fosc	Oscillation Frequency		-	0.90	1	1.05	MHz	V_{DD}=2.0~5.5V
				1.80	2	2.10		

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	224 nibbles RAM	
ROM	Program & data ROM	
ENV0~3	8-bit Envelope of channel 0~3	
Multiplier	Hardware multiplier for MIDI	
MIX	Mixer control register	T[0x3]
MIXER	Channels audio data mixer	
AUD	Audio output control register	T[0x4]
VOL	Volume control register	T[0x0]
PWM / DAC	PWM and D/A converter audio output	
INST	Instruction registers	
INST DEC	Instruction decoder	

Name	Function	Address
PFLG	Play flag register	T[0x1]
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	T[0x2]
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	
LVDT	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	T[0x8~0xF]

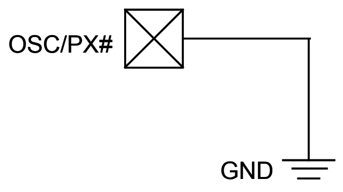
M[ ] : Memory register and the hex number 0x? between the brackets means its address.

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

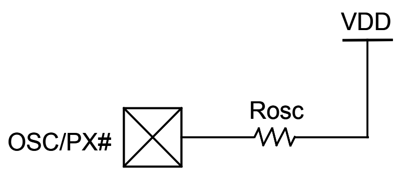
2.2 Clock Generator

The clock generator is a Ring oscillator, and users can select the internal resistor (INT-R) or the external resistor (EXT-R). A precise INT-R oscillator is provided, and its accuracy is up to ±1%.

If OSC/PX# pad is optioned as PX# I/O function, there is only INT-R, and EXT-R is disabled. When this pad is optioned as OSC function oppositely, INT-R or EXT-R can be determined by the configurations below.

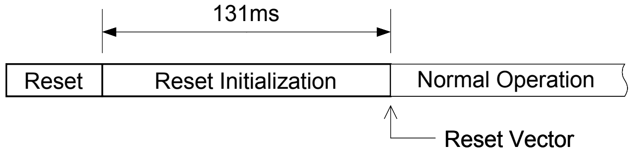


INT-R Oscillator Connection



EXT-R Oscillator Connection

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.

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 6ms. Users can move a 0xE value to the 0x2 INT system register to clear WDT.

2.3.4 IO Port External Reset

The PX0 (PA0, PB0, PC0, PD0 and PE0) 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 is at address 0x000010.

Inst./Event	Function
JMP	Changes the LSB 14-bit of PC, and the reminder MSB bits keep their value.
CALL	Pushes PC+2 to RPT.
LDPC	Loads RPT to PC, so users can execute a long jump.
RBPC	Reads back PC+1 to RPT.
Interrupt	Pushes PC to STK automatically.
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.
LDPC	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.
MMODE	PLAY instruction dedicated to play mute.
RBVP	Reads back VPR0~3 to RPT.
RBRO	Use RPT as address to read ROM data.
XMD	Use RPT[7:0] 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 RBVP instruction, because RBVP 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
XORM m	$M[m] \leftarrow M[m] \oplus A$	Z
ANDM m	$M[m] \leftarrow M[m] \& A$	Z
ORM m	$M[m] \leftarrow M[m] \mid A$	Z
XORL L	$A \leftarrow A \oplus L$	Z
ANDL L	$A \leftarrow A \& L$	Z
ORL L	$A \leftarrow A \mid L$	Z

2.5.1.2 Arithmetic Instruction

Instruction	Function	Flag Influenced
INCM m	$M[m] \leftarrow M[m] + 1$	C, Z
DECM m	$M[m] \leftarrow M[m] - 1$	C, Z
ADDM m	$M[m] \leftarrow A + M[m] + C$	C, Z
ADDL L	$A \leftarrow A + L + C$	C, Z
INCA	$A \leftarrow A + 1$	C, Z
DECA	$A \leftarrow A - 1$	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]	Z
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 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
CPAM	Skip if A = M[m]	
CPMZ	Skip if M[m] = 0	
CPAL	Skip if A = L	
CPCZ	Skip if C = 0	
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

T[t] : 4-bit System register data at register address t

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.

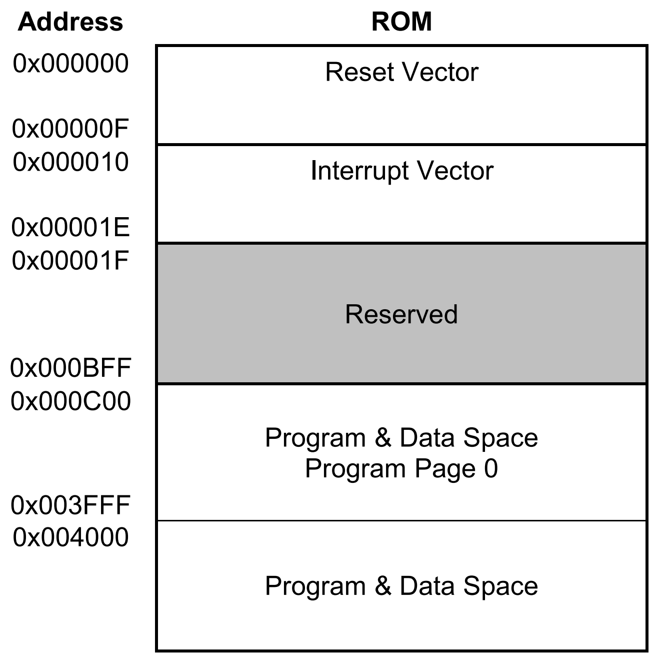
Besides RSTC and SETC commands directly assign the value of the carry flag, C is influenced by the arithmetic result. C means carry and also means the complement of borrow. If the addition operation is larger than 0xF, C=1, and C=0 if the result ≦15. If the subtraction operation is smaller than 0, C=0,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

2.6 Memory Organization

There are maximum 1.5M words ROM, 224 nibbles of RAM and 19 nibbles of dedicated system control register. The registers are divided into 11 nibbles of system registers and 8 nibbles of memory registers. Besides, there are several registers without address allocation, and they can only be accessed by the special instructions. One of the registers is RAM page register (PG), and the others are audio control 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 14-bit length address of ROM, the program page size is 16K words.



2.6.2 RAM

Each page of RAM contains 56 nibbles. NY5A, NY5B and NY5C provide 4 pages. The page number (PG) register of RAM defined by the MPG instruction, and its initial value is 0. Because the memory space is shared with the memory registers (address=0x00~0x07), the address for RAM is 0x08~0x3F.

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 RPT1 and RPT0 first, and users can read from or write in the XMD memory register to realize the indexed ROM access.

2.6.3 Memory Register

8 nibbles of memory register share the address `m’ 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]
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. MVTA reads the register value of the address operand `t’ to the ACC. MVAT writes the ACC value to the register of the address `t’.

Address	Name	Description
0	VOL	Volume control register
1	PFLG	Play flag register
2	INT	Interrupt control register
3	MIX	Mixer control register
4	AUD	Audio output control register
8	PA	PA I/O port control register
9	PB	PB I/O port control register
A	PC	PC I/O port control register
B	PD	PD I/O port control register
C	PE	PE I/O port control register

2.6.5 Register without Memory Allocation

Name	Description	Instruction
PG	2-bit RAM page register	MPG
CH#	3-bit channel number register of working channel	CHNO
MD	3-bit channel mode register of CH#	CHMD
TCS	3-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 20 I/O ports, designated as PAx through PEx, and x=0~3. All the I/O ports can be configured as input, output, or IO port (bi-direction). For the input port, we provide an internal pull-high register option for convenience. For the output port, users can also option its initial value as low or high according to your application circuit. Besides, users can also enable the large current option for each output port to get a larger sink current. The bi-direction IO port can be an input or output by its register value, and users can option the bi-direction IO with a pull-high resister or without a pull-high resister (Open-Drain). When the register equals 0, it is an output and can only output zero. When the register equals 1, it is a weak pull-high or floating (Open-Drain) so that it also can be considered as an input port with/without a pull-high resistor. Users also can enable the large sink current option of an IO port.

The PX0 port means the PA0, PB0, PC0, PD0 or PE0 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.

The pull-high resister of all the I/O ports has two kinds of option: weak and strong. The weak one is about 850KΩ @3V for normal application and the strong one is about 480KΩ @3V usually for key matrix function. When users configure the weak or strong pull-high resister, the pull-high resisters of all I/O ports are set as the option value.

Category	Option	Description
PXx (X=A~F, x=1~3)	Bi-direction with pull-high	choose 1 of 6
	Bi-direction without pull-high (Open-Drain)	
	Floating input	
	Pull-high input	
	Initial low output	
	Initial high output	
	Large sink current	Disable
		Enable

Category	Option	Description
PX0 (X=A~F)	Bi-direction with pull-high	choose 1 of 10
	Bi-direction without pull-high (Open-Drain)	
	Floating input	
	Pull-high input	
	Initial low output	
	Initial high output	
	Floating reset	
	Pull-high reset	
	Initial low IR	
	Initial high IR	
	Large sink current	Disable
		Enable
All I/O	I/O port pull-high resister	Weak
		Strong

*** NY5B112C, NY5B132C, NY5B158C, NY5B185C : no IR output function.**

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, or PE0) 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	31.25~58.82 KHz
	IR low/high carrier	Low
		High

*** NY5B112C, NY5B132C, NY5B158C, NY5B185C : no IR output function.**

2.9 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). 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.

2.10 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. The mixer contains a mixer control register MIX. 1-ch ~ 4-ch voice and/or MIDI are all configurable by programming the MIX Two audio output stages: DAC and PWM are supported.

2.10.1 Speech

A voice channel includes a PFLG, a VPR, a voice decoder, a QIO control code generator and an 8-bit sample rate timer (TM) whose timer clock source (TCS) is fixed to 1MHz. It supports PCM and encoded ADPCM speech data.

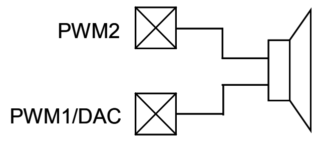
2.10.2 MIDI

A MIDI channel includes a PFLG, a VPR, a TM, an ENV, a timbre skipper and a multiplier, which multiplies the MIDI data and the envelope value held by the ENV. The timbre skipper is used to fulfill the higher octave pitch playing. The hardware multiplier is dedicated to the MIDI channel, and users can’t operate it by any instruction.

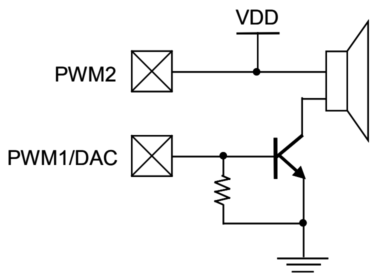
2.10.3 Audio Output

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.

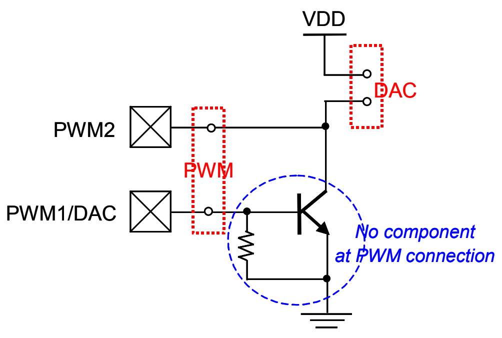
PWM2 Pad	Audio Output Initialization
Speaker (Floating)	PWM
VDD	DAC



PWM Output Connection



DAC Output Connection



PWM/DAC Connection Together

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

2.10.5 Volume Control

Both PWM and DAC supports 16 steps hardware volume control by the VOL register, 0x0~0xF. NY5P(J) does not support Volume Control in DAC mode

Chapter 3. System Control

3.1 Introduction

The VOL, PFLG, MIX, AUD, CH#, MD0~3, TCS0~3, TM0~3 and ENV0~3 are audio control related registers. The PA~F 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
0	VOL	R/W	[3:0]		16 steps volume control level	0xF	U
1	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		
2	INT	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]	0000	Interrupt ~= 0.128 ms	0.128ms	U
				0001	Interrupt ~= 0.256 ms		
				0010	Interrupt ~= 0.512 ms		
				0011	Interrupt ~= 1.024 ms		
				0100	Interrupt = TOF		
				0101	Interrupt = QIOF		
				0110	Interrupt OFF	OFF	U
				0111	Interrupt ON		
				1000	TOF = Timer 0 overflow	Timer 0	U
				1001	TOF = Timer 1 overflow		
				1010	TOF = Timer 2 overflow		
				1011	TOF = Timer 3 overflow		
				1100	Clear TOF	Clear	U
				1101	Clear QIOF	Clear	U
				1110	Clear WDT	Clear	U
				1111	Reserved		
3	MIX	R	[3:0]	0/1	MSB 4-bit audio amplitude data	0x0	U
		W	[0]	0	CH01 = ½CH0 + ½CH1	0x0	U
				1	CH01 = CH0		

Addr	Name	R/W	Bit	Data	Description	Initial	Wake-up
			[1]	0	CH23 = ½CH2 + ½CH3	0x0	U
				1	CH23 = CH2		
			[3:2]	0X	CHDT = ½CH01 + ½CH23	0x0	U
				10	CHDT = CH01		
				11	CHDT = CH23		
4	AUD	R	[0]	0/1	TOF (Timer Overflow Flag)	0x0	U
			[1]	0/1	QIOF (QIO Flag)	0x0	U
			[2]	0/1	Now Audio Output is DAC / PWM	(~AUD2)	U
			[3]		Reserved		
		W	[3:0]	X000	Audio output = DAC	(AUD2)	U
				X001	Reserved		
				X010	Audio output = PWM		
				X011	Reserved		
				X1X0	Audio Output OFF	OFF	U
				X1X1	Audio Output ON		
5		R/W	[3:0]		Reserved		
6		R/W	[3:0]		Reserved		
7		R/W	[3:0]		Reserved		
8	PA	R	[3:0]		Read port A input pad data	X	X
		W	[3:0]		Write to port A output register	0xF	U
9	PB	R	[3:0]		Read port B input pad data	X	X
		W	[3:0]		Write to port B output register	0xF	U
A	PC	R	[3:0]		Read port C input pad data	X	X
		W	[3:0]		Write to port C output register	0xF	U
B	PD	R	[3:0]		Read port D input pad data	X	X
		W	[3:0]		Write to port D output register	0xF	U
C	PE	R	[3:0]		Read port E input pad data	X	X
		W	[3:0]		Write to port E output register	0xF	U
E		R/W	[3:0]		Reserved		
F		R/W	[3:0]		Reserved		

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
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

Addr	Name	R/W	Bit	Description	Initial	Wake-up
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
PG	W	2	RAM page	0x0	U
CH#	W	2	Active channel number	0x0	U
MD0	W	3	Channel mode of channel 0	0x1	U
MD1	W	3	Channel mode of channel 1	0x1	U
MD2	W	3	Channel mode of channel 2	0x1	U
MD3	W	3	Channel mode of channel 3	0x1	U
TCS0	W	3	Timer clock source selection of TM0	0x6	U
TCS1	W	3	Timer clock source selection of TM1	0x6	U
TCS2	W	3	Timer clock source selection of TM2	0x6	U
TCS3	W	3	Timer clock source selection of TM3	0x6	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 NY5A and NY5B is 18-bit long, and the NY5C's RPT is 20-bit except the NY5C640's and NY5C520's are 21-bit. The redundant bits of RPT (PPT[21] of NY5C640 and NY5C520, RPT[21:20] of other NY5C parts, and RPT[21:18] of NY5A and NY5B) 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, the ENV and the XMD only access bit [7:0] of the RPT, others access all available bits. The RPT will be frequently accessed because of its multi-functionality, so the NY5 series 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 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 PG

The PG register is 2-bit register in NY5A, NY5B and NY5C. The PG is not a system register or a memory register, so it can only be written by the MPG instruction and can’t be read. The 2-bit operand means the page users want to write to the PG register. The PG has a hardware stack for interrupt. When an interrupt occurs, the PG is pushed to the stack automatically, and popped from the stack as an IRET executed.

3.4.2 XMD

As mentioned in section 2.6.2, users access XMD taking the RPT[7:6] as the RAM page and the RPT[5:0] as the address. Users have to watch out that the NY5 series does not support using XMD to access memory registers, so the RPT[7:0] can’t be 0x0~0x7 when accessing XMD.

3.5 I/O Ports Register

Each I/O port has its corresponding register. Reading from the register reveals the pad status, and writing to it means writing to the I/O register. If 4 ports of the same group (ex: PE0~3) are redundant, reading them acquires unknown, or reading the redundant bit acquires 1.

The 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 or slow mode, users have to read the pad status and save back to the I/O register before entering the halt or slow mode.

The register value of an output port simply means the output data. If the port 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 outputs 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. Users have to note that reading from an output port also getting the pad potential level, not the register value.

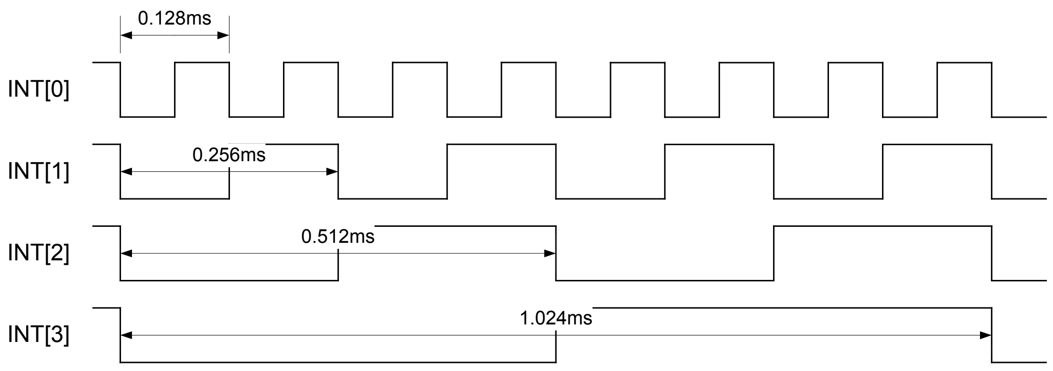
The register value of a bi-directional IO port controls the direction. If the I/O register is 0, the port is an output and outputs 0. If it is 1, the port outputs 1 by the pull-high resister so it simultaneously can be an input port. By the way, since the IO port can be an input only when the register value is 1, users can wake the IC up only by pulling an IO port to 0.

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 INT 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 2.9, the only one interrupt has 6 interrupt sources including 4 different BT intervals, the TOF and the QIOF. Writing 0x0 to 0x5 to INT selects the interrupt source. Writing 0x6 to INT turns off the interrupt generator, and writing 0x7 to INT turns it on. 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 INT 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 0xC to the INT clears the TOF.

Writing 0xD to the INT clears the QIOF.

Writing 0xE to the INT clears the WDT.

3.7 Audio Control Register

3.7.1 VOL

The VOL register dominates the hardware volume control of PWM and DAC. The VOL has 16 steps. 0x0 means the smallest volume and 0xF is the loudest level. Remember the VOL has to be 0xF before ramping up or down of DAC. (NY5P(J) does not support Volume Control in DAC mode)

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.

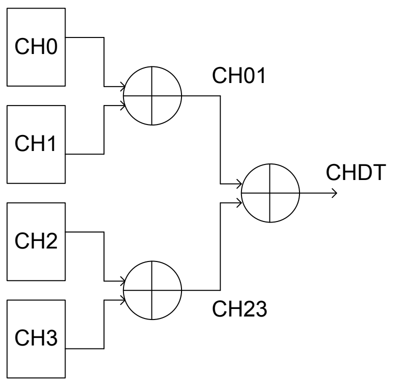
3.7.3 MIX

The reading source and the writing destination of the system register of address 0x3 are different.

3.7.3.1 Audio Data MSB

Reading the 4-bit data of MIX 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.

3.7.3.2 Mixer Control Register



Mixer Structure

Writing to the MIX register selects the mixing channel of the mixer. For example:

MIX=0x5 $\Rightarrow$

$CHDT = ½CH01 + ½CH23 = ½CH0 + ½(½CH2 + ½CH3) = ½CH0 + ¼CH2 + ¼CH3$

MIX=0xF $\Rightarrow$

CHDT = CH23 = CH2

3.7.4 AUD

The reading source and the writing destination of the system register of address 0x4 are different.

3.7.4.1 Audio Related Flag

Reading from bit [0] 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 [1] 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 [2] 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 selects DAC, and writing 0x2 to the AUD selects PWM. The initial value shows (AUD2) also means the pad detecting mechanism.

Writing 0x4 to the AUD turns the selected audio output off, and writing 0x5 to the AUD 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 CHMD, CHTCS, LDTM, RBVP, LDEN, RBEN, STOP, STPL, PLAY and MMODE instruction operation. So the access of the MDx, TCSx, TMx and ENVx register has relations with the CH#.

CHNO loads the ACC to the CH#, and the CH# can’t be read.

Remember to set the CH# before doing anything about the audio control.

3.7.6 MDx

The MDx registers indicate the speech or MIDI mode of the `x’ channel. The sample number of the timbre showed below affects the MIDI octave, we will describe in section 4.4.2. CHMD loads the ACC to the MD of the channel which the CH# indicates, and the MDx can’t be read.

As NY5A/5B/5C are 4-channel speech/melody synthesizer, it can play 4 channels speech or 4 channel MIDI melody at the same time. Other kinds of combination, for example 1 channel speech and 3 channels MIDI melody, are also allowable. While NY5 is used to play speech, user should program MDx with value 0x1.

While NY5 is used to play MIDI melody, it can support three modes to play Midi which are Head, ADSR and Tail. While Head mode is used, user need to provide sampled waveform file (*.wav) and Envelop information will be ignored in Head mode. Moreover, MDx has to be programmed as 0x1 to play this sampled waveform for Head mode.

While ADSR mode is used, user needs to provide sampled waveform file (*.wav) for AD portion and 8/16/32/64/128/256 points sampled waveform file for SR portion. While sampled waveform file for AD portion is played, MDx has to be programmed as 0x1. But NY5 will not apply Envelop information to AD portion. While 8/16/32/64/128/256 points sampled waveform file for SR portion is played, MDx has to be programmed as 0x2, 0x3, 0x4, 0x5, 0x6 or 0x7 which is described as below table. Moreover, NY5 will multiply Envelop information with this waveform for SR portion.

While Tail mode is used, user needs to provide 8/16/32/64/128/256 points sampled waveform file and NY5 will multiply Envelop information with this waveform. As consequence, user has to program MDx according to below table to have correct playback for Tail mode.

MDx	Mode
0x0	Not allowed
0x1	Speech
0x2	8 points Tail
0x3	16 points Tail
0x4	32 points Tail
0x5	64 points Tail
0x6	128 points Tail
0x7	256 points Tail

While NY5 play any noted with any one of these three modes, user has to program TCSx and TMx correctly in order to provide correct pitch during playback. In Section 3.7.7 and 3.7.8, how to setup TCSx and TMx are described. As to how to apply Envelop information by using ENVx register is described in Section 3.7.9.

3.7.7 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.

Considering ADSR mode, because AD portion is played with 1M Hz clock rate, it is recommended to program TCSx as 0x6 for SR portion in order to get better result.

TCSx	MDx=0x1	MDx=0x2, 0x3, 0x4, 0x5, 0x6, 0x7
0x0	1M	4M
0x1		8M
0x2		X
0x3		X
0x4		X
0x5		X
0x6		1M
0x7		2M

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

3.7.8.1 TM Table of Speech

TMx	S.R. (KHz)
0xF9	4

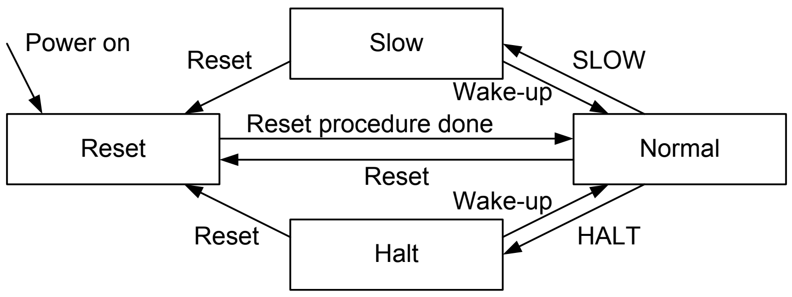
0xA6	6
0x7C	8
0x63	10
0x52	12
0x46	14
0x3E	16
0x31	20

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 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.

3.8.2 Slow Mode

The system enters the slow mode if the SLOW command executed. The system clock in the slow mode slows down about 16 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 PIOF 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.

Chapter 4. Audio Control

4.1 Introduction

The NY5A, NY5B and NY5C IC 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 Control

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 (0x200). Setting the TM to 0 pauses the speech, but keeps the speech data unchanged.

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

Step	Process	Instruction
1	Set CH#	CHNO
2	Set MD if the channel supports multi-mode	CHMD
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 10-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. Remember to align the starting address of the ADPCM data to 0x7 (the last 3 bits of the address have to be 1), and the PCM data has no issue about this.

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.

4.2.4 Mute

The NY5 series supports another special mute mode for speech. When a speech like the vocal or talk has a lot of suspension or silence, using the mute mode saves much ROM space. Turn on the mute mode option of the encode software to save your cost. Similar to the QIO, playing a mute mode encoded data, and the QIOF arises as a mute code occurs. Users have to execute the MMODE instruction according to the mute table. MMODE is a dedicated instruction for mute playing.

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 or STPL. 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 (0x200) immediately. STPL pushes the PFLG to 0 and stops the playing MIDI after end of waveform or sample point are met.

RBVP reads the playing address of the MIDI data.

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

4.3.2 MIDI Octave

If user adopts Head and ADSR modes to synthesize MIDI, below formula will apply to derive correct TM value in order to play any note of any Octave based on the same waveform:

$TM = ((F_{TCS} / F_{SR})*(\text{note pitch frequency} / \text{Root pitch frequency}) – 1$

TM : TM value in decimal, only 0~255 is allowed.

F_{TCS} : Frequency of the timer clock source.

F_{TCS} is 1,000, 000 while MDx is 0x1.

F_{SR} : Frequency of the sample rate

If Tail mode is used to synthesize MIDI, user can program TCSx and MDx in order to derive correct note frequency. Below tables is an example of how to determine MDx and TCSx for note C of each Octave and TM value for the twelve notes of each Octave based note C.

Octave	MDx	TCSx	Freq. of C
1	256 samples	2M	32.7 Hz
2	256 samples	4M	65.4 Hz
3	256 samples	8M	130.8 Hz
4	128 samples	8M	261.6 Hz
5	64 samples	8M	523.3 Hz
6	32 samples	8M	1046.5 Hz
7	16 samples	8M	2093 Hz
8	8 samples	8M	4186 Hz

The note pitch is controlled by the TM and recommended 8-bit TM value is described in below table.

TM	Note in the same Octave
0xEE	C
0xE0	C#
0xD4	D
0xC8	D#
0xBD	E
0xB2	F
0xA8	F#
0x9E	G
0x95	G#
0x8D	A
0x85	A#
0x7E	B

Chapter 5. Instruction Set

5.1 Instruction Classified Table

Item	Inst.	Op1	Op2	Operation	Inst. Length	Exec. Cycle	Oper. Flag	Flag Affected
Arithmetic Instructions								
1	INCM	6m		M = M +1	1	1		C, Z
2	DECM	6m		M = M - 1	1	1		C, Z
3	ADDM	6m		M = A + M + C	1	1	C	C, Z
4	XORM	6m		M = A ^ M	1	1		Z
5	ANDM	6m		M = A & M	1	1		Z
6	ORM	6m		M = A | M	1	1		Z
7	MVAM	6m		Move A to M	1	1		
8	MVMA	6m		Move M to A	1	1		Z
9	MVRM	4m	2r	Move R to M	1	1		
10	MVMR	4m	2r	Move M to R	1	1		
11	MVLR	4L	2r	Move L to R	1	1		
12	ADDL	4L		A = A + L + C	1	1	C	C, Z
13	XORL	4L		A = A ^ L	1	1		Z
14	ANDL	4L		A = A & L	1	1		Z
15	ORL	4L		A = A | L	1	1		Z
16	MVLA	4L		Move L to A	1	1		
17	MVAT	4t		Move A to T	1	1		
18	MVTA	4t		Move T to A	1	1		Z
19	RSTC			Reset C = 0	1	1		C
20	SETC			Set C = 1	1	1		C
21	INCA			A = A + 1	1	1		C, Z
22	DECA			A = A - 1	1	1		C, Z
Conditional Instructions								
23	CPAM	6m		Skip if A = M	1	1		
24	CPMZ	6m		Skip M = 0x0	1	1		
25	CPAL	4L		Skip if A = L	1	1		
26	CPCZ			Skip if C = 0	1	1	C	
27	CPZZ			Skip if Z = 0	1	1	Z	
Audio Instructions								
28	CHNO			Move A[1:0] to CH#	1	1		
29	CHMD			Move A[2:0] to MD[CH#]	1	1		
30	CHTCS			Move A[2:0] to TCS[CH#]	1	1		
31	LDTM			Move RPT[7:0] to TM[CH#]	1	1		
32	RBVP			Move VPR[CH#] to RPT	2	3		
33	LDEN			Move RPT[7:0] to ENV[CH#]	1	1		
34	RBEN			Move ENV[CH#] to RPT[7:0]	1	1		

Item	Inst.	Op1	Op2	Operation	Inst. Length	Exec. Cycle	Oper. Flag	Flag Affected
35	STOP			STOP playing CH# immediately	1	1		
36	STPL			STOP playing CH# after timbre loop	1	1		
37	PLAY			Move RPT to VPR[CH#]	1	1/3		
38	MMODE			Move RPT to VPR[CH#] in playing mute	1	3		
Other Instructions								
39	MPG	2p		Set RAM page	1	1		
40	RBRO	1n		Move ROM[RPT] data to ROD and ACC	1	3		
41	JMP	14a		Jump to Address	2	2		
42	CALL	14a		Jump to Address, and Move PC+2 to RPT	2	2		
43	IRET			Move STK to PC	2	2		
44	LDPC			Move RPT to PC	1	2		
45	RBPC			Move PC+1 to RPT	1	2		
46	HALT			Enter HALT mode	1	1		
47	SLOW			Enter SLOW mode	1	1		
48	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
T : 4-bit System register data
Z : 1-bit zero flag data
CH# : 2-bit channel number register
MD : 3-bit channel mode register of channel CH#
TCS : 3-bit channel clock source selection register of channel 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

a : ROM address
m : RAM or memory register address
n : data address Plus/Not plus 1
p : RAM page
r : Memory register address
t : System register address

5.2 Instruction Descriptions

5.2.1 Arithmetic Instructions

INCM m

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

Operation: M ← M + 1

Operand: 0x0 ≦ 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

ADDM m

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

Operation: M ← 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=0x7, M0=0x1, C=1, Z=0

DECM m

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

Operation: M ← M - 1

Operand: 0x0 ≦ 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

XORM m

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

Operation: M ← 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=0x3, M0=0x8, Z=0

ANDM m

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

Operation: M ← 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=0x7, M0=0x2, Z=0

ORM m

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

Operation: M ← 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=0x3, M0=0xB, 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

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

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 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

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

MVAT t

Function: Move A to T of address t.

Operation: T ← A

Operand: 0x0 ≤ t ≤ 0xF

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example:MVAT 0x0

Before Instruction

A=0x8

After Instruction

VOL=0x8

MVTA t

Function: Move T of address t to A.

Operation: A ← T

Operand: 0x0 ≤ t ≤ 0xF

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: MVTA 0x3

Before Instruction

MSB 4-bit audio amplitude data = 0x8

After Instruction

A=0x8

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

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

CPAM m

Function: Skip the next instruction if A 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: CPAM m0

Inst1

Inst2

After Instruction

If A ≠ M0, `Inst1' is executed.

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

CPMZ m

Function: Skip the next instruction if M of address m equals zero.

Operation: Skip next if M=0x0

Operand: 0x0 ≤ m ≤ 0x3F

Words: 1

Cycles: 1, (2, 3)

Operative Flags: None

Flags Affected: None

Example: CPMZ m0

Inst1

Inst2

After Instruction

If M0 ≠ 0x0, `Inst1' is executed.

If M0=0x0, `Inst1' is discarded, and `Inst2' is executed.

<u>CPAL L</u>

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

<u>CPCZ</u>

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

<u>CPZZ</u>

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

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

CHMD

Function: Move A[2:0] to MD of CH#.

Operation: MD ← A[2:0]

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CHMD

Before Instruction

CH#=0, A=0xE

After Instruction

MD0=0x6

CHTCS

Function: Move A[2:0] to TCS of CH#.

Operation: TCS ← A[2:0]

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CHTCS

Before Instruction

CH#=2, A=0x5

After Instruction

TCS2=0x5

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

RBVP

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: RBVP

Before Instruction

CH#=0, VPR0=0xABCDE, MD0≠0x0

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, MD1≠0x1

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, MD2≠0x1

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

STPL

Function: Stop the MIDI playing of CH# after the playing timbre loop.

Operation: Stop playing after this timbre
PFLG[CH#] ← 0

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: STPL

Before Instruction

CH=0x1, MD1=0x3

After Instruction

CH1 will be stopped and PFLG[1]=0 after this playing timbre finished.

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

MPG p

Function: Change the RAM page to page p.

Operation: RAM PAGE ← p

Operand: p ∈ [0,1,2,3]

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MPG 3
MVAM 0x12

Before Instruction

A=0xC

After Instruction

Md2=0xC

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: $0 \leq n \leq 1$

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

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[21:14] will not be changed

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[21:14] 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.

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

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.

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.