九齊科技股份有限公司

Nyquest Technology Co., Ltd.

NY6 Series

Single-Chip 4-bit MCU with 8~24 I/O,
6-ch Speech/MIDI Synthesizer

USER MANUEL

Version 1.3

Mar. 28, 2019

Revision History

Version	*Date*	*Description*	*Modified Page*
1.0	2016/08/31	Formal release.	-
1.1	2016/11/25	1. Modify pad description. 2. Update DC characteristics. 3. Fix typos.	10 11 -
1.2	2017/05/17	1. Add description that a 0.1uF power capacitor nearby PB_VDD is necessary if LDO regulator is enabled.	26
1.3	2019/03/28	Remove NY6C450A / NY6C520A / NY6C640A / NY6C720A.	-

Table of Contents

Chapter 1. Introduction

1.1 General Description

The NY6 series IC is a powerful 4-bit micro-controller based sound processor. There are 6 channels that are configured as speech or MIDI, and all of these 6 channels or part of them can be played with speech or MIDI simultaneously. By using the high fidelity 4-bit, 5-bit mixed ADPCM or PCM speech/ MIDI timbre synthesis algorithm with up to 44.1KHz sample rate, NY6 produces high quality voices. As NY6 is specially designed for MIDI synthesis application, it provides Attack-Decay-Sustain-Release method (ADSR) with 256-level envelope for Patch (instrument) synthesis. NY6 can precisely synthesize any tone frequency of MIDI with +/- 0.5% accurate internal oscillation and automatic Tone-Calibration. Therefore NY6 melody quality is very close to real instrument.

Moreover, NY6 is equipped with new Nyquest's developed high-quality noise filtering algorithm of 250KHz over-sampling, which can remove noise in order to improve speech and melody quality greatly. Up to 16-level digital volume can be applied to final synthetic speech or melody that is tailored for applications of volume adjustment. NY6 provides two kinds of audio outputs with fine resolution, one is 12-bit current-type D/A converter (DAC) and the other is 12-bit Pulse-Width-Modulation (PWM). Therefore NY6 speech/ melody quality is the best choice among all solutions.

Hardware SPI (Mode0/Mode3) is supported for Channel-0 voice data automatically (auto mode, 24-bit addressing capability) and user data (user mode) fetch from external SPI memory. Voltage comparator is built-in for analog signal detecting applications. Software low voltage reset counting mechanism is provided for low power management.

The RISC MCU architecture is very easy to program and control, various applications can be easily implemented. There are 75 instructions, and most of them are executed in single cycle. Besides normal operation mode, NY6 also provides Halt mode (or Sleep mode) and 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 75 instructions.
- NY6A have 10 items. 160K x 10-bit ROM is the maximum size.
- NY6B have 11 items. 208K x 10-bit ROM is the maximum size.
- NY6C have 12 items. 1728K x 10-bit ROM is the maximum size.
- 336x4-bit RAM, divided into 6 pages.
- 2MHz system clock for instruction execution.

- Slow mode to operate with low power consumption (+/-3.0% accuracy).
- 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=1.9V) and watch-dog reset (WDT) are supported to protect the system.
- Special hardware for LVR occurrence counting by program to manage low battery system operation.
- One interrupt entrance for multiple interrupt sources with an independent stack.
- 8-bit timer counter is applied to multiple clock source for various application.
- Low Voltage Detector (LVD) is built-in for monitoring the status of power and protect malfunction if unstable power is given. *(NY6A doesn't support LVD function)*
- LDO regulator is supported for the power supply of SPI flash. *(NY6A doesn't support LDO function)*
- Hardware SPI for external SPI devices data access. Dual power system operation supported (ex: NY6 @ 5V, SPI @ 3V).
- Up to 24 flexible Bi-direction I/Os. Direction of each I/O is independently controlled by individual register bit.
- Each Bi-direction I/O pin can be optioned as different input and output function. For the input option, users can select one of three kinds of option: input with pull-high resistor, input without pull-high resistor, or input with register-controlled pull-high resistor (high-to-low wakeup only). For the output option, users can select one of three kinds of option: output with normal drive/sink sink current, large sink current or constant sink current. *(Mask option)*
- Shared pins to provide IR carrier, comparator, SPI interface and external reset feature. *(Mask option)*

Shared Pin Function	NY6A	NY6B/NY6C
External reset (Reset)	PA3/Reset	PA3/Reset
IR carrier (IR)	PA2/IR	PA2/IR
Comparator	*N/A*	PA1/VIN
	N/A	PA0/VIP
SPI	*N/A*	PB0/CSb
	N/A	PB1/SCK
	N/A	PB2/SDO *(MOSI)*
	N/A	PB3/SDI *(MISO)*

(NY6A doesn't support the Comparator and SPI function)

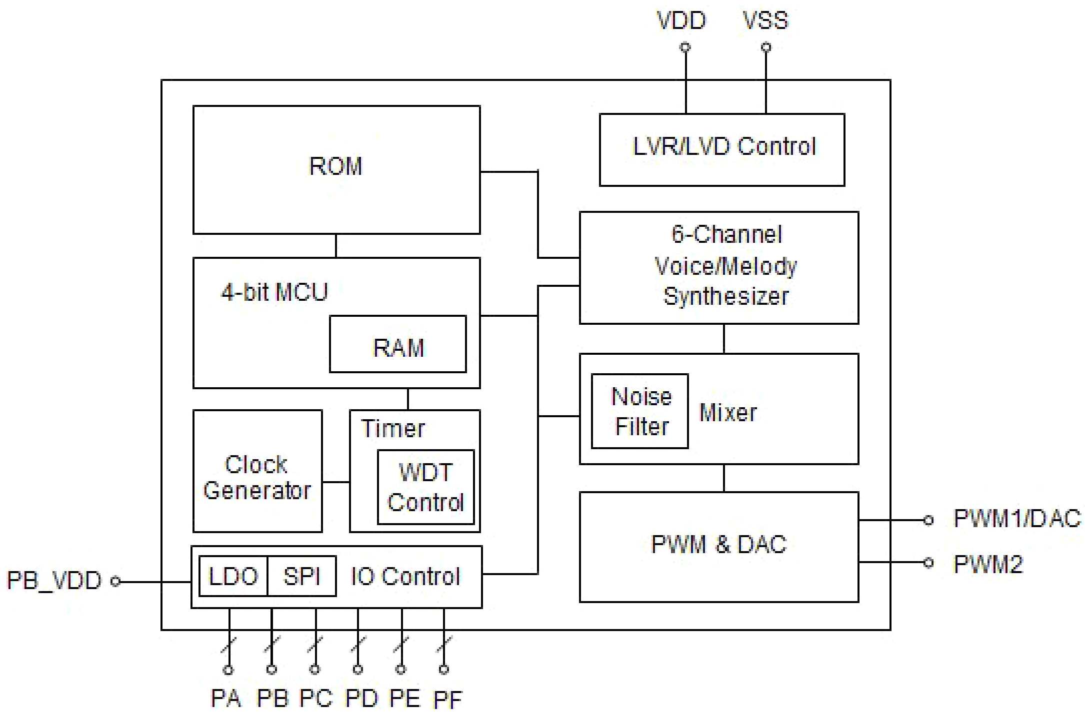
- Selection of IR carrier frequency and data high/low IR output is supported.

- Built-in voltage comparator for analog signal detection applications. Comparator output flag can be configured to generate interrupt and wakeup. Also, the flag can be used for timer counter clock source for specific application. *(NY6A doesn't support Comparator function)*
- Maximum of 6 channels can be played simultaneously, and each channel can be arbitrarily assigned as speech or MIDI channel.
- New high fidelity 4-bit / 5-bit mixed ADPCM or 10-bit PCM speech synthesis algorithm and ADSR with 256-step envelope for MIDI synthesis.
- Patented noise filtering algorithm with 250KHz over sampling to enhance signal-to-noise ratio and provide excellent sound quality without ROM size increase.
- 16-level digital volume control for synthetic speech/melody.
- Built-in hardware automatic Tone-Calibration of near-zero frequency deviation for precise tone frequency.
- High quality 12-bit D/A converter or 12-bit PWM driver. *(NY6A doesn't support DAC output)*
- PWM driver can be normal PWM or Ultra PWM.

1.3 Product List

P/N	Voice Duration @6KHz (sec)	ROM Size (bit)	Program ROM Size (bit)	I/O	PWM	DAC
NY6A003A	3.3	12K x 10	12K x 10	8	12-bit	-
NY6A005A	5	16K x 10	16K x 10	8	12-bit	-
NY6A008A	8.3	24K x 10	24K x 10	8	12-bit	-
NY6A011A	11.7	32K x 10	32K x 10	8	12-bit	-
NY6A018A	18.3	48K x 10	48K x 10	8	12-bit	-
NY6A025A	25	64K x 10	64K x 10	8	12-bit	-
NY6A035A	35	88K x 10	64K x 10	8	12-bit	-
NY6A045A	45	112K x 10	64K x 10	8	12-bit	-
NY6A055A	55	136K x 10	64K x 10	8	12-bit	-
NY6A065A	65	160K x 10	64K x 10	8	12-bit	-
NY6B005A	5	16K x 10	16K x 10	16	12-bit	12-bit
NY6B008A	8.3	24K x 10	24K x 10	16	12-bit	12-bit
NY6B011A	11.7	32K x 10	32K x 10	16	12-bit	12-bit
NY6B018A	18.3	48K x 10	48K x 10	16	12-bit	12-bit
NY6B025A	25	64K x 10	64K x 10	16	12-bit	12-bit
NY6B035A	35	88K x 10	64K x 10	16	12-bit	12-bit
NY6B045A	45	112K x 10	64K x 10	16	12-bit	12-bit
NY6B055A	55	136K x 10	64K x 10	16	12-bit	12-bit
NY6B065A	65	160K x 10	64K x 10	16	12-bit	12-bit
NY6B075A	75	184K x 10	64K x 10	16	12-bit	12-bit
NY6B085A	85	208K x 10	64K x 10	16	12-bit	12-bit
NY6C112A	111.7	272K x 10	64K x 10	24	12-bit	12-bit
NY6C132A	131.7	320K x 10	64K x 10	24	12-bit	12-bit
NY6C158A	158.3	384K x 10	64K x 10	24	12-bit	12-bit
NY6C185A	185	448K x 10	64K x 10	24	12-bit	12-bit
NY6C225A	225	544K x 10	64K x 10	24	12-bit	12-bit
NY6C265A	265	640K x 10	64K x 10	24	12-bit	12-bit
NY6C305A	305	736K x 10	64K x 10	24	12-bit	12-bit
NY6C345A	345	832K x 10	64K x 10	24	12-bit	12-bit

1.4 Block Diagram



1.5 Pad Description

Pin	ATTR.	Description
VDD1~3	Power	Positive power.
GND1~4	Power	Negative power.
PA0/VIP	I/O	Bit 0 for Port A, or positive input of comparator.
PA1/VIN	I/O	Bit 1 for Port A, or negative input of comparator.
PA2/IR	I/O	Bit 2 for Port A, or IR carrier output.
PA3/Reset	I/O	Bit 3 for Port A, or external reset input.
PB_VDD	Power	Power for PBx and external component. *(Not available for NY6A)*
PB0/CSb	I/O	Bit 0 for Port B, or chip select pin for SPI interface.
PB1/SCK	I/O	Bit 1 for Port B, or serial clock pin for SPI interface.
PB2/SDO	I/O	Bit 2 for Port B, or serial data output pin (MOSI) for SPI interface.
PB3/SDI	I/O	Bit 3 for Port B, or serial data input pin (MISO) for SPI interface.
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.
PWM1/DAC	O	PWM1 output or DAC output.
PWM2	O	PWM2 output.

* NY6A: PA0~PB3. *(There is no Comparator, LVD, LDO, SPI, and DAC function.)*

1.6 Electrical Characteristics

The following table lists the electrical characteristics. 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
V_{IN}	Input voltage	VSS–0.3V ~ VDD+0.3	V
T_{OP}	Operating Temperature	0 ~ +70	°C
T_{ST}	Storage Temperature	-25 ~ +85	°C

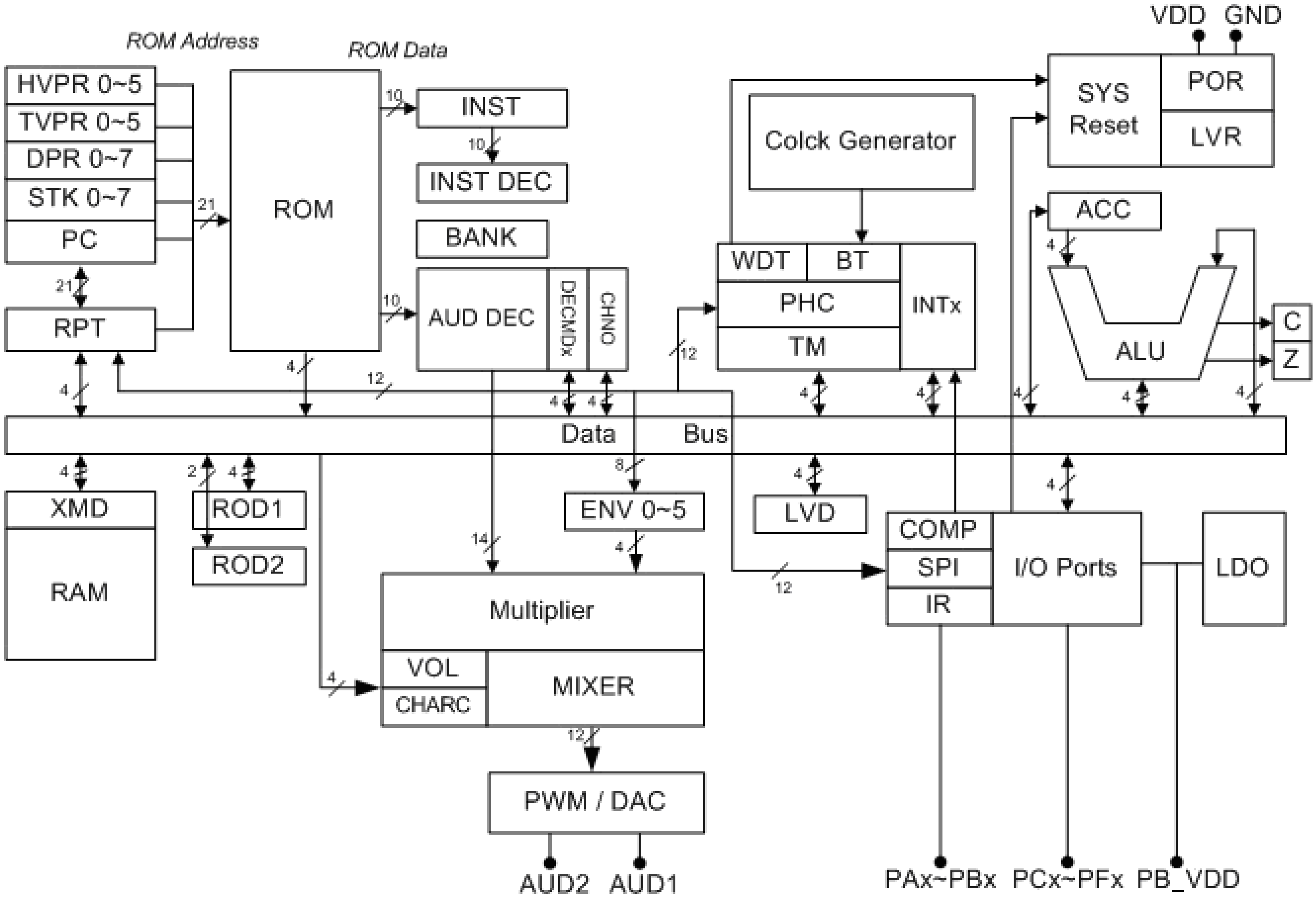
1.6.2 DC Characteristics

Symbol	Parameter		VDD	Min.	Typ.	Max.	Unit	Condition
V_{DD}	Operating voltage		--	2.0	3.0	5.5	V	2MHz.
I_{SB}	Supply current	Halt mode	3.0		0.1	0.5	uA	Sleep, no load.
			4.5		0.1	0.5		
I_{SL}		Slow mode	3.0		50		uA	BT=16.384ms, no load
			4.5		80			
I_{OP}		Normal mode	3.0		1.8		mA	2MHz, no loading
			4.5		5.0			
I_{IL}	Input current (Internal pull-high)	Weak (1.2M ohms)	3.0		2.5		uA	V_{IL}=0V
			4.5		7.4			
		Strong (100k ohms)	3.0		30		uA	
			4.5		75			
I_{OH}	Output high current		3.0		-7		mA	V_{OH}=2.0V
			4.5		-11			V_{OH}=3.5V
I_{OL}	Output low current (Normal current)		3.0		10		mA	V_{OL}=1.0V
			4.5		16			
	Output low current (Large current)		3.0		20		mA	
			4.5		30			
	Output low current (Constant current)		3.0		18		mA	
			4.5		21			
I_{DAC}	DAC output current		3.0		1.4		mA	Half-scale
I_{PWM}	PWM output current (Normal PWM)		3.0		60		mA	Load=8 ohms
			4.5		100			
	PWM output current (Ultra PWM)		3.0		80		mA	
			4.5		125			
ΔF/F	Frequency deviation by voltage drop		3.0		-0.5		%	$\frac{Fosc(3.0v)-Fosc(2.4v)}{Fosc(3v)}$
			4.5		0.5			$\frac{Fosc(4.5v)-Fosc(3.0v)}{Fosc(4.5v)}$
ΔF/F	Frequency lot deviation		3.0	-0.5		0.5	%	$\frac{Fmax(3.0v)-Fmin(3.0v)}{Fmax(3.0v)}$
Fosc	Oscillation Frequency		-	1.90	2	2.05	MHz	VDD=2.0~5.5V

Chapter 2. Hardware Architecture

2.1 Overview

2.1.1 Function Block Diagram



2.1.2 Hardware Summary Table

Name	Function	Address
HVPR0~5	Voice Head pointer according to CHNM.	
TVPR0~5	Voice Tail pointer according to CHNM	
DPR0~7	Data pointer (share with STK7~0)	
STK0~7	8-level interrupt dedicated stack(share with DPR0~7)	
PC	Program counter	
RPT	Multi-function register pointer	M[0x0 ~ 0x5]
XMD	Indexed RAM data access register	T[0xE~0xF]
RAM	336 nibbles RAM (6 pages, each 56 nibbles)	
ROM	Program & data ROM	
ROD1	ROM[7:4] data access register	M[0x6]
ROD2[1:0]	ROM[9:8] data access register	M[0x7]
INST	Instruction register	

Name	Function	Address
INST DEC	Instruction decoder	
BANK	Program Bank Register	
AUD DEC	Audio decoder	
DECMDx	PCM / ADPCM control register	T[0x7 ~ 0x8]
CHNM	Active channel select	
ENV0~5	8-bit Envelope of CHNM	
Multiplier	Hardware multiplier for MIDI	
VOL	Digital volume control register	T[0x9]
CHARC	Mix Channel#, Output choice	T[0x6]
Mixer	Channels audio data mixer	
PWM / DAC	PWM, D/A audio output	
Clock Generator	Ring oscillator clock generator	
WDT	Watch-dog timer and reset generator	
BT	System base timer	
PHC	PH Counter	
TM	Timer Counter	T[0xC ~ 0xD]
INTx	Interrupt generator	T[0x0 ~ 0x1]
LVD	Low Voltage Detector	T[0x0A]
SYS Reset	System reset generator	
POR	Power reset generator	
LVR	Low Voltage Reset	
ACC	4-bit accumulator	
ALU	4-bit arithmetic logic unit	
C	Carry flag for arithmetic	
Z	Zero flag for arithmetic	
COMP	Comparator	
SPI	SPI Control Interface	T[0x10 ~ 0x11]
IR	Infrared transmit block	
LDO	LDO Regulator for PB port (SPI Application)	T[0x10]
I/O Ports	I/O port register	T[0x14 ~ 0x1F]

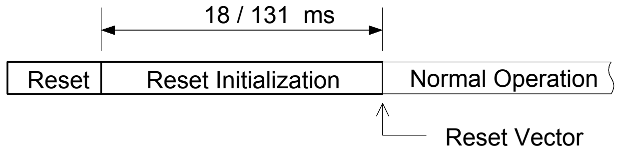
*T[] : System register and the hex number 0x? Between the brackets means its address.

*M[] : Memory 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 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 NY6 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 NY6 IC is approximate 1.9V. RAM (Pages5 $3E, $3F) are optioned to be protected for the record of LVR occurrence times of system low voltage management.

2.3.3 Watch-Dog Timer Reset (WDTR)

To recover from program function, the NY6 IC supports an embedded watch-dog timer reset. The WDTR function always works with the program executing. Users have to clear the WDT periodically to prevent from timing up with a reset generation. Typically, the minimum time-up period of the WDT is about 240ms and users can clear WDT through instruction CWDT0 and CWDT1.

2.3.4 I/O Port External Reset

The PA3/Reset I/O pin of the NY6 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 pull-high resister on PCB with internal pull-high resister option disabled. When the reset pin falls to the ground level, it generates an external reset.

2.4 Address Pointer

The NY6 micro-controller contains a program counter (PC), a multi-function register pointer (RPT), 6 head pointers (HVPR0~5) and 6 tail pointers (TVPR0~5) for channel 0~5 and 8 data pointers (DPR) which are shared with interrupt stack (STK). Particularly, the address of DPR is indexed by CHNM register, but the STK is nested type which starts from index 7 for STK0, ex. DPR0 is same address as STK7. The length of each address pointer is 21-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 and RPT.

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. PC is 18-bit wide for NY6A/NY6B and 21-bit wide for NY6C. 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 instructions which are listed in table below. The interrupt vector is at address 0x000010.

Inst./Event	Function
JMP Addr	Jump to {BANK, Addr}.
CALL Addr	Push the PC+2 to the STK and load {BANK, Addr} to PC.
Interrupt	Push PC+1 to STK automatically.
RET	Pop STK back to PC. Return to the main program from subroutine
IRET	Pop STK back to PC. Return to the main program from the interrupt routine.

Addr : 16-bit immediate address.

2.4.2 Stack (STK)

Eight level hardware push/pop stacks are available which are reacted to CALL or interrupt occurrence. When an interrupt/CALL takes apart, the system pushes PC+1/PC+2 to STK automatically. When the program returns to the main program from subroutine / the interrupt routine by RET / IRET instruction, the system pops the STK back to the PC. Unused STK can be used as DPR. The STK max width is 18 bits for NY6A and 21 bits for NY6B/NY6C.

2.4.3 Multi-function Register Pointer (RPT)

As implied in the name, RPT are multi-function registers. There are at most six RPT that are RPT0, RPT1, RPT2, RPT3, RPT4 and RPT5. RPT0~RPT4 are 4-bit wide and RPT5 is 1-bit wide, i.e. RPT5[0]. The RPT max width is 18 bits for NY6A/6B and 21 bits for NY6C. Users have to operate RPT in coordination with instructions below.

Inst./Event	Function
LDEN	Load RPT[7:0] to ENV, according to CHNM.
RBEN	Load ENV to RPT[7:0], according to CHNM
PLAY	Play RPT to HVPR, according to CHNM.
LDSEC	Load RPT to TVPR, according to CHNM.
LDPH	Load RPT[11:0] to PH, according to CHNM.
RBVPR	Read HVPR to RPT, according to CHNM.
RBNVPR	Read TVPR to RPT, according to CHNM.
LDPR	Load RPT to DPR/STK, according to CHNM.
RBPR	Read DPR/STK to RPT, according to CHNM.
RBSPRH	Read SPR[23:12] to RPT[11:0]
RBSPRL	Read SPR[11:0] to RPT[11:0]
LDSPRH	Load RPT[11:0] to SPR[23:12]
LDSPRL	Load RPT[11:0] to SPR[11:0]
RBDA	Read DAC[11:8] data to RPT2(RPT[11:8])
LDPC	Move RPT to PC
RBPC	Move PC to RPT
XMD0	Use {PAGE, RPT1[1:0], RPT0} as address to access indexed RAM data.
XMD1	Use {PAGE, RPT3[1:0], RPT2} as address to access indexed RAM data.

2.4.4 Head Voice Pointer (HVPR) & Tail Voice Pointer (TVPR)

Because NY6 is a 6-channel sound processor, 6 voice pointers each with 21-bit width are necessary for playing speech or MIDI of each channel. When PLAY is executed, the system loads RPT to HVPR of the channel that assigned by the CHNM register. When LDSEC is executed, the system loads RPT to TVPR of the channel that assigned by the CHNM register. Therefore, users have to move the start address of the speech or MIDI data to RPT first. Moreover, users can read HVPR/TVPR back by RBVPR/RBNVPR instruction, because RBVPR/RBNVPR moves HVPR/TVPR of the channel that assigned by the CHNM register to RPT. The HVPR/TVPR max width is 18 bits for NY6A/6B and 21 bits for NY6C.

2.4.5 Data Pointer (DPR)

Eight data pointers each with 21-bit width are necessary for reading ROM data of each channel. When LDPR is executed, the system loads RPT to DPR of the channel that assigned by the CHNM register. The read back ROM data is placed on ROD2[1:0], ROD1, ACC. ACC is the 4 LSB of ROM data. Besides, users can read DPR back by RBPR instruction, because RBPR moves DPR of the channel that assigned by the CHNM register to RPT. The DPR max width is 18 bits for NY6A and 21 bits for NY6B/NY6C. Unused DPR can be used as STK.

2.5 Arithmetic Logic Unit (ALU)

The NY6 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 status bits (carry and zero) to indicate the result of the operation. One or two operands will be the data sources of the ALU operation. The operands can be ACC, RAM, register, or literal constant data.

2.5.1 ALU Instruction Summary

2.5.1.1 Logic Instruction

Instruction	Function	Flag Influenced
XORM m	A ← M[m] ⊕ A	Z
ANDM m	A ← M[m] & A	Z
XORL L	A ← L ⊕ A	Z
ANDL L	A ← L & A	Z
ORL L	A ← L | A	Z
RRC	Right Rotate A with C	C, Z
RLC	Left Rotate A with C	C, Z
RRA	Right Rotate A	
RLA	Left Rotate A	

M[m] : 4-bit RAM data at memory address m1, 0x00≤ m ≤0x3F.

2.5.1.2 Arithmetic Instruction

Instruction	Function	Flag Influenced
ADDM m	{C, A} ← A + M[m] + C	C, Z
SUBM m	{C, A} ← A – M[m] - (~B)	C, Z
INCM m	{C, M[m]} ← M[m] + 1	C, Z
DECM m	{C, M[m]} ← M[m] - 1	C, Z
ADDL L	A ← A + L + C	C, Z
SUBL L	{C, A} ← A - L - (~B)	C, Z
INCA	A ← A + 1	C, Z
DECA	A ← A - 1	C, Z

M[m] : 4-bit RAM data at memory address m1, 0x00≤ m ≤0x3F.

B : 1-bit borrow flag data, shared with carry flag, B=~C.

2.5.1.3 Data Transfer Instruction

Instruction	Function	Flag Influenced
MVAM m	M[m] ← A	
MVMA m	A ← M[m]	Z
MVAT t1	T[t1] ← A	
MVTA t1	A ← T[t1]	Z
MVLA L	A ← L	
INTCB t2, b	Clear T[t2][b]	
INTSB t2, b	Set T[t2][b]	
SETC	C ← 1	C
CLRC	C ← 0	C

M[m] : 4-bit RAM data at memory address m1, 0x00≤ m ≤0x3F.

T[t1] : 4-bit system register data at address t1, 0x0≤ t1 ≤0x1F

T[t2] : 4-bit system register data at address t2, 0x0≤ t2 ≤0x3

b : bit address, 0x0≤ b ≤0x3

The width of the system register address ‘t1’ of MVAT and MVTA instruction is 5-bit (0x00~0x1F), and address ‘t2’ of INTCB and INTSB instruction is 2-bit, to access system register 0x0~0x3 related to interrupt execution. The width of ‘b’ is 2-bit for bit address of system register to execute clear or set desired bit by INTCB and INTSB. The width of the RAM address `m’ of instructions associated with memory operation is 6-bit. Only 0x00~0x07 registers are independent of SRAM page. Users can use memory-related instructions to handle RAM of address 0x08~0x3F, but the RAM page is still working.

2.5.1.4 Conditional Branch Instruction

Instruction	Function	Flag Influenced
SAGT L	Skip when A > L	
SALT L	Skip when A < L	
SANE L	Skip when A != L	
SCEZ	Skip if C = 0	
SZEZ	Skip if Z = 0	
SCNZ	Skip if C != 0	
SZNZ	Skip if Z !=0	
SBZ b	Skip when A[b] = 0	
SNP	Skip when Play = 0, according to CHNM	
SP	Skip when Play = 1, according to CHNM.	
SANP	Skip when ALL 6 channels Play = 0	
SNHP	Skip when head Play = 0, according to CHNM.	

A conditional branch instruction compares two operands and skips next instruction if expression is true. 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

Z : 1-bit zero flag data

L : 4-bit immediately literal data

A[b] : b-th bit of Accumulator, 0 ≤ b ≤3.

2.5.2 ALU Related Status Flag

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

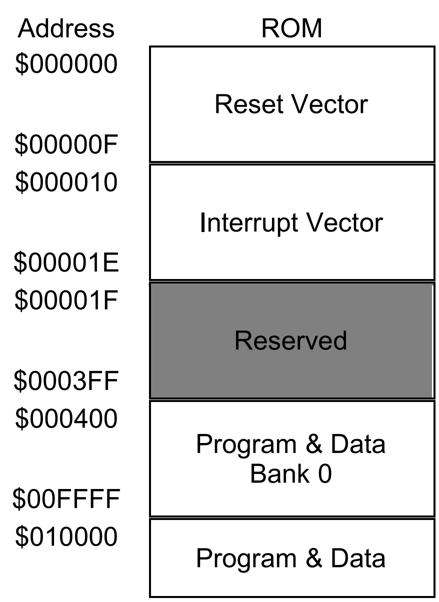
Besides CLRC and SETC commands directly assign the value of the carry flag, C is influenced by the arithmetic result. C means carry and also means the complement of borrow. If the addition operation result is larger than 0xF, C=1, and C=0 if the result is ≤15. If the subtraction operation smaller than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

2.6 Memory Organization

There are maximum 1728K words ROM, 6x56 nibbles of RAM and 32 nibbles of dedicated System Register.

2.6.1 ROM

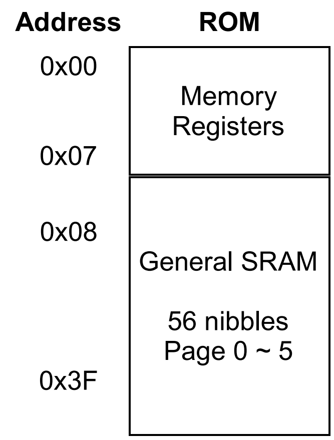
A large program/data/voice single ROM is provided, and its structure is shown below. The reserved region contains system information and can't be utilized by users. After reset process is completed, NY6 will start program execution from address 0x000.



Because program page size is 64K words defined by 16-bit length address of ROM, allowable range of unconditional branch instructions JMP and CALL are limited by program page size. However, combining with 3-bit BANK register, the total program size is 512K words. If users want to branch to program which is located beyond current program bank, user can change the BANK register first and then execute JMP or CALL instruction.

2.6.2 RAM

There are 6 pages of RAM, each page of RAM contains 56 nibbles. It's total 336 nibbles. The page of RAM defined by MPG (PAGE0~5), and its initial is PAGE0. Memory Registers of RPT0~5 and ROD1~2 will occupy address space from 0x00 to 0x07. Moreover, this address space of PAGE0~5 are mapped to the same dedicated registers. As consequence, the address space of PAGE0~5 RAM which can be used by programmer 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 {PAGE, RPT1[1:0], RPT0} or {PAGE, RPT3[1:0], RPT2} first, and users can read from or write in the XMD0/XMD1 memory register to realize the indexed RAM access.

2.7 I/O Ports

There are at most 24 I/O pins, designated as PAx through PFx, 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 (SFR $15), PBIO (SFR $17), PCIO (SFR $19), PDIO (SFR $1B), PEIO (SFR $1D) and PFIO (SFR $1F).

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 NY6. 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 NY6. 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, normal drive current and large sink current, or normal drive current and constant sink current.

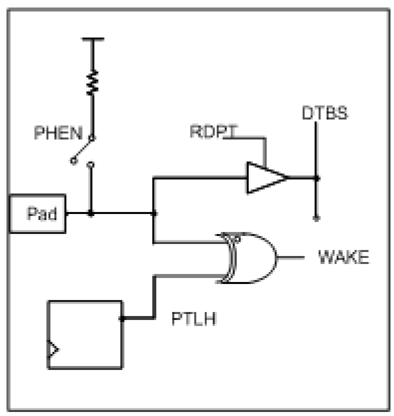
Some I/O ports can also be optioned as specific application, i.e. External reset pin (PA3), an infrared (IR) output pin (PA2), inputs of voltage comparator (PA0/PA1) or SPI associated control pins (PB0~3). A reset pin can possess a pull-high resister or not according to the mask option, which is used to enable/disable the pull-high resistor of I/O pin.

IR carrier polarity can be initial low or high according to data value. Moreover, the IR output can provide large sink current or not according to the mask option, which is used to determine output sink current described above.

The inputs of voltage comparator are designated to PA0/PA1 by option. Users can enable comparator through $ONOFF register for analog detection application.

SPI associated pins are allocated PB0 to CSb, PB1 to SCK, PB2 to MOSI and PB3 to MISO. NY6 plays as master part to control SPI flash. Also, NY6 supports two modes, User mode for normal read back or write in and Auto-play mode for automatically reading back speech data from SPI flash. For Auto-play Mode, NY6 offers this function to auto read back voice data to play, only for channel 0. Users have to finish a sequence of setting steps for voice pattern and set PLAY. For details, please refer to Chapter 2.10.

2.7.1 Pull-High Input Mode



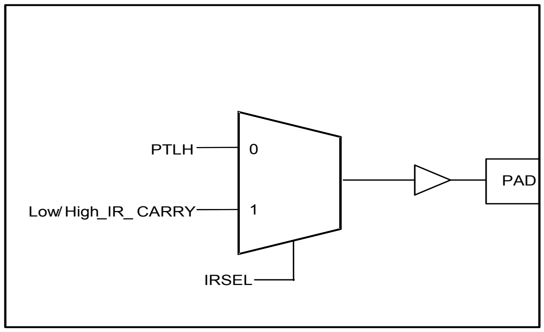
Pull-high Input Mode Configuration

Pad status of PA~PF, which are set as input mode, can be read in by MVTA. 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/constant sink current by setting related mask options. But drive current of NY6 is always weaker than normal sink current, about half the scale.

2.7.4 I/O Pin Mask Option

This Section will describe the summary of available functionalities for each I/O pin. All functionalities are determined by setting of corresponding mask options.

Category	I/O pin	Option
NY6A	PA0 PA1	Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.
	PA2/IR	IR carrier output or Normal I/O.
		Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.
	PA3/Reset	Reset input or Normal I/O.
		Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.
	PBx	Normal I/O
		Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.
	All I/O	Weak or strong input pull-high resistor.
NY6B	PA0/VIP PA1/VIN	Comparator input or Normal I/O.
		Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.
	PA2/IR	IR carrier output or Normal I/O.
		Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.

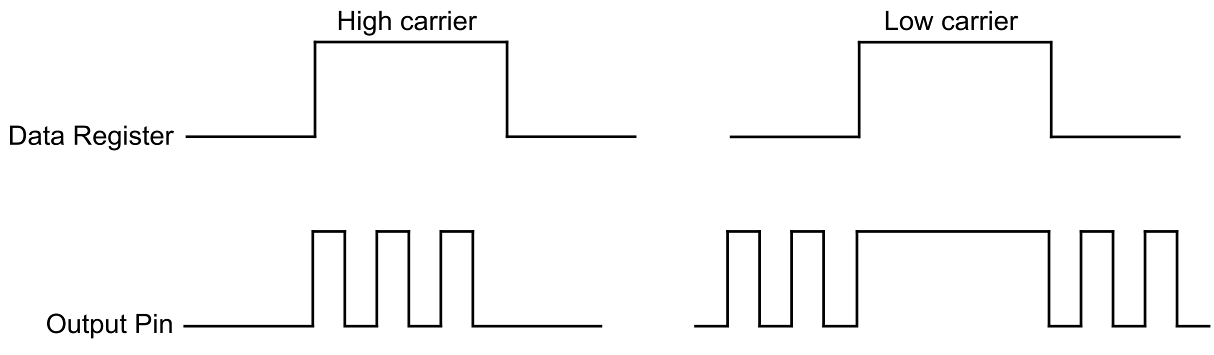
Category	I/O pin	Option
	PA3/Reset	Reset input or Normal I/O.
		Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.
	PBx PCx PDx	Normal I/O
		Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.
	All I/O	Weak or strong input pull-high resistor.
NY6C	PA0/VIP PA1/VIN	Comparator input or Normal I/O.
		Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.
	PA2/IR	IR carrier output or Normal I/O.
		Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.
	PA3/Reset	Reset input or Normal I/O.
		Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.
	PBx PCx PDx PEx PFx	Normal I/O
		Input with pull-high, floating or register-controlled pull-high.
		Normal, large or constant current output.
	All I/O	Weak or strong input pull-high resistor.

2.8 Infrared Transmitter

The NY6 series provides an infrared transmitter block, which is used to send infrared signal. Users can use PA2 as an IR output of NY6 series. Users can option to determine 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 pin sends infrared signal when the I/O port register value is low, and vice versa.

The IR frequency is programmable by 5 bits mask option, which can make frequency of IR carrier between 37.04KHz and 500KHz.

Category	Option	Description
IR	IR frequency	37.04 ~ 500KHz
	IR low/high carrier	Low
		High



2.9 Interrupt Generator

There is one hardware interrupt entrance in NY6 for multiple interrupt sources. The interrupt events are derived from system base timer (BT), Timer Counter(TM), PH counter (PHC), SPI status, and comparator status. Each one is enabled by SFR $INT0/INT1. The interrupt flag won't be reset automatically, but write 0 to its flag register, SFR $INTF0/INTF1. NY6 provides 4 kinds of fixed intervals from the system base timer for interrupt source: 0.256ms, 0.512ms, 1.024ms and 16.384ms. In Slow mode, those intervals will be multiplied by a fixed value, typically 14. Unless interrupt of BT is enabled by $INT0, the flag will keep initial state(low). However, the time interval from BT is enabled to first occurrence of interrupt may be not as accurate as specified due to NY6 characteristic.

As regards other interrupt sources, their function appliances should be turned on by $ONOFF, otherwise its block is disabled. Also, users have to enable entrances of interrupt of those appliances by $INT1. There is a difference from BT, those interrupt flags will be able to read back even if entrance is turned off. In other words, the flag is launched while the event is triggered and the procedure won't jump into interrupt subroutine.

For Timer Counter interrupt, timer value ($RTML/RTMH) and timer clock source($TMCS) have to be set first, then launch interrupt flag when timer counts to 0xFF and repeat from original timer value.

For PH counter interrupt, NY6 offers the PH of channel-1 to be a 12-bit counter and launch flag while PH value is over 0xFFF. Note that, PH counter has to be disabled before entering Halt mode

For SPI interrupt, there are two modes to launch the flag when frame data shifted out. For User mode, the flag goes high when the 8-bit data shifted out as common SPI protocol. For Play mode, the flag will be high when the 80-bit (PCM audio data) or 88-bit (ADPCM encoded data) frame data shifted out.

For comparator interrupt, the flag goes high from low while the level of VIP(PA0) higher than VIN(PA1) and keeps high until write 0 to clear flag through register $INTF1. Note that, the comparator flag will not be launched again even if VIP keeps higher than VIN. The level of VIP has to be lower than VIN for a while and turn high again to launch interrupt.

As an interrupt occurs, NY6 stores the accumulator (ACC), carry flag (C), zero flag (Z), RAM page (PAGE) and RPT0~5 automatically. PAGE is controlled by the command (MPG). Then NY6 move PC+1 to STK, and jump to the interrupt vector (0x000010). An interrupt routine finishes with an IRET instruction. The IC draws back ACC, C, Z, PAGE and RPT back, and moves STK to PC back to jump back the main program.

2.10 SPI Control Interface

Port PB is assigned for SPI interface, PB0 to CSb, PB1 to SCK, PB2 to MOSI and PB3 to MISO. For the connection with external Flash, the applied pins are MOSI, MISO and SCK. The MISO is the input pin to receive data from the external device, and the MOSI is the output pin to deliver data. The SCK is the output pin to offer the clock signal, and configured as Option (8M/4M/2M/1M Hz). However, the CSb of enable pin for the external device can share with one of IO ports and define it as output.

SPI interface is built-in to communicate with external flash through Port PB. In order to keep the same voltage level as external devices, PB_VDD is designated for Port PB power, and it can be also the power for external SPI devices. There are two typical applications for PB_VDD connections. Case 1: PB_VDD is connected to VDD when PB_VDD status is set as floating. Case 2: PB_VDD is connected to external SPI VDD pin when PB_VDD status is set as internal LDO regulator (3.3V).

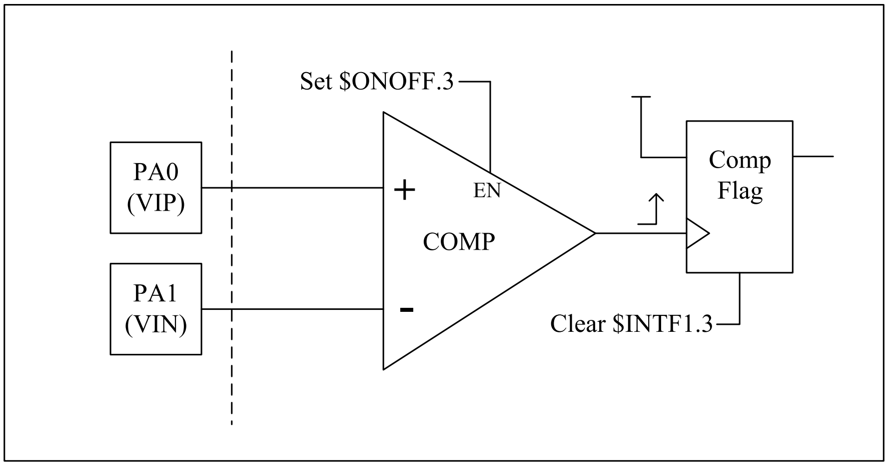
As the mentioned, the internal LDO regulator, it powers external flash and Port PB 10mA @3V. Users have to enable internal LDO by $SPIV before transmitting data or else the power for SPI interface sill be abnormal. Unless power is supplied by another source, external VDD, or the communication with SPI flash will be failed.

There are two modes designed in NY6 to communicate with SPI flash. User Mode is to access serial flash based on common SPI 8-bit protocol. NY6 acts as Master side to write command or address and read back data through serial flash.

Auto-Play mode is built-in protocol to communicate with SPI flash automatically. It supports users to access SPI flash and perform voice data to channel 0. Users just follow the specific start-up procedure, NY6 will automatically playback the voice data stored in flash and maximum of SPI size supported is up to 128M bits.

2.11 Comparator

A voltage comparator is built-in for analog signal detection applications. Users can apply PA0 / PA1 to inputs of comparator by mask option. Basically, the output of comparator is represented for the level difference between VIP (PA0) and VIN (PA1). If output of comparator goes high, VIP is with higher level than VIN; low, VIP with smaller level than VIN. The comparator flag will be set to high while the level of VIP is bigger than VIN and won’t be kept high even if the flag is clean and VIP is still bigger than VIN. Because the flag is controlled by a positive-edge clock source of register, the level of VIP has to be lower than VIP and the flag is clean by writing 0 to $INTF1. The flag will be launched while the level of VIP is bigger than VIN again.



2.12 LDO Regulator

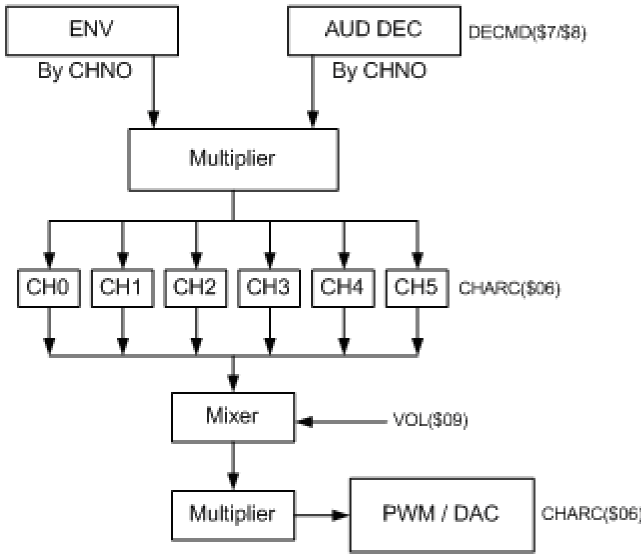
A LDO regulator is built-in as the power of PB0~3 to support SPI applications. Users can set 3.3V and internal power from LDO regulator through register $SPIV for SPI interface. The LDO regulator supplies enough power consumption for reading data from SPI flash, but programming SPI flash won't be supported due to less supply current of LDO (10mA). Particularly, the LDO regulator is designated for SPI interface, not normal IO. If users apply to control high-consumption device, i.e. LED, the application won't work as expected. Please note that a 0.1uF capacitor nearby PB_VDD pin is necessary to stabilize the voltage if LDO regulator is enabled.

2.13 Low Voltage Detector (LVD)

There is one hardware voltage detector in NY6. It offers four levels for various application, 2.4V, 2.7V, 3.6V and 4.1V 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 higher than selected level. User can check power status by setting different level and monitoring the flag. In general, for 2-battery application, 2.4V/2.7V will be chosen; 3-battery application, 3.6V/4.1V.

2.14 Audio Synthesizer Structure

NY6 provides a built-in speech/MIDI synthesizer. The synthesizer consists of six channels for voice or MIDI synthesis. The allowable simultaneous synthesis channel can be 2, 4 and 6. The block diagram of the synthesizer unit is shown in figure below.



2.14.1 Speech Synthesis

NY6 supports 10-bit PCM and encoded 4-bit / 5-bit mixed ADPCM speech data. The PCM voice has higher quality, but it occupies double ROM space at least than ADPCM. By cooperating with embedded noise filter of 250KHz over-sampling, it could decode high fidelity voice data even if you adapt ADPCM voice. It means you could store longer voice duration or provide more kinds of patch at lower sampling rate but enrich user's applications without degradation of sound quality.

2.14.2 MIDI Synthesis

There are three combinations to form a patch in NY6. The first way (called Head-Only) is to record a complete waveform, then play it by playing whole wave only. This is the best way to represent a high quality patch, but the price has to pay is the ROM cost. In contrast, users can extract the periodic part of a patch (called Tail-Only), then play it by playing the periodic wave repeatedly. The ROM occupied by this kind of patch is minimal; however, sound quality is sacrificed.

The compromise architecture is "Head+Tail" with envelope information, which is called ADSR. During MIDI synthesis, the Head wave is played only once and the Tail wave is always repeated to generate the synthesis output. Generally, the Head wave is used to represent the non-regular part at the beginning of a patch or to represent a whole of general voice or sound effect. The Tail wave is to represent a periodic cycle in the regular and periodic part in a patch. The Head wave and Tail wave are usually extracted from the same waveform and Tail wave is immediately successive to Head wave. This patch synthesis method can dramatically reduce ROM size needed to store the patch data.

Besides, a hardware circuit of automatic Tone-Calibration is built-in. It can result in near-zero frequency deviation for precise generation of tone frequency.

Note: There is a limitation about Tail waves that sample number of Tail waves must be integer multiple of 4 or 5 according to 5/4-bit data compression of ADPCM.

2.14.3 PH Value

User should set PH value in program to meet voice's sample rate or note's frequency. The PH value is derived by formula below:

$$\text{PH for voice synthesis (in Hex)} = \frac{SR \times 8 \times CH \times 4096}{F_{INST}}$$

$$\text{PH for MIDI synthesis (in Hex)} = \frac{SR \times 8 \times CH \times 4096}{F_{INST}} \times \frac{F_{NOTE}}{F_{PATCH}}$$

SR: sample rate of speech waveform or Head/Tail waveform. SR unit is hertz.

CH: the allowable value of CH is listed in table below.

Active Voice Channel	CH Value
2	2
4	4
6	6

Active MIDI Channel	CH Value
2	2
4	4
6	6

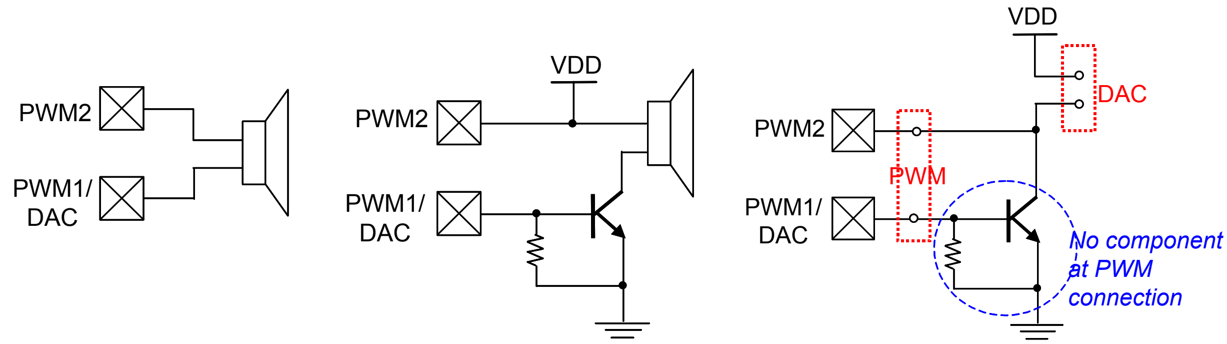
F_{INST}: Instruction frequency, 2,000,000 Hz ($2*10^6$).
F_{NOTE}: Frequency of the note which is being played.
F_{PATCH}: Frequency of key note on which patch waveform is based.

2.14.4 Audio Output

Before using the audio output, user can choose one of the 12-bit DAC or 12-bit PWM as the audio output for NY6 series. If DAC is selected, ramp-up process has to be implemented by user's application program. If PWM is selected, there is no need of ramp-up.

Besides in NY6, it provides a pad detecting mechanism to detect whether DAC or PWM is used. 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 Ultra PWM current for different customer demand. The Ultra PWM current consumes more current but makes sound louder.

2.14.5 Envelope Control

During speech synthesis or melody synthesis, there is one set of 8-bit envelope register, which can store the envelope information. Therefore NY6 can provide 256-level envelope control for each channel and users can use it as alternative of volume control for each channel. If user wants to have largest volume, value 0xFF is recommended.

As NY6 is a 6-channel synthesizer but there is only one set of envelope register physically, user has to load desired channel number to ACC and execute CHNO instruction to select a specific channel. Then, load the 256-level envelope data to RPT0~1 and execute LDEN instruction to update to this selected channel. Moreover, as NY6 micro-controller is 4-bit but envelope information is 8-bit, the envelope data of selected channel will not be updated until LDEN is executed. User can refer Chapter 3.8.6 for details.

2.14.6 Volume Control

NY6 supports 16-step digital volume control by the VOL register. Default value of VOL register is 0x8. In order to have suitable volume, VOL=0x3 is recommended for 6-ch speech/MIDI synthesis. VOL=0x4 is recommended for 4-ch speech/MIDI synthesis. VOL=0x8 is recommended for 2-ch speech/MIDI synthesis. VOL=0xF is recommended for 1-ch speech synthesis.

As sampled waveform of speech or Head/Tail may not fully occupy between maximum and minimum value, user may consider using larger value as digital volume than above recommended value for VOL register in order to have satisfied loudness. Moreover, because there is a Limiter after Mixer to saturate multi-channel synthetic result, it can prevent quality degradation of synthetic result.

In addition, NY6 supports instruction VOLX2 to multiply VOL by 2 for specific application. For instance, the maximum volume for 1-ch speech is 0xF, but if the sound is not loud enough, VOLX2 will double this volume 0xF to 0x1E to make it louder.

Chapter 3. System Control

3.1 Introduction

The INTx registers are used to enable the interrupt entrance for base timer(BT), Timer Counter(TM), PH Counter, SPI and Comparator application. The INTFx registers are for reading flag originated from those interrupt sources and written 0 to reset its flag individually. The BTF register is used to read BT signal of different interval 0.256ms, 0.512ms, 1.024ms and 16.384ms. Moreover, the BTF is also memory lock function for specific SRAM address 0x3E, 0x3F. The ONOFF register is to turn on block function such as Timer Counter(TM), PH Counter (PHC), SPI and Comparator. The CHARC, DECMD0, DECMD1 and VOL are audio control related registers. The LVD register is used to monitor IC power with four level setting, 2.4V, 2.7V, 3.6V and 4.1V, the output flag goes high while power is higher. The TMCS register is to select 8 clock sources (4MHz, 2MHz, 1MHz, 500KHz, 250KHz, 125KHz, 62.5KHz and Comparator output) for timer counter and timer value represented for timer counter or capture timer. The RTML/RTMH registers are used to access timer data. The XMDx are for indirect RAM access with different addressing composed by RPT and PAGE. The SPIV register is for power control for SPI interface, and SPIC register is for further SPI control. The SPIDx registers are used to access data between SPI flash. The Px and PxIO are I/O ports registers, here x could be A, B, C, D, E or F. As PA, PB, PC, PD, PE and PF are bi-directional I/O ports, PAIO, PBIO, PCIO, PDIO, PEIO and PFIO are used to determine the direction of each I/O pin.

3.1.1 System Register Address Map

Addr	Name	R/W	Bit	Data	Description	Default
$00	INT0	R/W	[0]	0/1	Disable / Enable BT 0.256ms Interrupt	Disable
			[1]	0/1	Disable / Enable BT 0.512ms Interrupt	Disable
			[2]	0/1	Disable / Enable BT 1.024ms Interrupt	Disable
			[3]	0/1	Disable / Enable BT 16.384ms Interrupt	Disable
$01	INT1	R/W	[0]	0/1	Disable / Enable Timer Interrupt	Disable
			[1]	0/1	Disable / Enable PH Interrupt	Disable
			[2]	0/1	Disable / Enable SPI Interrupt	Disable
			[3]	0/1	Disable / Enable COMP. Interrupt	Disable
$02	INTF0	R/W	[0]	0/1	BT 0.256ms Interrupt Flag, write 0 to clear flag	0
			[1]	0/1	BT 0.512ms Interrupt Flag, write 0 to clear flag	0
			[2]	0/1	BT 1.024ms Interrupt Flag, write 0 to clear flag	0
			[3]	0/1	BT 16.384ms Interrupt Flag, write 0 to clear flag	0
$03	INTF1	R/W	[0]	0/1	Timer Flag, write 0 to clear flag	0
			[1]	0/1	PH Flag, write 0 to clear flag	0
			[2]	0/1	SPI Flag, write 0 to clear flag	0
			[3]	0/1	COMP. Flag, write 0 to clear flag	0

Addr	Name	R/W	Bit	Data	Description	Default
$04	BTF	R	[0]	0/1	BT = 0.256 ms	
			[1]	0/1	BT = 0.512 ms	
			[2]	0/1	BT = 1.024 ms	
			[3]	0/1	BT = 16.384 ms	
$05	ONOFF	R/W	[0]	0/1	Timer Disable / Enable	Disable
			[1]	0/1	PH Counter Disable / Enable	Disable
			[2]	0/1	SPI Disable / Enable	Disable
			[3]	0/1	COMP. Disable / Enable	Disable
$06	CHARC	R/W	[1:0]	00	Audio Output Disable	Disable
				01	6 channels	
				10	4 channels	
				11	2 channels	
			[2]	0/1	POR Flag	1
			[3]	0/1	Audio output = DAC / PWM	(~AUD2)
$07	DECMD0	R/W	[0]	0/1	Noise Filter OFF / ON	OFF
			[1]	0/1	Tail Disable / Enable	Disable
			[3:2]	--	Reserved	0
$08	DECMD1	R	[1:0]	00	Head ADPCM 4-bit Mode	4-bit
				01	Head ADPCM 5-bit Mode	
				1x	Head PCM Mode	
			[3:2]	00	Tail ADPCM 4-bit Mode	4-bit
				01	Tail ADPCM 5-bit Mode	
				1x	Tail PCM Mode	
$09	VOL	R/W	[3:0]	0/1	16-level Volume [3:0]	0x8
$0A	LVD	R/W	[0]	0/1	LVD Function Disable / Enable	Disable
		R/W	[2:1]	00	VDD > 2.4V, Flag to High	> 2.8V
				01	VDD > 2.7V, Flag to High	
				10	VDD > 3.6V, Flag to High	
				11	VDD > 4.1V, Flag to High	
		R	[3]	0/1	LVD Flag	
$0B	TMCS	R	[2:0]	000	Timer Clock Source : 4M Hz	2MHz
				001	Timer Clock Source : 2M Hz	
				010	Timer Clock Source : 1M Hz	
				011	Timer Clock Source : 500K Hz	
				100	Timer Clock Source : 250K Hz	
				101	Timer Clock Source : 125K Hz	
				110	Timer Clock Source : 62.5 Hz	
				111	Timer Clock Source : Comparator Output	
			[3]	0/1	RTMx = Timer Counter / Capture Timer	Timer Counter

Addr	Name	R/W	Bit	Data	Description	Default
$0C	RTML	R/W	[3:0]	0/1	W: Load Timer Data R: Timer Counter / Capture Timer	
$0D	RTMH	R/W	[3:0]	0/1		
$0E	XMD0	R/W	[3:0]	0/1	Indexed SRAM data, {PAGE, RPT1[1:0], RPT0}	xxxx
$0F	XMD1	R/W	[3:0]	0/1	Indexed SRAM data, {PAGE, RPT3[1:0], RPT2}	xxxx
$10	SPIV	R	[0]	0/1	SPI Power = VDD / LDO	By Option
		R/W	[1]	0/1	Internal SPI Power Disable / Enable	Disable
		R	[2]	0/1	Comparator flag output	X
			[3]	0/1	Reserved	0
$11	SPIC	R/W	[0]	0/1	SCK initial Low / High	Low
		R	[1]	0/1	SPI Shift Done / Processing	Done
		R/W	[2]	0/1	SPID Resume / Pause	Resume
		R/W	[3]	0/1	User / Play Mode	User
$12	SPIDL	R/W	[3:0]	0/1	SDI[7:0] = {SPIDH, SPIDL} W : TX, R:RX	xxxx
$13	SPIDH	R/W	[3:0]	0/1		xxxx
$14	PA	R	[3:0]	0/1	PAIO = 1: Read port A input pad data	xxx
					PAIO = 0: Read port A output register	xxxx
		W	[3:0]	0/1	PAIO = 1: Wake-up Status (Option Disable) PAIO = 1: Floating / Pull-high (Option Enable)	wakeup status
					PAIO = 0: Write to port A output register	xxxx
$15	PAIO	R/W	[3:0]	0/1	Port A direction = Output / Input	Input
$16	PB	R	[3:0]	0/1	PBIO = 1: Read port B input pad data	xxxx
					PBIO = 0: Read port B output register	xxxx
		W	[3:0]	0/1	PBIO = 1: Wake-up Status (Option Disable) PBIO = 1: Floating / Pull-high (Option Enable)	wakeup status
					PBIO = 0: Write to port A output register	xxxx
$17	PBIO	R/W	[3:0]	0/1	Port B direction = Output / Input	Input
$18	PC	R	[3:0]	0/1	PCIO = 1: Read port C input pad data	xxxx
					PCIO = 0: Read port C output register	xxxx
		W	[3:0]	0/1	PCIO = 1: Wake-up Status (Option Disable) PCIO = 1: Floating / Pull-high (Option Enable)	wakeup status
					PCIO = 0: Write to port A output register	xxxx
$19	PCIO	R/W	[3:0]	0/1	Port C direction = Output / Input	Input
$1A	PD	R	[3:0]	0/1	PDIO = 1: Read port D input pad data	xxxx
					PDIO = 0: Read port D output register	xxxx
		W	[3:0]	0/1	PDIO = 1: Wake-up Status (Option Disable) PDIO = 1: Floating / Pull-high (Option Enable)	wakeup status
					PDIO = 0: Write to port A output register	xxxx
$1B	PDIO	R/W	[3:0]	0/1	Port D direction = Output / Input	Input
$1C	PE	R	[3:0]	0/1	PEIO = 1: Read port E input pad data	xxxx
					PEIO = 0: Read port E output register	xxxx

Addr	Name	R/W	Bit	Data	Description	Default
		W	[3:0]	0/1	PEIO = 1: Wake-up Status (Option Disable) PEIO = 1: Floating / Pull-high (Option Enable)	wakeup status
					PEIO = 0: Write to port A output register	xxxx
$1D	PEIO	R/W	[3:0]	0/1	Port E direction = Output / Input	Input
$1E	PF	R	[3:0]	0/1	PFIO = 1: Read port F input pad data	xxxx
					PFIO = 0: Read port F output register	xxxx
		W	[3:0]	0/1	PFIO = 1: Wake-up Status (Option Disable) PFIO = 1: Floating / Pull-high (Option Enable)	wakeup status
					PFIO = 0: Write to port A output register	xxxx
$1F	PFIO	R/W	[3:0]	0/1	Port F direction = Output / Input	Input

3.1.2 Memory Register Address Map

Addr	Name	R/W	Bit	Data	Description	Default
$0	RPT0	R/W	[3:0]	0/1	Multi-function register pointer [3:0]	‘b0000
$1	RPT1	R/W	[3:0]	0/1	Multi-function register pointer [7:4]	‘b0000
$2	RPT2	R/W	[3:0]	0/1	Multi-function register pointer [11:8]	‘b0000
$3	RPT3	R/W	[3:0]	0/1	Multi-function register pointer [15:12]	‘b0000
$4	RPT4	R/W	[3:0]	0/1	Multi-function register pointer [19:16]	‘b0000
$5	RPT5	R/W	[0]	0/1	Multi-function register pointer [20]	‘b0
$6	ROD1	R/W	[3:0]	0/1	ROM[7:4] data access register	xxxx
$7	ROD2	R/W	[1:0]	0/1	ROM[9:8] data access register	xx

3.2 RPT

As RPT have 6 registers and memory access may need up to 21 bits, RPT[3:0] is mapped to RPT0, RPT[7:4] is mapped to RPT1, RPT[11:8] is mapped to RPT2, RPT[15:12] is mapped to RPT3, RPT[19:16] is mapped to RPT4, RPT[20] is mapped to RPT5[0] and RPT5[3:1] are not used and read back “0”.

The RPT of NY6A and NY6B is 18-bit long, and the NY6C’s RPT is 21-bit. The redundant bits of RPT (RPT[20:18] of NY6A and NY6B) are un-writable and un-know if users read them. The RPT5 is 1-bit and its allocation is [0]. The functions of RPT are listed in the section 2.4.3.

Besides the instructions related to the LDPH only access bit [11:0] of the RPT, the RBDA only access bit [11:8] of the RPT, the LDEN only access bit [7:0] of the RPT, the XMD0 access bit [5:0] of RPT and the XMD1 only access bit [13:8] of the RPT, others instructions require all 18 or 21 bits available at RPT registers. The RPT will be frequently accessed because of its multi-functionality.

The SPI-related instructions, such RBSPRH, RBSPRL, LDSPRH and LDSPRL, access RPT twice for 24-bit addressing allocation for SPI flash. The RBSPRx is for reading 24-bit address, LDSPRx is for writing 24-bit address to RPT, “H” is for MSB 12-bit and ”L” is for LSB 12-bit data access.

3.3 ROD

When reading data from data ROM by table read instructions, these two registers will be used to store the higher bits of the obtained ROM data. After executing the RDN and RDNI instructions, bits [9:4] of the obtained 10-bit ROM data will be placed in ROD2[1:0] and ROD1[3:0] and bits [3:0] of ROM data will be placed in ACC.

3.4 INTx / INTFx ($0 ~ $03)

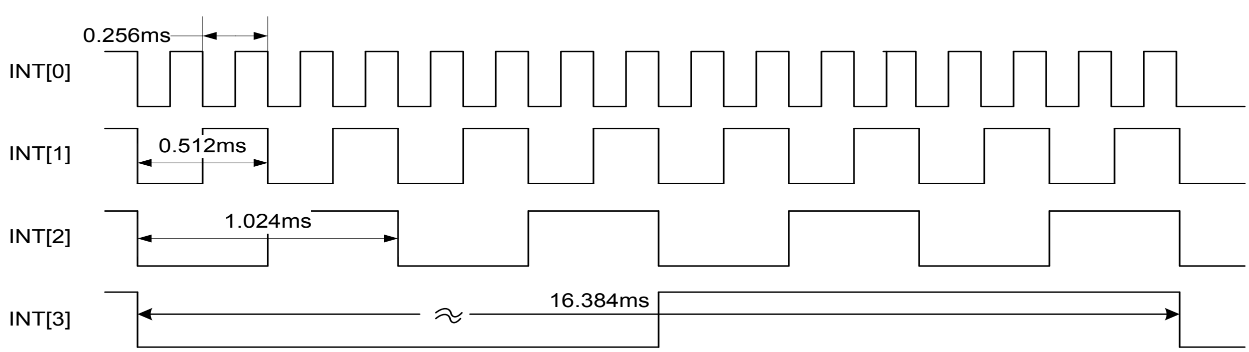
The INTx represents INT0 and INT1 register and controls the interrupt entrance reacts to base timer(BT), timer counter (TM), PH counter (PHC), SPI and comparator application. The program will get into the interrupt subroutine according to the occurrence of those interrupt sources. Users have to enable the corresponding interrupt source first and react as the event happens.

The INTFx represents INTF0 and INTF1 register and output high if the related interrupt is issued. Those interrupt flags can be clear by writing 0 to its bit, it won't be reset automatically. The INTF0 register is only reacted with four kinds of interval of base timer, 0.256ms, 0.512ms, 1.024ms, and 16.384ms.The INTF1 register is for four kinds of interrupt source, Timer counter(TM) flag issued as counter overflow, PHC flag issued as PH counter overflow and only set for channel-1, SPI flag issued as SPI shift process is done, and comparator flag issued as the level of VIP is higher than VIN and VIN is higher in the beginning.

Besides, those flags will keep to be launched while their events occur even if the corresponding interrupt is disabled. The INT0/INT1 registers are to permit those events for interrupt entrance, but their flags are still valid.

3.5 BTF ($04)

The reading source and the writing destination of the SFR[0x04] (BTF register) are different. Reading the 4-bit data of BTF acquires the value of the BT counter. The NY6 series provides 4 different base timer intervals for polling: 0.256ms, 0.512ms, 1.024ms and 16.384ms. The value of time means the period, so polling and finding data toggle means half time of the interval. Writing BTF is to apply for memory lock function only for 0x3E and 0x3F address in SRAM. Writing 0x5 is to unlock for current address and lock after this cycle accomplished, users has to write 0x5 again to unlock for next address, 0x3E or 0x3F.



INT Timing Figure

3.6 ONOFF ($05)

The ONOFF register is used to enable the block functions for timer counter(TM), PH counter(PHC), SPI and Comparator. The ONOFF[0] bit is to control Timer counter, disable the timer will stop counting. The ONOFF[1] bit is to set PH counter of channel-1 as an application for counting time, but the channel-1 won't be able to play voice simultaneously. The ONOFF[2] bit is to switch on SPI function and turns PB0~3 into SPI corresponding ports, PB0 to CSb, PB1 to SCK, PB2 to MOSI, PB3 to MISO. The PB0~3 will be normal ports (GPIO) if the SPI function is disable. The ONOFF[3] bit is to switch on comparator functionality and turns PA0 and PA1 into input VIP and input VIN. In addition, the wake-up function of PA0 and PA1 will be disabled once they are set for comparator inputs.

3.7 LVD ($0A)

The LVD register sets four kinds of voltage level and provide a flag for reading to monitor the voltage level of VDD. The LVD[0] bit is to enable/disable voltage detection functionality, its default is disable. The LVD[2:1] bits are used to select four voltage levels, 2.4V, 2.7V, 3.6V, 4.1V for multiple applications. Usually, for 2-battery usage, it is recommended to choose 2.4V and 2.7V; for 3-battery, 3.6V and 4.1V are suggested. The LVD[3] is the flag which is shown out the VDD status. The flag will go to high if VDD is higher than selected voltage level. The precision of level at 4.1V will be controlled in +/-5% and other levels less than 5%.

3.8 TMCS ($0B)

The TMCS register is used to select timer clock source and define timer value (RTMx) applied for timer counter or comparator application. The TMCS[2:0] bits are used to select 8 kinds of clock sources for timer counter, such as 4MHz, 2MHzm 500KHz, 250KHz, 125KHz, 62.5KHz and flag of comparator. The TMCS[3] bit is used to select timer value (RTMx) controlled by timer counter or comparator application. For general timer application, there is the specified procedure to set. First, set timer clock source, and load desired timer data to RTMx. Then, turn the timer counter on, the timer value will start counting from loaded timer data up to 0xFF. The flag will be launched due to the timer overflow and the timer value will be reloaded by the data of RTMx. For comparator application, user can set TMCS[3] to high and apply for an external RC circuit to PA0(VIP) and a reference voltage to PA1(VIN). While VIP is charged and higher than the level of VIN, the flag of comparator will be high and capture the current timer data to timer value (RTMx).

3.9 RTMx ($0C/$0D)

The RTMx registers are used to store 8-bit timer counter value or reload data. Writing RTML/RTMH is four bit register for timer reload data, once overflow occurs it will reload automatically. For reading those registers, users have to read RTMH first and system will save LSB 4-bit data of timer counter value into RTML at same time. Due to the time for reading RTML will spend two system cycles (~1us) at least after reading back RTMH. The timer counter keeps running and the current value of RTMH might be not as same as the time for RTML is read back. This case will influence the precision of RTMx, especially for faster timer clock source, i.e. 4MHz(0.25us) or 2MHz(0.5us).

3.10 XMDx ($0E/$0F)

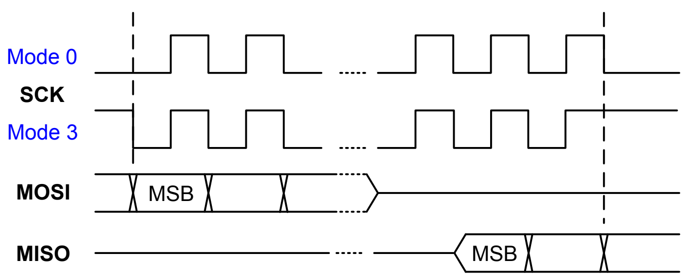
The NY6 series supports indirect-mode access of 336 nibbles SRAM data divided into 6 SRAM pages. There are two ways to access with XMD0 and XMD1 register. The XDM0 register is to access the 8-bit address comprised of {Page, RPT1[1:0], RPT0}; XMD1 is for the address {Page, RPT3[1:0], RPT2}, here Page is SRAM page decided by MPG instruction. After setting the SRAM address, the SRAM data can be read by reading data from the XMDx register and can be written by writing data to the XMDx register.

3.11 SPIV ($10)

The SPIV register is used to control power for SPI application. The SPIV[0] bit is read-only and defined by mask option, which represents SPI power supplied by internal LDO regulator or external VDD. The SPIV[1] bit is to enable LDO regulator, and the default is disable. The SPIV[2] bit is also read-only and represents the result of comparator and SPIV[3] bit is reserved.

3.12 SPIC ($11)

The SPIC is the register offers setting for SPI functionality. The SPIC[0] bit is to select SPI Mode 0 or Mode 3, the difference between the two modes is the serial clock polarity when the master is in Standby mode and not transferring data, as shown in the below.



Mode	Initial	Data Transfer	End
0	Low	X	Low
3	High	X	High

The SPIC[1] bit is represented the status of SPI data shifting and read only. If the bit is "1", that means SPI is busy doing shifting data serially into internal register; 0 means the process is done. The SPIC[2] bit is for Auto-Play Mode only to resume and pause the play procedure. The SPIC[3] bit is to select User / Auto-Play Mode, the User mode is 8-bit SPI protocol as normal to access SPI flash. For Auto-Play Mode, it's Read-only for receiving data related to 10-bit speech data, and basically it's hardware play mechanism. The format of speech data stored in SPI flash should be encoded in advance.

3.13 SPIDx ($12/$13)

The SPIDx is 8-bit registers divided into two 4-bit registers, SPIDL($12) and SPIDH($13). This set of two registers is to store SPI data for transferring or receiving to/from SPI flash. For transferring data, SPIDL register has to be executed first, then SPIDH. Once SPIDH is written, the procedure starts to shift out serially. For receiving data, there is no priority issue, once the shifting process is done and read both of them.

3.14 I/O Ports Register ($14 ~ $1F)

As PA, PB, PC, PD, PE and PF are bi-directional I/O ports, System register PAIO, PBIO, PCIO, PDIO, PEIO and PFIO are used to determine the direction of each I/O pin. Writing 1 to any bit of register PXIO (X=A~F), the corresponding I/O pin is configured as input pin. Writing 0 to any bit of PXIO (X=A~F), the corresponding I/O pin is configured as output pin. For I/O pin used as input pin, reading system register PX (X=A~F) 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 NY6 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~F, y=0~3). When 1 is written to register PX[y], high-to-low level change on this pin will wake up NY6. When 0 is written to register PX[y], low-to-high level change on this pin will wake up NY6. 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 NY6 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.

3.15 Audio Control Register

3.15.1 CHARC

The CHARC[1:0] set the active voice/MIDI channel number in NY6 voice/MIDI synthesizer as the following table:

CHARC		Total active channel number	Channels enabled
[1]	[0]		
0	0	0	All disable
0	1	6	channel 0~5
1	0	4	channel 0~3
1	1	2	channel 0~1

Thus the setting of fewer active channel numbers can achieve note synthesis of higher pitch frequency or higher octaves. The CHARC[2] bit is represented the power-on flag, once the power-on procedure is accomplished and the flag will keep high into program execution and wrote 0 to reset this flag. When power-on flag set to high as program execution, users can distinguish the reset event triggered by power-on reset (POR) or other events, such low-voltage reset (LVR), I/O reset or WDT reset. Only power-on reset (POR) will set this flag, others won’t. The CHARC[3] bit is to select DAC or PWM output mode. If the PWM mode is selected, all voice/MIDI channels will be mixed to PWM output. Switching between DAC and PWM modes during voice/MIDI playing should also be avoided.

3.15.2 DECMDx

The DECMDx registers are used to control audio format for each channel, x=0 or 1. For access of both registers, the referenced channel should be specified by instruction CHNO first to avoid setting to wrong channel. The DECMD0[0] bit is to control noise filter, OFF(=0) or ON(=1). The DECMD0[1] bit is used to enable (=1) or disable (=0) the inclusion of Tail wave in the voice synthesis procedure. The general situations to disable the Tail wave contain playing pure voice, sound effect or using a whole patch wave to synthesize MIDI (Head-Only). When “Tail-Only” mode is used in MIDI synthesis, it still takes advantage of “Head + Tail” mode. User has to enable Head waveform, which is the same as the Tail waveform. The other 2 bits of DECMD0 is reserved.

The DECMD1[1:0] bits are for Head wave to select audio format of the reference channel, there are 4-bit ADPCM, 5-bit ADPCM and PCM formats. The DECMD[3:2] are for Tail wave to select audio formats. When the voice decoder (ADCPM) is turned-off, NY6 plays the ROM data as pure PCM format. PCM format occupies twice the ROM space than ADPCM mode, and yield high quality voice. This setting is also specified for each channel individually. Therefore, specifies CHNM in advance before programming DECMDx.

As this feature can improve sound quality a lot, it is strongly recommended to enable noise filter for every application.

3.15.3 VOL

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

Total active channel	Recommended VOL value
6	0x3
4	0x4
2	0x8
1	0xF

3.16 Register Without Address Mapping

This Section will describe registers with implied addressing mode. There is no address assigned to this kind of registers, shows as below:

Name	R/W	Bit	Description	Initial
C	R/W	1	Arithmetic carry flag	0x0
Z	R/W	1	Arithmetic zero flag	0x0
BANK	R	3	ROM bank register	0x0
PAGE	R	3	3-bit RAM page register	0x0
CHNM	R/W	3	3-bit channel number register of desired channel	0x0
ENV	R/W	8	8-bit envelope of CH#	0x00
HPF	R	1	1-bit Head Play Flag of CH#	0x00
PFLG	R	1	1-bit Play Flag of CH#	0x00
PH	RW	12	12-bit PH value of CH#	0x000
MIXDT	R	12	12-bit Mixer data	0x000

3.16.1 BANK

The bank register is used to switch the program bank when the total program size has exceeded the capacity of single program bank. This register of NY6 series is 3-bit wide and the BANK instruction will write the specific number to bank register. Each program bank can address up to 64K words space and at most 8 banks are supported in NY6 chip. While the program execution will change to another program bank by JMP or CALL instruction, the bank register should be set with the program bank of targeting address in advance. Therefore, combining with bank register and 64K words program page, the total address space is 512K words.

3.16.2 PAGE

There are 6 memory pages in NY6 series. As PAGE register is not a system register or a memory mapped register, it can only be written by the MPG instruction and can’t be read. The MPG instruction will write the specific page number to PAGE register.

3.16.3 CHNM

The CHNM register is a channel selector that specifies which voice or MIDI channel will be referred to by the subsequently channel related register control or instruction execution. Before accessing the channel related registers or executing the channel related instructions, the CHNM register should be set correctly by instruction CHNO. The channel related registers include the ENV[7:0] and the DECMDx in SFR[0x7/0x8]. These registers have individual register settings for each channel. The channel related instructions include the PLAY, LDSEC, STOP, SNP, SP, SNHP, LDEN, RBEN and LDPH instructions.

The CHNM register is also used to specify the data or voice pointer register to be utilized by the LDPR, RDN, RDNI, RBPR, RBNVPR and RBVPR instructions. In NY6 chip, there are total 8 register locations shared by the data pointer registers and stack registers. The usage of stack registers grows from location 0x7 toward location 0x0 and the usage of data pointer registers should grow in the opposite direction.

3.16.4 ENV

This register is used to set the output voice envelope value (0x00 ~ 0xFF). Therefore digital volume of each channel is also controlled by 8-bit envelope. The channel to apply envelope setting is selected is by instruction LDEN, and the ENV will be loaded by RPT[7:0] (RPT1, RPT0). Also ENV is able to read back by instruction RBEN, and the RPT[7:0] is the register which saves envelope value. Note that the envelope data is a set of data dedicated for individual channel, that means the referenced channel has to be chosen before load or read back envelope date from ENV register toward/backward RPT[7:0].

3.16.5 Head Play Flag

HPF flag of a specific channel reflects the playback status of Head waveform at this specific channel, which can be Channel 0 to Channel 5. Therefore HPF flag is only associated with Head waveform. When HPF flag of a specific channel is 0, it means playback at this specific channel is completed. When HPF flag of a specific channel is 1, it means playback at this specific channel is on-going. The specific channel is determined by executing instruction CHNO and the ACC register should be written for CHNM. Users can obtain the status of HPF flag by instruction SNHP.

3.16.6 Play Flag

PFLG flag of a specific channel reflects the playback status of Head waveform or Tail waveform at this specific channel, which can be Channel 0 to Channel 5. Therefore PFLG flag is associated with Head waveform and Tail waveform. As long as either Head waveform is playing or Tail waveform is playing, PFLG flag of this specific channel will be 1. When Head waveform is end of play and Tail waveform is end of play, PFLG flag of this specific channel will be 0.

User can use PF flag and HPF flag together to decode the status of playback while Tail-Only mode or Head+Tail mode is used for MIDI synthesis. For example, PFLG=1 and HPF=0 means Head waveform is end of play and Tail waveform playback is on-going for a specific channel. The specific channel is determined by execution of instruction CHNO. Users can obtain the status of PF flag by instruction SP, SNP or SANP.

After instruction STOP is executed, the playback of specific channel determined by content of CHNM would stop immediately and PF flag will become 0.

3.16.7 PH Value Setting

PH is a 12-bit value, which represents how much relative time is elapsed from last playback sample based on ratio of sample rate to system clock. Therefore, this architecture will not produce accumulated error while counting sample rate in order to synthesize each note frequency precisely. Each channel has its own PH value. User can select a specific channel by executing instruction CHNO to select specific channel and utilize instruction LDPH to write value to PH.

3.16.8 Mixer Data

The Mixer output is temporarily stored to a 12-bit register, which is fed into Audio Output to produce audio signal. When Mixer is on (CHARC[1:0] ≠2’b00), user can utilize instruction RBDA to read this MSB 4-bit register to RPT2[3:0].

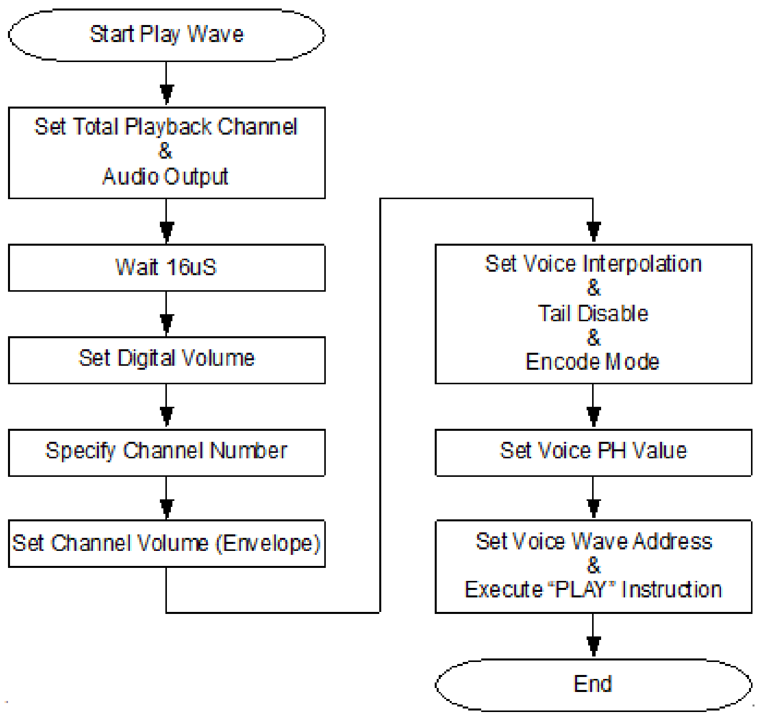
3.17 Audio Playback

This section will describe how to play voice and melody with example codes.

3.17.1 Voice Playback

3.17.1.1 Flow Chart

The flow chart of voice playback is depicted as graph below.



3.17.1.2 Programming Procedure

1. Setup Initial Starting Address of Voice File to Be Played

   The starting address must be aligned with specific address whose last 4 bits must be all zeros.

```
    ORGALIGN $, 0x10
L_Voice:
    #INCLUDATA  "Demo.v6x"
```

2. Setup Total Playback Channel

   The total number of playback channel can be 2, 4, 6 or disable, and this value will determine PH setting and volume setting accordingly. Once playback channel is set to non-disable, the audio output will be ready in 16us. Users have to wait the audio is fully turned on and execute “PLAY”.

3. Set Audio Output

   The audio output will be PWM or DAC for NY6 series. If DAC is selected, user has to implement ramp-up procedure by his program codes. If PWM is selected, it did not need ramp-up procedure.

4. Configure Digital Volume

   There are 16 kinds of digital volume could be applied, from 0x0 to 0xF.

5. Assign CHNM to Play

   Select specific NY6 channel to play voice before any further configuration.

6. Configure Envelop to Change Channel Volume

   By writing value to register {RPT0, RPT1} and executing instruction LDEN, there are at most 256 levels to adjust channel volume.

7. Set Interpolation

   User can select interpolation function enable or disable.

8. Setup “Head” Waveform and Voice File Format

   As voice is played, only is “Head” waveform allowed. The file format of voice file could be PCM, ADPCM4 or ADPCM5.

9. Determine PH value

   PH value is determined according to formula $\frac{SR \times 8 \times CH \times 4096}{F_{INST}}$.

   For example, SR=16,000 Hz, CH=2, F_{INST}=2,000,000, the PH value will be 0x20C.

10. Play Voice

    Instruction PLAY can be used to play voice and its usage is illustrated by the following piece of codes.

```
    MVLA        0x0             ; Set Voice Address
    MVAM        RPT0
    MVAM        RPT1
    MVLA        0x5
    MVAM        RPT2
    MVLA        0x2
    MVAM        RPT3
    MVLA        0x1
    MVAM        RPT4
    MVLR        0x0
    MVAM        RPT5
    PLAY                        ; Play

    ORG         0x12500
L_Voice:
    #INCLUDATA  "Demo.v6x"
```

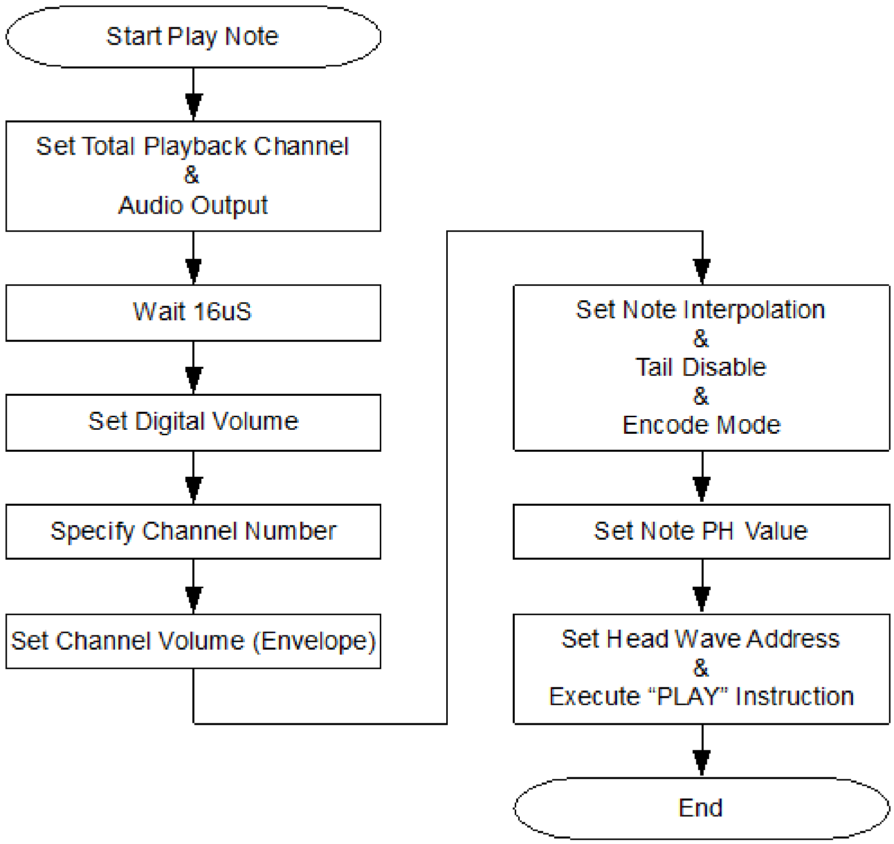
3.17.1.3 Example Code of Voice Playback

```
L_START:
        MVLA        0x09            ; Set Total Playback Channel & Audio Output
        MVAT        CHARC
        MVLA        0x0F            ; Set Digital Volume
        MVAT        VOL
        MVLA        0x00            ; Specify Channel Number
        CHNO
        MVLA        0xF             ; Set Channel Volume (Envelope)
        MVAM        RPT0
        MVAM        RPT1
        LDEN
        MVLA        0x01            ; Set Voice Interpolation & Tail Disable & Encode Mode
        MVAT        DECMD0
        MVLA        0x02
        MVAT        DECMD1
        MVLA        0x0B            ; Set Voice PH Value (Voice S.R.=12K)
        MVAM        RPT0
        MVLA        0x09
        MVAM        RPT1
        MVLA        0x04
        MVAM        RPT2
        MVLA        0x00
        MVAM        RPT3
        LDPH
        MVLA        0x00             ; Set Voice Wave Address & Execute “PLAY” Instruction
        MVAM        RPT0
        MVAM        RPT1
        MVLA        0x05
        MVAM        RPT2
        MVLA        0x02
        MVAM        RPT3
        MVLA        0x01
        MVAM        RPT4
        MVLA        0x00
        MVAM        RPT5
        PLAY                         ; Play

        ORG         0x12500
  L_Voice:
        #INCLUDATA "Demo.v6x"
```

3.17.2 Melody Playback, Head-Only Mode

3.17.2.1 Flow Chart

The flow chart of Head-Only melody playback is depicted as graph below.



3.17.2.2 Programming Procedure

1. Setup Initial Starting Address of Patch File to Be Played.

   The starting address must be aligned with specific address whose last 4 bits must be all zeros.

```
        ORGALIGN $, 0x10
L_Head:
        #INCLUDATA  "Piano_Head.v6x"
```

2. Setup Total Playback Channel

   The total number of playback channel can be 2, 4, 6 or disable, and this value will determine PH setting and volume setting accordingly. Once playback channel is set to non-disable, the audio output will be ready in 16us. Users have to wait the audio is fully turned on and execute "PLAY".

3. Set Audio Output

   The audio output will be PWM or DAC for NY6 series. If DAC is selected, user has to implement ramp-up procedure by his program codes. If PWM is selected, it did not need ramp-up procedure.

4. Configure Digital Volume

   There are 16 kinds of digital volume could be applied, from 0x0 to 0xF.

5. Assign CHNM to Play

   Select specific NY6 channel to play melody before any further configuration.

6. Configure Envelop to Change Channel Volume

   By writing value to register {RPT0, RPT1} and executing instruction LDEN, there are at most 256 levels to adjust channel volume.

7. Set Interpolation

   User can select interpolation function enable or disable.

8. Setup “Head” Waveform File Format

   As Head-Only mode is adopted, “Tail” waveform is disabled. The file format of patch file could be PCM or ADPCM5.

9. Determine PH value

   PH value is determined according to formula $\frac{SR \times 8 \times CH \times 4096}{F_{INST}} \times \frac{F_{NOTE}}{F_{PATCH}}$.

   For example, patch SR=22,050 Hz, CH=2, F_{INST}=2,000,000, F_{PATCH} is G3 (196.0 Hz), F_{NOTE} is B3 (246.9 Hz), the PH value will be 0x38E.

10. Play Melody

    Instruction PLAY can be used to play “Head” waveform and its usage is illustrated by the following piece of codes.

```
    MVLA        0x0             ; Set Head Wave Address
    MVAM        RPT0
    MVAM        RPT1
    MVLA        0x5
    MVAM        RPT2
    MVLA        0x2
    MVAM        RPT3
    MVLA        0x1
    MVAM        RPT4
    MVLR        0x0
    MVAM        RPT5
    PLAY                        ; Play

    ORG         0x12500
L_Head:
    #INCLUDATA  "Piano_Head.v6x"
```

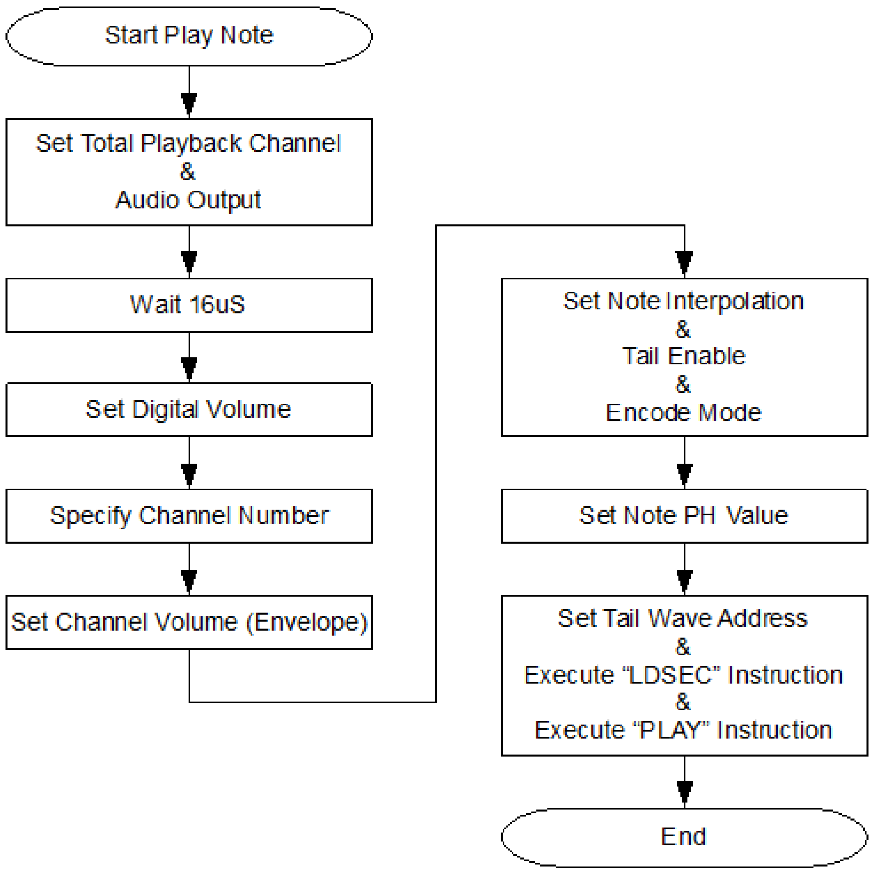
3.17.2.3 Example Code of Head-Only Melody Playback

```
L_START:
        MVLA    0x09        ; Set Total Playback Channel & Audio Output
        MVAT    CHARC
        MVLA    0x0F        ; Set Digital Volume
        MVAT    VOL
        MVLA    0x00        ; Specify Channel Number
        CHNO
        MVLA    0xF         ; Set Channel Volume (Envelope)
        MVAM    RPT0
        MVAM    RPT1
        LDEN
        MVLA    0x01        ; Set Note Interpolation & Tail Disable & Encode Mode
        MVAT    DECMD0
        MVLA    0x02
        MVAT    DECMD1
        MVLA    0x0C        ; Set Note PH Value
        MVAM    RPT0
        MVLA    0x02
        MVAM    RPT1
        MVLA    0x07
        MVAM    RPT2
        MVLA    0x00
        MVAM    RPT3
        LDPH
        MVLA    0x00        ; Set Head Wave Address & Execute "PLAY" Instruction
        MVAM    RPT0
        MVAM    RPT1
        MVLA    0x05
        MVAM    RPT2
        MVLA    0x02
        MVAM    RPT3
        MVLA    0x01
        MVAM    RPT4
        MVLA    0x00
        MVAM    RPT5
        PLAY

        ORG     0x12500
  L_ Head:
        #INCLUDATA  "Piano_Head.v6x"
```

3.17.3 Melody Playback, Tail-Only Mode

3.17.3.1 Flow Chart

The flow chart of Tail-Only melody playback is depicted as graph below.



3.17.3.2 Programming Procedure

1. Setup Initial Starting Address of Patch File to Be Played.

   The starting address must be aligned with specific address whose last 4 bits must be all zeros.

```
        ORGALIGN $, 0x10
    L_Tail:
        #INCLUDATA  "Piano_Tail.v6x"
```

2. Setup Total Playback Channel

   The total number of playback channel can be 2, 4, 6 or disable, and this value will determine PH setting and volume setting accordingly. Once playback channel is set to non-disable, the audio output will be ready in 16us. Users have to wait the audio is fully turned on and execute "PLAY".

3. Set Audio Output

   The audio output will be PWM or DAC for NY6 series. If DAC is selected, user has to implement ramp-up procedure by his program codes. If PWM is selected, it did not need ramp-up procedure.

4. Configure Digital Volume

   There are 16 kinds of digital volume could be applied, from 0x0 to 0xF.

5. Assign CHNM to Play

   Select specific NY6 channel to play melody before any further configuration.

6. Configure Envelop to Change Channel Volume

   By writing value to register {RPT0, RPT1} and executing instruction LDEN, there are at most 256 levels to adjust channel volume.

7. Set Interpolation

   User can select interpolation function enable or disable.

8. Setup Waveform File Format

   As Tail-Only mode is adopted, Head and Tail waveform are enabled. The so-called Tail-Only mode is to use same wave to Head and Tail. The file format of patch file could be PCM or ADPCM5.

9. Determine PH value

   PH value is determined according to formula $\frac{SR \times 8 \times CH \times 4096}{F_{INST}} \times \frac{F_{NOTE}}{F_{PATCH}}$.

   For example, patch SR=22,050 Hz, CH=2, F_{INST}=2,000,000, F_{PATCH} is G3 (196.0 Hz), F_{NOTE} is B3 (246.9 Hz), the PH value will be 0x38E.

10. Play Melody

    Although Tail-Only mode is adopted, NY6 still take advantage of Head+Tail mode to synthesize melody. In other words, the same waveform will be played as Head waveform and Tail waveform. What users have to do is to assign the starting address of Tail-only waveform as that of Head-Only waveform. Therefore both waveforms point to the same address. Instruction LDSEC can be used to play “Tail” waveform and its usage is illustrated by the following piece of codes.

```
MVLA    0x0     ; Set Tail Wave Address
MVAM    RPT0
MVAM    RPT1
MVLA    0x5
MVAM    RPT2
MVLA    0x2
MVAM    RPT3
MVLA    0x1
MVAM    RPT4
MVLR    0x0
MVAM    RPT5
LDSEC           ; Load Tail Address Data
PLAY            ; Play
```

```
        ORG             0x12500
    L_Tail:
        #INCLUDATA  "Piano_Tail.v6x"
```

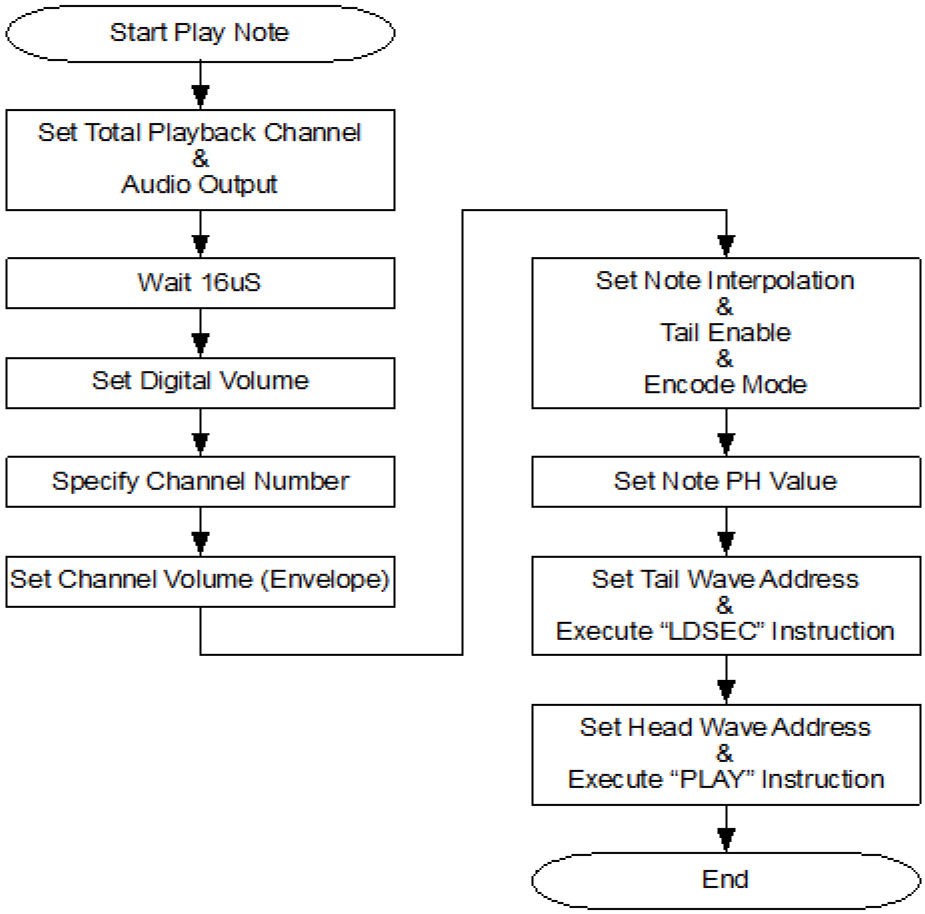
3.17.3.3 Example Code of Tail-Only Melody Playback

```
L_START:
        MVLA    0x09        ; Set Total Playback Channel & Audio Output
        MVAT    CHARC
        MVLA    0x0F        ; Set Digital Volume
        MVAT    VOL
        MVLA    0x00        ; Specify Channel Number
        CHNO
        MVLA    0xF         ; Set Channel Volume (Envelope)
        MVAM    RPT0
        MVAM    RPT1
        LDEN
        MVLA    0x03        ; Set Note Interpolation & Tail Enable & Encode Mode
        MVAT    DECMD0
        MVLA    0x0A
        MVAT    DECMD1
        MVLA    0x0C        ; Set Note PH Value
        MVAM    RPT0
        MVLA    0x02
        MVAM    RPT1
        MVLA    0x07
        MVAM    RPT2
        MVLA    0x00
        MVAM    RPT3
        LDPH
        MVLA    0x00        ; Set Tail Wave Address & Execute "LDSEC" Instruction
        MVAM    RPT0        ; & Execute "PLAY" Instruction
        MVAM    RPT1
        MVLA    0x05
        MVAM    RPT2
        MVLA    0x02
        MVAM    RPT3
        MVLA    0x01
        MVAM    RPT4
        MVLA    0x00
        MVAM    RPT5
        LDSEC
        PLAY
        ORG     0x12500
L_Tail:
        #INCLUDATA  "Piano_Tail.v6x"
```

3.17.4 Melody Playback, Head+Tail Mode

3.17.4.1 Flow Chart

The flow chart of Head+Tail melody playback is depicted as graph below.



3.17.4.2 Programming Procedure

1. Setup Initial Starting Address of Patch File to Be Played.

   The starting address must be aligned with specific address whose last 4 bits must be all zeros.

```
    ORGALIGN $, 0x10
L_Head:
    #INCLUDATA  "Piano_Head.v6x"

    ORGALIGN $, 0x10
L_Tail:
    #INCLUDATA  "Piano_Tail.v6x"
```

2. Setup Total Playback Channel

   The total number of playback channel can be 2, 4, 6 or disable, and this value will determine PH setting and volume setting accordingly. Once playback channel is set to non-disable, the audio output will be ready in 16us. Users have to wait the audio is fully turned on and execute "PLAY".

3. Set Audio Output

   The audio output will be PWM or DAC for NY6 series. If DAC is selected, user has to implement ramp-up procedure by his program codes. If PWM is selected, it did not need ramp-up procedure.

4. Configure Digital Volume

   There are 16 kinds of digital volume could be applied, from 0x0 to 0xF.

5. Assign CHNM to Play

   Select specific NY6 channel to play melody before any further configuration.

6. Configure Envelop to Change Channel Volume

   By writing value to register {RPT0, RPT1} and executing instruction LDEN, there are at most 256 levels to adjust channel volume.

7. Set Interpolation

   User can select interpolation function enable or disable.

8. Setup Waveform File Format

   As Head+Tail mode is adopted, “Tail” waveform is enabled too. The file format of patch file could be PCM or ADPCM5.

   The combination of file format of <Head, Tail> is <PCM, PCM>, <PCM, ADPCM5>, <ADPCM5, PCM> or <ADPCM5, ADPCM5>.

9. Determine PH value

   PH value is determined according to formula $\frac{SR \times 8 \times CH \times 4096}{F_{INST}} \times \frac{F_{NOTE}}{F_{PATCH}}$.

   For example, patch SR=22,050 Hz, CH=2, F_{INST}=2,000,000, F_{PATCH} is G3 (196.0 Hz), F_{NOTE} is B3 (246.9 Hz), the PH value will be 0x38E.

10. Play Melody

    Instruction PLAY can be used to play “Head” waveform. Instruction LDSEC can be used to play “Tail” waveform. Its usage is illustrated by the following piece of codes

```
MVLA    Low0(@@Tail)    ; Set Tail Wave Address
MVAM    RPT0
MVLA    Low1(@@Tail)
MVAM    RPT1
MVLA    Mid0(@@Tail)
MVAM    RPT2
MVLA    Mid1(@@Tail)
MVAM    RPT3
MVLA    High0(@@Tail)
MVAM    RPT4
MVLA    High1(@@Tail)
```

```
        MVAM        RPT5
        LDSEC                       ; Load Tail Address Data
        MVLA        Low0(@@Tail)    ; Set Head Wave Address
        MVAM        RPT0
        MVLA        Low1(@@Head)
        MVAM        RPT1
        MVLA        Mid0(@@Head)
        MVAM        RPT2
        MVLA        Mid1(@@Head)
        MVAM        RPT3
        MVLA        High0(@@Head)
        MVAM        RPT4
        MVLA        High1(@@Head)
        MVAM        RPT5
        PLAY                        ; Play

        ORGALIGN    $, 0x10
    L_Head:
        #INCLUDATA  "Piano_Head.v6x"

        ORGALIGN    $, 0x10
    L_Tail:
        #INCLUDATA  "Piano_Tail.v6x"
```

3.17.4.3 Example Code of Head+Tail Melody Playback

```
    L_START:
        MVLA        0x09            ; Set Total Playback Channel & Audio Output
        MVAT        CHARC
        MVLA        0x0F            ; Set Digital Volume
        MVAT        VOL
        MVLA        0x00            ; Specify Channel Number
        CHNO
        MVLA        0xF             ; Set Channel Volume (Envelope)
        MVAM        RPT0
        MVAM        RPT1
        LDEN
        MVLA        0x03            ; Set Note Interpolation & Tail Enable & Encode Mode
        MVAT        DECMD0
        MVLA        0x0A
        MVAT        DECMD1
        MVLA        0x0C            ; Set Note PH Value
        MVAM        RPT0
        MVLA        0x02
        MVAM        RPT1
```

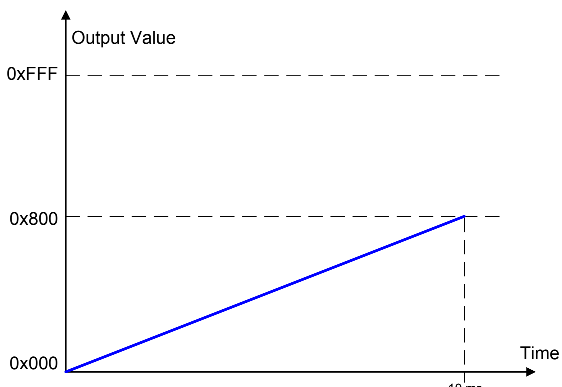
```
        MVLA        0x07
        MVAM        RPT2
        MVLA        0x00
        MVAM        RPT3
        LDPH
        MVLA        Low0(@@Tail)    ; Set Tail Wave Address & Execute "LDSEC" Instruction
        MVAM        RPT0
        MVLA        Low1(@@Tail)
        MVAM        RPT1
        MVLA        Mid0(@@Tail)
        MVAM        RPT2
        MVLA        Mid1(@@Tail)
        MVAM        RPT3
        MVLA        High0(@@Tail)
        MVAM        RPT4
        MVLA        High1(@@Tail)
        MVAM        RPT5
        LDSEC
        MVLA        Low0(@@Tail)    ; Set Head Wave Address & Execute "PLAY" Instruction
        MVAM        RPT0
        MVLA        Low1(@@Head)
        MVAM        RPT1
        MVLA        Mid0(@@Head)
        MVAM        RPT2
        MVLA        Mid1(@@Head)
        MVAM        RPT3
        MVLA        High0(@@Head)
        MVAM        RPT4
        MVLA        High1(@@Head)
        MVAM        RPT5
        PLAY

        ORGALIGN    $, 0x10
L_Head:
        #INCLUDATA  "Piano_Head.v6x"

        ORGALIGN    $, 0x10
L_Tail:
        #INCLUDATA  "Piano_Tail.v6x"
```

3.17.5 Ramp-up/Ramp-down Procedure for DAC

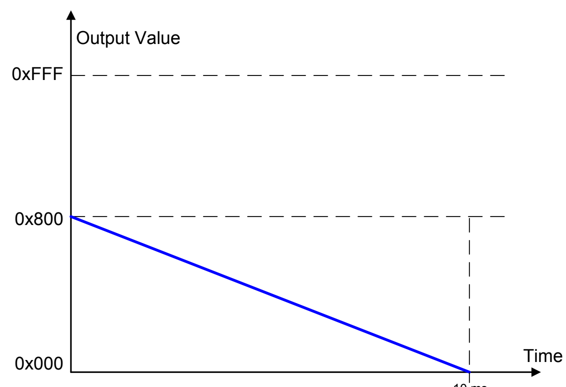
3.17.5.1 Operating Principle

While DAC is selected as audio output, the central point of DAC output is VDD/2 but DAC output is 0V before DAC is enabled. Therefore it needs a ramp-up process to make DAC output from 0V to VDD/2 before start of playback, and vice versa, a ramp-down process is necessary to make DAC output from VDD/2 to 0V after end of playback.

When user's voice data, either Speech, Head or Tail waveform, is encoded by *Voice_Encoder* and stored in NY6, users can take advantage of NY6 hardware to complete the ramp-up/ramp-down process. First of all, users have to provide one ramp-up and one ramp-down *.wav data and encode them with command "Encode Ramp Up/Down Table" in PCM format by *Voice_Encoder* and stored them in *.V6x format. It is recommended that length of ramp-up/ramp-down *.wav data is about 10ms and sample rate is higher than 8KHz. An example of ramp-up/ramp-down *.wav data is illustrated in the following graphs.



Ramp-up Waveform



Ramp-down Waveform

After ramp-up/ramp-down *.V6x files are ready, users can play this ramp-up/ramp-down *.V6x file by instruction PLAY as playing ordinary voice data to complete ramp-up/ramp-down process.

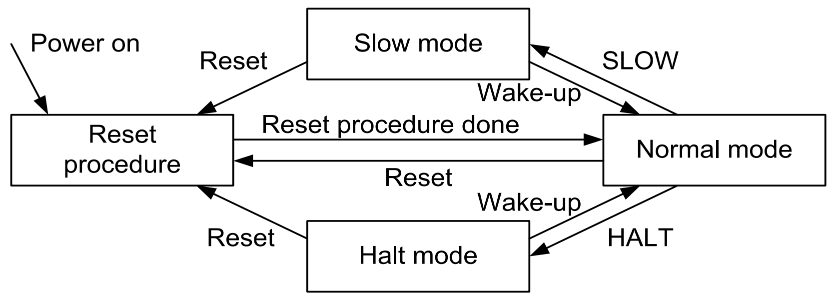
3.17.5.2 Example Code of Ramp-up/Ramp-down

```
L_START:
L_RampUp:
        MVLA        0x01            ; Set Total Playback Channel & DAC Output
        MVAT        CHARC
        MVLA        0x03            ; Set Digital Volume
        MVAT        VOL
        MVLA        0x00
L_RampUpChannelLoop:
        XORL        0x06
        SZEZ
        JMP         L_RampUpEnd
        XORL        0x06
```

```
        CHNO
        MVLA        0x01            ; Set Note Interpolation & Tail Disable & Encode Mode
        MVAT        DECMD0
        MVLA        0x02
        MVAT        DECMD1
        MVLA        0x05            ; Set Note PH Value
        MVAM        RPT0
        MVLA        0x02
        MVAM        RPT1
        MVLA        0x06
        MVAM        RPT2
        MVLA        0x00
        MVAM        RPT3
        LDPH
        MVLA        0x00            ; Set Head Wave Address & Execute "PLAY" Instruction
        MVAM        RPT0
        MVAM        RPT1
        MVLA        0x05
        MVAM        RPT2
        MVLA        0x02
        MVAM        RPT3
        MVLA        0x01
        MVAM        RPT4
        MVLA        0x00
        MVAM        RPT5
        PLAY
        RBCH
        INCA
        JMP         L_RampUpChannelLoop
L_RampUpEnd:
        ……………

        ORG         0x12500
L_RampUpFile:
        #INCLUDATA  "RampUp.v6x"
L_RampDown:
        MVLA        0x01            ; Set Total Playback Channel & DAC Output
        MVAT        CHARC
        Wait_16uS                   ; Wait 16uS for Ready of Audio Output
        MVLA        0x03            ; Set Digital Volume
        MVAT        VOL
        MVLA        0x00
L_RampDownChannelLoop:
        XORL        0x06
        SZEZ
        JMP         L_RampDownEnd
```

```
        XORL    0x06
        CHNO
        MVLA    0x01        ; Set Note Interpolation & Tail Disable & Encode Mode
        MVAT    DECMD0
        MVLA    0x02
        MVAT    DECMD1
        MVLA    0x05        ; Set Note PH Value
        MVAM    RPT0
        MVLA    0x02
        MVAM    RPT1
        MVLA    0x06
        MVAM    RPT2
        MVLA    0x00
        MVAM    RPT3
        LDPH
        MVLA    0x00        ; Set Head Wave Address & Execute "PLAY" Instruction
        MVAM    RPT0
        MVAM    RPT1
        MVLA    0x05
        MVAM    RPT2
        MVLA    0x02
        MVAM    RPT3
        MVLA    0x01
        MVAM    RPT4
        MVLA    0x00
        MVAM    RPT5
        PLAY
        RBCH
        INCA
        JMP     L_RampDownChannelLoop
L_RampDownEnd:
        ……………

        ORG     0x12500
L_RampDownFile:
        #INCLUDATA "RampDown.v6x"
```

3.18 Power Saving Mode



Power Saving Mode Flow Chart

3.18.1 Slow Mode

The system enters the Slow mode if the SLOW command is executed. The system clock in the Slow mode is about 14.3 (+/-3%) times slower than in the Normal mode. The difference between the Halt mode and the Slow mode 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 level change. In Slow mode, there are 4 kinds of base timer intervals for polling: 3.661ms, 7.322ms, 14.643ms and 234.291ms.The wake-up stable time from Slow mode is about 50us.

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 NY6 is waked-up from the Slow mode by an external reset signal, it goes into the reset procedure. After IC is waked-up by the input port level change, the next instruction after the SLOW instruction will be executed immediately. On the other hands, after IC is waked-up by the interrupt of BT, its interrupt service routine will be executed immediately. Remember to turn off the audio output before entering to the slow mode.

3.18.2 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 system from Halt mode is an input port level change 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 next instruction after the HALT instruction will be executed after the wake-up stable time (about 50us) is expired. 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 external reset signal, it goes into the reset procedure.

Note: There is a limitation about PH Counter under Halt and Slow mode. Users have to disable PH counter ($ONOFF[2]) before getting into Halt/Slow Mode.

Chapter 4. Instruction Set

4.1 Instruction Classified Table

Item	Inst.	Op1	Op2	Operation	Exec. Cycle	Inst. Length	Oper. Flag	Flag Affected
Arithmetic Instructions								
1	ADDM	6m		{C,A} = A + M + C	1	1	C	C, Z
2	SUBM	6m		{C,A} = A – M - (~B)	1	1	C	C, Z
3	INCM	6m		{C,M} = M + 1	1	1		C, Z
4	DECM	6m		{C,M} = M – 1	1	1		C, Z
5	ANDM	6m		A = A & M	1	1		Z
6	ORM	6m		A = A | M	1	1		Z
7	MVAM	6m		M = A	1	1		
8	MVMA	6m		A = M	1	1		Z
9	XORM	6m		A = A ⊕ M	1	1		Z
10	MVAT	5t		T = A	1	1		
11	MVTA	5t		A = T	1	1		Z
12	ADDL	4L		{C,A} = L + A + C	1	1	C	C, Z
13	SUBL	4L		{C,A} = A – L - (~B)	1	1	C	C, Z
14	ANDL	4L		A = A & L	1	1		Z
15	ORL	4L		A = A | L	1	1		Z
16	XORL	4L		A = A ⊕ L	1	1		Z
17	MVLA	4L		A = L	1	1		
18	INTCB	2t	2b	Clear T[b]	1	1		
19	INTSB	2t	2b	Set T[b]	1	1		
20	INCA			{C,A} = A + 1	1	1		C, Z
21	DECA			{C,A} = A – 1	1	1		C, Z
22	RRC			Right Rotate A with C	1	1	C	C, Z
23	RLC			Left Rotate A with C	1	1	C	C, Z
24	RRA			Right Rotate A	1	1		
25	RLA			Left Rotate A	1	1		
26	SETC			Set Carry flag	1	1		C
27	CLRC			Clear Carry flag	1	1		C
Conditional Instructions								
28	SAGT	4L		Skip when A > L	1-4	1		
29	SALT	4L		Skip when A < L	1-4	1		
30	SAEL	4L		Skip if A = L	1-4	1		
31	SCEZ			Skip if C = 0	1-4	1		
32	SZEZ			Skip if Z = 0	1-4	1		
33	SCNZ			Skip if C != 0	1-4	1		
34	SZNZ			Skip if Z != 0	1-4	1		
35	SBZ	2b		Skip when A[b] = 0	1-4	1		
Audio Instructions								
36	SNP			Skip when No Play of CHNM	1-4	1		
37	SP			Skip when Play of CHNM	1-4	1		
38	SANP			Skip when ALL 6 channels Play = 0	1-4	1		

Item	Inst.	Op1	Op2	Operation	Exec. Cycle	Inst. Length	Oper. Flag	Flag Affected
39	SNHP			Skip when head wave No Play of CHNM	1-4	1		
40	STOP			Stop wave play of CHNM	1	1		
41	CHNO			Load ACC to CHNM	1	1		
42	RBCH			Read CHNM to A	1	1		
43	VOLX1			VOL * 1	1	1		
44	VOLX2			VOL * 2	1	1		
45	LDEN			Load RPT[7:0] to ENV	1	1		
46	RBEN			Read ENV to RPT[7:0]	1	1		
47	PLAY			Load RPT to HVPR of CHNM	3	1		
48	LDSEC			Load RPT to TVPR of CHNM	3	1		
49	LDPR			Load RPT to DPR/STK	3	1		
50	LDPH			Load RPT[11:0] to PH of CHNM	1	1		
51	RBVPR			Read HVPR to RPT of CHNM	3	1		
52	RBNVPR			Read TVPR to RPT of CHNM	3	1		
53	RBPR			Read DPR/STK to RPT	3	1		
54	RDN			Data Table Read of CHNM	3	1		
55	RDNI			Data Table Read of CHNM	3	1		
Other Instructions								
56	CALL	16a		Call Adr	2	2		
57	JMP	16a		Jump Adr	2	2		
58	BANK	3bk		Set 3-bit Bank (512K)	1	1		
59	MPG	3p		Move Page to MPG	1	1		
60	RBSPRH			Read SPR[23:12] to RPT[11:0]	1	1		
61	RBSPRL			Read SPR[11:0] to RPT[11:0]	1	1		
62	LDSPRH			Load RPT[11:0] to SPR[23:12]	1	1		
63	LDSPRL			Load RPT[11:0] to SPR[11:0]	1	1		
64	SEI			Mask Interrupt	1	1		
65	CLI			Non-mask Interrupt	1	1		
66	RBDA			Read DAC[12:8] data to RPT2	1	1		
67	HALT			Enter sleep mode	1	1		
68	SLOW			Enter slow mode	1	1		
69	CWDT0			Clear WDT Step1	1	1		
70	CWDT1			Clear WDT Step2	1	1		
71	LDPC			Move RPT to PC	2	1		
72	RBPC			Move PC to RPT	2	1		
73	RET			Return from subroutine(CALL)	2	2		
74	IRET			Return from interrupt	2	2		
75	NOP			No Operation	1	1		

A, ACC : 4-bit Accumulator data

B : 1-bit borrow flag data, shared with carry flag, B=~C.

C : 1-bit carry flag data

M : 4-bit RAM or memory register data

T : 4-bit system function register data

L : 4-bit immediately literal data

Z : 1-bit zero flag data

RPT : Multi-function register data

CHNM : 3-bit channel number register

PH : 12-bit value for MIDI synthesis of CHNM

HVPR, TVPR : Head / Tail Voice address pointer of CHNM

ENV : 8-bit envelope data of CHNM

ROM : 10-bit ROM data

ROD : ROM data access register data

PC : Program counter address pointer

DPR : Data address pointer

STK : Interrupt dedicated stack address pointer

- a : ROM address
- b : bit address
- d : data pointer number
- m : RAM or memory register address
- t1 : 5-bit address of System Function Register
- t2 : 2-bit address of System Function Register
- p : 3-bit page pointer of RAM
- bk : 3-bit bank pointer of ROM

4.2 Instruction Descriptions

4.2.1 Arithmetic Instructions

ADDM m

Function : Add M to ACC with Carry and the result is saved back to ACC.

Operation : { C, ACC } ← ACC + M + C

Operand: m: 6-bit address of register or SRAM to ADD, 0x00 to 0x3F

Words : 1

Cycles : 1

Operative Flags: C

Flags Affected: C, Z

Example : ADDM m0

Before Instruction

ACC=0x7, [m0]=0xA, C=0

After Instruction

ACC=0x1, [m0]=0xA, C=1, Z=0

SUBM m

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

Operation : { C, ACC } = ACC - m - (~B)

Operand : m: 6-bit address of register or SRAM to subtract with, 0x0 to 0x3F

B: 1-bit borrow flag data, shared with carry flag, B=~C.

Words : 1

Cycles : 1

Operative Flags: C

Flags Affected: C, Z

Example : SUBM m0

Before Instruction

ACC=0xA, [m0]=0x2, C=1

After Instruction

ACC=0x8, [m0]=0x2, Z=0, C=1

INCM m

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

Operation:{ C, M } ← M + 1

Operand: m: 6-bit address of register or SRAM to increase, 0x00 to 0x3F

Words: 1

Cycles: 1

Operative Flags: C

Flags Affected: C, Z

Example: INCM m0

Before Instruction

[m0]=0x0

After Instruction

[m0]=0x1, C=0, Z=0

DECM m

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

Operation: { C, M } ← M - 1

Operand: m: 6-bit address of register or SRAM to decrease, 0x00 to 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

ANDM m

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

Operation: ACC ← ACC & M

Operand: m: 6-bit address of register or SRAM to AND with, 0x0 to 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: ANDM m0

Before Instruction

ACC=0x7, [m0]=0xA

After Instruction

ACC=0x2, [m0]=0xA, Z=0

ORM m

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

Operation :ACC ← ACC | M

Operand: m: 6-bit address of register or SRAM to OR with, 0x0 to 0x3F

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: Z

Example : ORM m0

Before Instruction

ACC=0x3, [m0]=0xB

After Instruction

ACC=0xB, [m0]=0xB, Z=0

MVAM m

Function: Move ACC to M of address m.

Operation: M ← ACC

Operand: m: 6-bit address of register or SRAM to move, 0x00 to 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MVAM m0

Before Instruction

ACC=0x8

After Instruction

[m0]=0x8

MVMA m

Function: Move M of address m to ACC.

Operation: ACC ← M

Operand: m: 6-bit address of register or SRAM to move, 0x00 to 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: MVMA m0

Before Instruction

[m0]=0x8

After Instruction

ACC=0x8

XORM m

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

Operation :ACC ← ACC ⊕ M

Operand: m: 6-bit address of register or SRAM to XOR, 0x00 to 0x3F

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: Z

Example : XORA m0

Before Instruction

ACC=0x3, [m0]=0xB

After Instruction

ACC=0x8, [m0]=0xB, Z=0

MVTA t

Function: Move M of address m to ACC.

Operation: ACC ← T

Operand: m: 6-bit address of SFR register, 0x00 to 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: MVTA t0

Before Instruction

[t0]=0x8

After Instruction

A=0x8

MVAT t

Function: Move ACC to T of address t.

Operation: T ← ACC

Operand: t: 6-bit address of System register, 0x00 to 0x3F

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MVAT t0

Before Instruction

ACC=0x8

After Instruction

[t0]=0x8

MVLA L

Function : Move immediate constant to ACC.

Operation : ACC ←L

Operand :L: 4-bit immediate constant value, 0x0 to 0xF

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : MVLA 0x7

Before Instruction

ACC=0x3

After Instruction

ACC=0x7

ADDL L

Function : Add immediate constant to ACC with Carry and the result is save back to ACC.

Operation : { C, ACC } ← ACC + L + C

Operand: L:4-bit immediate constant value, 0x0 to 0xF

Words : 1

Cycles : 1

Operative Flags: C

Flags Affected: C, Z

Example : ADDL 0xA

Before Instruction

A=0x7, L=0xA, C=0

After Instruction

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

ANDL L

Function : AND ACC with immediate constant, and the result is save back to ACC, i.e.

Operation :ACC ← ACC & L

Operand: L: 4-bit immediate constant value, 0x0 to 0xF

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: Z

Example : ANDL 0xB

Before Instruction

A=0x3, L=0xB

After Instruction

A=0x3, Z=0

SUBL L

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

Operation : { C, A } = A - L - (~B)

Operand: L: 4-bit immediate constant value, 0x0 to 0xF

B: 1-bit borrow flag data, shared with carry flag, B=~C.

Words : 1

Cycles : 1

Operative Flags: C

Flags Affected: C, Z

Example : SUBL 0x2

Before Instruction

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

After Instruction

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

ORL L

Function : OR ACC with immediate constant, and the result is save back to ACC, i.e.

Operation: ACC ← ACC | L

Operand: L: 4-bit immediate constant value, 0x0 to 0xF

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: Z

Example : ORL 0xB

Before Instruction

ACC=0x3, L=0xB

After Instruction

ACC=0xB, L=0xB, Z=0

INTCB t2, b

Function : Clear bit [b] of SFR(t2) to 0

Operation :0 ← t2[b]

Operand : t2: 2-bit address of SFR register to clear bit, 0x00 to 0x03

b: 2-bit bit location to clear to 0

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : INTCB t2, 0x2

Before Instruction

[t2]=0xF

After Instruction

[t2]=0xB

XORL L

Function : Exclusive OR ACC with immediate constant, and the result is save back to ACC.

Operation :ACC ← ACC ⊕ L

Operand: L: 4-bit immediate constant value, 0x0 to 0xF

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: Z

Example : XORL 0xB

Before Instruction

AC=0x3, L=0xB

After Instruction

AC=0x8, L=0xB, Z=0

INTSB t, b

Function : Set bit [b] of (t2) to 1

Operation :1 ← t2[b]

Operand :t2: 2-bit address of SFR register to clear bit, 0x00 to 0x03

b: 2-bit bit location to set to 1

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example :INTSB t2, 0x2

Before Instruction

[t2]=0x0

After Instruction

[t2]=0x4

INCA

Function: Add 1 to ACC, and save the result back to ACC.

Operation: { C, ACC } ← ACC + 1

Operand: None

Words: 1

Cycles: 1

Operative Flags: C

Flags Affected: C, Z

Example: INCA

Before Instruction

ACC=0x0

After Instruction

ACC=0x1, C=0, Z=0

DECA

Function: Subtract 1 from ACC, and save the result back to ACC.

Operation: { C, ACC } ← ACC - 1

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C, Z

Example: DECA

Before Instruction

ACC=0x0

After Instruction

ACC=0xF, C=0, Z=0

RLC

Function : Left rotate ACC with Carry.

Operation: { ACC[3:0], C } ← { C, ACC [3:0] }

Operand: None

Words : 1

Cycles : 1

Operative Flags: C

Flags Affected: C, Z

Example : RLC

Before Instruction

ACC =0xE ,C=0

After Instruction

ACC =0xC ,C=1 ,Z=0

RRC

Function : Right rotate ACC with Carry, i.e.

Operation: { C, ACC [3:0] } ← { ACC [3:0], C }

Operand: None

Words : 1

Cycles : 1

Operative Flags: C

Flags Affected: C, Z

Example : RRC

Before Instruction

ACC =0x3 ,C=1

After Instruction

ACC =0x9 ,C=1 ,Z=0

RRA

Function : Right rotate ACC.

Operation: { ACC[0], ACC [3:1] } ← { ACC [3:0] }

Operand: None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : RRA

Before Instruction

ACC =0xE ,Z=0

After Instruction

ACC =0x7 ,Z=0

RLA

Function : Left rotate ACC.

Operation: { ACC[2:0], ACC[3] } ← { ACC [3:0] }

Operand: None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : RLA

Before Instruction

ACC =0x3 ,Z=0

After Instruction

ACC =0x6 ,Z=0

CLRC

Function : Clear Carry bit 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 Carry bit 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

4.2.2 Conditional Instructions

SAGT L

Function: Skip the next instruction if ACC greater than immediate constant.

Operation: Skip next if ACC > L

Operand: L: 4-bit immediate constant value, 0x0 to 0xF

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: None

Flags Affected: None

Example: SAGT 0x8
Inst1
Inst2

After Instruction

If ACC = (or <) 0x8, `Inst1' is executed.

If ACC > 0x8, `Inst1' is discarded, and `Inst2' is executed.

SAEL L

Function: Skip the next instruction if ACC equal immediate constant.

Operation: Skip next if ACC = L

Operand: L: 4-bit immediate constant value, 0x0 to 0xF

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: None

Flags Affected: None

Example: SAEL 0x8
Inst1
Inst2

After Instruction

If ACC ≠ 0x8, `Inst1' is executed.

If ACC = 0x8, `Inst1' is discarded, and `Inst2' is executed.

SALT L

Function: Skip the next instruction if ACC less than immediate constant.

Operation: Skip next if ACC < L

Operand: L: 4-bit immediate constant value, 0x0 to 0xF

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: None

Flags Affected: None

Example: SALT 0x8
Inst1
Inst2

After Instruction

If ACC = (or >) 0x8, `Inst1' is executed.

If ACC < 0x8, `Inst1' is discarded, and `Inst2' is executed.

SCEZ

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

Operation: Skip next if C = 0

Operand: None

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: C

Flags Affected: None

Example: SCEZ
Inst1
Inst2

After Instruction

If C ≠ 0x0, `Inst1' is executed.

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

SZEZ

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

Operation: Skip next if Z = 0

Operand: None

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: Z

Flags Affected: None

Example: SZEZ

Inst1

Inst2

After Instruction

If Z ≠ 0x0, `Inst1’ is executed.

If Z = 0x0, `Inst1’ is discarded, and `Inst2’ is executed.

SZNZ

Function: Skip the next instruction if Zero equal to 1

Operation: Skip next if Z = 1

Operand: None

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: Z

Flags Affected: None

Example: SZNZ

Inst1

Inst2

After Instruction

If Z ≠ 0x 1, `Inst1’ is executed.

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

SCNZ

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

Operation: Skip next if C = 1

Operand: None

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: C

Flags Affected: None

Example: SCNZ

Inst1

Inst2

After Instruction

If C ≠ 0x 1, `Inst1’ is executed.

If C = 0x 1, `Inst1’ is discarded, and `Inst2’ is executed.

SBZ b

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

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

Operand : b: 2-bit bit location, 0x0 to 0x3

Words : 1

Cycles : 1, (2, 3, 4)

Example : SBZ 0x3

Inst1

Inst2

After Instruction

If ACC[3] = 1, `Inst1’ is executed.

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

4.2.3 Audio Instructions

PLAY

Function : Play voice (Head wave) on the channel indexed by the CHNM register. The voice (Head wave) address should be loaded in RPT firstly.

Operation : HVPR[CHNM] ← RPT

Operand : None

Words : 1

Cycles : 3

Operative Flags: None

Flags Affected: None

Example :
```
MVLA 0x1
CHNO
MVLA 0x4
MVAM RPT0
MVLA 0x3
MVLM RPT1
MVLA 0x2
MVAM RPT2
MVLA 0x0
MVAM RPT3
MVLA 0x1
MVAM RPT4
MVLA 0x0
MVAM RPT5
PLAY
```

Before Instruction

HVPR[0x1]= 0xXXXXX

After Instruction

HVPR[0x1]= 0x010234, PFLG[0x1]=1

LDSEC

Function : Load Tail wave address for the channel indexed by the CHNM register. The Tail wave address should be loaded in TREG[19:0] firstly.

Operation : TVPR[CHNM] ← RPT

Operand : None

Words : 1

Cycles : 3

Operative Flags: None

Flags Affected: None

Example :
```
MVLA 0x4
CHNO
MVLA 0x4
MVLR RPT0
MVLA 0x3
MVLR RPT1
MVLA 0x2
MVLR RPT2
MVLA 0x0
MVLR RPT3
MVLA 0x1
MVLR RPT4
MVLA 0x0
MVLR RPT5
LDSEC
```

Before Instruction

TVPR[0x1]= 0xXXXXX

After Instruction

TVPR[0x1]= 0x010234

Note : The LDSEC instruction will stop playing current voice first. Therefore, please always put the LDSCE instruction before the PLAY instruction while intending to start a new voice playing. Don't use LDSEC in ramp-up/ ramp-down.

LDPR

Function : Load ROM address to the DPR (data pointer) indexed by the CHNM register. The ROM address should be loaded in RPT firstly.

Operation : DPR [CHNO] ← RPT

Operand : None

Words : 1

Cycles : 3

Operative Flags: None

Flags Affected: None

Example :
```
MVLA    0x2
CHNO
MVLA 0x0
MVAM RPT0
MVLA    0x9
MVAM RPT1
MVLA    0x3
MVAM RPT2
MVLA    0xD
MVAM RPT3
MVLA    0x1
MVAM RPT4
MVLA    0x0
MVAM RPT5
LDPR
```

Before Instruction

DPR [0x2]= 0xXXXXX

After Instruction

DPR [0x2]= 0x01D390

LDPH

Function : Load PH value to the channel indexed by the CHNM register. The PH value should be loaded in RPT[11:0] firstly.

Operation : PH[CHNM] ← RPT[11:0]

Operand : None

Words : 1

Cycles : 3

Operative Flags: None

Flags Affected: None

Example :
```
MVLA    0x4
CHNO
MVLA 0x4
MVAM RPT0
MVLA    0x3
MVAM RPT1
MVLA    0x2
MVAM RPT2
LDPH
```

Before Instruction

PH [0x4]= 0xXXXX

After Instruction

PH[0x4]= 0x234

LDEN

Function : Load ENV value to channel indexed by the CHNM regiser. The ENV value should be load in RPT[7:0] firstly.

Operation : ENV[CHNM] ← RPT

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example :
```
MVLA   0x2
CHNO
MVLA   0xE
MVAM   RPT0
MVLA   0x2
MVAM   RPT1
LDEN
```
Before Instruction

ENV [0x2]= 0xXXXXX

After Instruction

ENV [0x2]= 0x2E

RDEN

Function : Read ENV value to the channel indexed by the CHNM register and the obtained content is put in RPT[7:0].

Operation : RPT[13:0] ← ENV[CHNM]

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example :
```
MVLA   0x2
CHNO
RBEN
```
Before Instruction

ENV [0x2]= 0x2E

After Instruction

RPT = 0x2E

RBVPR

Function : Read HVPR (voice pointer) content. The HVPR to read is indexed by the CHNM register and the obtained content is put in RPT.

Operation : RPT ← HVPR[CHNM]

Operand : None

Words : 1

Cycles : 3

Operative Flags: None

Flags Affected: None

Example :
```
MVLA   0x4
CHNO
RBVPR
```
Before Instruction

HVPR[0x4]= 0x010234

After Instruction

RPT = 0x010234

RBNVPR

Function : Read TVPR (voice pointer) content. The TVPR to read is indexed by the CHNM register and the obtained content is put in RPT.

Operation : RPT ← TVPR[CHNM]

Operand : None

Words : 1

Cycles : 3

Operative Flags: None

Flags Affected: None

Example :
```
MVLA   0x4
CHNO
RBNVPR
```
Before Instruction

TVPR[0x4]= 0x010234

After Instruction

RPT = 0x010234

RBPR

Function : Read DPR (data pointer) content. The DPR to read is indexed by the CHNM register and the obtained content is put in RPT.

Operation : RPT ← DPR[CHNM]

Operand : None

Words : 1

Cycles : 3

Operative Flags: None

Flags Affected: None

Example : MVLA 0x5
CHNO
RBDPR

Before Instruction

DPR[0x5]= 0x010234

After Instruction

RPT = 0x010234

RBCH

Function : Read CHNM register to ACC.

Operation : ACC ← CHNM

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : RBCH

Before Instruction

CHNM = 0x2

After Instruction

ACC = 0x2

CHNO

Function : Load ACC to CHNM register

Operation : CHNM ← ACC

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : MVLA 0x5
CHNO

Before Instruction

CHNM = 0xX

After Instruction

CHNM = 0x5

VOLX1

Function : Multiply the value of VOL register by 1.

Operation : VOL * 1

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : MVLA 0x2
MVAT VOL
VOLX1

Before Instruction

VOL= 0x2

After Instruction

VOL = 0x2

VOLX2

Function : Multiply the value of VOL register by 2.

Operation : VOL * 2

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : MVLA 0xF

MVAT VOL

VOLX2

Before Instruction

VOL = 0xF

After Instruction

VOL = 0x1E

RDN

Function : Read ROM data using the DPR (data pointer) indexed by the CHNM register.

Operation : ACC ← bit [3:0] of read data

ROD1 ← bit [7:4] of read data

ROD2 ← bit [11:8] of read data

Operand : None

Words : 1

Cycles : 3

Operative Flags: None

Flags Affected: None

Example : MVLA 0x1

CHNO

RDN

Before Instruction

data0 is 0x135

After Instruction

ACC=0x5, ROD1=0x3, ROD2=0x1, and DPR[0x1]= data0

RDNI

Function : Read ROM data using the DPR (data pointer) indexed by the CHNM register, and increase the DPR after data reading.

Operation : ACC ← bit [3:0] of read data

ROD1 ← bit [7:4] of read data

ROD2 ← bit [11:8] of read data

Operand : None

Words : 1

Cycles : 3

Operative Flags: None

Flags Affected: None

Example : MVLA 0x3

CHNO

RDNI

Before Instruction

data0 is 0x28B

After Instruction

ACC=0xB, ROD1=0x8, ROD2=0x2, and DPR[0x3]= data0+1

SNP

Function: Skip the next instruction if the channel (Head or Tail) indexed by the CHNM did not play.

Operation: Skip next if not play.

Operand: None

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: None

Flags Affected: None

Example: MVLA 0x1

CHNO

SNP

Inst1

Inst2

After Instruction

If CH1 play, \`Inst1’ is executed. If CH1 not play, \`Inst1’ is discarded, and \`Inst2’ is executed.

SP

Function: Skip the next instruction if the channel (Head or Tail) indexed by the CHNM register play.

Operation: Skip next if play.

Operand: None

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: None

Flags Affected: None

Example:
```
MVLA 0x1
CHNO
SP
Inst1
Inst2
```

After Instruction

If channel 1 not play, `Inst1’ is executed.

If channel 1 play, `Inst1’ is discarded, and `Inst2’ is executed.

STOP

Function : Stop voice (Head wave or Tail wave) playing on the channel indexed by the CHNM register.

Operation : None

Operand : None

Words : 1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :
```
MVLA 0x4
CHNO
STOP
```

After Instruction

Voice playing on channel 4 will be stopped, the channel data back to the middle.

SANP

Function: Skip the next instruction if no channel isn’t playing.

Operation: Skip next if All channel not play.

Operand: None

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: None

Flags Affected: None

Example:
```
SANP
Inst1
Inst2
```

After Instruction

If all channel play, `Inst1’ is executed.

If all channel not play, `Inst1’ is discarded, and `Inst2’ is executed.

SNHP

Function: Skip the next instruction if the channel (Head vice) indexed by the CHNM register did not play.

Operation: Skip next if Head waveform is not played.

Operand: None

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: None

Flags Affected: None

Example:
```
MVLA 0x1
CHNO
SNHP
Inst1
Inst2
```

After Instruction

If channel 1 (Head vice) play, `Inst1’ is executed.

If channel 1 (Head vice) not play, `Inst1’ is discarded, and `Inst2’ is executed.

4.2.4 Other Instructions

CALL a

Function : Call subroutine by direct address

Operation: STK ← PC+2

PC ← {BANK, a}

Operand : a: 16-bit program address to call, 0x0000 to 0xFFFF

Words : 2

Cycles : 2

Operative Flags: None

Flags Affected: None

Example : CALL a1

Before Instruction

PC=a0

After Instruction

PC={BANK, a1}, STK =a0+2

Note: PC[21:20] will not be changed.

BANK

Function :Set 3-bit value to Bank Register

Operation: BANK ← 3'bxxx

Operand : None

Words : 1

Cycles : 1

Example : BANK 0x2

After Instruction

BANK ← 0x2

JMP a

Function : Unconditional jump by direct address

Operation: PC ← {BANK, a}

Operand : a: 16-bit program address to jump, 0x0000 to 0xFFFF

Words : 2

Cycles : 2

Operative Flags: None

Flags Affected: None

Example : JMP a1

Before Instruction

PC=a0

After Instruction

PC={BANK, a1}

Note: PC[21:20] will not be changed.

MPG

Function : Set 3-bit value to SRAM page.

Operation:

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : MPG 0x3

Before Instruction

PAGE = 0x1

After Instruction

PAGE = 0x3

SEI

Function :Enable interrupt entrance

Operation:

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example :SEI

Before Instruction

Interrupt entrance is disable

After Instruction

Interrupt entrance is enable

RET

Function : Return from subroutine

Operation: PC ← STK

Operand : None

Words : 2

Cycles : 2

Example : RET

After Instruction

PC ← STK

CLI

Function : Disable interrupt entrance

Operation:

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : CLI

Before Instruction

Interrupt entrance is enable

After Instruction

Interrupt entrance is disable

IRET

Function : Return from interrupt routine

Operation: PC← STK

Operand : None

Words : 2

Cycles : 2

Operative Flags: None

Flags Affected: None

Example : IRET

After Instruction

PC ← STK, and

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

HALT

Function: Enter the halt (sleep) mode.

Operation: Stop system clock

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: HALT

After Instruction

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

CWDT0

Function: Clear Watch Dog Timer Step1.

Operation: Step1 for clear Watch Dog Timer

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CWDT0

Before Instruction

WDT counter = ???

After Instruction

WDT counter = ???

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, about 14.3 times.

CWDT1

Function: Clear Watch Dog Timer Step2.

Operation: Watch dog counter ← 0x0

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CWDT1

Before Instruction

WDT counter = ???

After Instruction

WDT counter = 0x0

Note : The CWDT0/CWDT1 instructions have to be executed step by step, othwise the watch dog timer won't be clear. The reason for two-step execution is to aviod unexpected system reset if IR decoded with unstable ROM data at low voltage.

RBDA

Function: Read Mixer data to RPT.

Operation: RPT[11:8] ← Mixer data

Operand: Mixer data : 4-bit

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: RBDA

Before Instruction

Mixer data =0x0234

After Instruction

RPT2=0x2

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.

LDPC

Function : Load program counter(PC) with RPT

Operation : PC ← RPT

Operand : None

Words : 1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :

```
MVLA 0x1
MVAM RPT0
MVLA 0x9
MVAM RPT1
MVLA 0x3
MVAM RPT2
MVLA 0xD
MVAM RPT3
MVLA 0x1
MVAM RPT4
MVLA 0x0
MVAM RPT5
LDPC
```

Before Instruction

PC = 0x?????

After Instruction

PC = 0x01D391

RDPC

Function : Read program counter (PC) to RPT

Operation : RPT[21:0] ← PC

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : RBPC

Before Instruction

RPT= 0x?????

PC = 0x012035

After Instruction

RPT = 0x012035

RDSPRH

Function : Read MSB 12-bit address for SPI flash. The address should be loaded in RPT firstly.

Operation : RPT[11:0] ← SPR [23:12]

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : RDSPRH

Before Instruction

RPT = 0x000

SPR = 0x432765

After Instruction

RPT = 0x432

LDSPRH

Function : Load MSB 12-bit address for SPI flash. The address should be loaded in RPT firstly.

Operation : SPR [23:12] ← RPT[11:0]

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example :

```
MVLA 0x2
MVAM RPT0
MVLA 0x3
MVAM RPT1
MVLA 0x4
MVAM RPT2
LDSPRH
```

Before Instruction

SPR = 0x000000

After Instruction

SPR = 0x432000

LDSPRL

Function : Load LSB 12-bit address for SPI flash. The address should be loaded in RPT firstly.

Operation : SPR [11:0] ← RPT[11:0]

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example :

```
MVLA 0x5
MVAM RPT0
MVLA 0x6
MVAM RPT1
MVLA 0x7
MVAM RPT2
LDSPRH
```

Before Instruction

SPR = 0x000000

After Instruction

SPR = 0x000765

RDSPRL

Function : Read LSB 12-bit address for SPI flash. The address should be loaded in RPT firstly.

Operation : RPT[11:0] ← SPR [11:0]

Operand : None

Words : 1

Cycles : 1

Operative Flags: None

Flags Affected: None

Example : RDSPRL

Before Instruction

RPT = 0x000

SPR = 0x432765

After Instruction

RPT = 0x765