九齊科技股份有限公司

Nyquest Technology Co., Ltd.

USER MANUAL

NY7 Series

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

Version 1.5

May 7, 2018

Revision History

Version	Date	Description	Modified Page
1.0	2013/06/28	Formal release.	--
1.1	2013/08/16	1. Add 4 mask options for Push-Pull analog volume adjustment. 2. Revise the description of *Chapter 2.10.1 Speech Synthesis*. 3. Provide the example code of ramp-up/ramp-down for DAC output. 4. Flag Z will not be affected by instruction MVLA. 5. Instruction LDPR, RBDPR, RDN and RDNI belong to category of “Other Instructions”.	7, 25 23 51 56 57
1.2	2014/01/02	Modify Data Transfer Instruction.	16, 63
1.3	2014/08/27	1. Modify the default value of weak input pull-high resistor. 2. Modify RRC and RLC command.	10, 21 66
1.4	2015/07/30	Modify Instruction “RET” is 2-word.	58, 81
1.5	2018/05/07	Add Instruction “MPG”.	58, 59, 83

Table of Contents

Chapter 1. Introduction

1.1 General Description

The NY7 series IC is a powerful 4-bit micro-controller based sound processor. There are 8 channels that are configured as speech or MIDI, and all of these 8 channels or part of them can be played with speech or MIDI simultaneously. By using the high fidelity 6-bit ADPCM speech/ MIDI timbre synthesis algorithm with up to 44.1KHz sample rate, NY7 can produce near-CD quality voices. As NY7 is specially designated for MIDI synthesis application, it provides Attack-Decay-Sustain-Release method (ADSR) with 256-level envelope for Patch (instrument) synthesis. NY7 can precisely synthesize any tone frequency of MIDI with +/- 0.5% accurate internal oscillation and automatic Tone-Calibration. Therefore NY7 melody quality is very close to real instrument.

Moreover, NY7 is equipped with new Nyquest's developed high-quality noise filtering algorithm of 128KHz 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. NY7A provides two kinds of audio outputs with fine resolution, one is 13-bit current-type D/A converter (DAC) and the other is 12-bit Pulse-Width-Modulation (PWM). NY7B/NY7C provide two kinds of audio outputs with fine resolution, one is 13-bit current-type D/A converter (DAC) and the other is 13-bit Push-Pull amplifier (PP). Therefore NY7 speech/melody quality is the best choice among all solutions.

The RISC MCU architecture is very easy to program and control, various applications can be easily implemented. There are 74 instructions, and most of them are executed in single cycle. Besides normal operation mode, NY7 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 74 instructions.
- NY7A have 9 items. 192K x 12-bit ROM is the maximum size.
- NY7B have 10 items. 256K x 12-bit ROM is the maximum size.
- NY7C have 19 items. 1536K x 12-bit ROM is the maximum size.
- 448x4-bit RAM, divided into 2 pages.
- Up to 4MHz system clock for instruction execution.
- Slow mode to operate with low power consumption (+/-3% 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.
- One interrupt entrance for multiple interrupt sources with an independent stack.
- 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 current and normal sink current, large sink current or constant sink current.
- Shared pins to provide IR carrier and external reset feature:

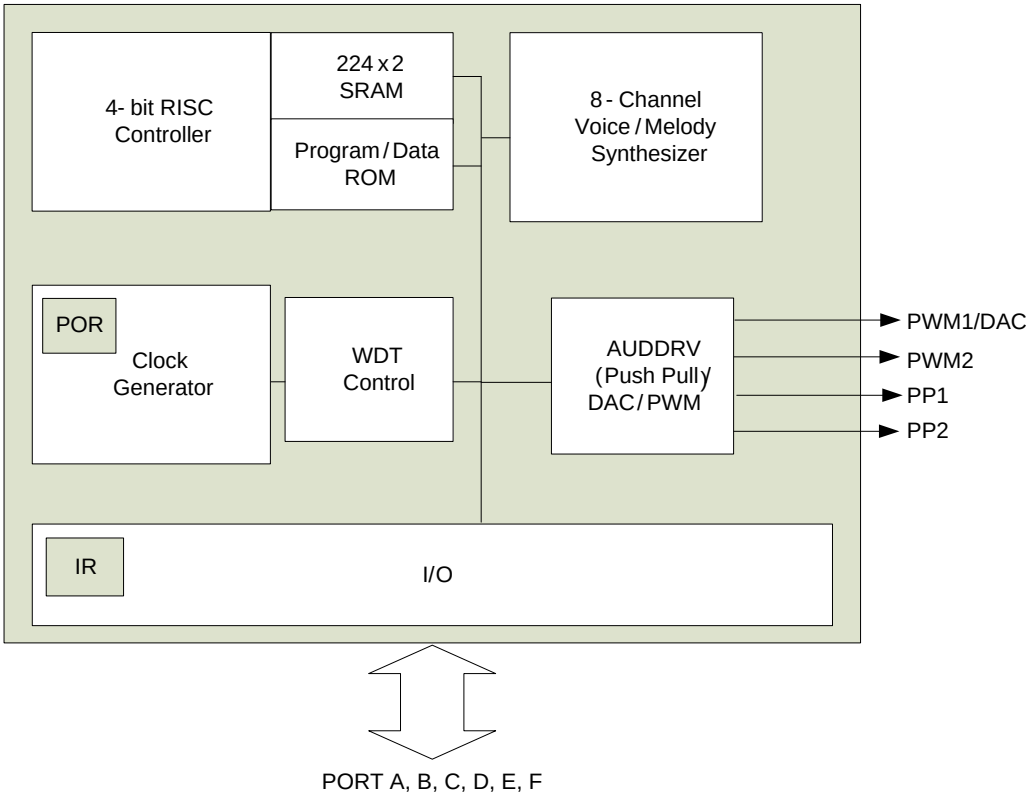
Shared Pin Function	NY7AxxxA	NY7BxxxA	NY7CxxxA
IR carrier (IR)	PB2/IR	PD2/IR	PF2/IR
External reset (Reset)	PB3/Reset	PD3/Reset	PF3/Reset

- Selection of IR carrier frequency and data high/low IR output is supported.
- Maximum of 8 channels can be played simultaneously, each channel can be arbitrarily assigned as speech or MIDI channel.
- New high fidelity 6-bit ADPCM speech synthesis algorithm and ADSR with 256-step envelope for MIDI synthesis.
- Patented noise filtering algorithm with 128KHz 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 13-bit D/A converter, 12-bit PWM driver or 13-bit Push-Pull amplifier audio output with 4-level mask-option analog volume of 100%, 83%, 66% and 50%. *(Push-Pull output power Pout=1.3W@V_{DD}=5V, THD+N=10%, F=1kHz, R_L=4Ω.)*
- 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	Push-Pull
NY7A004A	4.5	16K x 12	16K x 12	8	12-bit	13-bit	-
NY7A007A	7.2	24K x 12	24K x 12	8	12-bit	13-bit	-
NY7A010A	9.9	32K x 12	32K x 12	8	12-bit	13-bit	-
NY7A016A	15.4	48K x 12	48K x 12	8	12-bit	13-bit	-
NY7A021A	20.8	64K x 12	64K x 12	8	12-bit	13-bit	-
NY7A032A	31.8	96K x 12	64K x 12	8	12-bit	13-bit	-
NY7A043A	42.7	128K x 12	64K x 12	8	12-bit	13-bit	-
NY7A054A	53.6	160K x 12	64K x 12	8	12-bit	13-bit	-
NY7A065A	64.5	192K x 12	64K x 12	8	12-bit	13-bit	-
NY7B007A	7.2	24K x 12	24K x 12	16	-	13-bit	13-bit
NY7B010A	9.9	32K x 12	32K x 12	16	-	13-bit	13-bit
NY7B016A	15.4	48K x 12	48K x 12	16	-	13-bit	13-bit
NY7B021A	20.8	64K x 12	64K x 12	16	-	13-bit	13-bit
NY7B032A	31.8	96K x 12	64K x 12	16	-	13-bit	13-bit
NY7B043A	42.7	128K x 12	64K x 12	16	-	13-bit	13-bit
NY7B054A	53.6	160K x 12	64K x 12	16	-	13-bit	13-bit
NY7B065A	64.5	192K x 12	64K x 12	16	-	13-bit	13-bit
NY7B076A	75.5	224K x 12	64K x 12	16	-	13-bit	13-bit
NY7B087A	86.4	256K x 12	64K x 12	16	-	13-bit	13-bit
NY7C010A	9.9	32K x 12	32K x 12	24	-	13-bit	13-bit
NY7C016A	15.4	48K x 12	48K x 12	24	-	13-bit	13-bit
NY7C021A	20.8	64K x 12	64K x 12	24	-	13-bit	13-bit
NY7C032A	31.8	96K x 12	64K x 12	24	-	13-bit	13-bit
NY7C043A	42.7	128K x 12	64K x 12	24	-	13-bit	13-bit
NY7C054A	53.6	160K x 12	64K x 12	24	-	13-bit	13-bit
NY7C065A	64.5	192K x 12	64K x 12	24	-	13-bit	13-bit
NY7C076A	75.5	224K x 12	64K x 12	24	-	13-bit	13-bit
NY7C087A	86.4	256K x 12	64K x 12	24	-	13-bit	13-bit
NY7C110A	111.0	328K x 12	64K x 12	24	-	13-bit	13-bit
NY7C130A	130.1	384K x 12	64K x 12	24	-	13-bit	13-bit
NY7C150A	151.9	448K x 12	64K x 12	24	-	13-bit	13-bit
NY7C170A	173.8	512K x 12	64K x 12	24	-	13-bit	13-bit
NY7C220A	222.9	656K x 12	64K x 12	24	-	13-bit	13-bit
NY7C260A	261.1	768K x 12	64K x 12	24	-	13-bit	13-bit
NY7C305A	304.8	896K x 12	64K x 12	24	-	13-bit	13-bit
NY7C345A	348.5	1024K x 12	64K x 12	24	-	13-bit	13-bit
NY7C450A	457.8	1344K x 12	64K x 12	24	-	13-bit	13-bit
NY7C520A	523.3	1536K x 12	64K x 12	24	-	13-bit	13-bit

1.4 Block Diagram



1.5 Pad Description

Pin	ATTR.	Description
VDD#	Power	Positive power
GND#	Power	Negative power
PWM1/DAC	O	PWM1 output or DAC output
PWM2	O	PWM2 output
PP1	O	Push-Pull output
PP2	O	Push-Pull output
PA0~3	I/O	Bit 0~3 for Port A
PB0~3	I/O	Bit 0~3 for Port B
PC0~3	I/O	Bit 0~3 for Port C
PD0~3	I/O	Bit 0~3 for Port D
PE0~3	I/O	Bit 0~3 for Port E
PF0~3	I/O	Bit 0~3 for Port F

* NY7A: PA0~PB3, there is no Push-Pull output (PP1 & PP2).

* NY7B: PA0~PD3, there is no PWM output (PWM1 & PWM2).

* NY7C: PA0~PF3, there is no PWM output (PWM1 & PWM2).

1.6 Electrical Characteristics

The following table lists the electrical characteristics of the NY7 EV chip. All the product's properties must refer to each part's datasheet.

1.6.1 Absolute Maximum Rating

Symbol	Parameter	Rated Value	Unit
V_{DD} - V_{SS}	Supply voltage	-0.5 ~ +6.0	V
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

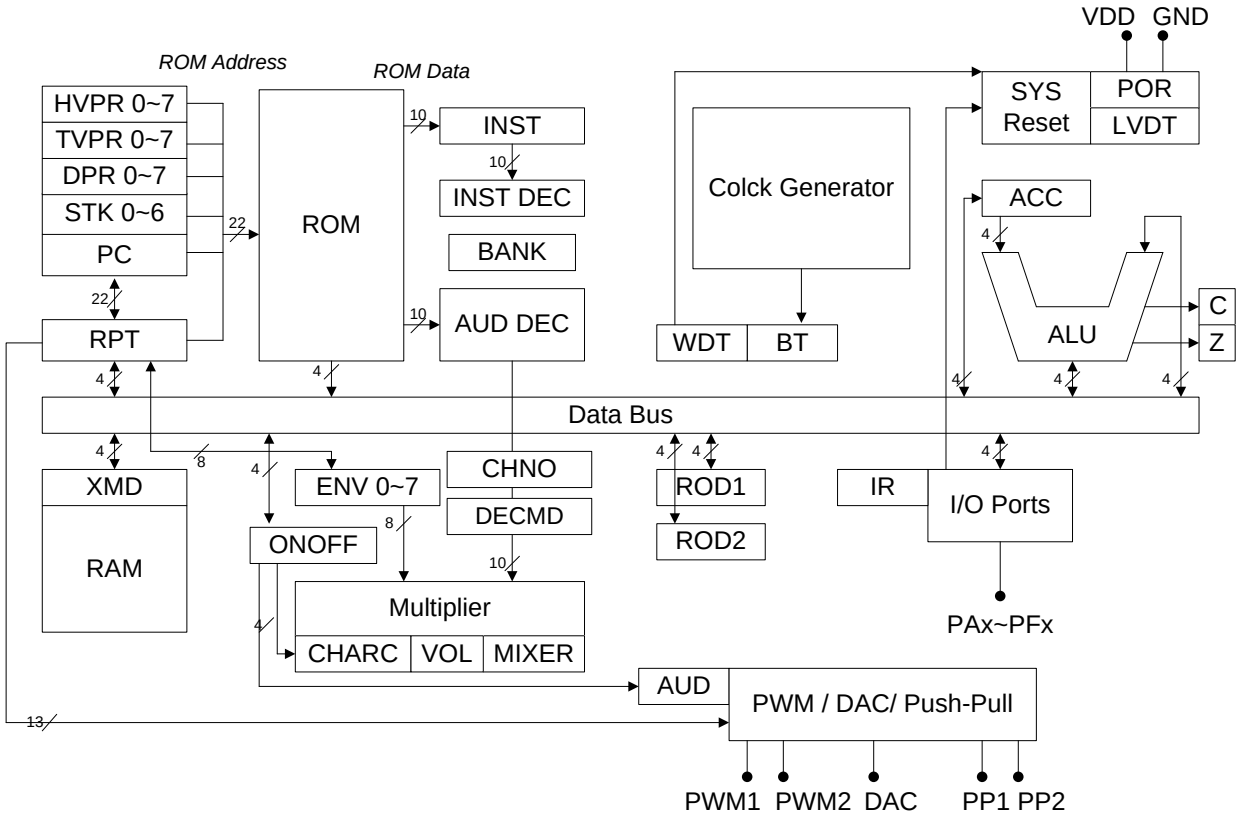
1.6.2 DC Characteristics

Symbol	Parameter		VDD	Min.	Typ.	Max.	Unit	Condition
V_{DD}	Operating voltage		--	2.0	3.0	5.5	V	4MHz.
I_{SB}	Supply current	Halt mode	3.0			1	uA	Sleep, no load.
			4.5			1		
I_{SL}		Slow mode	3.0		300		uA	Slow, no load.
			4.5		600			
I_{OP}		Normal mode	3.0		5		mA	4MHz, no load.
			4.5		8			
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		11		mA	V_{OL}=1.0V
			4.5		17			
	Output low current (Large current)		3.0		22		mA	
			4.5		33			
	Output low current (Constant current)		3.0		20		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			
I_{PP}	Push-Pull output current		3.0		180		mA	
			4.5		270			
ΔF/F	Frequency deviation by voltage drop		3.0		0.5		%	Fosc(3.0v)-Fosc(2.4v) / Fosc(3v)
			4.5		-0.5			Fosc(4.5v)-Fosc(3.0v) / Fosc(4.5v)
ΔF/F	Frequency lot deviation		3.0	-0.5		0.5	%	Fmax(3.0v)-Fmin(3.0v) / Fmax(3.0v)
Fosc	Oscillation Frequency		--	3.6	4	4.1	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
STK0~6	7-level interrupt dedicated stack	
PC	Program counter	
HVPR0~7	Voice Head pointer according to SFR(CHNO).	
TVPR0~7	Voice Tail pointer according to SFR(CHNO).	
DPR0~7	Data pointer	
RPT	Multi-function register pointer	SFR[0x0~0x5]
ENV0~7	8-bit Envelope of SFR (CHNO)	SFR[0x6~7]
ROD1	ROM[7:4] data access register	SFR[0x8]
ROD2	ROM[11:8] data access register	SFR[0x9]
CHARC	Mix Channel#, Output choice	SFR[0xA]
AUD	Audio output Data	SFR[0xB]
INT	Interrupt generator	SFR[0xC]
DECMD	PCM / ADPCM control register	SFR[0xD]
ONOFF	Interrupt and audio control register	SFR[0xE]
VOL	Digital volume control register	SFR[0xF]

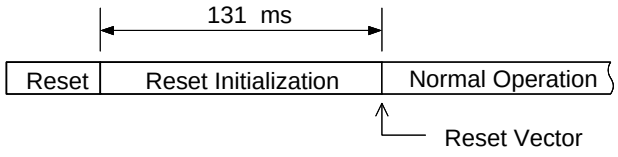
Name	Function	Address
BANK	Program Bank Register	SFR[0x10]
XMD	Indexed RAM data access register	SFR[0x11]
CHNO	Active channel select	SFR[0x12]
RAM	448 nibbles RAM	
ROM	Program & data ROM	
Multiplier	Hardware multiplier for MIDI	
Mixer	Channels audio data mixer	
PWM / DAC / PP	PWM, D/A or PP audio output	
INST	Instruction register	
INST DEC	Instruction decoder	
AUD DEC	Audio decoder	
Clock Generator	Ring oscillator clock generator	
WDT	Watch-dog timer and reset generator	
BT	System base timer	
SYS Reset	System reset generator	
POR	Power reset generator	
ACC	4-bit accumulator	
ALU	4-bit arithmetic logic unit	
C	Carry flag for arithmetic	
Z	Zero flag for arithmetic	
IR	Infrared transmit block	
I/O Ports	I/O port register	SFR[0x14~0x1F]

*SFR[] : System Function Register

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.

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 NY7 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 NY7 IC is approximate 1.9V.

2.3.3 Watch-Dog Timer Reset (WDTR)

To recover from program function, the NY7 IC supports an embedded watch-dog timer reset. The WDTR function always works with the program executing. Users have clear the WDT periodically to prevent from timing up with a reset generation. Typically, the minimum time-up period of the WDT is about 28ms and users can clear WDT through instruction CWDT.

2.3.4 I/O Port External Reset

The PB3/Reset (NY7A), PD3/Reset (NY7B) or PF3/Reset (NY7C) I/O pin of the NY7 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 disable. When the reset pin falls to the ground level, it generates an external reset.

2.4 Address Pointer

The NY7 micro-controller contains a program counter (PC), 8 data pointers, 7 interrupt dedicated stack (STK), a multi-function register pointer (RPT) and 8 head pointers (HVPR0~7) and 8 tail pointers (TVPR0~7) for channel 0~7. 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.

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 NY7A/NY7B and 21-bit wide for NY7C. The PC starts from the reset vector (address 0x000400) 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 0x000000.

Inst./Event	Function
JNC Addr	Jump to {BANK, Addr} if Carry = 0
JC Addr	Jump to {BANK, Addr} if Carry = 1
JNZ Addr	Jump to {BANK, Addr} if Zero = 0
JZ Addr	Jump to {BANK, Addr} if Zero = 1

Inst./Event	Function
JB b, Addr	Jump to {BANK, Addr} if A[b] = 1
JMP Addr	Jump to {BANK, Addr}.
CALL Addr	Push the PC+2 to the STK and load {BANK, Addr} to PC.
RJMP	Load RPT to PC, so users can execute a long jump.
RCALL	Push the PC+2 to the STK and load RPT to PC.
Interrupt	Push PC+2 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.

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

2.4.2 Stack (STK)

Seven level hardware push/pop stacks dedicated to the interrupt (CALL / RCALL) is available. When an interrupt takes apart, the system pushes the PC+2 (next instruction) to the STK automatically. STK occupy SFR from 0x8 to 0xE and STK is used from 0xE. 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 NY7A and 21 bits for NY7B/NY7C.

2.4.3 Multi-function Register Pointer (RPT)

As implied in the name, RPT are multi-function registers. There are at must six RPT that are RPT0, RPT1, RPT2, RPT3, RPT4 and RPT5. RPT0~RPT4 are 4-bit wide and RPT5 is 2-bit wide, i.e. RPT5[1:0]. The RPT max width are 18 bits for NY7A and 21 bits for NY7B/7C. Users have to operate RPT in coordination with instructions below.

Inst./Event	Function
RJUMP	Load RPT to PC, so users can execute a long jump.
RCALL	Push the PC+2 to the STK and load RPT to PC.
PLAY	Play RPT to HVPR, according to SFR(CHNO).
LDSEC	Load RPT to TVPR, according to SFR(CHNO).
LDPR	Load RPT to DPR/STK, according to SFR(CHNO).
LDPH	Load RPT[13:0] to PH, according to SFR(CHNO).
RBVPR	Read HVPR/TVPR to RPT, according to SFR(CHNO).
RBDPR	Read DPR/STK to RPT, according to SFR(CHNO).
RBDA	Read DAC data to RPT[12:0]
LDDA	Load RPT[12:0] to DAC reg.
XMD	Use RPT[7:0] as address to access indexed RAM data.

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

Because NY7 is an 8-channel sound processor, 8 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 CHNO register. When LDSEC is executed, the system loads RPT to TVPR of the channel that assigned by the CHNO register. When PLAYI is executed, the system loads immediately address to HVPR of the channel that assigned by the CHNO register. When LDSECI is executed, the system loads immediately address to TVPR of the channel that assigned by the CHNO register. So users have to move the start address of the speech or MIDI data to RPT first. Besides, users can read HVPR/TVPR back by RBVPR instruction, because RBVPR moves HVPR/TVPR of the channel that assigned by the CHNO register to RPT. The HVPR/TVPR max width is 18 bits for NY7A and 21 bits for NY7B/NY7C.

2.4.5 Data Pointer (DPR)

8 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 CHNO register. When LDPRI is executed, the system loads immediately address to DPR of the channel. The read back ROM data is placed on ROD2, ROD1, ACC. ACC is the 4 LSB of ROM data. Besides, users can read DPR back by RBDPR instruction, because RBDPR moves DPR of the channel that assigned by the CHNO register to RPT. The DPR max width is 18 bits for NY7A and 21 bits for NY7B/NY7C. Unused STK can be used as DPR.

2.5 Arithmetic Logic Unit (ALU)

The NY7 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, SFR register, or literal constant data.

2.5.1 ALU Instruction Summary

2.5.1.1 Logic Instruction

Instruction	Function	Flag Influenced
XORA m1	A ← M[m1] ⊕ A	Z
ANDA m1	A ← M[m1] & A	Z
ORA m1	A ← M[m1] | A	Z
RRM m1	Right Rotate M[m1] with C	C, Z
RLM m1	Left Rotate M[m1] with C	C, Z
XORL L	A ← L ⊕ A	Z
ANDL L	A ← L & A	Z

Instruction	Function	Flag Influenced
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[m1] : 4-bit RAM or SFR data at memory address m1, 0x00≦ m1 ≦0xFF.

2.5.1.2 Arithmetic Instruction

Instruction	Function	Flag Influenced
INCM m1	M[m1] ← M[m1] + 1	C, Z
DECM m1	M[m1] ← M[m1] - 1	C, Z
ADDA m1	{C, A} ← A + M[m1] + C	C, Z
ADDL L	A ← A + L + C	C, Z
SUBA m	{C,A} = A - M - (~B)	C, Z
SUBL L	{C,A} = A - L - (~B)	C, Z
INCA	A ← A + 1	C, Z
DECA	A ← A - 1	C, Z
CPLA	A ← 0 - A	

M[m1] : 4-bit RAM or SFR data at memory address m1, 0x00≦ m1 ≦0xFF.

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

2.5.1.3 Data Transfer Instruction

Instruction	Function	Flag Influenced
MVAM m1	M[m1] ← A	
MVMA m1	A ← M[m1]	Z
MVRM m2, r	M[m2] ← R[r]	
MVMR m2, r	R[r] ← M[m2]	
MVLR L, r	R[r] ← L	
MVLA L	A ← L	
BCLR m2, b	Clear M[m2][b]	
BSET m2, b	Set M[m2][b]	
SETC	C ← 1	C
CLRC	C ← 0	C

M[m1] : 4-bit RAM or SFR data at memory address m1, 0x00≦ m1 ≦0xFF.

M[m2] : 4-bit RAM at memory address m2, 0x00≦ m2 ≦0x1F, means address 0x20~0x3F.

R[r] : 4-bit SFR data at register address r, 0x0≦ r ≦0x7.

The width of the SFR address `r' of MVRM, MVMR, and MVLR command is 3-bit, and the MSB of the memory register is forced to be 0. So users can only use the three commands to handle RPT0~5 and

ENVL/ENVH. The width of the RAM or memory register address `m' of MVRM, and MVMR command is 5-bit, and the MSB 3-bit of the address is forced to be 0x1. Users can only use the two instructions (MVRM, MVMR) to handle RAM or memory register of address 0x20~0x3F, but the RAM page is still working.

2.5.1.4 Conditional Branch Instruction

Instruction	Function	Flag Influenced
JNC Addr	Jump to {BANK, Addr} if Carry = 0	
JC Addr	Jump to {BANK, Addr} if Carry = 1	
JNZ Addr	Jump to {BANK, Addr} if Zero = 0	
JZ Addr	Jump to {BANK, Addr} if Zero = 1	
JB b, Addr	Jump to {BANK, Addr} if A[b] = 1	
SAGT L	Skip when A > L	
SALT L	Skip when A < L	
SANE L	Skip when A != L	
SBZ b	Skip when A[b] = 0	
SBNZ b	Skip when A[b] = 1	
SNHP	Skip when head Play = 0, according to SFR(CHNO).	
SHP	Skip when head Play = 1, according to SFR(CHNO).	
SNP	Skip when Play = 0, according to SFR(CHNO).	
SP	Skip when Play = 1, according to SFR(CHNO).	
SANP	Skip when ALL 8 channels Play = 0	

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 logicaloperation

& : 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

M[m1] : 4-bit RAM or SFR data at memory address m1, 0x00≦ m1 ≦0xFF.

M[m2] : 4-bit RAM at memory address m2, 0x00≦ m2 ≦0x1F, means address 0x20~0x3F.

R[r] : 4-bit SFR data at register address r, 0x0≦ r ≦0x7.

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 1.5M words ROM, 448 nibbles of RAM and 32 nibbles of dedicated System Function Register (SFR).

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, NY7 will start program execution from address 0x400.

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 4-bit BANK register (address $10 of System Function Register), the total program size is 1M 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.

If destination address is beyond 1M words, instructions RJMP and RCALL associated with RPT[20:0] can be used and BANK register is ignored. Instruction LDPRI can handle 20-bit length address of ROM.

Address	ROM
0x000000	Interrupt Vector
0x00000F	
0x000010	Reserved
0x0003FF	
0x000400	Program & Data Page 0
0x00FFFF	
0x010000	Program & Data Page 1 ~ 23
0x17FFFF	

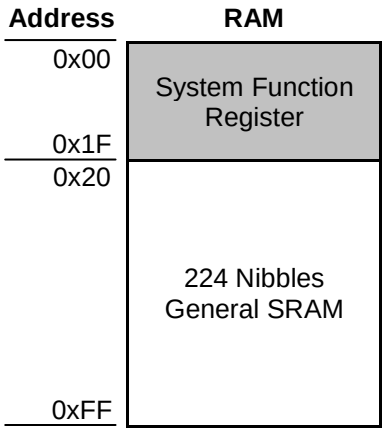
2.6.2 RAM

There are two pages of RAM, each page of RAM contains 224 nibbles. It's total 448 nibbles. The page of RAM defined by instruction (PAGE0, PAGE1), and its initial is PAGE0. System Function Registers will occupy address space from 0x00 to 0x1F. Moreover, this address space of PAGE0 and PAGE1 are

mapped to the same System Function Registers. As consequence, the address space of PAGE0 and PAGE1 RAM which can be used by programmer is 0x20~0xFF.

The address space from 0x20 to 0x3F of PAGE0 and PAGE1 can be used with four special instructions MVRM, MVMR, BSET and BCLR. These instructions can access this range of memory space in single instruction cycle.

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



(Page 0 & Page 1)

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 NY7. 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 NY7. Users can refer *Chapter 3.6 I/O Ports Register* for details.

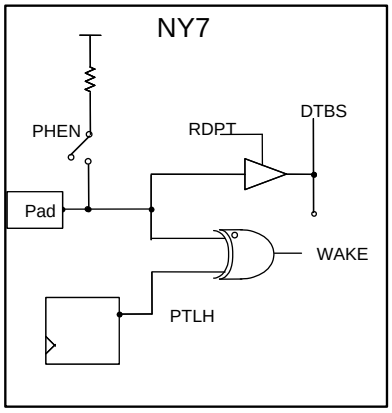
The pull-high resister of all the I/O pins has two kinds of option: weak and strong. The weak one is about 1.2MΩ@3V for normal application and the strong one is about 100KΩ@3V usually for key matrix function. When users decide this option, the same strength of pull-high resister will be applied to every 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 port can also be optioned as an external reset pin or an infrared (IR) output pin. 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 frequency can be determined by a 5-bit option. 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 describe above.

2.7.1 Pull-High Input Mode



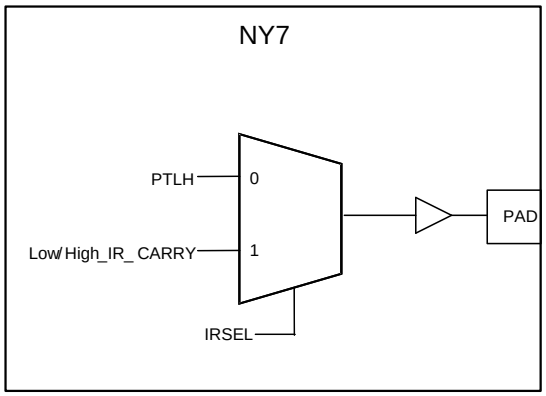
Pull-high Input Mode Configuration

Data of PA~PF, which are used as input mode, can be read in by MVMA. If the pads are not connected, a pull-high resistor will help to pull the pad toward supply voltage. All input or I/O pins can be used to wake-up the IC. In input mode, the system will be waked-up by comparing PTLH with pad voltage level. 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 NY7 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.

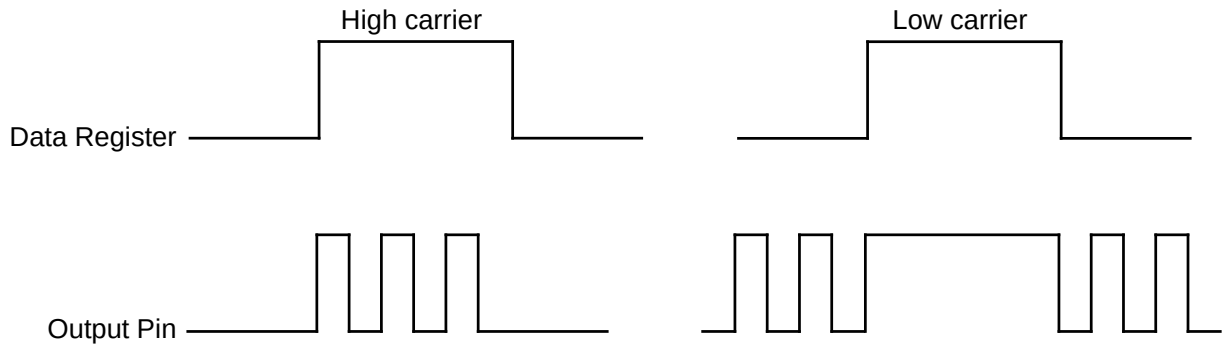
Category	I/O pin	Option	Default Value
NY7A	PAx PB0 ~ PB1	Normal I/O.	Enable
		Input with pull-high, floating or register-control pull-high.	Pull-high
		Normal, large or constant sink current output.	Normal sink
	PB2/IR	IR carrier output or Normal I/O.	User Selection
		Input with pull-high, floating or register-control pull-high.	Pull-high
		Normal, large or constant sink current output.	Normal sink
	PB3/Reset	Reset input or Normal I/O.	User Selection
		Input with pull-high, floating or register-control pull-high.	Pull-high
		Normal, large or constant sink current output.	Normal sink
	All I/O	Weak or strong input pull-high resistor.	Weak (1.2MΩ@3V)
NY7B	PAx PBx PCx PD0 ~ PD1	Normal I/O.	Enable
		Input with pull-high, floating or register-control pull-high.	Pull-high
		Normal, large or constant sink current output.	Normal sink
	PD2/IR	IR carrier output or Normal I/O.	User Selection
		Input with pull-high, floating or register-control pull-high.	Pull-high
		Normal, large or constant sink current output.	Normal sink
	PD3/Reset	Reset input or Normal I/O.	User Selection
		Input with pull-high, floating or register-control pull-high.	Pull-high
		Normal, large or constant sink current output.	Normal sink
	All I/O	Weak or strong input pull-high resistor.	Weak (1.2MΩ@3V)
NY7C	PAx, PBx PCx, PDx PEx PF1 ~ PF2	Normal I/O.	Enable
		Input with pull-high, floating or register-control pull-high.	Pull-high
		Normal, large or constant sink current output.	Normal sink
	PF2/IR	IR carrier output or Normal I/O.	User Selection
		Input with pull-high, floating or register-control pull-high.	Pull-high
		Normal, large or constant sink current output.	Normal sink
	PF3/Reset	Reset input or Normal I/O.	User Selection
		Input with pull-high, floating or register-control pull-high.	Pull-high
		Normal, large or constant sink current output.	Normal sink
	All I/O	Weak or strong input pull-high resistor.	Weak (1.2MΩ@3V)

2.8 Infrared Transmitter

The NY7 series provides an infrared transmitter block, which is used to send infrared signal. Users can use PB2, PD2 or PF2 as an IR output. 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 31.25KHz and 58.82KHz.

Category	Option	Description
IR	IR frequency	31.25 ~ 58.82KHz
	IR low/high carrier	Low
		High



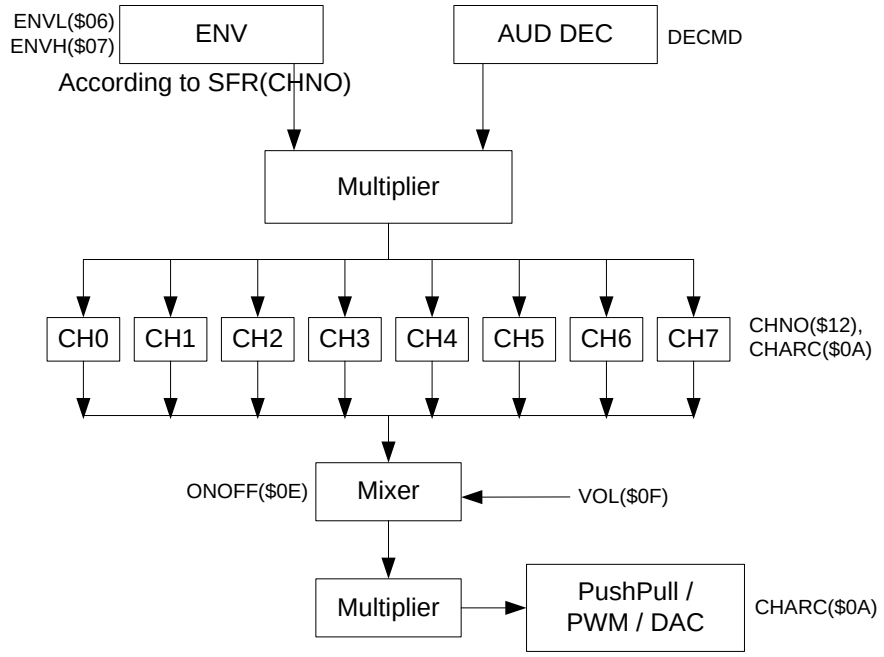
2.9 Interrupt Generator

There is one hardware interrupt in NY7. The interrupt event is a fixed interval, which is derived from system base timer (BT). There is a system base timer in the NY7 IC, which functions as long as the IC isn't in the halt mode. NY7 provides 4 kinds of fixed intervals from the system base timer for interrupt source: 0.064ms, 0.128ms, 0.256ms and 1.024ms. In Slow mode, there are 4 kinds of base timer interval for polling: 1.024ms, 2.048ms, 4.096ms and 16.384ms. When using interrupts, user must select interrupt source first and then turn on. However, the time interval from BT is enabled to first occurrence of interrupt may be not as accurate as specified due to NY7 characteristic.

As an interrupt occurs, NY7 stores the accumulator (ACC), carry flag (C), zero flag (Z) and RAM page (PG) automatically. PG is controlled by the command (PAGE0, PAGE1). Then NY7 move PC+2 to STK, and jump to the interrupt vector (0x000000). An interrupt routine finishes with an IRET instruction. The IC draws back ACC, C, Z and PG back, and moves STK to PC back to jump back the main program. The interrupt event of BT will be automatically cleared after entering the interrupt routine.

2.10 Audio Synthesizer Structure

NY7 provides a built-in speech/MIDI synthesizer. The synthesizer consists of eight channels for voice or MIDI synthesis. The allowable simultaneous synthesis channel can be 2, 4, 6 and 8. The block diagram of the synthesizer unit is shown in figure below.



2.10.1 Speech Synthesis

NY7 supports 10-bit PCM and encoded 6-bit ADPCM speech data. The PCM voice has higher quality, but it occupies double ROM space than the ADPCM one. By cooperating with embedded noise filter of 128KHz 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.

2.10.2 MIDI Synthesis

There are three combinations to form a patch in NY7. 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, user 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.

2.10.3 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:

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

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

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
1	2
2	2
4	4
6	4 for channel 2, 3 8 for channel 0, 1, 4, 5
8	8

Active MIDI Channel	CH Value
2	2
4	4
6	4 for channel 2, 3 8 for channel 0, 1, 4, 5
8	8

F_{INST}: instruction cycle. It is 4,000,000 or 2,000,000 by mask option.

Factor is 1 if noise filter is disabled, and 2 if noise filter is enabled.

F_{NOTE}: frequency of the note which is being played.

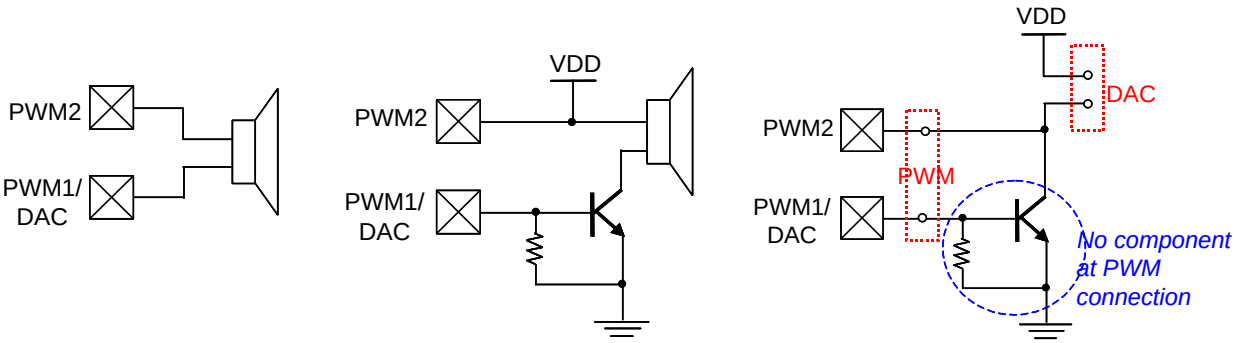
F_{PATCH}: frequency of key note on which patch waveform is based.

2.10.4 Audio Output

Before using the audio output, user can choose one of the 13-bit Push-Pull (PP), 13-bit DAC or 12-bit PWM as the audio output for NY7. NY7A provides PWM and DAC audio output and NY7B/NY7C provide Push-Pull (PP) and DAC audio output. If user selects Push-Pull, user has to enable the Push-Pull by clearing bit 2 and 3 as zero in CHARC($0A) first. It provides hardware ramp-up and ramp-up time is about 100us. Moreover, there are 4 mask options to select Push-Pull gain for volume adjustment. These 4 mask options correspond to 100%, 83%, 66% or 50% of maximum analog volume output. 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 NY7A, 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.

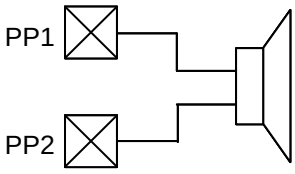
PWM2 Pad	Audio Output Initialization
Speaker (Floating)	PWM
VDD	DAC



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.

As to connection for Push-Pull output, it just needs to connect PP1 to one terminal of speaker and PP2 to another terminal of speaker.



Push-Pull Output Connection

2.10.5 Envelope Control

During speech synthesis or melody synthesis, there is one set of 8-bit envelope register (ENVH and ENVL), which can store the envelope information. Therefore NY7 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 NY7 is a 8-channel synthesizer but there is only one set of envelope register physically, user has to write value to SFR CHNO to select a specific channel and the 256-level envelope data will be updated to this selected channel. Moreover, as NY7 micro-controller is 4-bit but envelope information is 8-bit, the envelope data of selected channel will not be updated until SFR ENVH is written. User can refer Chapter 3.8.6 for details.

2.10.6 Volume Control

NY7 supports 16-step digital volume control by the VOL register. Default value of VOL register is 0x2. In order to have suitable volume, VOL=0x2 is recommended for 8-ch speech/MIDI synthesis. 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.

Chapter 3. System Control

3.1 Introduction of System Function Register

The combination of RPT0~5 are multi-function register pointers. The ENVL/H, CHARC, AUD, CHNO, DECMD, ONOFF and VOL are audio control related registers. INT register is used to control or access the system base timer (BT) and interrupt. The BANK register is used to switch the program bank when targeting branch address is located beyond current program bank. The XMD is RAM indirect access registers. The ROD1 and ROD2 registers are used to read the ROM data. The Px and PxIO are I/O ports registers, here x could A, B, C, D, E or F. As PA, PB, PC, PD, PE and PF are bi-directional I/O ports, SFR PAIO, PBIO, PCIO, PDIO, PEIO and PFIO are used to determine the direction of each I/O pin.

Addr	Name	R/W	Bit	Data	Description	Default
$00	RPT0	R/W	[3:0]	0/1	Multi-function register pointer [3:0]	xxxx
$01	RPT1	R/W	[3:0]	0/1	Multi-function register pointer [7:4]	xxxx
$02	RPT2	R/W	[3:0]	0/1	Multi-function register pointer [11:8]	xxxx
$03	RPT3	R/W	[3:0]	0/1	Multi-function register pointer [15:12]	xxxx
$04	RPT4	R/W	[3:0]	0/1	Multi-function register pointer [19:16]	xxxx
$05	RPT5	R/W	[0]	0/1	Multi-function register pointer [20]	xx
$06	ENVL	R/W	[3:0]	0/1	Envelope [3:0]	xxxx
$07	ENVH	R/W	[3:0]	0/1	Envelope [7:4]	xxxx
$08	ROD1	R	[3:0]	0/1	ROM Data Latch[7:4]	xxxx
$09	ROD2	R	[3:0]	0/1	ROM Data Latch[11:8]	xxxx
$0A	CHARC	R/W	[1:0]	00	8 channels	8 CH
				01	6 channels	
				10	4 channels	
				11	2 channels	
			[3:2]	0x	Push-Pull (PP)	PP
				10	DAC	
				11	PWM	
$0B	AUD	R	[3:0]	0/1	DAC[12:9]	xxxx
$0C	INT	R	[0]	0/1	BT = 0.064 ms	
			[1]	0/1	BT = 0.128 ms	
			[2]	0/1	BT = 0.256 ms	
			[3]	0/1	BT = 1.024 ms	
		W	[1:0]	00	Interrupt ~= 0.064 ms	0.064 ms
				01	Interrupt ~= 0.128 ms	
				10	Interrupt ~= 0.256ms	
				11	Interrupt ~= 1.024 ms	
$0D	DECMD	R/W	[0]	0/1	reserved	
			[1]	0/1	Tail Disable / Enable	Enable

Addr	Name	R/W	Bit	Data	Description	Default
			[2]	0/1	Head PCM / ADPCM	ADPCM
			[3]	0/1	Tail PCM / ADPCM	ADPCM
$0E	ONOFF	R/W	[0]	0/1	Interrupt Off / On	Off
			[1]	0/1	Audio Out Off / On	Off
			[2]	0/1	Noise filter Off / ON @ 16 cycles per channel	OFF
			[3]	0/1	Audio Mixer Off / On	OFF
$0F	VOL	R/W	[3:0]	0/1	Volume level of Mixer	0010
$10	BANK	R/W	[3:0]	0/1	Program Bank Register	0000
$11	XMD	R/W	[3:0]	0/1	Indexed SRAM data	xxxx
$12	CHNO	R/W	[3:0]	0/1	Active channel select	0000
$14	PA	R	[3:0]	0/1	PAIO = 1: Read port A input pad data	xxxx
					PAIO = 0: Read port A output register	xxxx
		W	[3:0]	0/1	PAIO = 1: wakeup status / pull-high control	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: wakeup status / pull-high control	wakeup status
					PBIO = 0: Write to port B 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: wakeup status / pull-high control	wakeup status
					PCIO = 0: Write to port C 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: wakeup status / pull-high control	wakeup status
					PDIO = 0: Write to port D 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
		W	[3:0]	0/1	PEIO = 1: wakeup status / pull-high control	wakeup status
					PEIO = 0: Write to port E 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: wakeup status / pull-high control	wakeup status
					PFIO = 0: Write to port F output register	xxxx
$1F	PFIO	R/W	[3:0]	0/1	Port F direction = Output / Input	Input

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.

The RPT of NY7A and NY7B is 18-bit long, and the NY7C's RPT is 21-bit. The redundant bits of RPT (RPT[20:18] of NY7A and NY7B) 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 [13:0] of the RPT, the LDDA and RBDA only access bit [12:0] of the RPT, and the XMD only access bit [7:0] 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, so the NY7 series provides 3 instructions to accelerate the access of RPT0~5, ENVL and ENVH. The three instructions are MVRM, MVMR and MVLR.

The RCALL instruction pushes the RPT to the PC and jump to the subroutine address. When the subroutine is finished, use RET to come back to the main program.

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 RD, RDI, RDN and RDNI instructions, bits [11:4] of the obtained 12-bit ROM data will be placed in ROD2[3:0] and ROD1[3:0] and bits [3:0] of ROM data will be placed in ACC.

3.4 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 NY7A and NY7B is 2-bit wide and that of NY7C is 4-bit wide. Each program bank can address up to 64K words space and at most 16 banks are supported in NY7 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 1M words. If users want to branch to program located beyond 1M words, instructions RJMP and RCALL with RPT[20:0] are used and BANK register will not be used in this case.

3.5 XMD

The NY7 series supports indirect-mode access of SRAM data. The RPT0 and RPT1 registers are used to set the 8-bit SRAM address within a SRAM page for indirect read/write. After setting the SRAM address, the

SRAM data can be read by reading data from the XMD register and can be written by writing data to the XMD register. Users have to watch out that the NY7 series does not support using XMD to access System Function Register, so the RPT[7:0] can't be 0x0~0x1F when accessing XMD.

3.6 I/O Ports Register

As PA, PB, PC, PD, PE and PF are bi-directional I/O ports, SFR PAIO, PBIO, PCIO, PDIO, PEIO and PFIO are used to determine the direction of each I/O pin. Writing 1 to any bit of SFR 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 SFR PX (X=A~F) will obtain current state on I/O pin. For I/O pin used as output pin, reading SFR PX (X=A~F) will obtain the value of output register.

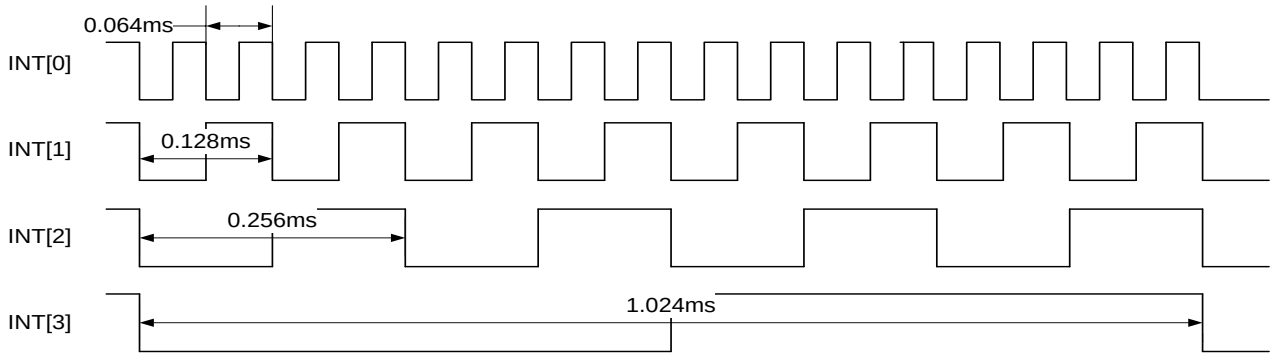
For I/O pin used as input pin, there is a mask option to define whether pull-high resistor of corresponding I/O pin can be enabled or disabled during program execution. When this mask option is disabled, either high-to-low or low-to-high level change on this pin can wake up NY7 from Halt mode or Slow mode. Therefore, user has to read the I/O pin state before entering Halt mode or Slow mode and write back to SFR PX[y] (X=A~F, y=0~3). When 1 is written to SFR PX[y], high-to-low level change on this pin will wake up NY7. When 0 is written to SFR PX[y], low-to-high level change on this pin will wake up NY7. However, user can not enable or disable pull-high resistor during program execution. When this mask option is enabled, only high-to-low level change on this pin can wake up NY7 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 SFR 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 SFR 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. Users have to note that reading from an output port also getting the pad potential level, not the register value.

3.7 INT

The reading source and the writing destination of the SFR[0x0C] (INT register) are different. Reading the 4-bit data of INT acquires the value of the BT counter. The NY7 series provides 4 different base timer intervals for polling: 0.064ms, 0.128ms, 0.256ms and 1.024ms. The value of time means the period, so polling and finding data toggle means half time of the interval. Writing INT[1:0] is selected interrupt source: 0.064ms, 0.128ms, 0.256ms and 1.024ms.



INT Timing Figure

3.8 Audio Control Register

3.8.1 CHARC

The CHARC[1:0] set the active voice/MIDI channel number in NY7 voice/MIDI synthesizer as the following table:

CHARC		Total active channel number	Channels enabled
[1]	[0]		
0	0	8	channel 0~7
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.

When the active channel number is set to 6, channel 6 and 7 are disabled but channel 2 and 3 will be synthesized twice in a whole sample synthesis cycle. In other words, there are 8 active channels which are channel 0,1,2,3,4,5,2,3 while CHARC[1:0] is 01b. Therefore, PH calculation for channel 0,1,4,5 should follow the rule of total 8 active channels but PH calculation for channel 2,3 should adopt the rule of total 4 active channels.

CHARC		Active	NY7		
[3]	[2]	Output	NY7A	NY7B	NY7C
0	X	Push-Pull	-	v	v
1	0	DAC	v	v	v
1	1	PWM	v	-	-

The CHARC[3:2] is used to select the Push-Pull or DAC or PWM output mode. If the Push-Pull mode is selected, all voice/MIDI channels will be mixed to Push-Pull output. Switching between Push-Pull, DAC and PWM modes during voice/MIDI playing should also be avoided.

3.8.2 VOL

The VOL register dominates 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
8	0x2
6	0x3
4	0x4
2	0x8
1	0xF

3.8.3 ONOFF

The ONOFF[0] bit is used to enable (=1) or disable (=0) in the Interrupt. The Interrupt setting is selected by writing INT[1:0] in SFR[0xC]. Before Interrupt is enabled, the interrupt time base must be selected first.

The ONOFF[1] bit is used to enable (=1) or disable (=0) in the Audio Output. Users have to program Audio Output setting by CHARC[1:0] in SFR[0xC] in advance.

The noise filtering of 128KHz over-sampling is enabled by setting bit[2] of ONOFF register to 1. It must be achieved before playback of voice/MIDI and this feature can not be disabled during playback. **As this feature can improve sound quality a lot, it is strongly recommended to enable this feature for every application**.

The ONOFF[3] bit is used to enable (=1) or disable (=0) channel Mixer.

3.8.4 AUD

Reading the 4-bit data of AUD will obtain value of the 4-bit MSB (Most Significant Bit) audio data, which is constituted by 1-bit sign and 3-bit MSB of amplitude. The 4-bit MSB is represented by 2's complement number. This information helps users to get the amplitude of the playing sound. 0x0 and 0xF means the smallest and 0x7 and 0x8 means the largest level of the output audio data.

3.8.5 CHNO

The CHNO 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 CHNO register should be set correctly. The channel related registers include the ENVL[3:0] in SFR[0x6], ENVH[3:0] in SFR[0x7] and the DECMD in SFR[0xD]. These registers have individual register settings for each channel. The channel

related instructions include the PLAY, PLAYI, LDSEC, LDSECI, STOP, SNP, SP, SNHP, SHP and LDPH instructions.

The CHNO register is also used to specify the data or voice pointer register to be utilized by the LDPR, RDN, RDNI, RBDPR and RBVPR instructions. In NY7 chip, there are total 15 register locations shared by the data pointer registers and stack registers. The usage of stack registers grows from location 0xE toward location 0x8 and the usage of data pointer registers should grow in the opposite direction.

3.8.6 ENVL & ENVH

These two registers are 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 setting CHNO[3:0] in SFR[0x12]. As envelope data is 8-bit but ENVL/ENVH are both 4-bit registers, ENVL must be written first and then ENVH. Moreover ENVH must be written otherwise the 8-bit envelope data will not be updated by NY7.

3.8.7 DECMD

The DECMD[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). Note that before writing the DECMD[1] bit, the referenced channel should be specified by setting CHNO[3:0] in SFR[0x12] in advance.

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 DECMD[3:2] are for user to turn-on (=1) or turn-off (=0) the embedded voice decoder of Head wave or/and Tail wave. When the voice decoder is turned-off, NY7 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 CHNO in SFR[0x12] in advance before programming DECMD[3:2].

3.9 Register Without Address Mapping

This Section will describe registers with implied addressing mode. There is no address assigned to this kind of registers.

3.9.1 PAGE

There are 2 memory pages in NY7 series. As PAGE register is not a system register or a memory mapped register, it can only be written by the PAGE0, PAGE1 instruction and can’t be read. The PAGE0、PAGE1 instruction will write the specific page number to PG register.

3.9.2 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 7. 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 writing value to SFR CHNO. Users can obtain the status of HPF flag by instruction SHP or SNHP.

3.9.3 Play Flag

PF 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 7. Therefore PF flag is associated with Head waveform and Tail waveform. As long as either Head waveform is playing or Tail waveform is playing, PF flag of this specific channel will be 1. When Head waveform is end of play and Tail waveform is end of play, PF flag of this specific channel will be 0.

User can use PF flag and HPF flag together to understand the playback status while Tail-Only mode or Head+Tail mode is used for MIDI synthesis. For example, PH=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 writing value to SFR 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 SFR CHNO would stop immediately and PF flag will become 0.

3.9.4 PH Value Setting

PH is a 14-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 writing value to SFR CHNO and utilize instruction LDPH to write value to PH.

It is recommended to keep PH value less than 0x0FFF in order to have better synthetic sound/melody quality. While PH value is larger than 0x1000, synthetic sound/melody quality may degrade a little.

3.9.5 Mixer Data

The Mixer output is temporarily stored to a 13-bit register, which is fed into Audio Output to produce audio signal.

When Mixer is on (SFR ONOFF[3]=1), user can utilize instruction RBDA to read this 13-bit register to RPT[12:0]. If user wants to write value to this register, it has to turn Mixer off in advance. Instruction LDDA can be used to write the contents of RPT[12:0] to this register.

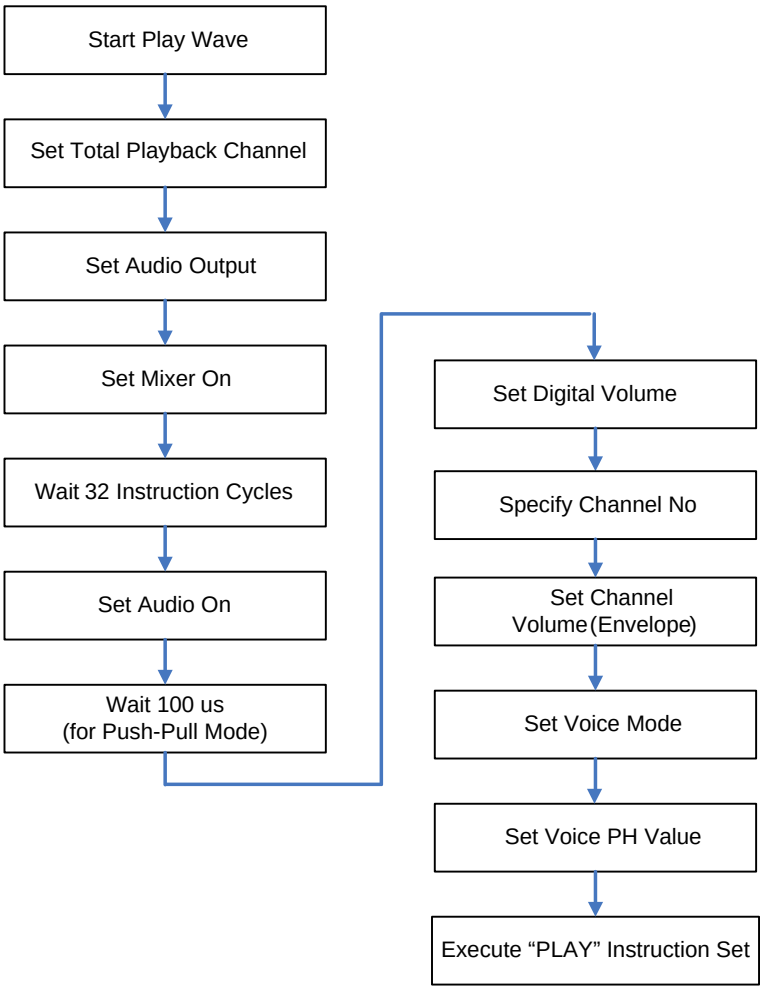
3.10 Audio Playback

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

3.10.1 Voice Playback

3.10.1.1 Flow Chart

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



3.10.1.2 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
   @@Voice0:
       #includata  "Demo.v7x"
   ```

2. Setup Total Playback Channel and Audio Output.

The total number of playback channel can be 2, 4, 6 or 8. This value will determine PH setting and volume setting accordingly. The audio output will depend on which NY7 series is used: NY7A can be PWM or DAC, and NY7B/NY7C can be DAC or Push-Pull.

3. Enable Mixer

   After Mixed is enabled, user cannot execute next step until 32 instruction cycles are expired.

4. Enable Audio Output

   If Push-Pull is selected, it needs to wait 100 us for ramp-up procedure, which is performed by NY7. 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.

5. Configure Digital Volume

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

6. Assign CHNO to Play

   Select specific NY7 channel to play voice before any further configuration.

7. Configure Envelop to Change Channel Volume

   By writing value to System Function Register ENVL and ENVH, there are at most 256 levels to adjust channel volume. ENVL must be written before ENVH is written. Moreover, ENVH must be written otherwise ENVL and ENVH will not be updated.

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 or ADPCM.

9. Determine PH value

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

   For example, SR=16,000 Hz, CH=2, Factor=2 (noise filter is enabled), F_{INST}=4,000,000, the PH value will be 0x20C.

10. Play Voice

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

```
…….
…….
PLAYI   @@Voice0        ;PLAY Demo Voice
```

```
……..
……..
mvlr    0x0,rpt0            ;PLAY Demo1 Voice
mvlr    0x0,rpt1
mvlr    0x5,rpt2
mvlr    0x2,rpt3
mvlr    0x1,rpt4
mvlr    0x0,rpt5
PLAY

Orgalign $, 0x10
@@Voice0:
    #includata  "Demo.v7x"

ORG   0x12500
@@Voice1:
    #includata  "Demo1.v7x"
```

3.10.1.3 Example Code of Voice Playback

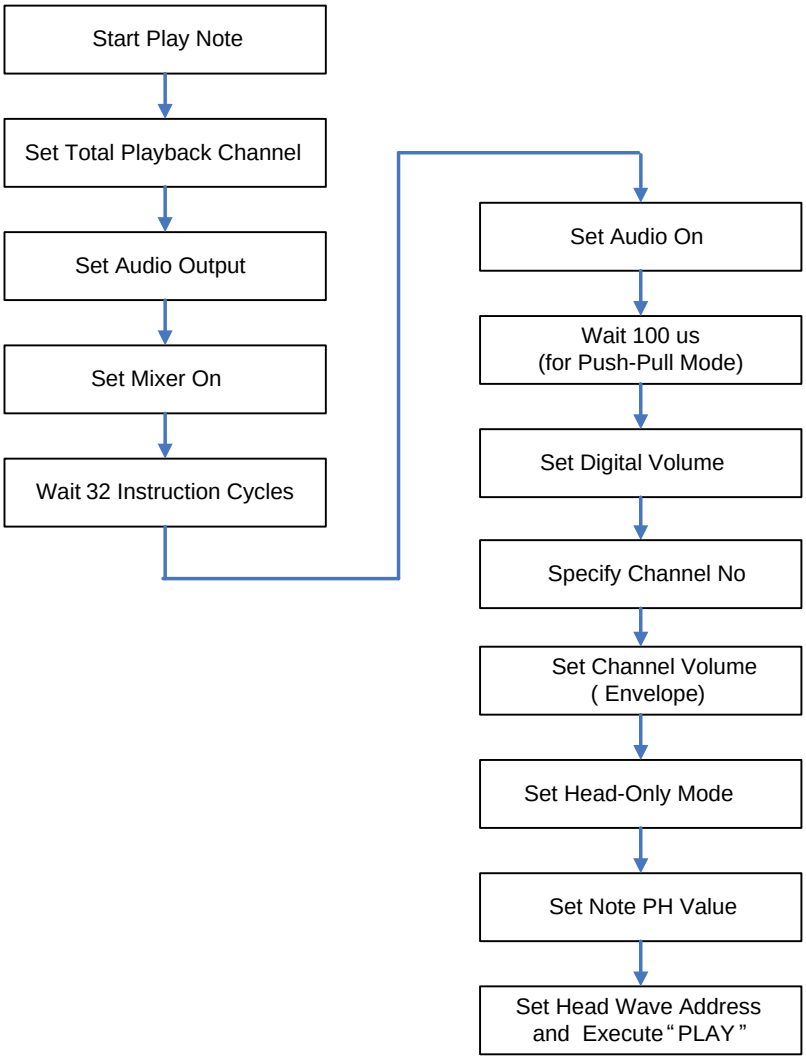
```
@@START:
    CLEAR_SFR           ; Clear System Function Register

    mvla    0x0         ;Set Push-Pull, Total Output Channel =8
    mvam    charc

    mvma    onoff       ;Set Mixer on
    orl     0x8
    mvam    onoff

    Wait_Mix_32Cycles   ;Wait Mixer on

    mvma    onoff       ;Set Audio On
    orl     0x2
    mvam    onoff

    Wait_PP_100us       ;Wait Push-Pull Ramp up

    mvla    0xF         ;Set Master Volume=0xF
```

```
mvam     vol

mvla      0x0             ;Set channel number=0
mvam    chno

mvla      0x0            ;Set Voice =PCM Mode, Full Wave Play
mvam    decmd

mvla     0xF             ;Set Channel0 Volume=0xFF
mvam   envl
mvam   envh

mvlr     0x6,rpt0         ;Set Voice PH (Voice S.R.=16K)
mvlr     0x0,rpt1
mvlr     0x1,rpt2
mvlr     0x0,rpt3
LDPH                      ;Load PH Value

PLAYI   @@Voice0     ;PLAY Demo Voice

Orgalign $, 0x10
@@Voice0:
      #includata  "Demo.v7x"
```

3.10.2 Melody Playback, Head-Only Mode

3.10.2.1 Flow Chart

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



3.10.2.2 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
@@Head0:
        #includata  "Piano_Head0.v7x"
```

2. Setup Total Playback Channel and Audio Output.

   The total number of playback channel can be 2, 4, 6 or 8. This value will determine PH setting and volume setting accordingly. The audio output will depend on which NY7 series is used: NY7A can be PWM or DAC, and NY7B/NY7C can be DAC or Push-Pull.

3. Enable Mixer

   After Mixed is enabled, user can not execute next step until 32 instruction cycles are expired.

4. Enable Audio Output

   If Push-Pull is selected, it needs to wait 100 us for ramp-up procedure, which is performed by NY7. 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.

5. Configure Digital Volume

   There are 16 kinds of digital volume could be applied, from 0 to 15.

6. Assign Channel # to Play

   Select specific NY7 channel to play melody before any further configuration.

7. Configure Envelop to Change Channel Volume

   By writing value to System Function Register ENVL and ENVH, there are at most 256 levels to adjust channel volume. ENVL must be written before ENVH is written. Moreover, ENVH must be written otherwise ENVL and ENVH will not be updated.

8. Setup Waveform File Format

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

9. Determine PH value

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

   For example, patch SR=22,050 Hz, CH=2, Factor=2 (noise filter is enabled), F_{INST}=4,000,000, 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 or PLAYI can be used to play “Head” waveform and its usage is illustrated by the following piece of codes.

```
…….
…….
PLAYI   @@Head0   ;Set Head Wave Address and PLAY
…….
…….
```

```
        mvlr    0x0,rpt0        ;Set Head Wave Address
        mvlr    0x0,rpt1
        mvlr    0x5,rpt2
        mvlr    0x2,rpt3
        mvlr    0x1,rpt4
        mvlr    0x0,rpt5
        PLAY                    ;PLAY

    Orgalign $, 0x10
    @@Head0:
        #includata  "Piano_Head0.v7x"

    ORG   0x12500
    @@Head1:
        #includata  "Piano_Head1.v7x"
```

3.10.2.3 Example Code of Head-Only Melody Playback

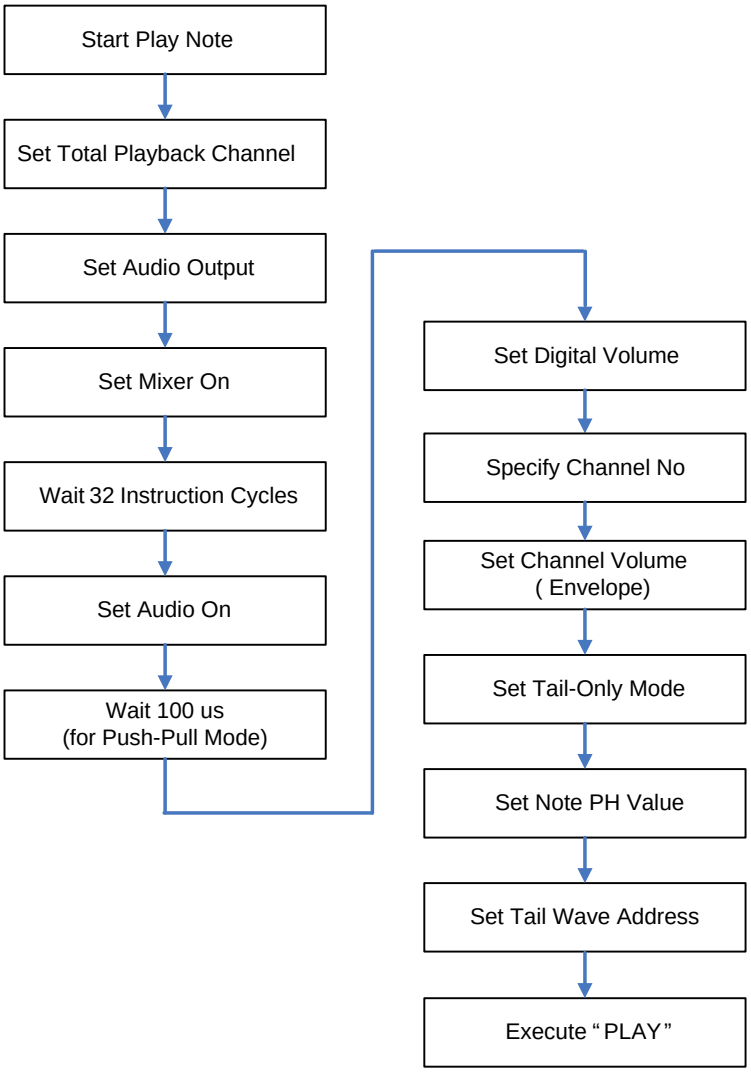
```
    @@START:
        CLEAR_SFR           ;Clear SRF Register

        mvla    0x0         ;Set Push-Pull, Total Output Channel =8
        mvam    charc

        mvma    onoff       ;Set Mixer on
        orl     0x8
        mvam    onoff

        Wait_Mix_32Cycles   ;Wait Mixer on

        mvma    onoff       ;Set Audio On
        orl     0x2
        mvam    onoff

        Wait_PP_100us       ;Wait Push-Pull Ramp up

        mvla    0xF         ;Set Master Volume=0xF
        mvam    vol
```

```
    mvla    0x0             ;Set channel number=0
    mvam    chno

    mvla    0x0             ;Set Voice =PCM Mode, Head Only
    mvam    decmd

    mvla    0xF             ;Set Channel0 Volume=0xFF
    mvam    envl
    mvam    envh

    mvlr    0x6,rpt0        ;Set Note PH
    mvlr    0x0,rpt1
    mvlr    0x1,rpt2
    mvlr    0x0,rpt3
    LDPH                    ;Load Note PH Value

    PLAYI   @@HEAD_WAVE     ;PLAY Head Wave

Orgalign $, 0x10
@@HEAD_WAVE:
    #includata   "Piano_Head.v7x"
```

3.10.3 Melody Playback, Tail-Only Mode

3.10.3.1 Flow Chart

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



3.10.3.2 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
   @@Tail0:
       #includata  "Piano_Tail0.v7x"
   ```

2. Setup Total Playback Channel and Audio Output.

   The total number of playback channel can be 2, 4, 6 or 8. This value will determine PH setting and volume setting accordingly. The audio output will depend on which NY7 series is used: NY7A can be PWM or DAC, and NY7B/NY7C can be DAC or Push-Pull.

3. Enable Mixer

   After Mixed is enabled, user can not execute next step until 32 instruction cycles are expired.

4. Enable Audio Output

   If Push-Pull is selected, it needs to wait 100 us for ramp-up procedure, which is performed by NY7. 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.

5. Configure Digital Volume

   There are 16 kinds of digital volume could be applied, from 0 to 15.

6. Assign Channel # to Play

   Select specific NY7 channel to play melody before any further configuration.

7. Configure Envelop to Change Channel Volume

   By writing value to System Function Register ENVL and ENVH, there are at most 256 levels to adjust channel volume. ENVL must be written before ENVH is written. Moreover, ENVH must be written otherwise ENVL and ENVH will not be updated.

8. Setup Waveform File Format

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

9. Determine PH value

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

   For example, patch SR=22,050 Hz, CH=2, Factor=2 (noise filter is enabled), F_{INST}=4,000,000, 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, NY7 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 or LDSECI can be used to play “Tail” waveform and its usage is illustrated by the following piece of codes.

    ……

    ……

```
        LDSECI    @@Tail0       ;Set Tail Wave Address
        PLAYI     @@Tail0       ;Set Tail Wave Address and Play
        …….
        …….
        mvlr      0x0,rpt0      ;Set Tail Wave Address
        mvlr      0x0,rpt1
        mvlr      0x5,rpt2
        mvlr      0x2,rpt3
        mvlr      0x1,rpt4
        mvlr      0x0,rpt5
        LDSEC

        Mvlr      0x0,rpt0      ;Set Tail Wave Address
        mvlr      0x0,rpt1
        mvlr      0x5,rpt2
        mvlr      0x2,rpt3
        mvlr      0x1,rpt4
        mvlr      0x0,rpt5
        PLAY                    ;PLAY

Orgalign $, 0x10
@@Tail0:
        #includata  "Piano_Tail0.v7x"

ORG   0x12500
@@Tail1:
        #includata  "Piano_Tail1.v7x"
```

3.10.3.3 Example Code of Tail-Only Melody Playback

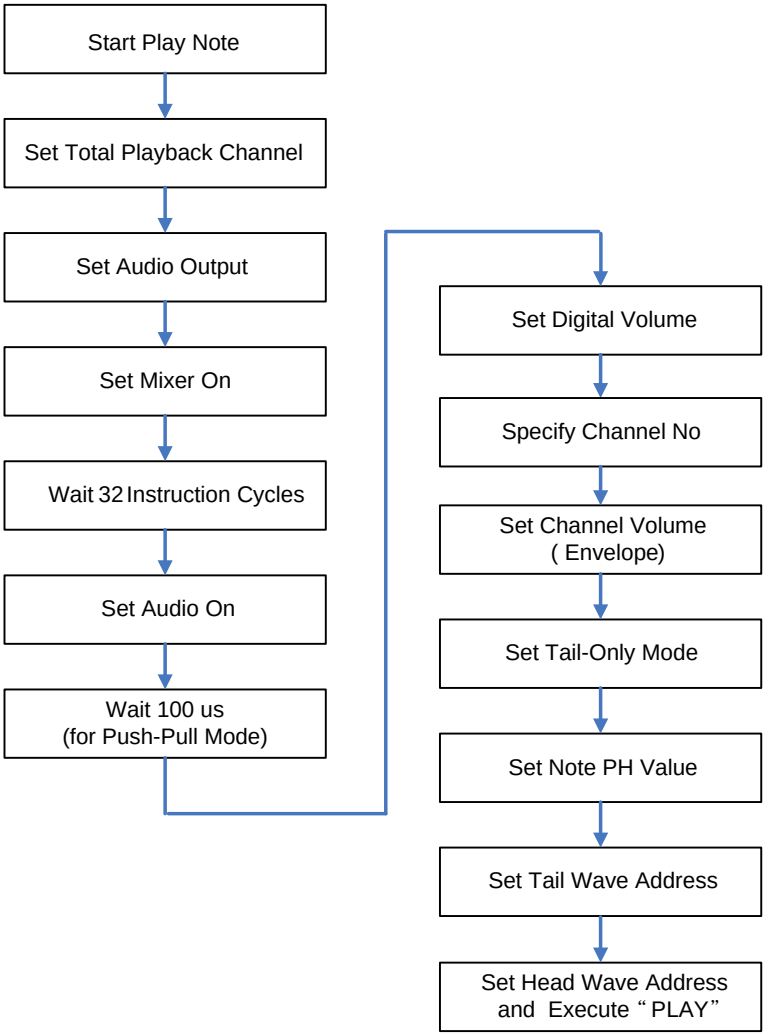
```
@@START:
        CLEAR_SFR           ; Clear System Function Register

        mvla    0x0         ;Set Push-Pull, Total Output Channel =8
        mvam    charc

        mvma    onoff       ;Set Mixer on
        orl     0x8
        mvam    onoff
```

```
        Wait_Mix_32Cycles   ;Wait Mixer on

        Mvma    onoff       ;Set Audio On
        orl     0x2
        mvam    onoff

        Wait_PP_100us       ;Wait Push-Pull Ramp up

        mvla    0xF         ;Set Master Volume=0xF
        mvam    vol

        mvla    0x0         ;Set channel number=0
        mvam    chno

        mvla    0xE         ;Set Head/Tail = ADPCM Mode, Set Tail Wave Enable
        mvam    decmd

        mvla    0xF         ;Set Channel0 Volume=0xFF
        mvam    envl
        mvam    envh

        mvlr    0x6,rpt0    ;Set Note PH
        mvlr    0x0,rpt1
        mvlr    0x1,rpt2
        mvlr    0x0,rpt3
        LDPH                ;Load Note PH Value

        LDSECI  @@TAIL_Wave     ;Load Tail Wave Address
        PLAYI   @@TAIL_Wave     ;PLAY Tail Wave

Orgalign $, 0x10
@@TAIL_WAVE:
        #includata  "Piano_TAIL.v7x"
```

3.10.4 Melody Playback, Head+Tail Mode

3.10.4.1 Flow Chart

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



3.10.4.2 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
   @@Head0:
       #includata  "Piano_Head0.v7x"
   ```

2. Setup Total Playback Channel and Audio Output.

   The total number of playback channel can be 2, 4, 6 or 8. This value will determine PH setting and volume setting accordingly. The audio output will depend on which NY7 series is used: NY7A can be PWM or DAC, and NY7B/NY7C can be DAC or Push-Pull.

3. Enable Mixer

   After Mixed is enabled, user can not execute next step until 32 instruction cycles are expired.

4. Enable Audio Output

   If Push-Pull is selected, it needs to wait 100 us for ramp-up procedure, which is performed by NY7. 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.

5. Configure Digital Volume

   There are 16 kinds of digital volume could be applied, from 0 to 15.

6. Assign Channel # to Play

   Select specific NY7 channel to play melody before any further configuration.

7. Configure Envelop to Change Channel Volume

   By writing value to System Function Register ENVL and ENVH, there are at most 256 levels to adjust channel volume. ENVL must be written before ENVH is written. Moreover, ENVH must be written otherwise ENVL and ENVH will not be updated.

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 ADPCM.

   Moreover, the file format of Head and Tail waveform must be the same. In other words, the allowable combination of file format of <Head, Tail> is <PCM, PCM> or <ADPCM, ADPCM>.

9. Determine PH value

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

   For example, patch SR=22,050 Hz, CH=2, Factor=2 (noise filter is enabled), F_{INST}=4,000,000, 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 or PLAYI can be used to play “Head” waveform. Instruction LDSEC or LDSECI can be used to play “Tail” waveform. Its usage is illustrated by the following piece of codes.

```
…….
…….
LDSECI   @@Tail0        ;Set Tail Wave Address
PLAYI    @@Head0        ;Set Head Wave Address and Play
```

```
……..
……..
mvlr    0x0,rpt0        ;Set Tail Wave Address
mvlr    0x0,rpt1
mvlr    0x5,rpt2
mvlr    0x2,rpt3
mvlr    0x1,rpt4
mvlr    0x0,rpt5
LDSEC

mvlr    0x0,rpt0        ;Set Head Wave Address
mvlr    0x0,rpt1
mvlr    0x0,rpt2
mvlr    0x5,rpt3
mvlr    0x2,rpt4
mvlr    0x0,rpt5
PLAY                    ;PLAY

Orgalign $, 0x10
@@Tail0:
    #includata  "Piano_Tail0.v7x"
ORG   0x12500
@@Tail1:
    #includata  "Piano_Tail1.v7x"

Orgalign $, 0x10
@@Head0:
    #includata  "Piano_Head0.v7x"

ORG   0x25000
@@Head1:
    #includata  "Piano_Head1.v7x"
```

3.10.4.3 Example Code of Head+Tail Melody Playback

```
@@START:
    CLEAR_SFR           ; Clear System Function Register

    mvla    0x0         ;Set Push-Pull, Total Output Channel =8
```

```
        mvam    charc

        mvma    onoff           ;Set Mixer on
        orl     0x8
        mvam    onoff

        Wait_Mix_32Cycles   ;Wait Mixer on

        mvma    onoff           ;Set Audio On
        orl     0x2
        mvam    onoff

        Wait_PP_100us       ;Wait Push-Pull Ramp up

        mvla    0xF             ;Set Master Volume=0xF
        mvam    vol

        mvla    0x0             ;Set channel number=0
        mvam    chno

        mvla    0x2             ;Set Head/Tail =PCM Mode, Set Tail Wave Enable
        mvam   decmd

        mvla    0xF             ;Set Channel0 Volume=0xFF
        mvam    envl
        mvam    envh

        mvlr    0x6,rpt0        ;Set Note PH
        mvlr    0x0,rpt1
        mvlr    0x1,rpt2
        mvlr    0x0,rpt3
        LDPH                    ;Load Note PH Value

        LDSECI   @@TAIL_Wave     ;Load Tail Wave Address
        PLAYI    @@HEAD_Wave    ;Load Head Wave Address and Play

Orgalign $, 0x10
@@HEAD_WAVE:
        #includata  "Piano_HEAD.v7x"
```

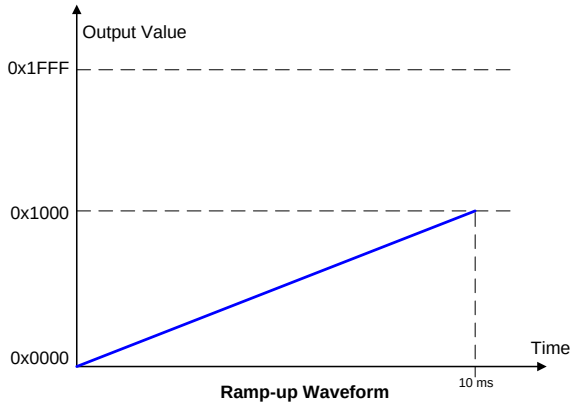
```
Orgalign $, 0x10
@@TAIL_WAVE:
    #includata  "Piano_TAIL.v7x"
```

3.10.5 Ramp-up/Ramp-down Procedure for DAC

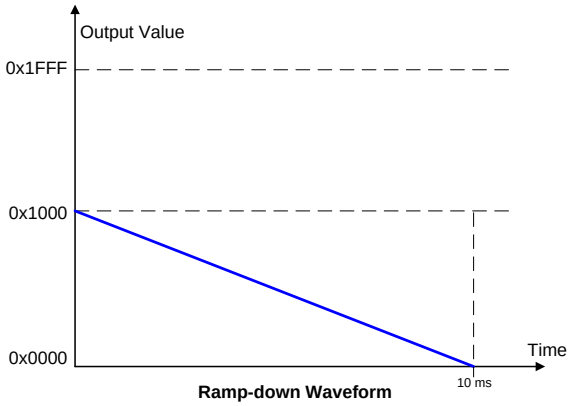
3.10.5.1 Operating Principle

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 NY7, users can take advantage of NY7 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 *.V7x 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 *.V7x files are ready, users can play this ramp-up/ramp-down *.V7x file by instruction PLAY/PLAYI as playing ordinary voice data to complete ramp-up/ramp-down process.

On the other hand, if user's voice data is stored outside NY7, for example external SPI Flash, users have to implement ramp-up/ramp-down process by firmware as following.

The steps to implement ramp-up process are described below.

1. Initialize RPT0~RPT3 (RPT[3:0]) by content of data buffer of DAC output with instruction RBDA.
2. Increase RPT[3:0] by 1 and write RPT[3:0] to data buffer of DAC output by instruction LDDA.
3. Repeat until RPT[3:0] is 0x1000.

The steps to implement ramp-down process are described below.

1. Initialize RPT0~RPT3 (RPT[3:0]) by content of data buffer of DAC output with instruction RBDA.
2. Decrease RPT[3:0] by 1 and write RPT[3:0] to data buffer of DAC output by instruction LDDA.
3. Repeat until RPT[3:0] is 0x0000.

3.10.5.2 Example Code of Ramp-up/Ramp-down by Hardware

```
@@START:
        CLEAR_SFR           ;Clear System Function Register
        mvla    0x8         ;Set DAC, Total Output Channel =8
        mvam    charc

        mvma    onoff       ;Set Mixer on
        orl     0x8
        mvam    onoff

        Wait_Mix_32Cycles   ;Wait Mixer on

        mvma    onoff       ;Set Audio On
        orl     0x2
        mvam    onoff

@@Ramp_up:
        mvla    0x2         ;Set Master Volume=0x2
        mvam    vol
        Voice_Play  @@Rampup,CH0
        Voice_Play  @@Rampup,CH1
        Voice_Play  @@Rampup,CH2
        Voice_Play  @@Rampup,CH3
        Voice_Play  @@Rampup,CH4
        Voice_Play  @@Rampup,CH5
        Voice_Play  @@Rampup,CH6
        Voice_Play  @@Rampup,CH7
@@check_rampup:
        cwdt
        sanp                ;check all 8 channels play =0
        jmp     @@check_rampup
        mvma    0xF
        mvam    vol
```

```
@@Ramp_down:
        Voice_Play    @@Rampdown,CH0
        Voice_Play    @@Rampdown,CH1
        Voice_Play    @@Rampdown,CH2
        Voice_Play    @@Rampdown,CH3
        Voice_Play    @@Rampdown,CH4
        Voice_Play    @@Rampdown,CH5
        Voice_Play    @@Rampdown,CH6
        Voice_Play    @@Rampdown,CH7
@@check_rumpdown:
        cwdt
        sanp                    ;check all 8 channels play =0
        jmp       @@check_rumpdown
        mvla      0x0           ;audio off
        mvamonoff
Orgalign $, 0x10
@@Rampup:
        #includata  "Rampup.v7x"
Orgalign $, 0x10
@@Rampdown:
        #includata  "Rampdown.v7x"
```

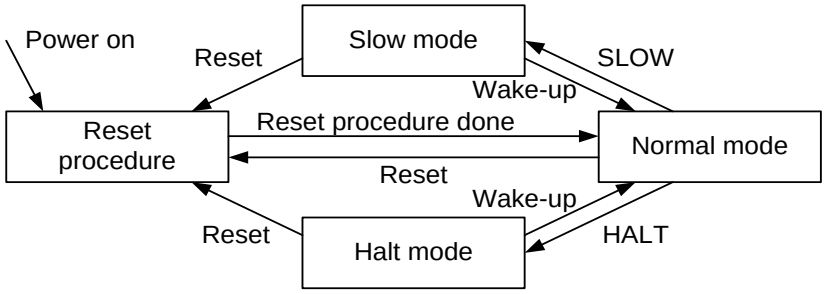
3.10.5.3 Example Code of Ramp-up/Ramp-down by Firmware

```
@@START:
        CLEAR_SFR           ;Clear SRF Register
        mvla      0x8       ;Set DAC, and don't care channel set.
        mvam      charc

        mvma      onoff     ;Set Audio On
        orl       0x2
        mvam      onoff

        rbda                ;Read DAC to RPT[12:0]
@@Ramp_up:
        clrc
        incm      rpt0      ;Increase rpt[12:0] by 1 from 0 to 0x1000.
        jnc       $+9
```

```
        incm    rpt1
        jnc     $+6
        incm    rpt2
        jnc     $+3
        incm    rpt3
        mvma    rpt3
        jnz     L_Rampup_End
        ldda                ;Load RPT[12:0] to DAC reg.
        jmp     L_WaitRampup
L_Rampup_End:
        mvla    0x00
        mvam    rpt0
        mvam    rpt1
        mvam    rpt2
        mvla    0x01
        mvam    rpt3
        ldda

@@Ramp_down:
        rbda                ;Read DAC to RPT[12:0]
L_WaitRampDN:
        cwdt                ;Decrease RPT[12:0] by 1 to 0.
        setc
        decm    rpt0
        jc      $+9
        decm    rpt1
        jc      $+6
        decm    rpt2
        jc      $+3
        decm    rpt3
        jnc     L_RampDN_End
        ldda
        jmp     L_WaitRampDN
L_RampDN_End:
        mvla    0
        mvam    rpt0
        mvam    rpt1
```

```
mvam    rpt2
mvam    rpt3
ldda

mvla    0               ;audio off
mvam    onoff
```

3.11 Power Saving Mode



Power Saving Mode Flow Chart

3.11.1 Slow Mode

The system enters the Slow mode if the SLOW command is executed. The system clock in the Slow mode is about 16 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: 1.024ms, 2.048ms, 4.096ms and 16.384ms.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 NY7 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.

3.11.2 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.

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	ADDA	8m		{C,A} = A + M + C	1	1	C	C, Z
2	XORA	8m		A = A ⊕ M	1	1		Z
3	INCM	8m		{C,M} = M + 1	1	1		C, Z
4	DECM	8m		{C,M} = M – 1	1	1		C, Z
5	RRM	8m		Right Rotate M with C	1	1	C	C, Z
6	RLM	8m		Left Rotate M with C	1	1	C	C, Z
7	MVAM	8m		M = A	1	1		
8	MVMA	8m		A = M	1	1		Z
9	ANDA	8m		A = A & M	1	1		Z
10	ORA	8m		A = A | M	1	1		Z
11	SUBA	8m		{C,A} = A – M - (~B)	1	1	C	C, Z
12	BCLR	5m	2b	Clear M[b]	1	1		
13	BSET	5m	2b	Set M[b]	1	1		
14	MVRM	5m	3r	M = R	1	1		
15	MVMR	5m	3r	R = M	1	1		
16	MVLR	4L	3r	R = L	1	1		
17	MVLA	4L		A = L	1	1		
18	ADDL	4L		{C,A} = L + A + C	1	1	C	C, Z
19	SUBL	4L		{C,A} = A – L - (~B)	1	1	C	C, Z
20	ANDL	4L		A = A & L	1	1		Z
21	ORL	4L		A = A | L	1	1		Z
22	XORL	4L		A = A ⊕ L	1	1		Z
23	INCA			{C,A} = A + 1	1	1		C, Z
24	DECA			{C,A} = A – 1	1	1		C, Z
25	RRC			Right Rotate A with C	1	1	C	C, Z
26	RLC			Left Rotate A with C	1	1	C	C, Z
27	RRA			Right Rotate A	1	1		Z
28	RLA			Left Rotate A	1	1		Z
29	CPLA			A = 0 – A	1	1		Z
30	SETC			Set Carry flag	1	1		C
31	CLRC			Clear Carry flag	1	1		C
Conditional Instructions								
32	JNC	16a		Jump when Carry = 0	2	2	C	
33	JC	16a		Jump when Carry = 1	2	2	C	
34	JNZ	16a		Jump when Zero = 0	2	2	Z	
35	JZ	16a		Jump when Zero = 1	2	2	Z	
36	JB	16a	2b	Jump Adr when A[b] = 1	2	2		
37	SAGT	4L		Skip when A > L	1-4	1		

Item	Inst.	Op1	Op2	Operation	Exec. Cycle	Inst. Length	Oper. Flag	Flag Affected
38	SALT	4L		Skip when A < L	1-4	1		
39	SANE	4L		Skip when A != L	1-4	1		
40	SBZ	2b		Skip when A[b] = 0	1-4	1		
41	SBNZ	2b		Skip when A[b] = 1	1-4	1		
Audio Instructions								
42	PLAY			Play RPT to HVPR of CHNO	3	1		
43	LDSEC			Load RPT to TVPR of CHNO	3	1		
44	LDPH			Load RPT[13:0] to PH of CHNO		1		
45	RBVPR			Read HVPR/TVPR to RPT of CHNO	3	1		
46	SNP			Skip when No Play of CHNO	1-4	1		
47	SP			Skip when Play of CHNO	1-4	1		
48	SANP			Skip when ALL 8 channels Play = 0	1-4	1		
49	STOP			Stop wave play of CHNO		1		
50	SNHP			Skip when head wave No Play of CHNO	1-4	1		
51	SHP			Skip when head wave Play of CHNO	1-4	1		
52	PLAYI	22a		Play Immediately Adr to HVPR of CHNO	4	3		
53	LDSECI	22a		Load Immediately Adr to TVPR of CHNO	4	3		
Other Instructions								
54	LDPR			Load RPT to DPR/STK of CHNO	3	1		
55	RBDPR			Read DPR/STK to RPT of CHNO	3	1		
56	RDN			Data Table Read of CHNO	3	1		
57	RDNI			Data Table Read of CHNO	3	1		
58	RD	4d		ROM Data Read of DPR/STK	3	1		
59	RDI	4d		ROM Data Read of DPR/STK	3	1		
60	LDPRI	4d	20a	Load Immediately Adr to DPR/STK	4	3		
61	CALL	16a		Call Adr	2	2		
62	JMP	16a		Jump Adr	2	2		
63	RCALL			Call RPT	2	1		
64	RJMP			Jump RPT	2	1		
65	RET			Return from subroutine(CALL)	2	2		
66	IRET			Return from interrupt	2	2		
67	HALT			Enter sleep mode		1		
68	SLOW			Enter slow mode		1		
69	CWDT			Clear WDT		1		
70	PAGE0			SRAM Page 0		1		
71	PAGE1			SRAM Page 1		1		
72	RBDA			Read DAC data to RPT[12:0]		1		
73	LDDA			Load RPT[12:0] to DAC reg.		1		
74	NOP			No Operation		1		
75	MPG	1p		Set SRAM Page	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, M=[m]

R : 4-bit memory register data

L : 4-bit immediately literal data

Z : 1-bit zero flag data

RPT : Multi-function register data

CHNO : 2-bit channel number register in SFR[0x12]

PH : 14-bit value for MIDI synthesis of CHNO

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

ENV : 8-bit envelope data of CHNO

ROM : 12-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

r: 3-bit address of System Function Register

p: 1-bit page pointer of RAM

4.2 Instruction Descriptions

4.2.1 Arithmetic Instructions

ADDA m

Function : Add (m) to ACC with Carry and the result is save back to ACC.

Operation : {C, ACC} ← ACC + M + C

Operand: :m: 8-bit address of register or SRAM to ADD, 0x00 to 0xFF

Words :1

Cycles :1

Operative Flags: C

Flags Affected: C, Z

Example :ADDA m0

Before Instruction

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

After Instruction

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

INCM m

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

Operation: M ← M + 1

Operand: m: 8-bit address of register or SRAM to increase, 0x00 to 0xFF

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C, Z

Example: INCM m0

Before Instruction

[m0]=0x0

After Instruction

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

XORA m

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

Operation :ACC ← ACC ⊕ M

Operand: m: 8-bit address of register or SRAM to XOR, 0x00 to 0xFF

Words :1

Cycles :1

Operative Flags:

Flags Affected: Z

Example : XORA m0

Before Instruction

A=0x3, [m0]=0xB

After Instruction

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

DECM m

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

Operation: M ← M - 1

Operand: m: 8-bit address of register or SRAM to decrease, 0x00 to 0xFF

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

RRM m

Function :Right rotate (m) with Carry.

Operation :{ (m)[3:0], C } ← { C, (m)[3:0] }

Operand: m: 8-bit address of register or SRAM to rotate, 0x00 to 0xFF

Words :1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :RRM m0

Before Instruction

[m0]=0x3 and Carry=1

After Instruction

[m0]=0x9 and Carry=1

RLM m

Function :Left rotate (m) with Carry, i.e.

Operation :{ C, (m)[3:0] } ← { (m)[3:0], C }

Operand: m: 8-bit address of register or SRAM to rotate, 0x00 to 0xFF

Words :1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :RLM m0

Before Instruction

[m0]=0xE and Carry=0

After Instruction

[m0]=0xC and Carry=1

MVAM m

Function: Move A to M of address m.

Operation: M ← A

Operand: m: 8-bit address of register or SRAM to move, 0x00 to 0xFF

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MVAM m0

Before Instruction

A=0x8

After Instruction

[m0]=0x8

MVMA m

Function: Move M of address m to A.

Operation: A ← M

Operand: m: 8-bit address of register or SRAM to move, 0x00 to 0xFF

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: MVMA m0

Before Instruction

[m0]=0x8

After Instruction

A=0x8

ANDA m

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

Operation: A ← A & M

Operand: m: 8-bit address of register or SRAM to AND with, 0x0 to 0xFF

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: Z

Example: ANDM m0

Before Instruction

A=0x7, [m0]=0xA

After Instruction

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

SUBA 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,A} = A－ m －(~B)

Operand: m: 8-bit address of register or SRAM to subtract with, 0x0 to 0xFF

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

Words :1

Cycles :1

Operative Flags: None

Flags Affected: None

Example : SUBA m0

Before Instruction

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

After Instruction

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

ORA m

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

Operation: ACC ← ACC | M

Operand: m: 8-bit address of register or SRAM to OR with, 0x0 to 0xFF

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: ORA m0

Before Instruction

A=0x3, [m0]=0xB

After Instruction

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

BCLR m2, b

Function: Clear bit [b] of (m2) to 0

Operation: 0 ← m2[b]

Operand: m2: 5-bit address of SRAM to clear bit, 0x0 to 0x1F

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

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: BCLR m20, 0x2

Before Instruction

[m20]=0xF

After Instruction

[m20]=0xB

Note: The BCLR instruction can only be applied to 32 general SRAMs (0x20~0x3F) of each page.

BSET m, b

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

Operation :1 ← m2[b]

Operand : m2: 5-bit address of SRAM to clear bit, 0x0 to 0x1F

b: 2-bit bit location to set to 1, 0x0 to 0x3

Words :1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :BSET m20, 0x2

Before Instruction

[m20]=0x0

After Instruction

[m20]=0x4

Note : The BSET instruction can only be applied to 32 general SRAMs (0x20~0x3F) of each page.

MVRM m, r

Function : Move SFR(r) to M of address m2.

Operation :M(m2) ← SFR[r]

Operand : m2: 5-bit address of SRAM to set bit, 0x20 to 0x3F

r: 3-bit bit address of SFR to set bit, 0x00 to 0x07

Words :1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :MVRM m20, 0x2

Before Instruction

[m20]=0x0, SFR[0x2]=0x3

After Instruction

[m20]=0x3, SFR[0x2]=0x3

Note : The MVRM instruction can only be applied to 32 general SRAMs (0x20~0x3F) of each page.

MVMR m, r

Function :Move M of address m2 to SFR(r).

Operation : SFR[r] ←M(m2)

Operand : m2: 5-bit address of SRAM to set bit, 0x20 to 0x3F

r: 3-bit bit address of SFR to set bit, 0x00 to 0x07

Words :1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :MVMR m20, 0x2

Before Instruction

[m20]=0x0, SFR[0x2]=0x3

After Instruction

[m20]=0x0, SFR[0x2]=0x0

Note : The MVMR instruction can only be applied to 32 general SRAMs (0x20~0x3F) of each page.

MVLR L, r

Function :Move immediate constant to SFR(r)

Operation : SFR[r] ←L

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

r: 3-bit bit address of SFR to set bit, 0x00 to 0x07

Words :1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :MVLR 0x7, 0x2

Before Instruction

SFR[0x2]=0x3

After Instruction

SFR[0x2]=0x7

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

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

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

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: None

Example : ORL 0xB

Before Instruction

A=0x3, L=0xB

After Instruction

A=0xB, L=0xB, Z=0

INCA

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

Operation: ACC ← ACC + 1

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: C, Z

Example: INCA

Before Instruction

ACC=0x0

After Instruction

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

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: None

Example : XORL 0xB

Before Instruction

A=0x3, L=0xB

After Instruction

A=0x8, L=0xB, Z=0

DECA

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

Operation: 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

RRC

Function :Left 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: Z

Example :RRA

Before Instruction

ACC =0xE ,Z=0

After Instruction

ACC =0x7 ,Z=0

RLC

Function :Right 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

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: Z

Example :RLA

Before Instruction

ACC =0x3 ,Z=0

After Instruction

ACC =0x6 ,Z=0

CPLA

Function :Take complement with ACC.

Operation: ACC← 0 －ACC

Operand: None

Words :1

Cycles :1

Operative Flags: None

Flags Affected: Z

Example :CPLA

Before Instruction

ACC =0xB ,Z=0

After Instruction

ACC =0x5 ,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

JNC a

Function :JNC branches to the address indicated by a if the carry bit is not set.

Operation : When C=0, PC ← {BANK, a}

Operand :a: 16-bit Address, 0x0 to 0xFFFF

Words :2

Cycles :2

Operative Flags: None

Flags Affected: None

Example :JNC a1

Before Instruction

PC=next ADR, C=0

After Instruction

PC={BANK, a1}

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

JNZ a

Function :JNZ branches to the address indicated by a if the Zero bit is not set.

Operation : When Z=0, PC ← {BANK, a}

Operand :a: 16-bit Address, 0x0 to 0xFFFF

Words :2

Cycles :2

Operative Flags: None

Flags Affected: None

Example :JNZ a1

Before Instruction

PC= next ADR, Z=0

After Instruction

PC={BANK, a1}

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

JC a

Function :JC branches to the address indicated by a if the carry bit is set.

Operation : When C=1, PC ← {BANK, a}

Operand : a: 16-bit Address, 0x0 to 0xFFFF

Words :2

Cycles :2

Operative Flags: None

Flags Affected: None

Example :JC a1

Before Instruction

PC= next ADR, C=1

After Instruction

PC={BANK, a1}

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

JZ a

Function :JZ branches to the address indicated by a if the Zero bit is set.

Operation : When Z=1, PC ← {BANK, a}

Operand :a: 16-bit Address, 0x0 to 0xFFFF

Words :2

Cycles :2

Operative Flags: None

Flags Affected: None

Example :JZ a1

Before Instruction

PC= next ADR, Z=1

After Instruction

PC={BANK, a1}

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

JB b, a

Function : JB branches to the address indicated by a if ACC[b] is set.

Operation : When ACC[b]=1, PC ← {BANK, a}

Operand : a: 16-bit Address, 0x0 to 0xFFFF
b: 2-bit bit location, 0x0 to 0x3

Words :2

Cycles :2

Operative Flags: None

Flags Affected: None

Example :JB 0x3, a1
Before Instruction
ACC = 0x8, PC= next ADR
After Instruction
ACC = 0x8, PC={BANK, a1}

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

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.

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.

SANE L

Function: Skip the next instruction if ACC not 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: SANE 0x8
Inst1
Inst2
After Instruction
If ACC = 0x8, `Inst1' is executed.
If ACC < (or >) 0x8, `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 :2

Cycles :2

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.

SBNZ b

Function :Skip the next instruction if ACC[b] is set.

Operation : Skip next if ACC[b]=1.

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

Words :2

Cycles :2

Example :SBNZ 0x3

Inst1

Inst2

After Instruction

If ACC[3] = 0, `Inst1' is executed.

If ACC[3] = 1, `Inst1' is discarded, and `Inst2' is executed.

4.2.3 Audio Instructions

PLAY

Function : Play voice (Head wave) on the channel indexed by the CHNO($12) register. The voice (Head wave) address should be loaded in RPT firstly.

Operation : HVPR[CHNO] ← RPT

Operand :None

Words :1

Cycles :3

Operative Flags: None

Flags Affected: None

Example :
```
MVLA 0x1
MVAM CHNO
MVLR 0x4, RPT0
MVLR 0x3, RPT1
MVLR 0x2, RPT2
MVLR 0x0, RPT3
MVLR 0x1, RPT4
MVLR 0x0, 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 CHNO($12) register. The Tail wave address should be loaded in TREG[19:0] firstly.

Operation : TVPR[CHNO] ← RPT

Operand :None

Words :1

Cycles :3

Operative Flags: None

Flags Affected: None

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

TVPR[0x1]= 0xXXXXX

After Instruction

TVPR[0x1]= 0x010234

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

LDPR

Function : Load ROM address to the DPR (data pointer) indexed by the CHNO($12) 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
MVAM CHNO
MVLR 0x0, RPT0
MVLR 0x9, RPT1
MVLR 0x3, RPT2
MVLR 0xD, RPT3
MVLR 0x1, RPT4
MVLR 0x0, RPT5
LDPR

Before Instruction
DPR [0x2]= 0xXXXXX

After Instruction
DPR [0x2]= 0x01D390

LDPH

Function : Load PH value to the channel indexed by the CHNO($12) register. The PH value should be loaded in RPT[13:0] firstly.

Operation : PH[CHNO] ← RPT[13:0]

Operand :None

Words :1

Cycles :3

Operative Flags: None

Flags Affected: None

Example : MVLA 0x4
MVAM CHNO
MVLR 0x4, RPT0
MVLR 0x3, RPT1
MVLR 0x2, RPT2
MVLR 0x0, RPT3
LDPH

Before Instruction
PH [0x4]= 0xXXXX

After Instruction
PH[0x4]= 0x0234

RBVPR

Function : Read HVPR/TVPR (voice pointer) content. The HVPR/TVPR to read is indexed by the CHNO($12) register and the obtained content is put in RPT.

Operation : RPT ← HVPR/TVPR[CHNO]

Operand :None

Words :1

Cycles :3

Operative Flags: None

Flags Affected: None

Example : MVLA 0x4

MVAM CHNO

RBVPR

Before Instruction

HVPR[0x4]= 0x010234

After Instruction

RPT = 0x010234

RBDPR

Function : Read DPR (data pointer) content. The DPR to read is indexed by the CHNO($12) register and the obtained content is put in RPT.

Operation : RPT ← DPR[CHNO]

Operand :None

Words :1

Cycles :3

Operative Flags: None

Flags Affected: None

Example : MVLA 0x5

MVAM CHNO

RBDPR

Before Instruction

DPR[0x5]= 0x010234

After Instruction

RPT = 0x010234

RDN

Function : Read ROM data using the DPR (data pointer) indexed by the CHNO($12) register.

Operation : ACC ← bit [3:0] of read data
ROD1($08) ← bit [7:4] of read data
ROD2($09) ← bit [11:8] of read data

Operand :None

Words :1

Cycles :3

Operative Flags: None

Flags Affected: None

Example :
```
MVLA 0x1
MVAM CHNO
MVLR (LOW0 data0),RPT0
MVLR (LOW1 data0),RPT1
MVLR (MID0 data0),RPT2
MVLR (MID1 data0),RPT3
MVLR (HIGH0 data0),RPT4
MVLR (HIGH1 data0),RPT5
LDPR
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 CHNO($12) register, and increase the DPR after data reading.

Operation : ACC ← bit [3:0] of read data
ROD1($08) ← bit [7:4] of read data
ROD2($09) ← bit [11:8] of read data

Operand :None

Words :1

Cycles :3

Operative Flags: None

Flags Affected: None

Example :
```
MVLA 0x3
MVAM CHNO
MVLR (LOW0 data0),RPT0
MVLR (LOW1 data0),RPT1
MVLR (MID0 data0),RPT2
MVLR (MID1 data0),RPT3
MVLR (HIGH0 data0),RPT4
MVLR (HIGH1 data0),RPT5
LDPR
RDNI
```
Before Instruction
data0 is 0x82B
After Instruction
ACC=0xB, ROD1=0x2, ROD2=0x8, and DPR[0x3]= data0+1

SNP

Function: Skip the next instruction if the channel (Head or Tail) indexed by the CHNO($12) 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

MVAM CHNO

SNP

Inst1

Inst2

After Instruction

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

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.

SP

Function: Skip the next instruction if the channel (Head or Tail) indexed by the CHNO($12) 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

MVAM 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 CHNO($12) register.

Operation : None

Operand :None

Words :1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :MVLA 0x4

MVAM CHNO

STOP

After Instruction

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

SNHP

Function: Skip the next instruction if the channel (Head vice) indexed by the CHNO($12) 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

MVAM 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.

SHP

Function: Skip the next instruction if the channel (Head vice) indexed by the CHNO($12) register is playing.

Operation: Skip next if Head waveform is playing.

Operand: None

Words: 1

Cycles: 1, (2, 3, 4)

Operative Flags: None

Flags Affected: None

Example: MVLA 0x1

MVAM CHNO

SHP

Inst1

Inst2

After Instruction

If channel 1 (Head vice)not play, `Inst1' is executed.

If channel 1 (Head vice)play, `Inst1' is discarded, and `Inst2' is executed.

PLAYI a

Function : Play voice (Head wave) on the channel indexed by the CHNO($12) register. The voice (Head wave) address is specified by a.

Operation: HVPR[CHNO] ← a

Operand : a: 22-bit HVPR address to load for playing, 0x000000 to 0x3FFFFF

Words :3

Cycles :4

Operative Flags: None

Flags Affected: None

Example : MVLA 0x2
MVAM CHNO
PLAYI 0x010234

Before Instruction
HVPR[0x2]= 0xXXXXX

After Instruction
HVPR[0x2]= 0x010234, PFLG[0x1]=1

LDSECI a

Function : Load immediate Tail wave address for the channel indexed by the CHNO($12) register. The voice (Tail wave) address is specified by a.

Operation: TVPR[CHNO] ← a

Operand : a: 22-bit Tail wave address to load, 0x000000 to 0x3FFFFF

Words :3

Cycles :4

Operative Flags: None

Flags Affected: None

Example : MVLA 0x6
MVAM CHNO
LDSECI 0x010C34

Before Instruction
TVPR[0x6]= 0xXXXXX

After Instruction
TVPR[0x6]= 0x10C34

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

4.2.4 Other Instructions

RD d

Function : Read ROM data using the selected DPR (data pointer)

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

ROD1($08) ← bit [7:4] of read data

ROD2($09) ← bit [11:8] of read data

Operand : d: 4-bit DPR number to select the DPR for data reading, 0x0 to 0xF

Words :1

Cycles :3

Operative Flags: None

Flags Affected: None

Example : LDPRI 0x1, TBLADR

RD 0x1

Before Instruction

TBLADR is 0x135

After Instruction

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

RDI d

Function : Read ROM data using the selected DPR (data pointer), and increase the DPR after data reading.

Operand :ACC ← bit [3:0] of read data

ROD1 ($08) ← bit [7:4] of read data

ROD2 ($09) ← bit [11:8] of read data

Operand : d: 4-bit DPR number to select the DPR for data reading, 0x0 to 0xF

Words :1

Cycles :3

Operative Flags: None

Flags Affected: None

Example : LDPRI 0x3, TBLADR

RDI 0x3

Before Instruction

TBLADR is 0x135

After Instruction

ACC=0x5, ROD1=0x3, ROD2=0x1, and DPR[0x3]= TBLADR+1

LDPRI d, a

Function : Load immediate ROM address to the selected DPR (data pointer).

Operation: DPR[d] ← a

Operand : d: 4-bit DPR number to load the ROM address, 0x0 to 0xF

a: 20-bit ROM address to load into the selected DPR, 0x00000 to 0xFFFFF

Words :3

Cycles :4

Operative Flags: None

Flags Affected: None

Example : LDPRI 0x7, 0x20D39

After executing the LDPRI instruction:

DPR7=0x20D39

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.

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.

RCALL

Function : Call subroutine by indirect address. The call address should be loaded in RPT firstly.

Operation: STK ← PC+2

PC ← RPT

Operand :None

Words :1

Cycles :2

Operative Flags: None

Flags Affected: None

Example :
```
MVLR(LOW0 data0),RPT0
MVLR(LOW1 data0),RPT1
MVLR(MID0 data0),RPT2
MVLR(MID1 data0),RPT3
MVLR(HIGH0 data0),RPT4
MVLR(HIGH1 data0),RPT5
RCALL
```
Before Instruction

PC=a0, RPT= ADR[data0]

After Instruction

PC=RPT, STK =a0+2

RJMP

Function :Unconditional jump by indirect address. The jump address should be loaded in RPT firstly.

Operation: PC ← RPT

Operand :None

Words :1

Cycles :2

Operative Flags: None

Flags Affected: None

Example :
```
MVLR(LOW0 data0),RPT0
MVLR(LOW1 data0),RPT1
MVLR(MID0 data0),RPT2
MVLR(MID1 data0),RPT3
MVLR(HIGH0 data0),RPT4
MVLR(HIGH1 data0),RPT5
RJMP
```
Before Instruction

PC=a0, RPT= ADR[data0]

After Instruction

PC=RPT

RET

Function :Return from subroutine

Operation: PC ← STK

Operand: None

Words: 2

Cycles: 2

Example: RET

After Instruction

PC)← STK

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.

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

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 16 times.

CWDT

Function: Clear Watch Dog Timer.

Operation: Watch dog counter ← 0x0

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: CWDT

Before Instruction

WDT counter = ???

After Instruction

WDT counter = 0x0

PAGE1

Function: Set SRAM page to 1

Operation:

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: PAGE1

Before Instruction

PAGE = 0x0

After Instruction

PAGE = 0x1

PAGE0

Function :Set SRAM page to 0

Operation:

Operand :None

Words :1

Cycles :1

Operative Flags: None

Flags Affected: None

Example :PAGE0

Before Instruction

PAGE = 0x1

After Instruction

PAGE = 0x0

RBDA

Function: Read Mixer data to RPT.

Operation: RPT[12:0] ← Mixer data

Operand: Mixer data : 13-bit

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example:RBDA

Before Instruction

Mixer data =0x01234

After Instruction

RPT=0x01234

LDDA

Function: Load RPT to Mixer data.

Operation: Mixer data ← RPT

Operand: Mixer data: 13-bit

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: LDDA

Before Instruction

RPT[12:0]=0x321

After Instruction

Mixer data =0x321

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.

MPG

Function: Set 1-bit value to SRAM page.

Operation:

Operand: None

Words: 1

Cycles: 1

Operative Flags: None

Flags Affected: None

Example: MPG 0x1

Before Instruction

PAGE = 0x0

After Instruction

PAGE = 0x1