



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

User Manual

Q-Code

Power-Speech Format Programmer

Version 8.4

Aug. 29, 2025

NYQUEST TECHNOLOGY CO., Ltd. reserves the right to change this document without prior notice. Information provided by NYQUEST is believed to be accurate and reliable. However, NYQUEST makes no warranty for any errors which may appear in this document. Contact NYQUEST to obtain the latest version of device specifications before placing your orders. No responsibility is assumed by NYQUEST for any infringement of patent or other rights of third parties which may result from its use. In addition, NYQUEST products are not authorized for use as critical components in life support devices/systems or aviation devices/systems, where a malfunction or failure of the product may reasonably be expected to result in significant injury to the user, without the express written approval of NYQUEST.

Table of Content

1 Introduction.....	24
2 Q-Code Interface.....	25
2.1 Function Menu	26
2.1.1 [File] Menu	26
2.1.2 [Edit] Menu	26
2.1.3 [Search] Menu	27
2.1.4 [View] Menu	27
2.1.5 [Compile] Menu	28
2.1.6 [Debug] Menu.....	28
2.1.7 [Tool] Menu.....	29
2.1.8 [Setting] Menu	30
2.1.9 [Help].....	30
2.2 Section	30
2.3 Text Edit Area	31
2.4 Debug Mode.....	31
2.4.1 Watch Window	31
2.4.2 Register Window	32
2.5 Message Window	32
2.5.1 Error List.....	32
2.5.2 Build Message.....	32
2.6 Information Window.....	32
2.6.1 Information	32
2.6.2 Resource	33
2.6.3 Pin	33
2.7 Q-Code Status Bar	34
2.7.1 Build Result and ICE Status.....	34
2.7.2 ICE Connect Status.....	34
2.7.3 Row, Column, Characters	34
3 The Architecture of Q-Code.....	35
3.1 Option	36
3.1.1 IC Body	36
3.1.2 Client	36
3.1.3 Voltage	37
3.1.4 Package Test	37
3.1.5 Oscillator	38
3.1.6 Voice Output.....	43
3.1.7 Touch Key	49
3.1.8 Reset.....	53

3.1.9	IR.....	55
3.1.10	SPI Flash.....	60
3.1.11	Serial Control TX.....	65
3.1.12	Serial Control RX	66
3.1.13	I2C.....	67
3.1.14	UART.....	68
3.1.15	FD.....	70
3.1.16	Random.....	70
3.1.17	Debounce.....	71
3.1.18	Interrupt Service.....	75
3.1.19	LVD	76
3.1.20	PWM-IO	77
3.1.21	RAM Usage	78
3.1.22	RFC.....	79
3.1.23	WaveID.....	81
3.1.24	Sound Localization.....	81
3.1.25	Sound Detection.....	83
3.1.26	Picth Detection	83
3.1.27	Maximum Single Note.....	84
3.1.28	Melody_OKON_BgNote.....	84
3.1.29	Variable Compatible	84
3.1.30	Realtime Play.....	85
3.1.31	Sound Effect.....	85
3.1.32	Audio Filter.....	86
3.1.33	Common ROM Code.....	87
3.1.34	ISP.....	87
3.1.35	Storage Modual.....	88
3.1.36	ADC Input.....	88
3.2	Voice File	89
3.2.1	NY4	90
3.2.2	NY5 / NY5+	91
3.2.3	NY6.....	92
3.2.4	NY7	92
3.2.5	NX1	93
3.2.6	How to Play Files Processed by Q-Sound in Q-Code	94
3.3	Action File	95
3.3.1	NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1.....	95
3.4	Melody Database	95
3.4.1	NY5.....	95
3.4.2	NY5+ / NY6 / NY7 / NX1	96
3.5	Touch-Key File	96
3.5.1	NY9T	96
3.6	Memory	96

3.6.1	NX1 OTP	96
3.6.2	NX1 EF	97
3.7	SPI Flash	97
3.7.1	NY6	97
3.8	Record	98
3.8.1	NX1	98
3.9	LED Strip	101
3.9.1	Option	101
3.9.2	Single String	101
3.9.3	Sync-Play	102
3.9.4	Scrolling-Text	103
3.9.5	Add File	103
3.10	QFID	104
3.10.1	NY5+ / NY7 / NX1	104
3.11	I/O	108
3.11.1	Matrix Key	108
3.11.2	Direct Key	108
3.11.3	Touch Key	115
3.11.4	Push-High Resistor	116
3.11.5	Input Voltage	118
3.12	Input State	118
3.13	Output State	120
3.13.1	NY4 / NY5	121
3.13.2	NY5+	122
3.13.3	NY6 / NY7	122
3.13.4	NX1	122
3.14	I/O Expander	123
3.14.1	I/O Expander	123
3.14.2	Matrix	124
3.14.3	Direct	124
3.14.4	Input State	125
3.14.5	Output State	126
3.15	QIO	126
3.15.1	Configure	126
3.15.2	QIO Table	127
3.16	PWM-IO	128
3.16.1	Configure	128
3.16.2	Multi Signal	129
3.16.3	Single Signal	130
3.17	Action	130
3.17.1	Configure	130
3.17.2	Multi Signal	131

3.17.3	Single Signal	132
3.18	VR.....	132
3.18.1	VR File	133
3.18.2	VR Combo.....	134
3.18.3	VR State	136
3.19	Action Mark	137
3.19.1	NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1.....	137
3.20	Wave Mark	138
3.20.1	NY4 / NY5 / NY5+ / NX1	138
3.21	Melody Mark.....	139
3.21.1	NY5 / NY5+ / NY6 / NY7 / NX1	139
3.22	Note On.....	140
3.22.1	NY5 / NY5+ / NY6 / NY7 / NX1	140
3.23	PWM-IO Mark	141
3.23.1	NY5+	141
3.24	IR Receive	142
3.24.1	NY4 / NY5 / NY5+ / NY6 / NY7 / NX1	142
3.25	Symbol	143
3.25.1	NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1.....	143
3.26	Variable	143
3.26.1	NX1	143
3.27	Storage Variable	144
3.27.1	NX1	144
3.28	Table	144
3.28.1	NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1.....	144
3.29	ASM.....	145
3.29.1	NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T.....	145
3.30	C-Code.....	146
3.30.1	NX1	146
3.31	Macro.....	147
3.31.1	NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1.....	147
3.32	Sentence.....	147
3.32.1	NY4 / NY5	148
3.32.2	NY5+ / NY6 / NY7 / NX1	148
3.32.3	NY9T	149
3.33	Subroutine.....	150
3.33.1	NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1.....	150
3.34	QIO Custom	150
3.34.1	NY4 / NY5 / NY5+	150
3.35	Path	150

3.35.1	NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1	150
3.36	Background1	151
3.36.1	NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1	151
3.37	Background2	151
3.37.1	NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1	151
3.38	Background3	152
3.38.1	NX1	152
3.39	Background4	152
3.39.1	NX1	152
3.40	Background5	152
3.40.1	NX1	152
3.41	Interrupt	153
3.41.1	NX1	153
4	Q-Code Special Paths	155
4.1	General Paths	155
4.1.1	PowerOn	155
4.1.2	Before_PowerOn	155
4.1.3	Main	156
4.1.4	Sleep	156
4.1.5	WakeUp	156
4.2	Timer Paths	157
4.2.1	512us	157
4.2.2	1ms	157
4.2.3	2ms	157
4.2.4	4ms	158
4.2.5	8ms	158
4.2.6	500ms	159
4.2.7	1sec	159
4.2.8	4sec	159
4.3	Interrupt Paths	160
4.3.1	Interrupt	160
4.3.2	Int_256us	160
4.3.3	Int_512us	161
4.3.4	Int_1ms	161
4.3.5	Int_16ms	162
4.3.6	Int_CMP	162
4.3.7	Int_TMR	162
4.3.8	Int_TMR0	163
4.3.9	Int_TMR1	163
4.3.10	Int_TMR2	164
4.3.11	Int_TMR3	164
4.3.12	Int_PWMA	165

4.3.13	<i>Int_PWMb</i>	166
4.3.14	<i>RTC Interrupt (RTC_2Hz / RTC_64Hz / RTC_1024Hz / RTC_4096Hz / RTC_16384Hz)</i>	166
4.4	LVD Paths	167
4.4.1	<i>LVD_mVn / LVD_Max</i>	167
4.4.2	<i>LVD</i>	167
4.5	Communication Paths	168
4.5.1	<i>IR_RX</i>	168
4.5.2	<i>SC_RX</i>	168
4.5.3	<i>I2C_RX</i>	168
4.5.4	<i>I2C_TX_Ok</i>	169
4.5.5	<i>I2C_Error</i>	169
4.5.6	<i>UART_RX</i>	169
4.5.7	<i>WaveID_RX</i>	169
4.5.8	<i>SL_RX</i>	170
4.6	Touchkey Paths	170
4.6.1	<i>Enforce_Calibrate</i>	170
4.7	SPI Flash Paths	171
4.7.1	<i>SPI_EraseEnd</i>	171
4.7.2	<i>SPI_EraseTimeout</i>	171
4.8	Voice Recognition Paths	172
4.8.1	<i>VRGC_Timeout</i>	172
4.8.2	<i>VR_Unknown</i>	172
4.8.3	<i>VT_BeforeRecord</i>	172
4.8.4	<i>VT_BeforeTraining</i>	172
4.8.5	<i>VT_AddTagFail</i>	173
4.9	Record Paths	173
4.9.1	<i>KRecord_Timeout</i>	173
4.10	Sound Detect Paths	173
4.10.1	<i>Sound_Detected</i>	173
4.10.2	<i>Sound_Muted</i>	174
4.11	Pitch Detect Paths	174
4.11.1	<i>Pitch_Detected</i>	174
4.12	I/O Expander Path	175
4.12.1	<i>IoExp_CommFail</i>	175
5	Q-Code Command	176
5.1	Arithmetic Logic Command	176
5.1.1	<i>Variable Operation</i>	176
5.1.2	<i>Var=RandomL</i>	178
5.1.3	<i>Var=RandomH</i>	178
5.1.4	<i>Var=Random</i>	178
5.2	Flow Control Command	179

5.2.1	Comparison	180
5.2.2	Px = data?Path	181
5.2.3	Px != data?Path	181
5.2.4	Px.n = 0?Path	181
5.2.5	Px.n != 0?Path	181
5.2.6	Px.n = 1?Path	182
5.2.7	Px.n != 1?Path	182
5.2.8	Vol = n?Path	182
5.2.9	Vol != n?Path	182
5.2.10	MixCtrl = data?Path	182
5.2.11	MixCtrl != data?Path	182
5.2.12	RandomL = data?Path	183
5.2.13	RandomH = data?Path	183
5.2.14	RandomL != data?Path	183
5.2.15	RandomH != data?Path	183
5.2.16	Random = data?Path	184
5.2.17	Random != data?Path	184
5.2.18	Px[1 X 0 X]?Path	185
5.2.19	ChUsed(Ch)?Path	185
5.2.20	Voice(Ch)?Path	186
5.2.21	PauseV(Ch)?Path	186
5.2.22	Melody?Path	187
5.2.23	PauseM?Path	187
5.2.24	Delay(n)?Path	187
5.2.25	PauseD(n)?Path	187
5.2.26	Action(Ch)?Path	188
5.2.27	PauseA(Ch)?Path	188
5.2.28	PWMIO?Path	188
5.2.29	PausePWM?Path	189
5.2.30	HoldPWM(n)?Path	190
5.2.31	SPIPlay(Ch)?Path	190
5.2.32	SPIPause(Ch)?Path	191
5.2.33	Pause(n)?Path	191
5.2.34	Record?Path	191
5.2.35	KRecord?Path	192
5.2.36	Recorded(\$Rec)?Path	192
5.2.37	PlayK?Path	192
5.2.38	EraseR?Path	192
5.2.39	VR_VAD?Path	192
5.2.40	LEDStr?Path	193
5.2.41	LEDSync?Path	193
5.2.42	LEDText?Path	193
5.2.43	Animaltalks_Record?Path	193
5.2.44	Animaltalks_Play?Path	194
5.2.45	Animalsings_Record?Path	194

5.2.46	<i>Animalsings_Play?Path</i>	194
5.2.47	<i>Calibrate?Path</i>	194
5.2.48	<i>IoExp_Exists?Path</i>	194
5.2.49	<i>Checksum?Path</i>	195
5.2.50	<i>SPI_CheckSum?Path</i>	195
5.2.51	<i>!Px[1 X 0 X]?Path</i>	195
5.2.52	<i>!ChUsed(Ch)?Path</i>	196
5.2.53	<i>!Voice(Ch)?Path</i>	196
5.2.54	<i>!PauseV(Ch)?Path</i>	197
5.2.55	<i>!Melody?Path</i>	197
5.2.56	<i>!PauseM?Path</i>	197
5.2.57	<i>!Delay(n)?Path</i>	198
5.2.58	<i>!PauseD(n)?Path</i>	198
5.2.59	<i>!Action(Ch)?Path</i>	198
5.2.60	<i>!PauseA(Ch)?Path</i>	198
5.2.61	<i>!PWMIO?Path</i>	199
5.2.62	<i>!PausePWM?Path</i>	200
5.2.63	<i>!HoldPWM(n)?Path</i>	200
5.2.64	<i>!SPIPlay(Ch)?Path</i>	201
5.2.65	<i>!SPIPause(Ch)?Path</i>	201
5.2.66	<i>!Pause(n)?Path</i>	202
5.2.67	<i>!Record?Path</i>	202
5.2.68	<i>!KRecord?Path</i>	202
5.2.69	<i>!Recorded(\$Rec)?Path</i>	202
5.2.70	<i>!PlayK?Path</i>	203
5.2.71	<i>!EraseR?Path</i>	203
5.2.72	<i>!VR_VAD?Path</i>	203
5.2.73	<i>!LEDStr?Path</i>	203
5.2.74	<i>!LEDSync?Path</i>	203
5.2.75	<i>!LEDText?Path</i>	204
5.2.76	<i>!Animaltalks_Record?Path</i>	204
5.2.77	<i>!Animaltalks_Play?Path</i>	204
5.2.78	<i>!Animalsings_Record?Path</i>	204
5.2.79	<i>!Animalsings_Play?Path</i>	205
5.2.80	<i>!Calibrate?Path</i>	205
5.2.81	<i>!IoExp_Exists?Path</i>	205
5.2.82	<i>!Checksum?Path</i>	205
5.2.83	<i>!SPI_CheckSum?Path</i>	206
5.2.84	<i>If-Else</i>	206
5.2.85	<i>Switch(Ri) = [Path0, Path1, Path2,... Path15]</i>	207
5.2.86	<i>Switch(Px) = [Path0, Path1, Path2,... Path15]</i>	208
5.2.87	<i>Switch(RandomL) = [Path0, Path1, Path2,... Path15]</i>	208
5.2.88	<i>Switch(RandomH) = [Path0, Path1, Path2,... Path15]</i>	208
5.2.89	<i>Switch(Xi) = [Path0, Path1, Path2,..Path255]</i>	208
5.2.90	<i>Switch(Random) = [Path0, Path1, Path2,... Path255]</i>	209

5.2.91	<i>Switch(Px[d x d x]) = [Path0, Path1, Path2,... Path15]</i>	209
5.2.92	<i>While</i>	209
5.2.93	<i>Do-While</i>	210
5.2.94	<i>For</i>	211
5.3	<i>I/O Command</i>	212
5.3.1	<i>Ri = Px</i>	213
5.3.2	<i>Ri = PxM</i>	213
5.3.3	<i>Ri.n = Px.n</i>	213
5.3.4	<i>PxM = data</i>	213
5.3.5	<i>PxM = Ri</i>	213
5.3.6	<i>PxM.n = 0</i>	213
5.3.7	<i>PxM.n = 1</i>	214
5.3.8	<i>Px = data</i>	214
5.3.9	<i>Px = Ri</i>	214
5.3.10	<i>Px = Py</i>	214
5.3.11	<i>!Px</i>	214
5.3.12	<i>!Px.n</i>	214
5.3.13	<i>Px.n = 0</i>	215
5.3.14	<i>Px.n = 1</i>	215
5.3.15	<i>Px.n = Ri.n</i>	215
5.3.16	<i>Px.n = Py.n</i>	215
5.3.17	<i>Px = Py + Ri</i>	215
5.3.18	<i>Px = Py - Ri</i>	215
5.3.19	<i>Px = Py Ri</i>	216
5.3.20	<i>Px = Py ^ Ri</i>	216
5.3.21	<i>Px = Py & Ri</i>	216
5.3.22	<i>Ri = Px + Rj</i>	216
5.3.23	<i>Ri = Px - Rj</i>	216
5.3.24	<i>Ri = Px Rj</i>	216
5.3.25	<i>Ri = Px ^ Rj</i>	217
5.3.26	<i>Ri = Px & Rj</i>	217
5.3.27	<i>Px = Py + data</i>	217
5.3.28	<i>Px = Py - data</i>	217
5.3.29	<i>Px = Py data</i>	217
5.3.30	<i>Px = Py ^ data</i>	217
5.3.31	<i>Px = Py & data</i>	217
5.3.32	<i>Ri = Px + data</i>	218
5.3.33	<i>Ri = Px - data</i>	218
5.3.34	<i>Ri = Px data</i>	218
5.3.35	<i>Ri = Px ^ data</i>	218
5.3.36	<i>Ri = Px & data</i>	218
5.3.37	<i>Px = [1 X 0 FD A P Q]</i>	218
5.3.38	<i>[Px.n ...] = [1 0 X Q]</i>	220
5.3.39	<i>Px.n = 1KHz(time)</i>	220
5.3.40	<i>XiL = Px</i>	220

5.3.41	$XiH = Px$	220
5.3.42	$XiL.n = Px.n$	221
5.3.43	$XiH.n = Px.n$	221
5.3.44	$Xi.n = Px.n$	221
5.3.45	$Xi = [Px, Py]$	221
5.3.46	$Px = XiL$	221
5.3.47	$Px = XiH$	221
5.3.48	$Px.n = XiL.n$	221
5.3.49	$Px.n = XiH.n$	222
5.3.50	$Px.n = Xi.n$	222
5.3.51	$[Px, Py] = Xi$	222
5.4	Path Command	222
5.4.1	ASM	222
5.4.2	C-Code	223
5.4.3	BG	224
5.4.4	BreakFG	225
5.4.5	StopFG	226
5.4.6	StopBG	226
5.4.7	StopBG1	226
5.4.8	StopBG2	227
5.4.9	StopBG3	227
5.4.10	StopBG4	227
5.4.11	StopBG5	228
5.4.12	Subroutine	228
5.4.13	Label(Pathname)	228
5.4.14	Macro	228
5.4.15	1ms_RET	229
5.4.16	4ms_RET	229
5.5	Voice Command	230
5.5.1	PlayV / PlayVS	230
5.5.2	WaitVN(Ch)	234
5.5.3	PauseV(Ch)	234
5.5.4	ResumeV(Ch)	235
5.5.5	StopV(Ch)	235
5.5.6	FreqCH = nK	236
5.5.7	96V_Chx_Freq = nK	236
5.5.8	$V\_Chx\_Vol = n / V\_Chx\_Vol = Xi$	237
5.5.9	$Xi = V\_Chx\_Vol$	237
5.5.10	SBC_Loop_On	237
5.5.11	SBC_Loop_Off	238
5.5.12	ADPCM_Loop_On	238
5.5.13	ADPCM_Loop_Off	238
5.5.14	ADPCM_UpSampling	238
5.5.15	ADM_Loop_On	239

5.5.16	ADM_Loop_Off	239
5.5.17	ADM_UpSampling.....	239
5.5.18	PCM_Loop_On	240
5.5.19	PCM_Loop_Off	240
5.5.20	ReadFileCountV.....	240
5.6	Record Command	241
5.6.1	Record / RecordS.....	241
5.6.2	WaitRN	241
5.6.3	StopR	242
5.6.4	EraseR / EraseRS.....	242
5.6.5	WaitEN	242
5.7	Sentence Command.....	243
5.7.1	PlayS(Parameter).....	243
5.7.2	WaitSN(n).....	243
5.7.3	PauseS(n)	244
5.7.4	ResumeS(n)	244
5.7.5	StopS(n)	244
5.8	SPI Play Command	245
5.8.1	SPIPlay / SPIPlayS	245
5.8.2	SPIWaitN(Ch).....	247
5.8.3	SPIStop(Ch)	247
5.8.4	SPIPause(Ch)	247
5.8.5	SPIResume(Ch)	248
5.8.6	SPIVol = n /SPIVol = Ri.....	248
5.8.7	Ri = SPIVol.....	248
5.8.8	SPIGetIndex.....	248
5.9	SPI Flash Command	249
5.9.1	SPI_WREN	249
5.9.2	SPI_WRDIS	250
5.9.3	SPI_RDID(Result, SPIGroup)	250
5.9.4	SPI_CE	251
5.9.5	SPI_SE(Addr, Count, SPIGroup)	252
5.9.6	SPI_BE(Addr, Count, SPIGroup)	253
5.9.7	SPI_DP(SPIGroup)	255
5.9.8	SPI_RDP(Result, SPIGroup)	255
5.9.9	SPI_WRSR(Data, SPIGroup)	256
5.9.10	SPI_RDSR(Result, SPIGroup).....	257
5.9.11	SPI_WRD(Addr, Data, SPIGroup)	257
5.9.12	SPI_RDD(Addr, Result, SPIGroup).....	259
5.9.13	SPI_GetAddr(Index, Result, SPIGroup).....	260
5.10	SPI Command.....	261
5.10.1	SPI_CS_On(SpiGroup).....	261
5.10.2	SPI_CS_Off(SpiGroup).....	262

5.10.3	<i>SPI_TX(Data, SPIGroup)</i>	262
5.10.4	<i>SPI_RX(Result, SPIGroup)</i>	263
5.10.5	<i>SPI_CLKDIV(Divisor, SPIGroup)</i>	263
5.10.6	<i>SPI_CPOL(Polarity, SPIGroup)</i>	264
5.10.7	<i>SPI_CPHA(Phase, SPIGroup)</i>	264
5.11	Embedded Flash Command	264
5.11.1	<i>EF_SE</i>	264
5.11.2	<i>EF_WRD</i>	265
5.11.3	<i>EF_RDD</i>	265
5.11.4	<i>EF_GetAddr</i>	266
5.12	Storage Command	267
5.12.1	<i>Storage_Save</i>	267
5.13	Comparator Command	267
5.13.1	<i>CMP_ON / CMP_ON(Source)</i>	267
5.13.2	<i>CMP_OFF</i>	268
5.13.3	<i>CMP_Read(Rj:Ri) / CMP_Read(Xi)</i>	268
5.13.4	<i>CMP_CNT_ON (Count)</i>	268
5.13.5	<i>CMP_CNT_OFF</i>	268
5.14	Counter Command	269
5.14.1	<i>TMR_ON</i>	269
5.14.2	<i>TMR_OFF</i>	270
5.14.3	<i>TMR_Read(Rj:Ri) / TMR_Read(Xi)</i>	270
5.15	MIDI Command	270
5.15.1	<i>PlayM / PlayMS</i>	270
5.15.2	<i>WaitMN</i>	274
5.15.3	<i>PauseM</i>	274
5.15.4	<i>ResumeM</i>	274
5.15.5	<i>StopM</i>	274
5.15.6	<i>Instrument(Ch, i)</i>	275
5.15.7	<i>M_Chx_Vol = n / M_Chx_Vol = Ri</i>	275
5.15.8	<i>Ri = M_Chx_Vol</i>	276
5.15.9	<i>Tempo + n</i>	276
5.15.10	<i>Tempo - n</i>	276
5.15.11	<i>Tempo++</i>	276
5.15.12	<i>Tempo--</i>	277
5.15.13	<i>TempoRst</i>	277
5.15.14	<i>Tempo = n</i>	277
5.15.15	<i>Tempo(Rj:Ri) / Tempo(Xi)</i>	278
5.15.16	<i>ReadTempo(Rj:Ri) / ReadTempo(Xi)</i>	278
5.15.17	<i>Mute_On(Ch)</i>	278
5.15.18	<i>Mute_Off(Ch)</i>	279
5.15.19	<i>OKON_On / SingleNote_On</i>	279
5.15.20	<i>OKON_Off / SingleNote_Off</i>	280

5.15.21	OKON_Play / SinglePlay	280
5.15.22	OKON_SustainOn	281
5.15.23	OKON_SustainOff	281
5.15.24	OKON_SustainEnd	281
5.15.25	DynamicOn	281
5.15.26	DynamicOff	282
5.15.27	StopMNote	282
5.15.28	MIDI_Pitch(Semitone)	282
5.15.29	Mask_On(Ch)	283
5.15.30	Mask_Off(Ch)	283
5.15.31	ReadFileCountM	283
5.15.32	MIDI_Loop_On	284
5.15.33	MIDI_Loop_Off	284
5.16	Keyboard Command	284
5.16.1	InstNoteOn(Index, Note, Vol) & InstNoteOff(Note)	285
5.16.2	InstNoteAllOff	285
5.16.3	DrumNoteOn(Index, Vol) & DrumNoteOff(Index)	285
5.16.4	NoteVibrato = n / NoteVibrato = Ri	286
5.16.5	Gliss(Note, Semitone, Time)	286
5.16.6	MaxSingleNote = n / MaxSingleNote = Ri	287
5.16.7	LongInst_HoleTime	287
5.16.8	ShortInst_HoldTime	288
5.16.9	KRecord / KRecordS	288
5.16.10	WaitKRN	289
5.16.11	StopKR	289
5.16.12	PlayK / PlayKS	289
5.16.13	WaitKN	290
5.16.14	StopK	290
5.17	Volume Command	291
5.17.1	Vol_Max	291
5.17.2	Vol_Min	291
5.17.3	Vol = n	291
5.17.4	Vol = Ri	291
5.17.5	Vol++	291
5.17.6	Vol--	292
5.17.7	Ri = Vol	292
5.17.8	Px = Vol	292
5.17.9	Vol += n	292
5.17.10	Vol -= n	292
5.17.11	VolX1	292
5.17.12	VolX2	293
5.17.13	CHx_VOL = n	293
5.17.14	PP_Gain = n	293
5.17.15	PP_Gain = Ri	294

5.17.16	PP_Gain++.....	294
5.17.17	PP_Gain--	294
5.17.18	Ri = PP_Gain	294
5.17.19	PGA_Gain = n	294
5.17.20	PGA_Gain = Ri	296
5.17.21	Ri = PGA_Gain	296
5.17.22	Ri = MixCtrl.....	296
5.17.23	Px = MixCtrl.....	296
5.17.24	MixCtrl	296
5.18	TouchKey Command	297
5.18.1	TouchKey_ON.....	297
5.18.2	TouchKey_OFF	297
5.18.3	TouchKey_CLR.....	298
5.18.4	TouchKey_Scan_Slow	298
5.18.5	TouchKey_Scan_Normal	298
5.18.6	TouchKey_Sensitivity (Level).....	299
5.18.7	Calibrate_ON	300
5.18.8	Calibrate_OFF.....	300
5.18.9	AutoJudge_Calibrate.....	300
5.18.10	Enforce_Calibrate_Normal.....	300
5.18.11	Enforce_Calibrate_Sleep	301
5.18.12	Ri = TouchKey(Px)	301
5.18.13	Var = Touchkey_Count(TouchKey)	302
5.18.14	Var = Touchkey_BGCount(TouchKey)	302
5.19	Table Command	302
5.19.1	TableL(TableName, Rx/Xx, Ry/Yy, Ri)	302
5.19.2	TableM(TableName, Rx/Xx, Ry/Yy, Ri)	303
5.19.3	TableH(TableName, Rx/Xx, Ry/Yy, Ri).....	303
5.19.4	Table(TableName, Rx/Xx, Ry/Yy, Rh, Rm, Ri).....	303
5.19.5	TableL(TableName, X, Y, Ri).....	304
5.19.6	TableM(TableName, X, Y, Ri).....	304
5.19.7	TableH(TableName, X, Y, Ri)	305
5.19.8	Table(TableName, X, Y, Rh, Rm, Ri).....	305
5.19.9	TableL(TableName, Rx/Xx, Ry/Yy, Xi)	306
5.19.10	TableH(TableName, Rx/Xx, Ry/Yy, Xi).....	306
5.19.11	Table(TableName, Rx/Xx, Ry/Yy, Xh, Xi)	306
5.19.12	TableL(TableName, X, Y, Xi)	307
5.19.13	TableH(TableName, X, Y, Xi).....	307
5.19.14	Table(TableName, X, Y, Xh, Xi).....	307
5.19.15	Table(TableName, VarX, VarY, VarR)	308
5.20	IR Command	309
5.20.1	IR_TX=data	309
5.20.2	IR_TX(Ri:Rk:Rj:Ri)	309
5.20.3	IR_TX(Xj:Xi)	310

5.20.4	<i>IR_TX_WaitN</i>	310
5.20.5	<i>IR_RX_ON</i>	310
5.20.6	<i>IR_RX_OFF</i>	310
5.20.7	<i>[Rl, Rk, Rj, Ri] = IR_RX</i>	310
5.20.8	<i>[Xj, Xi] = IR_RX</i>	311
5.20.9	<i>IR_RX = data?Path</i>	311
5.20.10	<i>IR_RX != data?Path</i>	311
5.20.11	<i>IR_TX_Busy?Path</i>	311
5.20.12	<i>IR_Carrier_On</i>	311
5.20.13	<i>IR_Carrier_Off</i>	312
5.21	Serial Control TX Command	312
5.21.1	<i>SC_TX(Mode, Parameter)</i>	312
5.21.2	<i>SC_TX(Mode, Rl:Rk:Rj:Ri)</i>	313
5.21.3	<i>SC_TX(Mode, Xj:Xi)</i>	313
5.22	Serial Control Command	314
5.22.1	<i>SC_RX_ON</i>	314
5.22.2	<i>SC_RX_OFF</i>	314
5.22.3	<i>[Rl, Rk, Rj, Ri] = SC_RX</i>	314
5.22.4	<i>[Xj, Xi] = SC_RX</i>	314
5.22.5	<i>SC_RX = data?Path</i>	315
5.22.6	<i>SC_RX != data?Path</i>	315
5.23	I2C Command	315
1.1.1	<i>I2C_TX</i>	315
1.1.2	<i>I2C_RX</i>	315
1.1.3	<i>I2C_RX=data?Path</i>	316
1.1.4	<i>I2C_RX!=data?Path</i>	316
5.23.1	<i>I2C_Ack?Path</i>	316
5.23.2	<i>I2C_Start</i>	316
5.23.3	<i>I2C_Stop</i>	316
5.23.4	<i>I2C_MReadAck</i>	317
5.23.5	<i>I2C_MReadNAck</i>	317
5.23.6	<i>I2C_Reset</i>	317
5.24	UART Command	317
5.24.1	<i>UART_TX</i>	317
5.24.2	<i>UART_TX_WaitN</i>	318
5.24.3	<i>UART_RX</i>	318
5.24.4	<i>UART_RX=data?Path</i>	318
5.24.5	<i>UART_RX!=data?Path</i>	318
5.24.6	<i>UART_TX_Busy?Path</i>	318
5.25	PWMIO Command	318
5.25.1	<i>PWMOut / PWMOutS</i>	319
5.25.2	<i>PlayPWM / PlayPWMS</i>	320
5.25.3	<i>WaitPN</i>	323

5.25.4	StopPWM	324
5.25.5	PausePWM	324
5.25.6	ResumePWM	325
5.25.7	HoldPWM	325
5.25.8	PWMCtrl(@PWME _n , Extension)	326
5.25.9	PWMDuty	326
5.26	Interrupt Command	327
5.26.1	INT_ON	327
5.26.2	INT_OFF	327
5.26.3	INT_RET	328
5.26.4	INT = n	328
5.27	Delay Command	328
5.27.1	Delay(Time)	328
5.27.2	Delay(Rk:Rj:Ri)	329
5.27.3	WaitDN(n)	329
5.27.4	StopD(n)	330
5.27.5	Paused(n)	330
5.27.6	ResumeD(n)	330
5.27.7	SDelay(time)	331
5.28	Action Command	331
5.28.1	PlayA / PlayAS	331
5.28.2	WaitAN(Ch)	333
5.28.3	PauseA(Ch)	333
5.28.4	ResumeA(Ch)	333
5.28.5	HoldA(Ch)	333
5.28.6	StopA(Ch)	333
5.29	LED Strip Command	334
5.29.1	LEDStr_Play / LEDStr_PlayS	334
5.29.2	LEDStr_Stop	335
5.29.3	LEDStr_Clear	335
5.29.4	LEDStr_Brightness	335
5.29.5	LEDSync_Play / LEDSync_PlayS	336
5.29.6	LEDSync_Stop	336
5.29.7	LEDSync_Clear	337
5.29.8	LEDSync_Brightness	337
5.29.9	LEDText_Play / LEDText_PlayS	337
5.29.10	LEDText_Stop	338
5.29.11	LEDText_Clear	338
5.29.12	LEDText_Brightness	338
5.30	QFID Command	338
5.30.1	QFID_GroupID(Ri)	339
5.30.2	QFID_TagId (GroupID, Ri:Rk:Rj:Ri)	339
5.30.3	QFID_TagInput (ID, Ri)	339

5.30.4	QFID_On	339
5.30.5	QFID_Off	340
5.30.6	QFID_SlowOn	340
5.30.7	QFID_SlowOff	340
5.31	WaveID Command	340
5.31.1	WaveID_TX(Ri)	340
5.31.2	WaveID_RX_ON	341
5.31.3	WaveID_RX_OFF	341
5.31.4	WaveID_RX	341
5.32	Sound Localization Command	342
5.32.1	SL_On	342
5.32.2	SL_Off	342
5.32.3	Var=SL_RX	342
5.33	Sound Detect Command	342
5.33.1	SoundDetect_On	343
5.33.2	SoundDetect_Off	343
5.34	Pitch Detect Command	343
5.34.1	PitchDetect_On	343
5.34.2	PitchDetect_Off	344
5.34.3	ReadPitch	344
5.35	RFC Command	344
5.35.1	RFC_On	344
5.35.2	RFC_Off	344
5.35.3	RFC_Level(Ri)	344
5.36	ADC Input Command	345
5.36.1	ADC_Input	345
5.37	VR Command	345
5.37.1	VR State	345
5.37.2	VR_ON	346
5.37.3	VR_OFF	346
5.37.4	VR_VAD=n	347
5.37.5	VR_VAD_On	348
5.37.6	VR_VAD_Off	348
5.37.7	VRGC_Timeout_CLR	348
5.37.8	Ri = VR_HitScore	348
5.37.9	Ri = VR_HitID	349
5.37.10	VR_Loading	349
5.37.11	VT_Training	349
5.37.12	VT_Delete	349
5.37.13	VT_DeleteAll	349
5.37.14	VT_TrainingNum	350
5.38	Sound Effect Command	350
5.38.1	PitchChange	350

5.38.2	PitchChange_Off.....	351
5.38.3	SpeedChange	351
5.38.4	SpeedChange_Off	353
5.38.5	Robot1.....	353
5.38.6	Robot1_Off.....	353
5.38.7	Robot2.....	353
5.38.8	Robot2_Off.....	354
5.38.9	Robot3.....	354
5.38.10	Robot3_Off.....	354
5.38.11	Robot4.....	354
5.38.12	Robot4_Off.....	355
5.38.13	Echo	355
5.38.14	Echo_Off	355
5.38.15	Reverb.....	356
5.38.16	Reverb_Off.....	356
5.38.17	Darth	356
5.38.18	Darth_Off.....	357
5.38.19	AnimalRoar	357
5.38.20	AnimalRoar_Off.....	357
5.39	Real Time Play Command.....	357
5.39.1	RT_Play.....	358
5.39.2	RT_Play_Off.....	358
5.39.3	RT_PitchChange.....	358
5.39.4	RT_PitchChange_Off.....	359
5.39.5	RT_Robot1.....	359
5.39.6	RT_Robot1_Off.....	359
5.39.7	RT_Robot2.....	359
5.39.8	RT_Robot2_Off.....	360
5.39.9	RT_Robot3.....	360
5.39.10	RT_Robot3_Off.....	360
5.39.11	RT_Robot4.....	360
5.39.12	RT_Robot4_Off.....	360
5.39.13	RT_Echo	361
5.39.14	RT_Echo_Off.....	361
5.39.15	RT_Reverb.....	361
5.39.16	RT_Reverb_Off.....	361
5.39.17	RT_Ghost.....	361
5.39.18	RT_Ghost_Off.....	362
5.39.19	RT_Darth.....	362
5.39.20	RT_Darth_Off.....	362
5.39.21	RT_Chorus.....	362
5.39.22	RT_Chorus_Off.....	363
5.39.23	RT_Vol = n / RT_Vol = Var.....	363
5.39.24	Var = RT_Vol.....	363

5.40	Animaltalks Command	363
5.40.1	Animaltalks_On	364
5.40.2	Animaltalks_Off	364
5.40.3	Animaltalks_Record / Animaltalks_RecordS	364
5.40.4	Animaltalks_StopR	365
5.40.5	Animaltalks_Play / Animaltalks_PlayS	365
5.40.6	Animaltalks_Stop	365
5.40.7	Animaltalks_SetVoice	365
5.40.8	Animaltalks_LongSound_On	366
5.40.9	Animaltalks_LongSound_Off	366
5.41	Animalsings Command	367
5.41.1	Animalsings_On	367
5.41.2	Animalsings_Off	367
5.41.3	Animalsings_Record / Animalsings_RecordS	367
5.41.4	Animalsings_StopR	368
5.41.5	Animalsings_Play / Animalsings_PlayS	368
5.41.6	Animalsings_Stop	368
5.41.7	Animalsings_SetVoice	369
5.41.8	Animalsings_LongSound_On	369
5.41.9	Animalsings_LongSound_Off	369
5.41.10	Animalsings_NC_On	370
5.41.11	Animalsings_NC_Off	370
5.41.12	Animalsings_NC_Auto	370
5.42	I/O Expander Command	370
5.42.1	IoExp Input State	370
5.42.2	IoExp Output State	371
5.42.3	IoExp_Key_CLR	371
5.42.4	IoExp_Key_ON	371
5.42.5	IoExp_Key_OFF	372
5.42.6	IoExp_Sleep	372
5.42.7	IoExp_ErrId	372
5.42.8	IoExp_ReadInfo	372
5.42.9	IoExp_ReadSN	373
5.42.10	IoExp_SetInputPullHigh	373
5.42.11	IoExp_SetInputPullLow	373
5.42.12	IoExp_SetInputFloating	373
5.42.13	IoExp_SetOutputHigh	373
5.42.14	IoExp_SetOutputLow	374
5.43	MISC Command	374
5.43.1	Input State	374
5.43.2	Output State	375
5.43.3	Action Mark State	375
5.43.4	Wave Mark State	375
5.43.5	Melody Mark State	376

5.43.6	NoteOn State.....	377
5.43.7	PWM-IO Mark State	377
5.43.8	QFID State	378
5.43.9	Key_CLR.....	378
5.43.10	Key_ON.....	379
5.43.11	Key_OFF.....	379
5.43.12	Stop.....	379
5.43.13	Pause(n).....	380
5.43.14	Resume(n)	380
5.43.15	ReadChannel(Rj:Ri) / ReadChannel(Xi)	380
5.43.16	PauseDown.....	381
5.43.17	ResumeUp	381
5.43.18	Audio_Loop_On	381
5.43.19	Audio_Loop_Off.....	382
5.43.20	Audio_ON.....	382
5.43.21	Audio_OFF.....	382
5.43.22	AudioMode=n.....	382
5.43.23	NoiseFilter_ON(Ch).....	383
5.43.24	NoiseFilter_OFF.....	383
5.43.25	EQ_Filter.....	384
5.43.26	EQ_Filter_Off.....	384
5.43.27	AGC_On.....	385
5.43.28	AGC_Off.....	385
5.43.29	RampUp	385
5.43.30	RampDown	386
5.43.31	AutoSleep_On.....	386
5.43.32	AutoSleep_Off.....	386
5.43.33	Sleep.....	387
5.43.34	WDT_CLR.....	387
5.43.35	Repeat.....	387
5.43.36	Background	387
5.43.37	Slow	388
5.43.38	SlowOff.....	389
5.43.39	END.....	389
5.43.40	ReadRollingCode	389
5.43.41	ChMode(n)	389
5.43.42	ReadRadj(Ri)	390
5.43.43	SW_Reset.....	391
5.43.44	Debounce(Time)	391
5.43.45	Direct_Debounce(Time).....	391
5.43.46	Matrix_Debounce(Time).....	392
5.43.47	Ri = LVD	392
5.43.48	Enforce_Wakeup.....	393
5.43.49	GetMicVol.....	393
5.43.50	Millis	394

5.43.51	<i>Srand</i>	394
5.44	Debug Command	394
5.44.1	<i>Printf</i>	394
6	Appendix	396
6.1	New File Window	396
6.2	Tools	397
6.2.1	<i>Related Software Tools</i>	397
6.2.2	<i>Related Hardware Tools</i>	397
6.3	Insert Control Code to WAV File by Using Quick-IO	399
6.4	Using Q-Sound Functions in <i>Q-Code</i>	400
6.5	Command Table of <i>Q-Code</i>	404
6.5.1	<i>Command Table of NY4</i>	404
6.5.2	<i>Command Table of NY5</i>	407
6.5.3	<i>Command Table of NY5+</i>	410
6.5.4	<i>Command Table of NY6</i>	415
6.5.5	<i>Command Table of NY7</i>	420
6.5.6	<i>Command Table of NY9T</i>	425
6.5.7	<i>Command Table of NX1 OTP</i>	428
-	435	
6.5.8	<i>Command Table of NX1 EF</i>	435
-	443	
6.6	NX1 Audio Channel.....	443
6.7	RAM Resources	445
6.6.1	<i>Command for NY4 RAM Resources</i>	445
6.6.2	<i>Command for NY5 RAM Resources</i>	447
6.6.3	<i>NY5+Memory Allocation</i>	449
6.6.4	<i>NY6 Memory Allocation</i>	449
6.6.5	<i>NY7 Memory Allocation</i>	449
6.6.6	<i>NY9T Memory Allocation</i>	449
6.8	Corresponding Table of Ri and Xi.....	453
6.9	Sample Rate Table	454
6.8.1	<i>NY4 / NY5 Sample Rate Table</i>	454
6.10	Unusable Commands for Special Path	456
6.11	Serial Control Protocol	457
6.12	NY5 / NY5+ Volume Mapping.....	459
6.13	Development Flowchart of <i>Q-Code</i>	460
6.12.1	<i>NY4 Flowchart</i>	460
6.12.2	<i>NY5 Flowchart</i>	460
6.12.3	<i>NY5+ Flowchart</i>	461
6.12.4	<i>NY6 Flowchart</i>	461

6.12.5 NY7 Flowchart.....462

6.12.6 NY9T Flowchart 462

6.12.7 NX1 Flowchart.....463

6.14 Convert To NY5+ 464

7 Revision History 465

1 Introduction

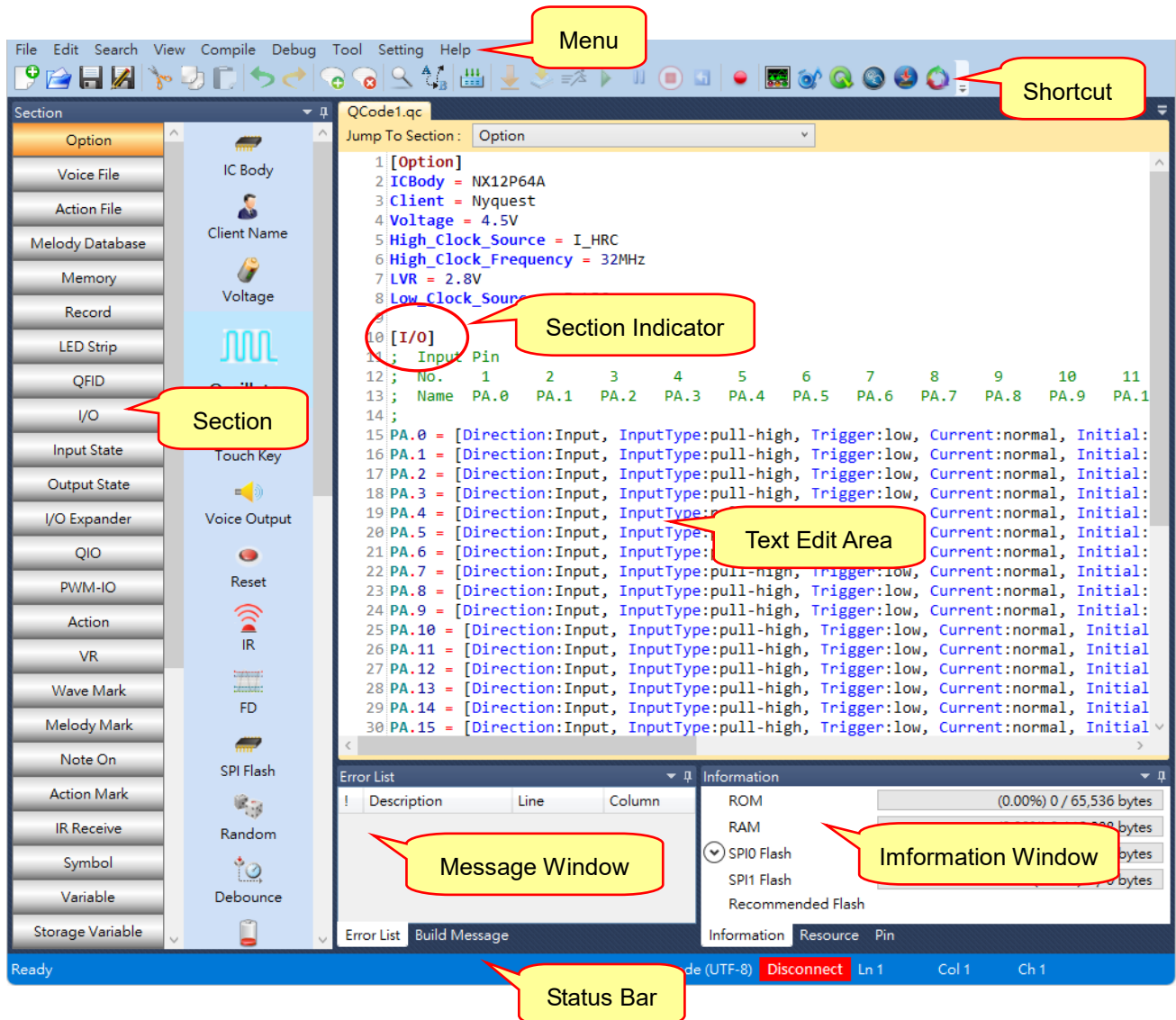
Q-Code is an application development tool for NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1 series IC of Nyquest Technology Co., Ltd. It provides a simple graphical user interface to complete the development of application program. Engineers will be able to utilize the powerful features to complete the development of application program even without the understanding of hardware structure and assembly language. This way the product development process is simplified and the product development efficiency is enhanced.

Please contact Nyquest Technology to obtain the latest installation program. Double click on the installation program to activate installation wizard, and follow the instructions to complete the installation process.

System requirements:

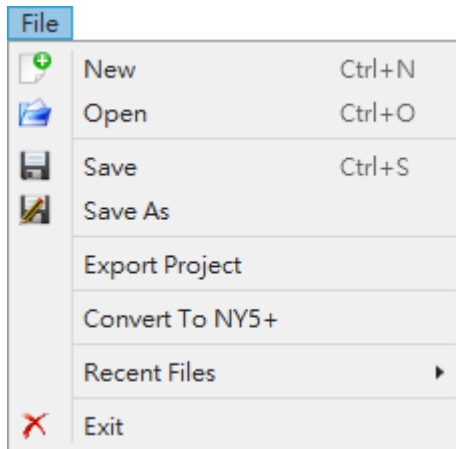
- ◆ A PC compiled with Pentium 1.3GHz or higher CPU, Win7 / Win8 / Win10 / Win11.
- ◆ At least 1G of RAM.
- ◆ At least 2G free space on the hard disk.
- ◆ A display card and monitor that support 1366x768 or higher resolution.
- ◆ Microsoft .Net Framework 4.8 installed.

2 Q-Code Interface



2.1 Function Menu

2.1.1 [File] Menu



New: Create a new .qc file.

Open: Open a .qc file.

Save: Save .qc file.

Save As: Save .qc file as another name or to another path.

Export Project: Export .qc file and the used files to another directory.

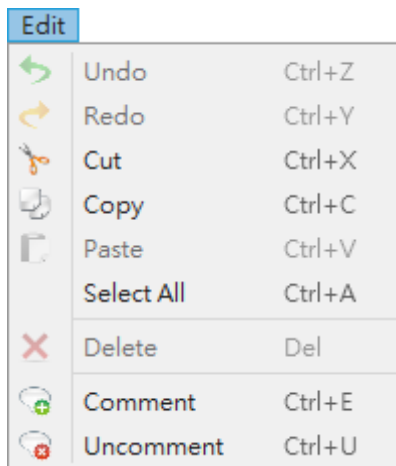
Convert To NY5+: Convert .qc file to NY5+ project. The detail in 6.13.

Note: Only NY5 supports.

Recent Files: Open a recently accessed .qc file. When a file is closed, it will automatically be added to the "Recent Files" list.

Exit: Exit Q-Code.

2.1.2 [Edit] Menu



Undo: Recover the previous step during editing.

Redo: Cancel the previous Undo action during editing.

Cut: Cut out a certain text block during editing.

Copy: Copy out text in a certain block during editing.

Paste: This feature is for pasting cut or copied block.

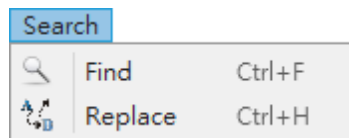
Select All: This feature is for selecting all contents.

Delete: Delete text currently selected by the cursor.

Comment Selection: For making comments on cursor location or selected multiple lines.

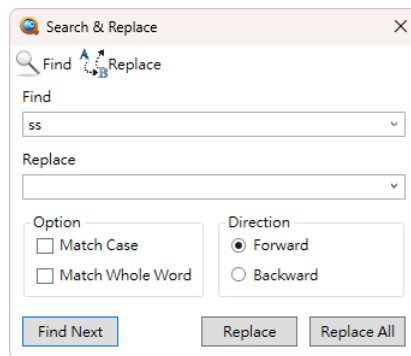
Uncomment Selection: For canceling comments on cursor location or selected multiple lines.

2.1.3 [Search] Menu



Find: This text area is for typing in string or number to be found in program.

Replace: This text area is for replacing current matching string.



Match Case: This feature is for case-sensitive searching.

Match Whole Word: When this function is selected, it will seek words that match to the input text.

(Partially matched words will not be sought).

Forward: This feature is for finding the forward matching string.

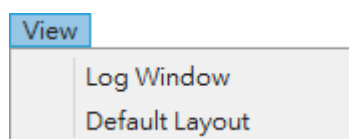
Backward: This feature is for finding the backward matching string.

Find Next: This feature is for finding the next string.

Replace: This button is for replacing current matching string.

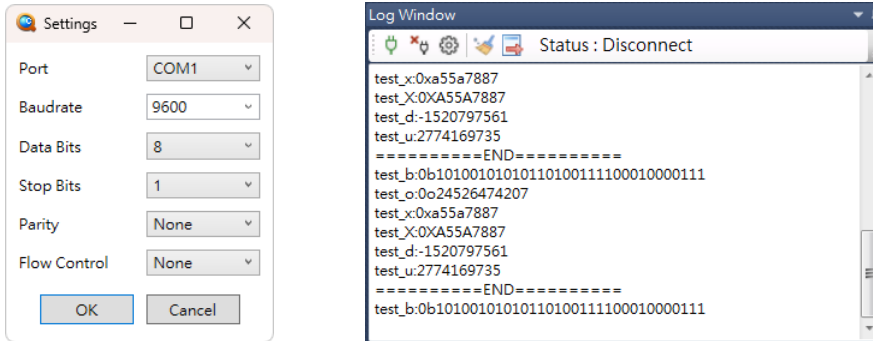
Replace All: This button is for replacing all matching strings.

2.1.4 [View] Menu

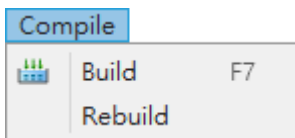


Default Layout: Tick to restore default screen layout.

Log Window: Receive information sent by UART. Use the setting screen (as shown on the below) to set the Port / Baudrate / Parity and other related settings, click the connect button to start data reception.



2.1.5 [Compile] Menu



Build: For compiling current .qc file into .bin file to be downloaded to ICE or Flash Demo Board for verification. In order to speed up compiling time, if the resource file has been built and remains unchanged, it won't be built again. After the completion of compilation, the result will be shown in Build Message window and the ROM usage will be shown in "Information".

Rebuild: No matter the resource files are built or not, the data will be clear and rebuild to the new .bin file.

2.1.6 [Debug] Menu

This feature is for the need of using ICE to debug while users are using Q-Code. For using this feature, ICE or NX_Programmer must be connected to the USB port of the computer first.

Download To SPI Flash: Download data to SPI Flash.

Note: Only NY6 supports.

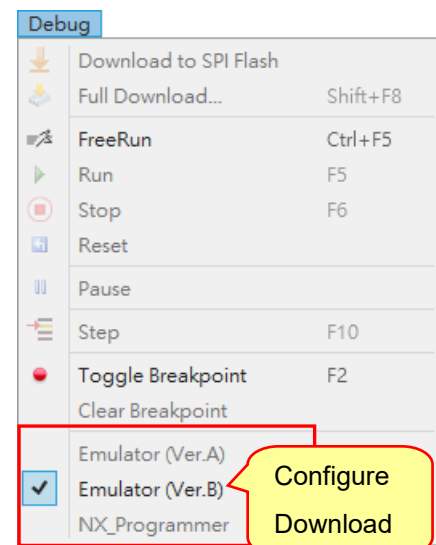
Full Download: Download the full program and data to ICE and SPI Flash. It needs computer to connect the NX_Programmer. User can choose download to ROM, SPI Flash or both.

Note: Only NX1 supports.

Free Run: Execute ICE, and the process cannot be interrupted.

Run: Execute ICE, and the process can be interrupted for debugging.

Note: Only NX1 supports.



Stop: Stop ICE.

Reset: Reset without power-off.

Pause: Pause ICE.

Step: Execute ICE step by step.

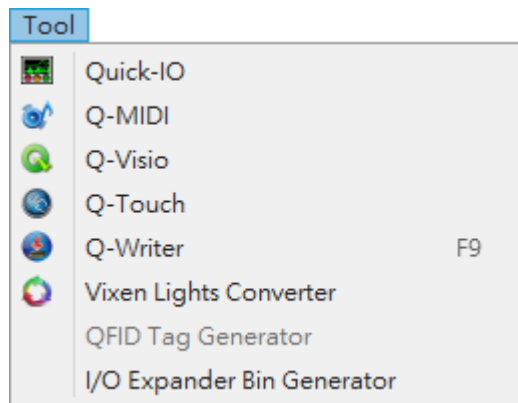
Toggle Breakpoint: Add or delete the breakpoint.

Clear Breakpoint: Remove all breakpoints.

Configure Download: User could download built program to different kinds of demo kits by this setting.

This function is supported by Emulator Ver.A, Emulator Ver.B, and NX_Programmer. After setting, files will be downloaded in accordance with settings when Download executed.

2.1.7 [Tool] Menu



Q-MIDI: This software is used to edit MIDI and timbre. Generated file's extension is .qmd, which can be used in NY5 / NY5+ / NY6 / NY7 / NX1.

Quick-IO: This software is for drawing the signals of output pin. It must be used together with WAV file and the extension name of the generated file is .nyq.

Q-Visio: This software is for drawing the signals of output pin and the extension name of the generated file is .vio.

Q-Touch: This software is for Touch-Key calibration, and the extension of generated file is .t9x and .tnx.

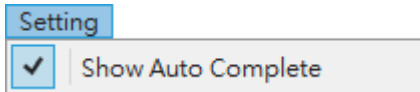
Q-Writer: This software is for programming the BIN file into Flash Demo Board, Romter or OTP for verification.

Vixen Lights Converter : This software is used to convert the .csv file into multi .csv and .vio files.

QFID Tag Generator: This option is for generating the .bin file of the QFID application which is used by Tag. Tag Generator is for NY7 / NX1 series only.

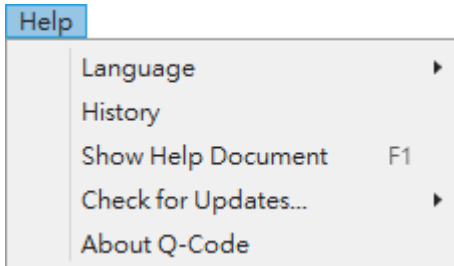
I/O Expander Bin Generator: This tool is used to generate .bin files used cooperates with I/O Expander function.

2.1.8 [Setting] Menu



Show Auto Complete: Tick to enable the auto complete function.

2.1.9 [Help]



Language: Switch the language of Q-Code.

History: See the revision history of Q-Code.

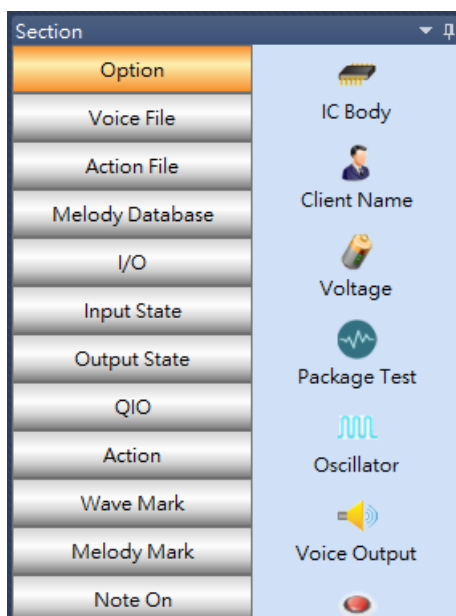
Show Help Document: Display Q-Code Reference Manual.

Check for Updates...: Check for the latest version of Q-Code, NYASM, NYC_NX1. The function needs Internet connection.

About Q-Code: Show current Q-Code version and the contact information for technical support.

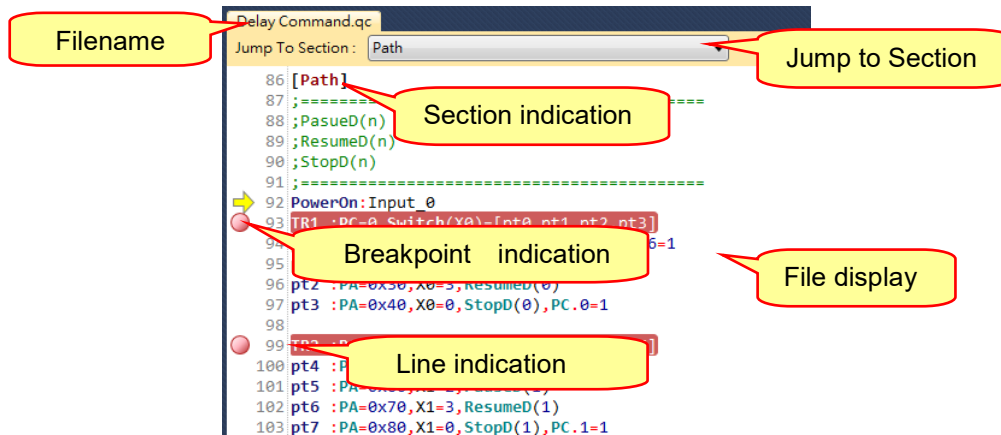
2.2 Section

Sections group multiple user interfaces to help user easily develop Q-Code applications. For detailed operating procedure please refer to the description of individual section.



2.3 Text Edit Area

This area is for user to edit Q-Code program. Similar to the operation of general text editing software, line numbers are shown on the left side of the editing interface, and color-changing function will be carried out according to commands and keywords defined in Q-Code language. Other than main menu and hotkeys, user can also utilize all kinds of editing features through the menu accessed by right-clicking the mouse, and edit Q-Code program using "Section". For detailed information of developing Q-Code application, please refer to Q-Code structure.



Filename: Shows filename.

Breakpoint Indication: The icon of program operation or breakpoint.

Line Indication: Shows the line number.

Jump To Section: Move the cursor to the selected section.

File Display: The file content.

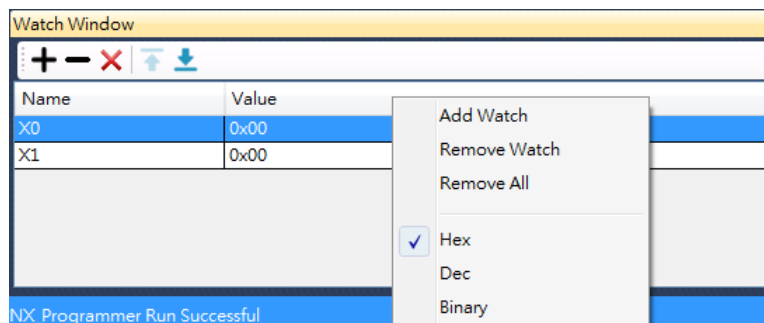
Section Indication: Shows the current sections.

2.4 Debug Mode

This function only supports NX1 series. Press "Run" to enter the debug mode, and then the Watch window and the Register window will appear.

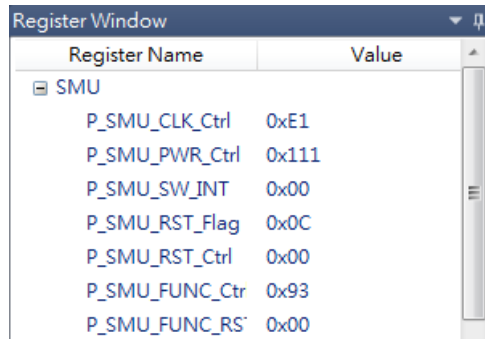
2.4.1 Watch Window

Users can add the target registers or variables to observe for easy tracking.



2.4.2 Register Window

The Register window will show the value of register in the process of debugging, and it updates value only in Pause status.

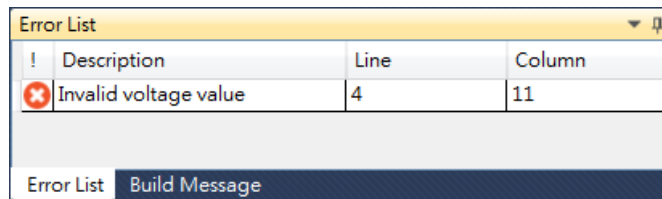


2.5 Message Window

This window is divided into two pages: Error List and Build Message. User can left-click the mouse to switch between two pages.

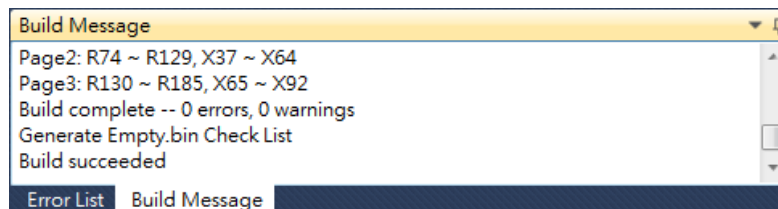
2.5.1 Error List

Error List is for showing all messages delivered to user by editing interface including syntax errors and error messages. User can select and double click an error message and the editing interface will highlight the location at which error might have occurred.



2.5.2 Build Message

Build Message showing all messages for user during processes of compiling and downloading to ICE.



2.6 Information Window

2.6.1 Information

When the compilation is completed, Q-Code will display the status of memory usage in the Information

window. If the ROM capacity is insufficient, it will be shown in red font. Recommended IC is based on the used ROM capacity of the project and provides a suitable IC from the same IC series.

Information	
ROM	(2.40%) 10,213 / 425,984 words
Program ROM	(7.56%) 4,953 / 65,536 words
Data ROM	5,260 words
Voice Files	5,256 words
Recommended IC	NY5Q046A
Information Resource Pin	

If the project IC is NX1 series, the Information window will display Recommended Flash. This is based on the SPI Flash capacity of SPI_Encoder project setting and displays the suitable N25Q series IC body.

Information	
ROM	(24.22%) 63,488 / 262,144 bytes
Records (UDR)	20,480 bytes
RAM	(59.26%) 6,068 / 10,240 bytes
SPI0 Flash	(51.71%) 138,815,821 / 268,435,456 bytes
XP	(0.00%) 0 / 2,096,128 bytes
Data	138,815,821 bytes
Voice Files	138,757,200 bytes
Melody Database	55,750 bytes
SPI1 Flash	(52.68%) 276,218 / 524,288 bytes
Melody Database	275,120 bytes
Recommended Flash	N/A
Information Resource Pin	

2.6.2 Resource

NX1 series, Q-Code will display Timer usage in this window.

Resource	
Name	Description
RTC	None
Timer 0	ADC
Timer 1	None
Timer 3	None
PWMA	None
PWMB	None
Information Resource Pin	

Note: All timers except RTC are exclusively allocated. Manual allocation is required if conflict is occurred.

2.6.3 Pin

Q-Code will display the status of pin usage in this window.

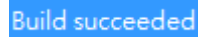
Pin	
Pin	Description
PA.0	Output
PA.1	Output
PA.2	Output
PA.3	Output
PA.4	Output
PA.5	Output
PA.6	Output
PA.7	Output
PA.8	Reset
Information Resource Pin	

2.7 Q-Code Status Bar

This feature is for showing the current status of Q-Code system including Build Message, ICE Status, Row, Column, Character Count and INS.

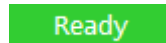
2.7.1 Build Result and ICE Status

After compiling current .qc file into binary file. The result is shown in the following figure.

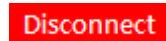
A blue rectangular status bar with the text "Build succeeded" in white.

2.7.2 ICE Connect Status

Show "ICE has connected" as the following figure.

A green rectangular status bar with the text "Ready" in white.

Show "ICE has disconnected" as the following figure.

A red rectangular status bar with the text "Disconnect" in white.

2.7.3 Row, Column, Characters

Show the information of cursor location including Row, Column and Character Count.

A blue rectangular status bar with the text "Ln 4 Col 14 Ch 61" in white.

3 The Architecture of Q-Code

A Q-Code program consists of different sections, which stands for different function. Support of section may vary by IC series. The following table illustrates the supportness of sections for each IC series.

Sections	NY4	NY5	NY5+	NY6	NY7	NY9T	NX1
Option	☒	☒	☒	☒	☒	☒	☒
Voice File	☒	☒	☒	☒	☒	🔒	☒
Action File	☒	☒	☒	☒	☒	☒	☒
Melody Database	🔒	☒	☒	☒	☒	🔒	☒
TouchKey File	🔒	🔒	🔒	🔒	🔒	☒	🔒
Memory	🔒	🔒	🔒	🔒	🔒	🔒	☒
SPI Flash	🔒	🔒	🔒	☒	🔒	🔒	🔒
Record	🔒	🔒	🔒	🔒	🔒	🔒	☒
LED Strip	🔒	🔒	🔒	🔒	🔒	🔒	☒
QFID	🔒	🔒	☒	🔒	☒	🔒	☒
I/O	☒	☒	☒	☒	☒	☒	☒
Input State	☒	☒	☒	☒	☒	☒	☒
Output State	☒	☒	☒	☒	☒	☒	☒
I/O Expander	🔒	🔒	☒	🔒	🔒	🔒	☒
Input State Exp0~3	🔒	🔒	☒	🔒	🔒	🔒	☒
Output State Exp0~3	🔒	🔒	☒	🔒	🔒	🔒	☒
QIO	☒	☒	☒	🔒	🔒	🔒	☒
PWM-IO	🔒	🔒	☒	🔒	🔒	🔒	☒
Action	☒	☒	☒	☒	☒	☒	☒
VR	🔒	🔒	🔒	🔒	🔒	🔒	☒
Action Mark	☒	☒	☒	☒	☒	☒	☒
Wave Mark	☒	☒	☒	🔒	🔒	🔒	☒
Melody Mark	🔒	☒	☒	☒	☒	🔒	☒
Note On	🔒	☒	☒	☒	☒	🔒	☒
PWM-IO Mark	🔒	🔒	☒	🔒	🔒	🔒	🔒
IR Receive	☒	☒	☒	☒	☒	🔒	☒
Symbol	☒	☒	☒	☒	☒	☒	☒
Variable	🔒	🔒	🔒	🔒	🔒	🔒	☒
Storage Variable	🔒	🔒	🔒	🔒	🔒	🔒	☒
Table	☒	☒	☒	☒	☒	☒	☒
ASM	☒	☒	☒	☒	☒	☒	🔒
C-Code	🔒	🔒	🔒	🔒	🔒	🔒	☒
Macro	☒	☒	☒	☒	☒	☒	☒
Sentence	☒	☒	☒	☒	☒	☒	☒

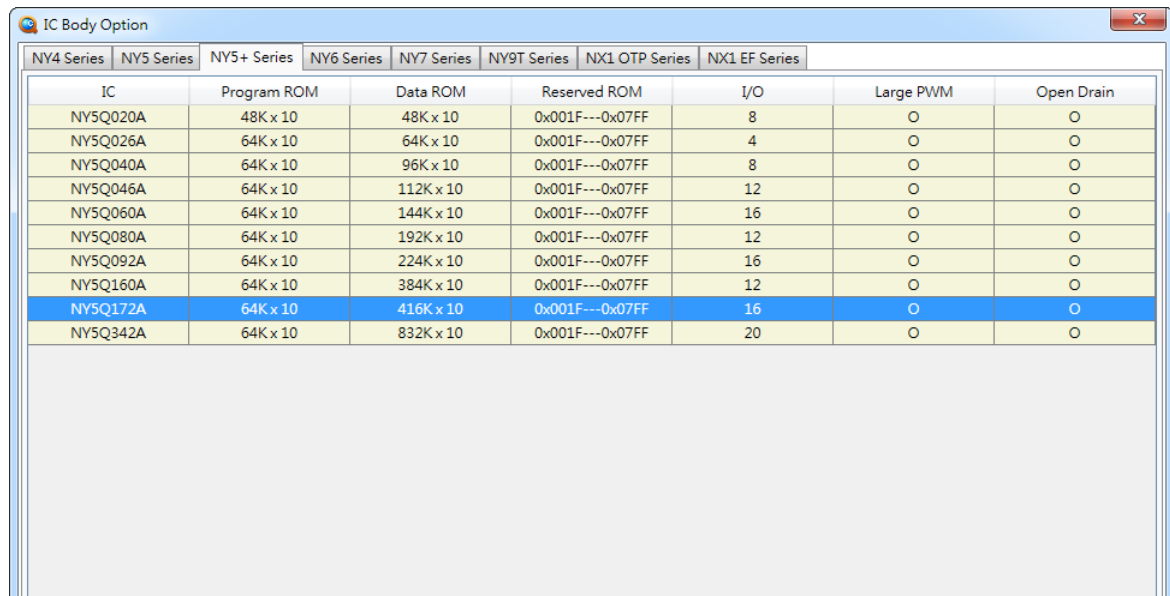
Sections	NY4	NY5	NY5+	NY6	NY7	NY9T	NX1
Subroutine	☑	☑	☑	☑	☑	☑	☑
QIO Custom	☑	☑	☑	🔒	🔒	🔒	🔒
Path	☑	☑	☑	☑	☑	☑	☑
Background1	☑	☑	☑	☑	☑	☑	☑
Background2	☑	☑	☑	☑	☑	☑	☑
Background3	🔒	🔒	🔒	🔒	🔒	🔒	☑
Interrupt	🔒	🔒	🔒	🔒	🔒	🔒	☑

3.1 Option

3.1.1 IC Body

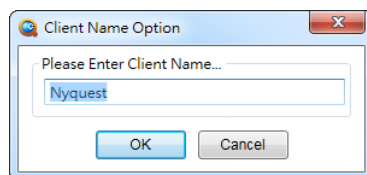
The IC body is selected here.

Ex. ICBODY = NY5Q172A



NY4 Series	NY5 Series	NY5+ Series	NY6 Series	NY7 Series	NY9T Series	NX1 OTP Series	NX1 EF Series
IC	Program ROM	Data ROM	Reserved ROM	I/O	Large PWM	Open Drain	
NY5Q020A	48K x 10	48K x 10	0x001F---0x07FF	8	O	O	
NY5Q026A	64K x 10	64K x 10	0x001F---0x07FF	4	O	O	
NY5Q040A	64K x 10	96K x 10	0x001F---0x07FF	8	O	O	
NY5Q046A	64K x 10	112K x 10	0x001F---0x07FF	12	O	O	
NY5Q060A	64K x 10	144K x 10	0x001F---0x07FF	16	O	O	
NY5Q080A	64K x 10	192K x 10	0x001F---0x07FF	12	O	O	
NY5Q092A	64K x 10	224K x 10	0x001F---0x07FF	16	O	O	
NY5Q160A	64K x 10	384K x 10	0x001F---0x07FF	12	O	O	
NY5Q172A	64K x 10	416K x 10	0x001F---0x07FF	16	O	O	
NY5Q342A	64K x 10	832K x 10	0x001F---0x07FF	20	O	O	

3.1.2 Client



Client Name Option

Please Enter Client Name...

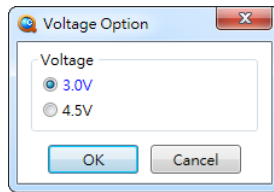
OK Cancel

The name of user or company is shown here. The compilation will not be completed without filling out client name (this is for the protection of rights of program developers).

Ex. CLIENT = Nyquest

3.1.3 Voltage

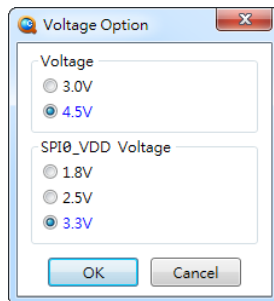
3.1.3.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1EF



Voltage: The working voltage of application on product chip is selected here. (the working frequency of IC at 3.0V will be a little different from that of 4.5V).

Ex. Voltage = 3.0V

3.1.3.2 NX1 OTP



Voltage: The working voltage of application on product chip is selected here. There are options of 3.0V and 4.5V (the working frequency of IC at 3.0V will be a little different from that of 4.5V).

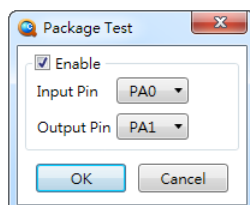
SPI0_VDD Voltage: Select the external SPI flash operating voltage through the SPI0 pin (PB.0~PB.3). There are options of 1.8V, 2.5V, or 3.3V. SPI Flash power will be provided by the internal LDO.

Ex. Voltage = 4.5V

SPI0_VDD_Voltage = 3.3V

3.1.4 Package Test

3.1.4.1 NY4 / NY5 / NY7



This option enables user to verify the Open / Short, Standby Current, OSC frequency deviation and Checksum of packaged ICs.

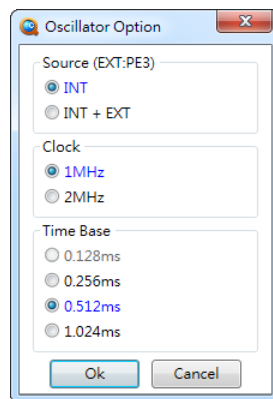
Note:

1. The power-on initialization time of the IC will increase by an additional 80ms after enabling this option.
2. The test program will occupy a small amount of ROM size after enabling the option.
3. To perform IC Package Testing flow, user package method must fit Nyquest standard pin out, user codes must enable Package_Test option with both Package_Test_Input and Package_Test_Output pins available.
4. Because the pin configuration of NY5 cannot be changed after set, the input and output pin configuration must meet the following table

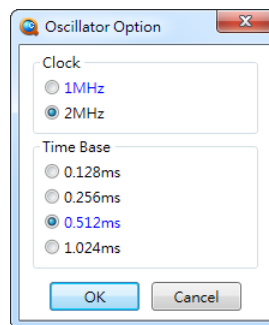
Package test pin	Configuration
Command Input Pin	Direct Input
	Direct Input IO
	Matrix Input
	Matrix Output
	Serial Control (SPI_Like) Clock
	Serial Control (SPI_Like) Data
	Serial Control (IR_Trigger) Data
	IR RX
Signal Output Pin	Direct Input IO
	Direct Output
	Matrix Output

3.1.5 Oscillator

3.1.5.1 NY5



NY5



NY5PxxxB / NY5PxxxJ

Source : INT indicates using internal resistor for OSC, and INT+EXT indicates that OSC can utilize either internal resistor or external resistor. (default INT).

Note: Only NY5 supports the EXT option.

Ex. Oscillator = INT + EXT

Clock: System clock rate. NY5 accepts 1MHz / 2MHz (default 1MHz).

Ex. Clock = 2MHz

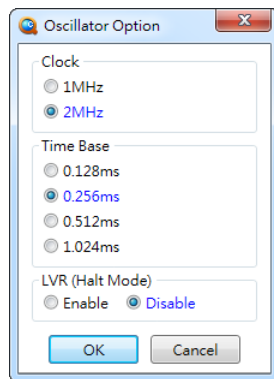
Time Base: Time Base is time counter when Q-Code executes instructions, all the time counts are on the basis of the Time Base. NY5 accepts 0.128ms / 0.256ms / 0.512ms / 1.024ms (default 0.512ms).

Ex. TimeBase = 0.512ms

Note:

1. Due to performance issue, only Clock = 2MHz supports 0.128ms timebase in NY5.
2. Faster timebase counter occupies more CPU loading.
3. Timebase counter is shared with Action counter. When user changes Time Base setting, the Action time counter is also affected.
4. When user changes Time Base setting, it doesn't affect the Melody time counter.

3.1.5.2 NY5+



Clock: System clock rate. NY5+ accepts 1MHz / 2MHz (default 2MHz).

Ex. Clock = 2MHz

Time Base: Time Base is time counter when Q-Code executes instructions, all the time counts are on the basis of the Time Base. NY5+ accepts 0.128ms / 0.256ms / 0.512ms / 1.024ms (default 0.256ms).

Ex. TimeBase = 0.512ms

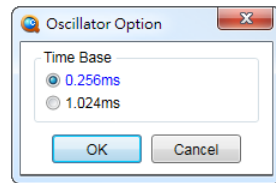
LVR (Halt Mode) : Where to enable or disable LVR function when IC is in Halt Mode.

Ex. LVR_When_Halt = Enable

Note:

1. *Faster timebase counter occupies more CPU loading.*
2. *Timebase counter is shared with Action counter. When user changes Time Base setting, the Action time counter is also affected.*
3. *When user changes Time Base setting, it doesn't affect the Melody time counter.*
4. *HALT mode with LVR ON will increase Standby Current.*

3.1.5.3 NY6



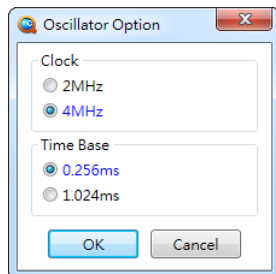
Time Base: Time Base is time counter when Q-Code executes instructions, all the time counts are on the basis of the Time Base. NY6 accepts 0.256ms / 1.024ms (default 0.256ms).

Ex. TimeBase = 0.256ms

Note:

1. *Faster timebase counter occupies more CPU loading.*
2. *Timebase counter is shared with Action counter. When user changes Time Base setting, the Action time counter is also affected.*
3. *When user changes Time Base setting, it doesn't affect the Melody time counter.*

3.1.5.4 NY7



Clock : System clock rate. NY7 accepts 2MHz / 4MHz (default 4MHz)

Ex. Clock = 4MHz

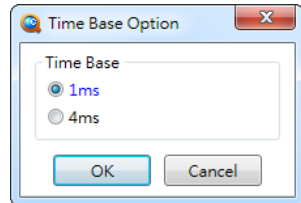
Time Base : Time Base is time counter when Q-Code executes instructions, all the time counts are on the basis of the Time Base. NY7 accepts 0.128ms / 0.256ms / 1.024ms (default 0.256ms).

Ex. TimeBase = 0.256ms

Note:

1. *Faster timebase counter occupies more CPU loading.*
2. *Timebase counter is shared with Action counter. When user changes Time Base setting, the Action time counter is also affected.*
3. *When user changes Time Base setting, it doesn't affect the Melody time counter.*

3.1.5.5 NY9T



Time Base: Time Base is time counter when Q-Code executes instructions, all the time counts are on the basis of the Time Base. NY9T accepts 1ms / 4ms (default 1ms).

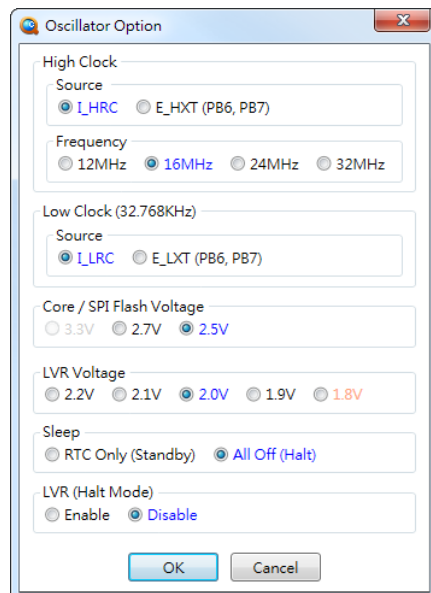
Ex. TimeBase = 1ms

Note:

1. *Faster timebase counter occupies more CPU loading.*
2. *Timebase counter is shared with Action counter. When user changes Time Base setting, the Action time counter is also affected.*
3. *When user changes Time Base setting, it doesn't affect the Melody time counter.*

3.1.5.6 NX1

Oscillator options of NX1 are divided into two groups: High_Clock and Low_Clock. High_Clock and Low_Clock can be used with an external oscillator. However, There is only one external oscillator pin on the IC. Only one of them (High_Clock or Low_Clock) can use an external oscillator.



High_Clock_Source: The clock source of Normal mode. User can choose I_HRC or E_HXT.

High_Clock_Frequency: Normal mode system operating frequency.

Low_Clock_Source: The clock source of low speed mode. User can choose I_HRC or E_HXT and frequency fixed to 32.768KHz.

Core / SPI Flash Voltage: Working voltage of core and external SPI Flash. Accepts 3.3V / 2.8V / 2.7V / 2.5V. Power is provided by internal LDO of NX1.

LVR Voltage: Threshold of LVR Voltage. Available value varies according to High Clock Frequency.

Note: *NX11P21A / NX11S / NX11M / NX12M / NX13M supports Core / SPI Flash Voltage option.*

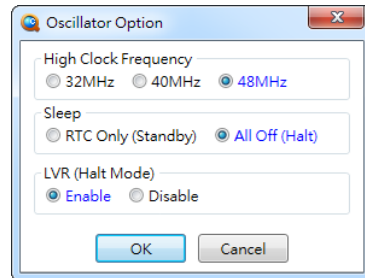
Sleep: Sleep Mode. When set to RTC Only, all timers except RTC_2Hz and RTC_64Hz are stopped. And set to All Off, all timers are stopped, and only IO interrupt can wakeup IC ◦

Ex. Sleep = All_Off

LVR (Halt Mode) : Where to enable or disable LVR function when IC is in Halt Mode. This options is only available when Sleep option is set to All Off.

Ex. LVR_When_Halt = Enable

3.1.5.7 NX1 EF



LVR (Halt Mode) : Where to enable or disable LVR function when IC is in Halt Mode.

Sleep: Sleep Mode. When set to RTC Only, all timers except RTC_2Hz and RTC_64Hz are stopped. And set to All Off, all timers are stopped, and only IO interrupt can wakeup IC ◦

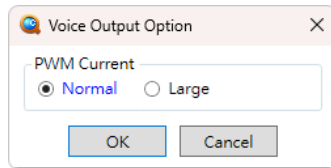
Ex. Sleep = All_Off

LVR (Halt Mode) : Whether to enable or disable LVR function. When IC is in Halt Mode. This options is only available when Sleep option is set to All Off.

Ex. LVR_When_Halt = Enable

3.1.6 Voice Output

3.1.6.1 NY4

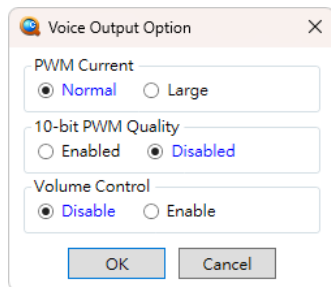


PWM Current: Pulse Width Modulation output current. NY4B series supports Normal / Large.

Note: NY4A does not support the PWM Current option.

Ex. PWM Current = Normal

3.1.6.2 NY4PxxxC



PWM Current: Pulse Width Modulation output current. Supports Normal / Large.

Ex. PWM Current = Normal

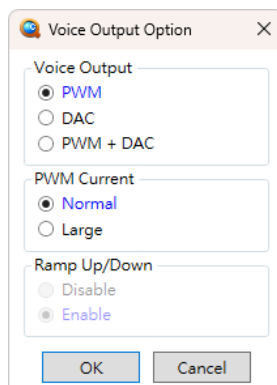
10-Bit PWM Quality: Enable the 10-bit PWM option or not.

Ex. PWM_10Bit = Enable

Volume Control: Enable the Volume Control or not.

Ex. Volume_Control = Enable

3.1.6.3 NY5



Voice Output : Set the driver type of the speaker. The PWM option means PWM output can be connected with speakers directly. The DAC voice output needs to be connected to an amplifier. The PWM+DAC feature is for automatically detecting external device as PWM output or DAC output.

Ex. Voice Output = PWM

PWM Current: Pulse Width Modulation output current.

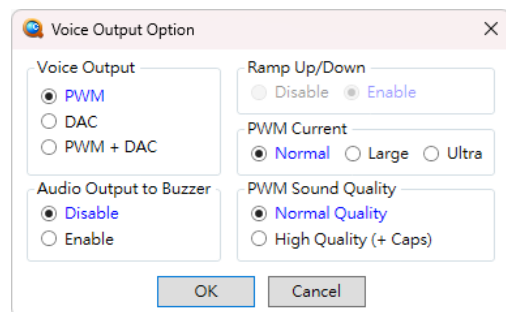
Ex. PWM Current = Normal

Ramp Up/Down: User can turn off Ramp Up/Down function under DAC mode.

Ex. RampUp/Down = Disable

Note: *Not all applications need to disable RampUp/RampDown feature, because it can prevent “pop” sound caused by RampUp/RampDown movements with external OP Amp output.*

3.1.6.4 NY5+



Voice Output : Set the driver type of the speaker. The PWM option means PWM output can be connected with speakers directly. The DAC voice output needs to be connected to an amplifier. The PWM+DAC feature is for automatically detecting external device as PWM output or DAC output.

Ex. Voice Output = PWM

PWM Current: Pulse Width Modulation output current.

Ex. PWM Current = Normal

PWM Sound Quality: Sound quality of PWM output.

Ex. PWM_Sound_Quality = Normal

Ramp Up/Down: User can turn off Ramp Up/Down function under DAC mode.

Ex. RampUp/Down = Disable

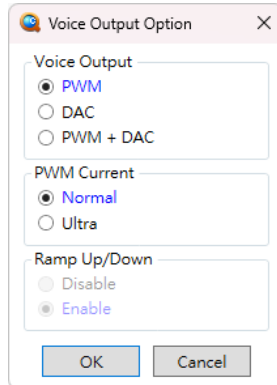
Note: *Not all applications need to disable Ramp Up/Down feature, because it can prevent*

“pop” sound caused by RampUp/RampDown movements with external OP Amp output.

Audio Output to Buzzer : User should enable this option when sound output to Buzzer.

Ex. Audio_Output_Buzzer = Enable

3.1.6.5 NY6



Voice Output : Set the driver type of the speaker. The PWM option means PWM output can be connected with speakers directly. The DAC voice output needs to be connected to an amplifier. The PWM+DAC feature is for automatically detecting external device as PWM output or DAC output. NY6A series only support PWM. NY6B / NY6C series provides the PWM / DAC / PWM+DAC option. (default is PWM).

Note: For NY6 series, the Voice Output only set the initial mode, user can change this setting by using AudioMode command

Ex. Voice Output = PWM + DAC

PWM Current: Pulse Width Modulation output current.

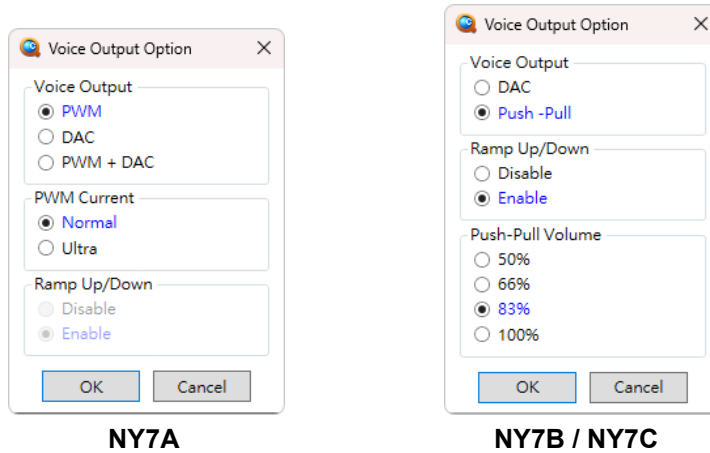
Ex. PWM Current = Ultra

RampUp/Down: User can turn off RampUp / RampDown function under DAC mode.

Ex. RampUp/Down = Disable

Note: Not all applications need to disable RampUp/RampDown feature, because it can prevent “pop” sound caused by RampUp/RampDown movements with external OP Amp output.

3.1.6.6 NY7



Voice Output: Set the driver type of the speaker. The PWM option means PWM output can be connected with speakers directly. The DAC voice output needs to be connected to an amplifier. The PWM+DAC feature is for automatically detecting external device as PWM output or DAC output. Push-Pull is push-pull output. NY7A series provides the PWM / DAC / PWM+DAC option (default is PWM). NY7B / NY7C series provides the DAC and Push-Pull option. Only one output is available in the same time (default is Push-Pull).

Note: For NY7 series, the Voice Output only set the initial mode, user can change this setting by using AudioMode command.

Ex. Voice Output = PWM

Ex. Voice Output = Push-Pull

PWM Current: Pulse Width Modulation output current. NY7A series supports Normal / Ultra.

Ex. PWM Current = Ultra

Note: NY7B / NY7C series do not support the PWM Current setting.

RampUp/Down: User can turn off RampUp / RampDown function under DAC mode.

Ex. RampUp/Down = Disable

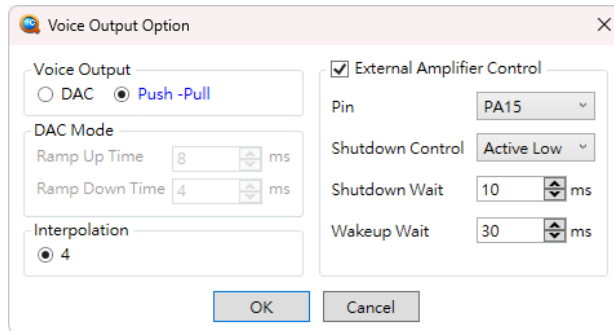
Note: Not all applications need to disable RampUp/RampDown feature, because it can prevent “pop” sound caused by RampUp/RampDown movements with external OP Amp output.

Push-Pull Volume: Built-in Push-Pull gain output, which can drivespeaker directly. Its options are: 100%, 83%, 66%, and 50%, The default is 83%.

Note: NY7A do not support the Push-Pull Volume setting.

Ex. Gain = 83%

3.1.6.7 NX1 OTP



The dialog box titled "Voice Output Option" contains the following settings:

- Voice Output:** Radio buttons for DAC and Push-Pull (selected).
- DAC Mode:**
 - Ramp Up Time: 8 ms
 - Ramp Down Time: 4 ms
- Interpolation:** Radio button for 4 (selected).
- External Amplifier Control:** Checkmark is checked.
 - Pin: PA15
 - Shutdown Control: Active Low
 - Shutdown Wait: 10 ms
 - Wakeup Wait: 30 ms

Buttons: OK, Cancel

Voice Output : Set the driver type of the speaker. The DAC voice output needs to be connected to an amplifier. Push-Pull is the push-pull output.

Ex. Voice Output = Push-Pull

Ramp Up Time: Set the time for Ramp Up. Ramp Up will be finished within the set time.

Ex. RampUp_Time = 8ms

Ramp Down Time: Set the time for Ramp Down. Ramp Down will be finished within the set time.

Ex. RampDown_Time = 4ms

External Amplifier Control: Enable External amplifier control.

Ex. ExtAmp = Enable

Pin: Pin connected to external amplifier.

Ex. ExtAmp_Pin = PA.15

Shutdown Control: Control type of external amplifier, accepts Active High / Active Low (default is Active Low).

Ex. ExtAmp_Shutdown_Control = Active_Low

Shutdown Wait: Set time to turn off the external amplifier.

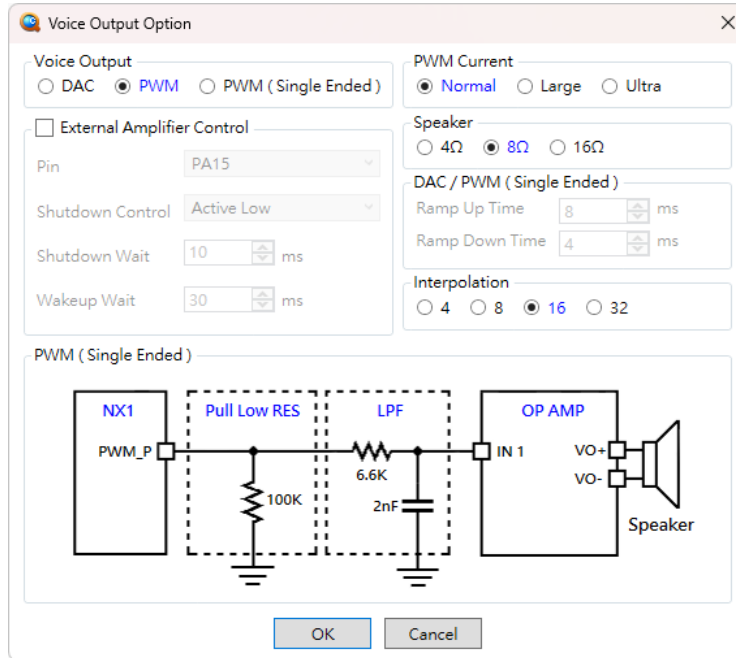
Ex. ExtAmp_Shutdown_Time = 10ms

Wakeup Wait: Set time to wake up the external amplifier.

Ex. ExtAmp_Wakeup_Time = 30ms

Interpolation: Set the samples of interpolation. For NX1 OTP IC, the parameter is 4.

3.1.6.8 NX1 EF



The dialog box titled "Voice Output Option" contains the following settings:

- Voice Output:** DAC (unselected), **PWM** (selected), PWM (Single Ended) (unselected).
- External Amplifier Control:** (unchecked)
 - Pin: PA15
 - Shutdown Control: Active Low
 - Shutdown Wait: 10 ms
 - Wakeup Wait: 30 ms
- PWM Current:** **Normal** (selected), Large (unselected), Ultra (unselected).
- Speaker:** 4Ω (unselected), **8Ω** (selected), 16Ω (unselected).
- DAC / PWM (Single Ended):**
 - Ramp Up Time: 8 ms
 - Ramp Down Time: 4 ms
- Interpolation:** 4 (unselected), 8 (unselected), **16** (selected), 32 (unselected).

Below the settings is a circuit diagram for "PWM (Single Ended)". It shows a block labeled "NX1" with a pin "PWM_P" connected to a "Pull Low RES" (100K resistor to ground). This is followed by an "LPF" (Low Pass Filter) consisting of a 6.6K resistor and a 2nF capacitor to ground. The output of the LPF is connected to the "IN 1" pin of an "OP AMP" (Operational Amplifier). The OP AMP's output is connected to a "Speaker" via "VO+" and "VO-" pins.

Voice Output: Set the driver type of the speaker. The PWM option means PWM output can be connected with speakers directly. The DAC voice output needs to be connected to an amplifier. The PWM (Single Ended) voice output needs to be connected to an amplifier with resistor and capacitors.

Ex. Voice Output = PWM

Ex. Voice Output = PWM_SE

External Amplifier Control: Enable External amplifier control.

Pin: Pin connected to external amplifier.

Shutdown Control: Control type of external amplifier, accepts Active High / Active Low (default is Active Low).

Shutdown Wait: Set time to turn off the external amplifier.

Wakeup Wait: Set time to wake up the external amplifier.

PWM Current: Pulse Width Modulation output current, it supports Normal / Large / Ultra.

Ex. PWM Current = Ultra

Speaker: Set the impedance of speaker.

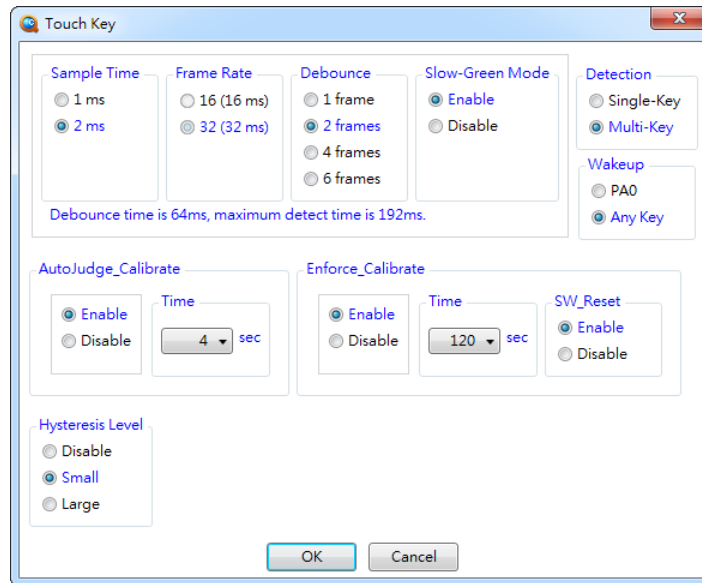
Ramp Up Time: Set the time for Ramp Up. Ramp Up will be finished within the set time.

Ramp Down Time: Set the time for Ramp Down. Ramp Down will be finished within the set time.

Interpolation: Set the samples of interpolation. This setting provides 4 sample options: 4, 8, 16 and 32. The more parameters, the more samples, and vice versa. This default parameter for is 16.

3.1.7 Touch Key

3.1.7.1 NY9T



Sample Time: The duration to scan a Touch-Key. Longer sample time gets more accurate result, but the time to detect a Touch-Key is being touched will be longer. There are 2 options 1ms / 2ms (default 2ms).

Frame Rate: Number of samples to be scanned in a frame, in other words, the period of time to scan all Touch-Keys. The more the number of samples, the less is the power consumption. However, the time to detect a Touch-Key is being touched will be longer. Available values : 8/16 samples (when Sample Time is 1ms), 16/32ms (when Sample Time is 2ms).

Debounce: The number of successive frames for scanning as to detect a Touch-Key has been pressed. More frames get more accurate result, but the time to detect a Touch-Key is being touched will be longer. For those environments with noise interference, it is recommended to select more frames.

Slow-Green Mode: The Slow-Green Mode is only available if there are more than 8 (including 8) Touch-Keys. When using TouchKey_Scan_Slow command to enter slow mode, IC will be in Slow-Green Mode after sleep if Slow-Green Mode is enabled, otherwise IC will be in Slow mode. (default Enabled)

TouchKey_DebounceTime is the time that a Touch-Key is detected as pressed in normal mode; whereas TouchKey_DetectTime is the time that a Touch-Key is detected as pressed in slow mode. The actual detecting time may be shorter. The formula for calculating a Touch-Key detecting time is shown below:

$$\text{Normal Mode} = \text{SampleTime} \times \text{FrameRate} \times \text{Debounce}$$

Slow Mode = SampleTime x FrameRate x (3 + Debounce)

Slow-Green Mode = SampleTime x FrameRate x (4 + Debounce)

Note:

1. *Frame Rate is fixed to 16 samples in NY9T016A.*
2. *NY9T001A / NY9T004A has no Slow-Green Mode setting.*
3. *The Slow-Green Mode is only available as IC enters sleep mode and switches to TouchKey_Scan_Slow.*
4. *In Normal mode, IC will keep in Normal Mode Scan, with or without using TouchKey_Scan_Slow.*

Ex.

TouchKey_DebounceTime = 64ms { SampleTime=2ms, FrameRate=32ms, Debounce=2,
Slow-Green_Mode=Enable }

;The detection time of the touch key in normal mode is 64ms ◦

TouchKey_DetectTime = 192ms

;The max detection time of the touch key in Sleep mode is 192ms ◦

Note:

1. *Sample Time, Frame Rate, Debounce, Slow-Green Mode and Scan Count (Q-Touch Option) will all affect the response time and current consumption of touch keys.*
2. *The Detect Time of the touch key will be 1~4 times the scan time due to the setting of the Scan Count in the Q-Touch option. The key response will be slower, but the key accuracy will be improved.*

Detection: There are 2 kinds of Touch-Keys tracking: Single-Key and Multi-Key. The Single-Key accepts one single touched key at the same time, whereas the Multi-Key accepts multiple touched keys. (Default is Multi-Key)

Ex. TouchKey_Detection = Multi-Key **; Detect multiple touched keys at the same time.**

Wakeup: User can select to wake IC up by PA.0 or any Touch-Keys. (Default is AnyKey)

Ex. TouchKey_Wakeup = PA0 **; Only PA0 key can wake IC up.**

Ex. TouchKey_Wakeup = AnyKey **; Any Touch-Keys can wake IC up.**

Note:

1. *The TouchKey_WakeUp function is not available for NY9T001A.*
2. *User can save energy better by using single key PA.0.*
3. *To activate PA.0 Wakeup function, its necessary to use the TouchKey_Scan_Slow command to enable the Touch-Key slow scan function.*

AutoJudge_Calibrate: Auto calibration of Touch-Keys for every certain period of time. After IC operated for a while, it is suggested to calibrate Touch-Keys. For NY9T,

User can select the auto calibration for every 4 seconds by option, or disable the auto calibration and manually use time path to set a longer time. (Time 0.5 ~ 60 seconds, the default is 4 seconds)

Ex. AutoJudge_Calibrate = 4s ; Auto calibrate Touch-Keys every 4 seconds.

Ex. AutoJudge_Calibrate = Disable ; Disable the auto calibration function.

; Disable the AutoJudge_Calibrate, and calibrate manually by program calculation. (Please refer to 4.2.8 for programmatical calibration time)

Note: When Touch-Key function is disabled, the AutoJudge_Calibrate function is also disabled.

Enforce_Calibrate: Enforce the Touch-Keys to calibrate. To avoid Touch-Keys are touched inadvertently and cause unexpected Touch-Keys triggered, it is suggested to do a Enforce_Calibrate after a certain period of time. User can disable the function and calibrate manually by using time path to count time. (Time 4 ~ 120 seconds, default is 120 seconds)

Ex. Enforce_Calibrate = 60s ; Calibrate Touch-Keys every 60 seconds.

Ex. Enforce_Calibrate = Disable ; Disable the Enforce Calibrate function.

Note: Enforce_Calibrate starts counting after Touch-Key is pressed, and reset counter if Touch-Key is released.

SW_Reset: Determine if Ram and IO is reset after Enforce_Calibrate is done. (default Enable).

Ex. SW_Reset = Enable ; Do a software reset after Enforce_Calibrate

Ex. SW_Reset = Disable ; Do not reset after Enforce_Calibrate

Note: If there is a mode or setting should be memorized after Enforce_Calibrate, user can disable SW_Reset, and manually reset only some status in Enforce_Calibrate path. (Please refer to 4.6.1 for examples of Enforce_Calibrate path)

Hysteresis Level: Set the Touch-Key Hysteresis level. User can adjust the sensitivity of the Touch-Key when the it is pressed, to avoid touching the edge of the pad may cause the Touch-Key in an unstable state. (default Small).

Ex. TouchKey_Hysteresis_Level = Disable

; Do not adjust the sensitivity level after Touch-Key is pressed.

Ex. TouchKey_Hysteresis_Level = Small

; Adjust the sensitivity one level higher after Touch-Key is pressed.

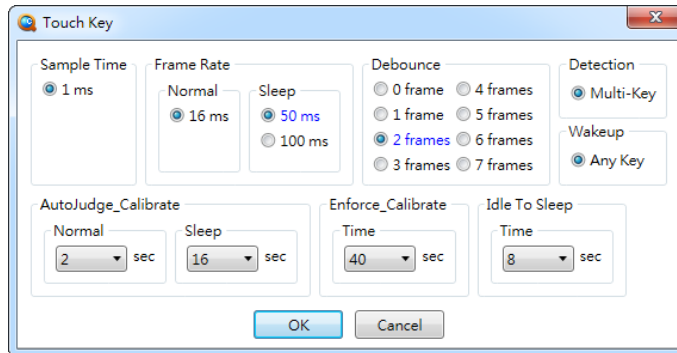
Ex. TouchKey_Hysteresis_Level = Large

; Increase the sensitivity up to 50% after Touch-Key is pressed.

Note:

1. Usually the touched site is far away from the touch pin of IC or the Touch-Key is sensitive, will need to execute the function. Users can actually use the product to decide which one to set.
2. NY9T004A does not support Hysteresis Level function.

3.1.7.2 NX1



Sample Time: The duration to scan a Touch-Key. Longer sample time gets more accurate result, but the time to detect a Touch-Key is being touched will be longer. Only supports 1ms.

Frame Rate: Number of samples to be scanned in a frame, in other words, the period of time to scan all Touch-Keys. The more the number of samples, the less is the power consumption. The Normal Mode only supports 16ms, the Sleep Mode supports 50ms / 100ms.

Debounce: The number of successive frames for scanning as to detect a Touch-Key has been pressed. More frames get more accurate result, but the time to detect a Touch-Key is being touched will be longer. For those environments with noise interference, it is recommended to select more frames.

Detection: The Multi-Key accepts multiple touched keys.

Wakeup: User can wake IC up by any Touch-Keys.

AutoJudge_Calibrate: Auto calibration of Touch-Keys for every certain period of time. After IC operated for a while, it is suggested to calibrate Touch-Keys. The Normal Mode can set every 1~8 seconds to execute calibration (default is 2 seconds). The Sleep Mode can set every 8~60 seconds (default is 16 seconds).

Ex. `TouchKey_AutoJudge_Calibrate = 4s` ; Auto calibrate Touch-Keys every 4 seconds.

Note: When Touch-Key function is disabled, the AutoJudge_Calibrate function is also disabled.

Enforce_Calibrate: Enforce the Touch-Keys to calibrate. To avoid Touch-Keys are touched inadvertently and cause unexpected Touch-Keys triggered, it is suggested to do a Enforce_Calibrate after a certain period of time. User can disable the function and calibrate manually by using time path to count time. (Time 4 ~ 120 seconds, default is 40 seconds)

Ex. TouchKey_Enforce_Calibrate = 60s ; Calibrate Touch-Keys every 60 seconds.

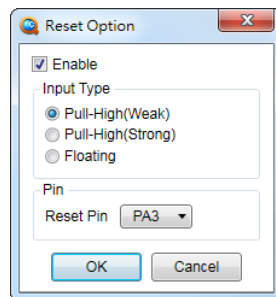
Note: Enforce_Calibrate starts counting after Touch-Key is pressed, and reset counter if Touch-Key is released.

Idle To Sleep: The period for entering the sleep mode. This option can adjust the idle time for entering sleep. NX1 supports 1~16 seconds (default is 8 seconds).

Ex. TouchKey_IdleToSleep = 7s ; After 7 seconds of un-triggered, enter the sleep mode

3.1.8 Reset

3.1.8.1 NY4

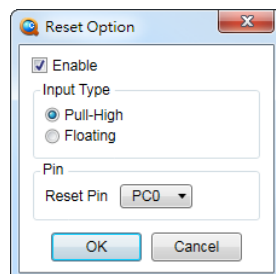


This feature is for configuring pin for externally resetting IC. User can select Pull-High (Weak / Strong) or Floating as Reset pin. If the “Floating” is selected, the external connection of 1M resistance is required (Default value is Disable). The pin of Reset pin is fixed.

Ex.Reset = P:W ; Pull-High Weak Resistor

Ex.Reset = P:S ; Pull-High Strong Resistor

3.1.8.2 NY5 / NY6 / NY7



This feature is for configuring pin for externally resetting IC. User can select Pull-High or Floating

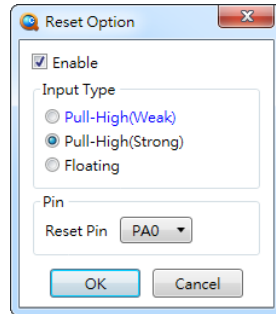
as Reset pin. If the “Floating” is selected, the external connection of 1M resistance is required (default value is Disable).

Ex.Reset = Pull-High

For NY5, user can select the target pin as Reset pin, but for NY6 / NY7, the pin of Reset pin is fixed.

Ex.Reset_Pin = PC.0

3.1.8.3 NY5+



This feature is for configuring pin for externally resetting IC. User can select Pull-High (Weak / Strong) or Floating as Reset pin. If the “Floating” is selected, the external connection of 1M resistance is required (default value is Disable). User can select the target pin as Reset pin,

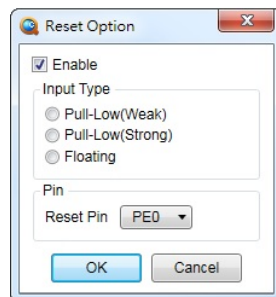
Ex.Reset = P:W ; Pull-High Weak Resistor

Ex.Reset = P:S ; Pull-High Strong Resistor

Ex.Reset = Floating ; Floating

Ex.Reset_Pin = PA.0

3.1.8.4 NY9T



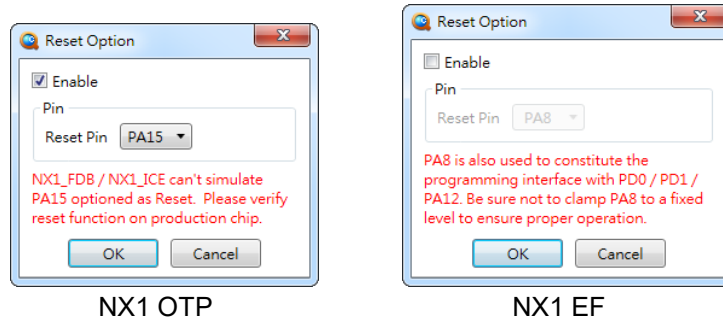
This feature is for configuring pin for externally resetting IC. User can select Pull-Low or Floating as Reset pin. If the “Floating” is selected, the external connection of 1M resistance is required (default value is Disable).

Ex.Reset = P:W ; Pull-Low Weak Resistor

Ex.Reset = P:S ; Pull-Low Strong Resistor

Ex.Reset = F ; Floating

3.1.8.5 NX1



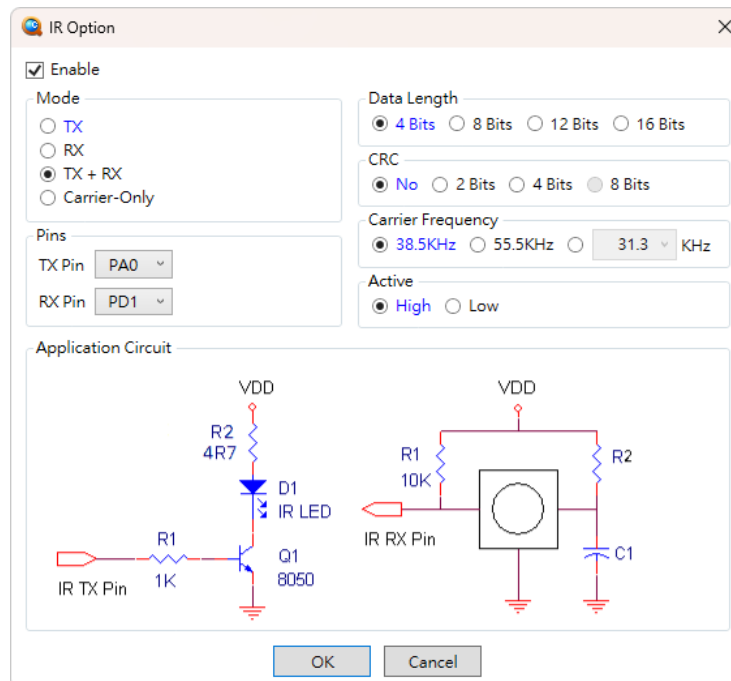
This feature is for configuring pin for externally resetting IC.

Ex. `Reset_Pin` = PA.15

3.1.9 IR

Set the transmission and reception of infrared. System has specific I/O assigned for the transmitting or receiving pin, but user must assign the length of IR coded data and the IR carrier frequency.

3.1.9.1 NY4 / NY5 / NY6 / NY7



Mode: select IR mode. TX / RX or Carrier-Only.

Ex. `IR_Mode` = TX + RX

TX Pin: select TX pin. Only in NY5 / NY5+ can be selected.

Ex. `IR_TX` = PA.0

RX Pin: select RX pin

Ex. IR_RX = PA.3

Data Length: select the length of IR coded data.

Ex. IR_Bit = 8

CRC: select the bit length of IR CRC (Cyclic Redundancy Check), the default is "No".

Ex. IR_CRC = 2

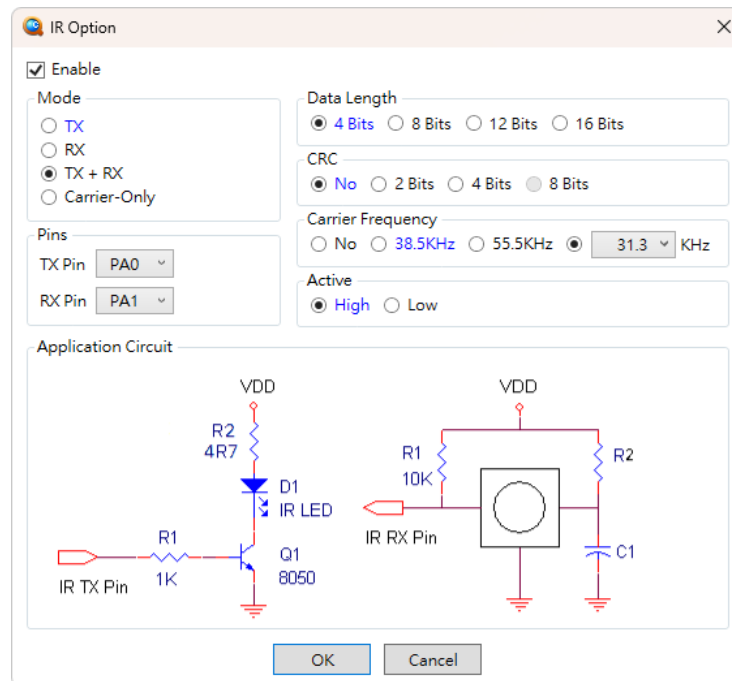
Carrier Frequency: select the IR carrier frequency.

Ex. IR_Carrier_Freq = 38.5K

Active: select NPN or PNP transistors as driver for IR LED.

Ex. IR_Active = High

3.1.9.2 NY5+



Mode: select IR mode. TX / RX or Carrier-Only.

Ex. IR_Mode = TX + RX

TX Pin: select TX pin.

Ex. IR_TX = PA.0

RX Pin: select RX pin

Ex. IR_RX = PA.3

Data Length: select the length of IR coded data.

Ex. IR_Bit = 8

CRC: select the bit length of IR CRC (Cyclic Redundancy Check), the default is “No”.

Ex. IR_CRC = 2

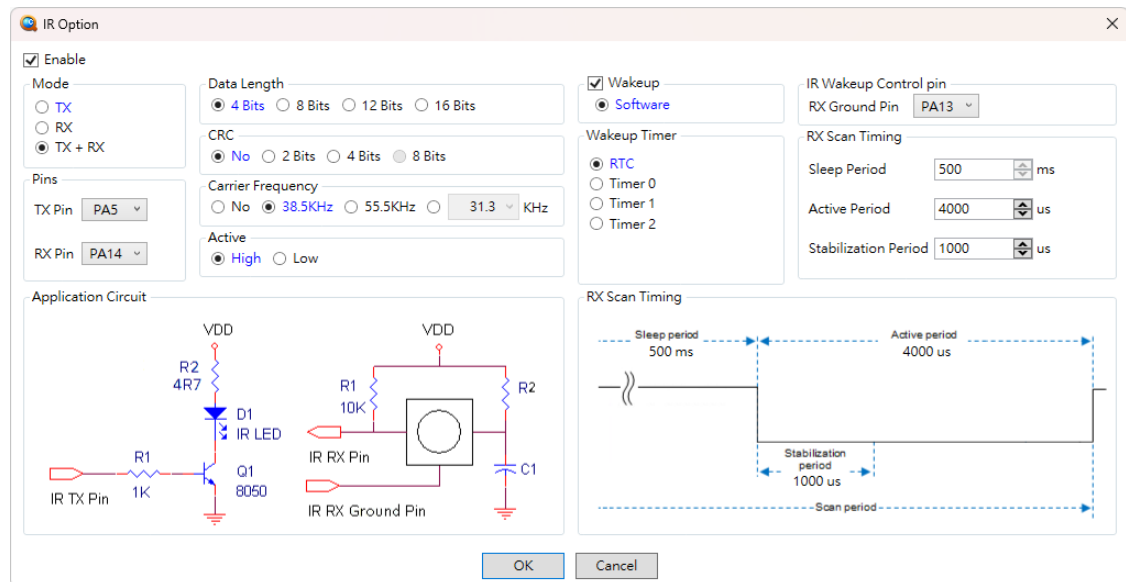
Carrier Frequency: select the IR carrier frequency.

Ex. IR_Carrier_Freq = Disable

Active: select NPN or PNP transistors as driver for IR LED.

Ex. IR_Active = High

3.1.9.3 NX1 OTP



Mode: select TX or RX mode.

Ex. IR_Mode = TX + RX

TX Pin: select TX pin.

Ex. IR_TX = PA.0

RX Pin: select RX pin

Ex. IR_RX = PA.3

Data Length: select the length of IR coded data.

Ex. IR_Bit = 8

CRC: select the bit length of IR CRC (Cyclic Redundancy Check), the default is “No”.

Ex. IR_CRC = 2

Carrier Frequency: select the IR carrier frequency.

Ex. IR_Carrier_Freq = Disable

Active: select NPN or PNP transistors as driver for IR LED.

Ex. IR_Active = High

Wakeup: enable IR Wakeup mode.

Ex. IR_Wakeup = Enable

IR_Wakeup_Type = SW

Wakeup Timer: select IR Wakeup timer.

Ex. IR_Wakeup_Timer = RTC

IR Wakeup Control Pin: select IR Wakeup timer.

Ex. IR_Wakeup_Pin = PA.3

Sleep Period (RX Scan Timing): select RX Sleep Period.

Ex. IR_Wakeup_Sleep_Period = 500ms

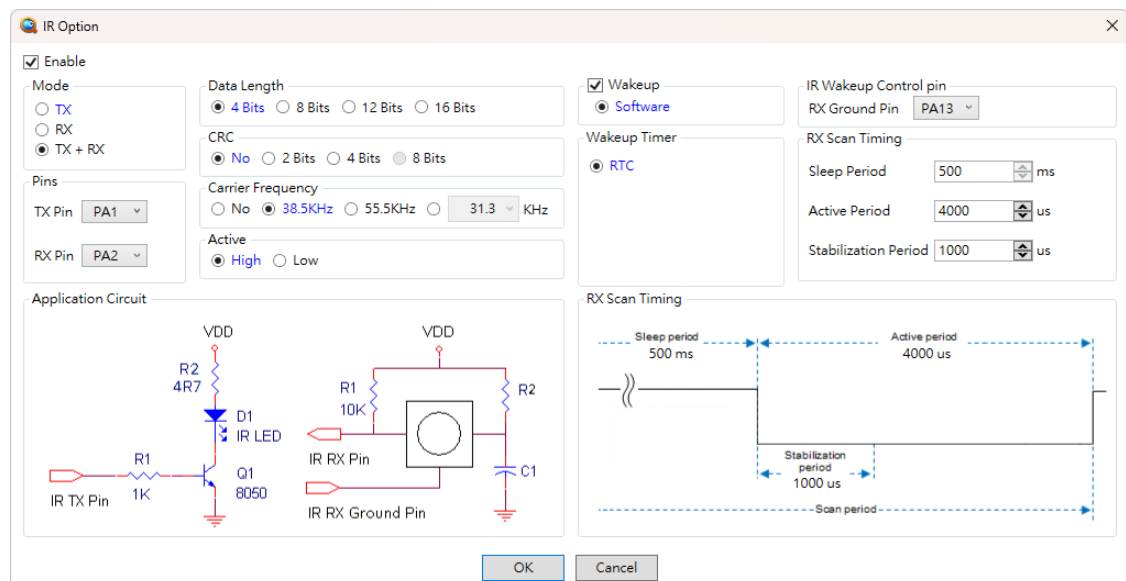
Active Period (RX Scan Timing): select RX Active Period.

Ex. IR_Wakeup_Active_Period = 1000us

Stabilization Period (RX Scan Timing): select RX Stabilization Period.

Ex. IR_Wakeup_Stabilization_Period = 4000us

3.1.9.4 NX1 EF



Mode: select TX or RX mode.

Ex. IR_Mode = TX + RX

TX Pin: select TX pin.

Ex. IR_TX = PA.0

RX Pin: select RX pin

Ex. IR_RX = PA.3

Data Length: select the length of IR coded data.

Ex. IR_Bit = 8

CRC: select the bit length of IR CRC (Cyclic Redundancy Check), the default is “No”.

Ex. IR_CRC = 2

Carrier Frequency: select the IR carrier frequency.

Ex. IR_Carrier_Freq = 38.5K

Active: select NPN or PNP transistors as driver for IR LED.

Ex. IR_Active = High

Wakeup: enable IR Wakeup mode.

Ex. IR_Wakeup = Enable

IR_Wakeup_Type = SW

Note: *NX11FS20A / NX11FS21A / NX11FS22B support HW IR Wakeup.*

Wakeup Timer: select IR Wakeup timer.

Ex. IR_Wakeup_Timer = RTC

Note:

- 1. NX11FS20A / NX11FS21A / NX11FS22B supports Timer 0 / Timer 1.*
- 2. HW IR Wakeup Timer is fixed in PWMA.*

IR Wakeup Control Pin: select RX Ground.

Ex. IR_Wakeup_Pin = PA.3

Sleep Period (RX Scan Timing): select RX Sleep Period.

Ex. IR_Wakeup_Sleep_Period = 500ms

Active Period (RX Scan Timing): select RX Active Period.

Ex. IR_Wakeup_Active_Period = 1000us

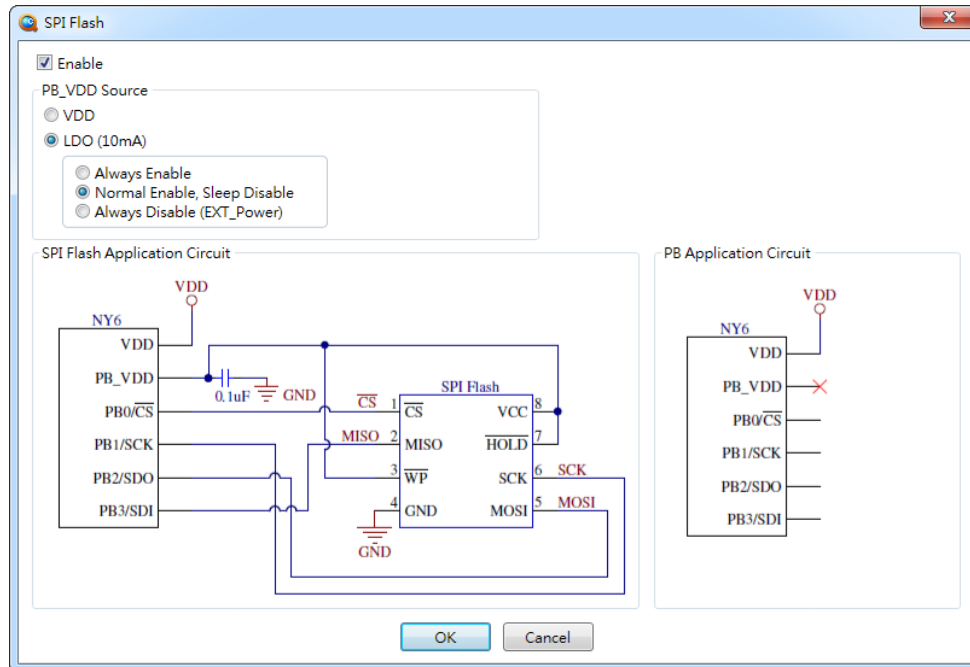
Stabilization Period (RX Scan Timing): select RX Stabilization Period.

Ex. IR_Wakeup_Stabilization_Period = 4000us

3.1.10 SPI Flash

The SPI (Serial Peripheral Interface) Flash. User can play external SPI voice data via the SPI functions in company with *SPI_Encoder* or *Voice_Encoder* v3.30 or newer version.

3.1.10.1 NY6



PB_VDD Source: source of PB power.

1. VDD: PB power is provided by VDD.

Note: User has to connect PB_VDD to VDD on real chip.

2. LDO Always Enable: PB power is provided by IC internal.
3. LDO Normal Enable, Sleep Disable: PB power is provided by IC internal, when IC enters Sleep mode, the LDO will be off.
4. LDO Always Disable (EXT_Power): PB power is provided by other external power. If the external power is larger than IC VDD, it may cause leakage current.

Note:

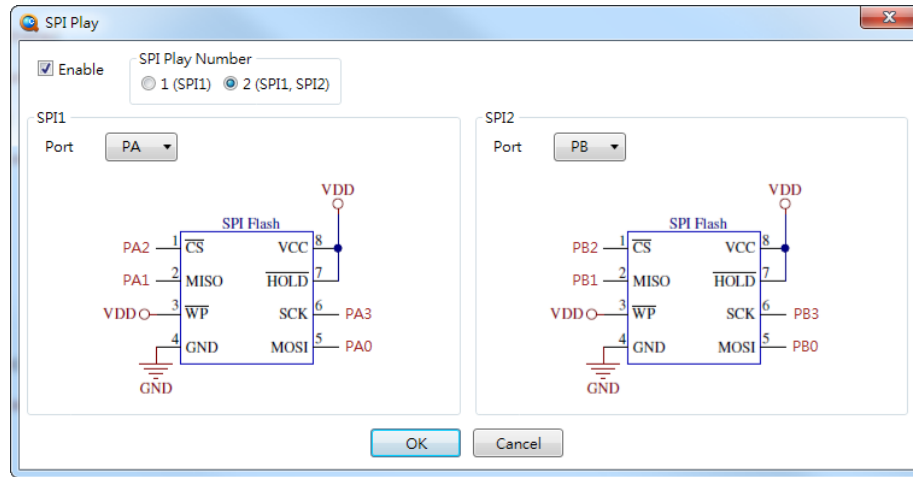
1. A 0.1uF capacitor nearby PB_VDD pin is necessary to stabilize the voltage if PB is applied to SPI Flash and using the power provided by IC.
2. Currently only PB pins support SPI, the pin setting is as follows:
CSB: PB.0 SCK: PB.1 MOSI: PB.2 MISO: PB.3
3. If PB is used as general IO instead of SPI, the source of PB power can also be selected.

Ex.

SPI = Enable

PB_VDD = VDD

3.1.10.2 NY7



SPI Paly Number: supports 2 set of SPI.

Note: Currently only supports Dual Mode SPI. SPI connect pins occupy a whole port, the pin setting is as follows:

MOSI: Px.0 MISO: Px.1 CSB: Px.2 CLK: Px.3

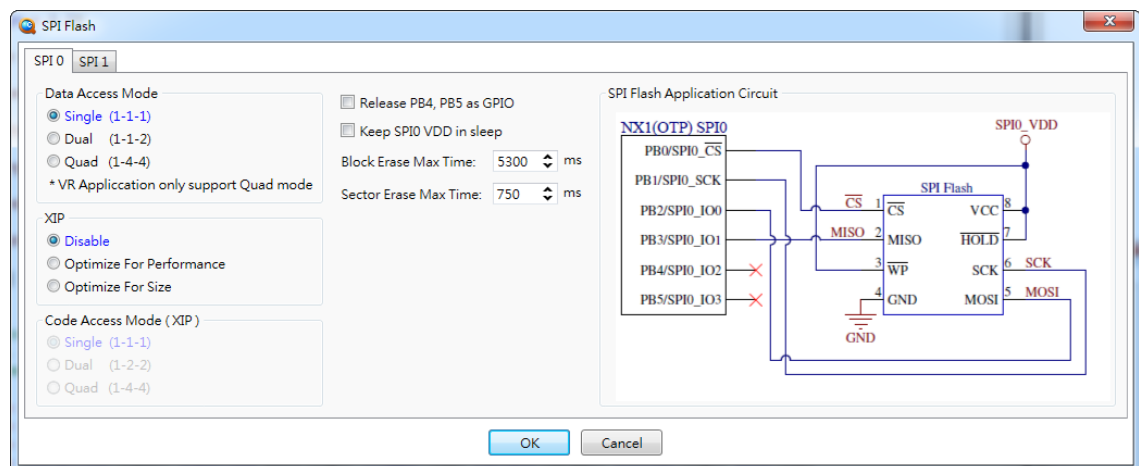
Ex.

SPI1 = PA

SPI2 = PB

3.1.10.3 NX1 OTP

• SPI 0



Data Access Mode : SPI0 can select Single / Dual / Quad.

Ex. SPI0_Data_Access_Mode = Single

XIP : There are 3 selections Disable / Optimze For Performance / Optimize For Size.

1. The Optimize For Performance options can place partial programs and data in SPI Flash.

- The Optimize For Size options can place more programs to SPI Flash but resulting in the increased loading of CPU.

Ex. XIP = Speed_Optimize

Code Access Mode : There are 3 selections Single / Dual / Quad.

Note: Please select Quad while using XIP. The execution speed will be faster than Single and Dual, and the loading of CPU could be relatively reduced

Ex. SPI_Code_Access_Mode = Quad

Release PB4, PB5 as GPIO : When the Quad option is not selected for setting Data Access Mode and Code Access Mode, the Release PB4, PB5 as GPIO will enable PB.4 and PB.5 for GPIO.

Ex. Release_PB4_PB5 = Disable

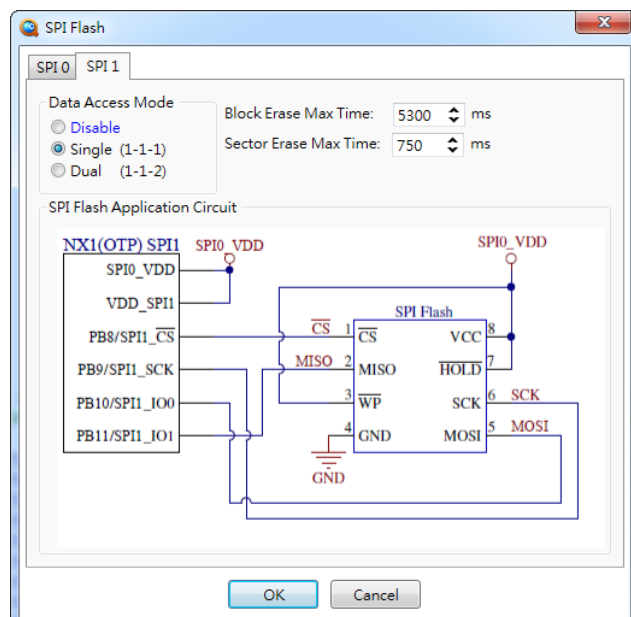
Block Erase Max Time : When the Flash wait timeout occurs using the Block Erase command, it will stop waiting and return immediately. It is recommended to set the appropriate time according to the Flash specification.

Ex. SPI0_BE_MaxTime = 5300ms

Sector Erase Max Time : When the Flash wait timeout occurs using the Sector Erase command, it will stop waiting and return immediately. It is recommended to set the appropriate time according to the Flash specification.

Ex. SPI0_SE_MaxTime = 750ms

● SPI 1



Data Access Mode : SPI1 can select Disable / Single / Dual.

Ex. SPI1_Data_Access_Mode = Single

Block Erase Max Time : When the Flash wait timeout occurs using the Block Erase command, it will stop waiting and return immediately. It is recommended to set the appropriate time according to the Flash specification.

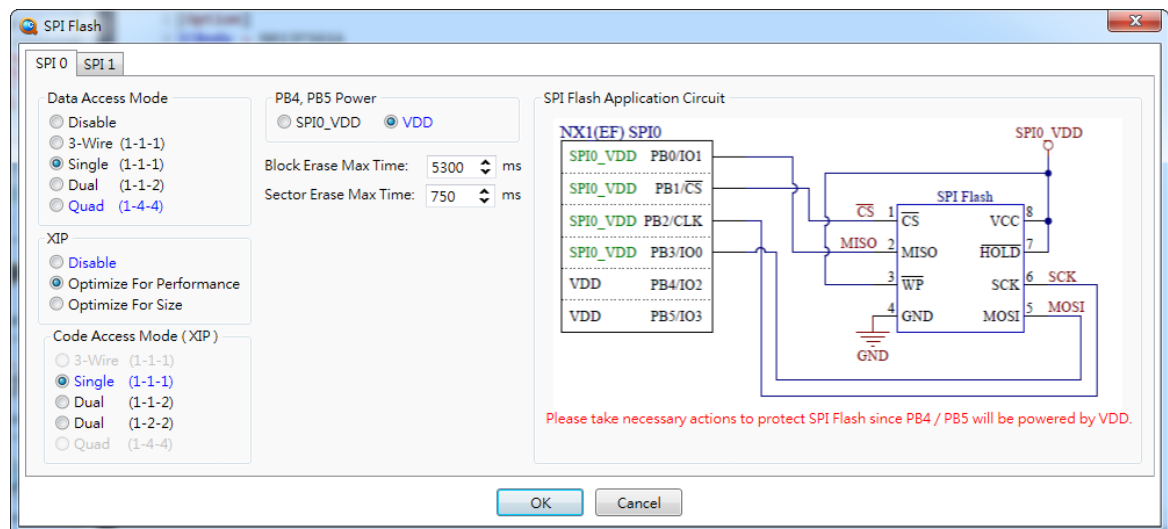
Ex. SPI1_BE_MaxTime = 5300ms

Sector Erase Max Time : When the Flash wait timeout occurs using the Sector Erase command, it will stop waiting and return immediately. It is recommended to set the appropriate time according to the Flash specification.

Ex. SPI1_SE_MaxTime = 750ms

3.1.10.4 NX1 EF

● SPI 0



Data Access Mode : SPI0 can select Disable / 3-Wire / Single / Dual / Quad mode.

Ex. SPI0_Data_Access_Mode = Single

XIP : There are 3 selections Disable / Optimize For Performance / Optimize For Size.

1. The Optimize For Performance options can place partial programs and data in SPI0 Flash.
2. The Optimize For Size options can place more programs to SPI Flash but resulting in the increased loading of CPU.

Ex. XIP = Speed_Optimize

Code Access Mode : There are 5 selections Disable / 3-Wire / Single / Dual / Quad.

Note: Please select Quad while using XIP. The execution speed will be faster than Single and Dual, and the loading of CPU could be relatively reduced

Ex. SPI0_Code_Access_Mode = Quad

PB4, PB5 Power : When the 3-Wire, Single or Dual option is selected for setting Data Access Mode and Code Access Mode, can select source of PB.4 and PB.5 power.

Ex. PB45_Power_Source = SPI0_VDD

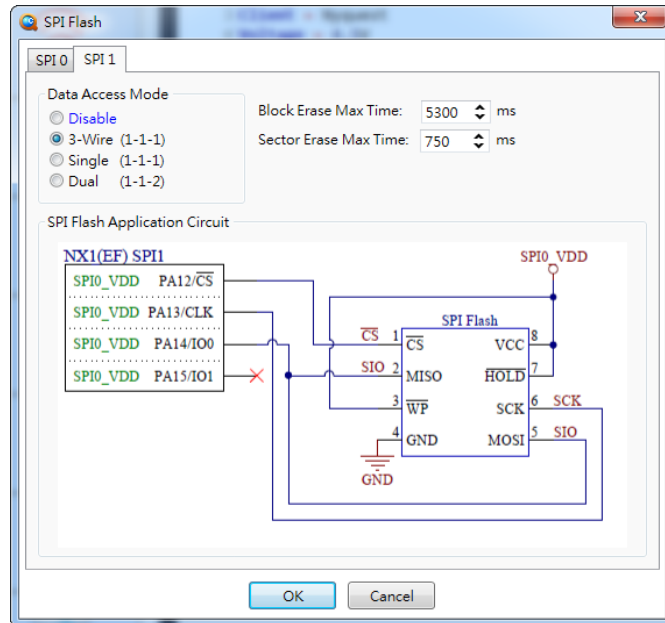
Block Erase Max Time : When the Flash wait timeout occurs using the Block Erase command, it will stop waiting and return immediately. It is recommended to set the appropriate time according to the Flash specification.

Ex. SPI0_BE_MaxTime = 5300ms

Sector Erase Max Time : When the Flash wait timeout occurs using the Sector Erase command, it will stop waiting and return immediately. It is recommended to set the appropriate time according to the Flash specification.

Ex. SPI0_SE_MaxTime = 750ms

● SPI 1



Data Access Mode : SPI1 can select Disable / 3-Wire / Single / Dual mode.

Ex. SPI1_Data_Access_Mode = Single

Block Erase Max Time : When the Flash wait timeout occurs using the Block Erase command, it will stop waiting and return immediately. It is recommended to set the appropriate time according to the Flash specification.

Ex. SPI1_BE_MaxTime = 5300ms

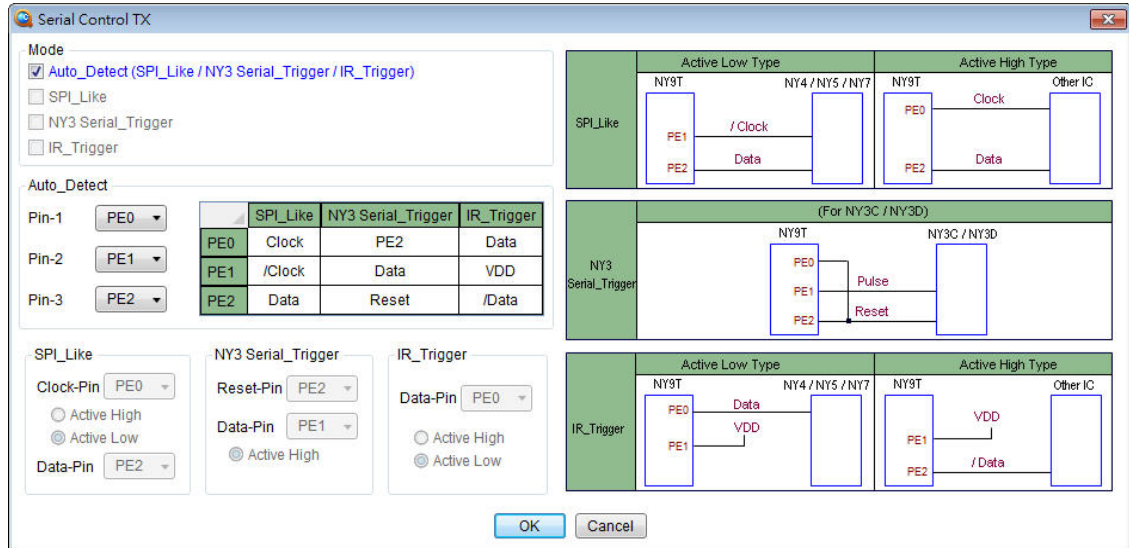
Sector Erase Max Time : When the Flash wait timeout occurs using the Sector Erase command, it will stop waiting and return immediately. It is recommended to set the appropriate time according to the Flash specification.

Ex. SPI1_SE_MaxTime = 750ms

3.1.11 Serial Control TX

3.1.11.1 NY9T

User can set the NY9T as the keys of MCUs. Q-Code provides 3 transfer protocols: SPI_Like, NY3 Serial Trigger, IR_Trigger. User can set to auto detect or specific transfer protocol



Auto Detect: Using setting of 3 pins to decide which transfer protocol NY9T uses. Auto detect pin settings are as below table.

AutoDetect Mode			
Mode	Pin-3	Pin-2	Pin-1
IR_Trigger	X	VDD	X
NY3 Serial_Trigger	Pin-3 connected to Pin-1	X	Pin-3 connected to Pin-1
SPI_Like	No Connected	No Connected	No Connected

Ex. SerialControl = Auto

SerialControl_Pin1 = PF.1

SerialControl_Pin2 = PF.2

SerialControl_Pin3 = PF.3 ; Auto detect communication protocol.

Note: When using Auto mode, 3 pins must use the same port.

SPI_Like: Use 2 pins to achieve a SPI-like communication protocol.

Ex. SerialControl = SPI_Like

SerialControl_SPI_Clock = PF.2

SerialControl_SPI_Data = PF.3

SerialControl_SPI_ClockTrigger = High or Low ; Triggered by rising or falling edge.

Note: When using SPI_Like mode, 2 pins must use the same port.

NY3 Serial_Trigger: User can connect NY3 via the serial signal, and NY9T acted as NY3 trigger buttons.

Ex. SerialControl = NY3

SerialControl_NY3_Reset = PF.3

SerialControl_NY3_Data = PF.2

IR_Trigger: Use a pin to simulate IR transmission signal, and can be directly used to connect current NY4/5/7 series with IR communication.

Ex. SerialControl = IR_Trigger

SerialControl_IR_Data = PF.3

SerialControl_IR_DataBusy = High or Low ; **triggered by rising or falling edge**

Note:

1. *Although there are multiple protocols can be selected, but only one protocol can be used for transmission at the same time.*
2. *If Auto-Detect function is chosen, Pin-1 will be set to Input Floating, and Pin-2/Pin-3 will set to Input Weak-Pull-Low.*
3. *If Auto-Detect function is chosen, the connecting pin of receiving IC must be set to Input floating.*
4. **NY9T001A / NY9T004A do not support Serial Control function.**

3.1.12 Serial Control RX

3.1.12.1 NY4 / NY5 / NY5+ / NY6 / NY7

Receive serial control data from external source. Q-Code supports SPI_Like and IR_Trigger protocol.

SPI_Like: uses 2 pins, use protocol similar to standard SPI.

Ex. Serial_Rx = SPI_Like

Serial_Rx_Data = PA.1

Serial_Rx_Clock = PA.0

Serial_Rx_ClockTrigger = High

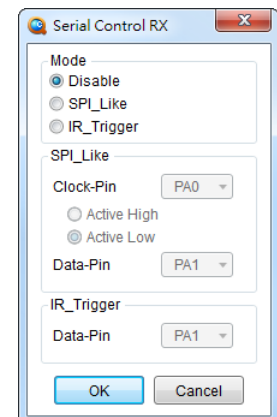
IR_Trigger: uses 1 pin, use IR protocol.

Ex. Serial_Rx = IR_Trigger

Serial_Rx_Data = PF.1

Note:

1. *When using SPI_like mode in NY5+ / NY6 / NY7 series, Clock / Data pin must use the same port.*
2. **IR_Trigger mode cannot be used with IR_RX.**



3.1.13 I2C

3.1.13.1 NY5+

Mode: Q-Code supports I2C, both Master / Slave modes. Q-Code supports transmit speed at 250Hz (about 36ms for transmitting single data).

Device Address: Address when acting as Slave. Accepts 0 ~ 127.

Clock: Clock (SCL) pin.

Data: Data (SDA) pin.

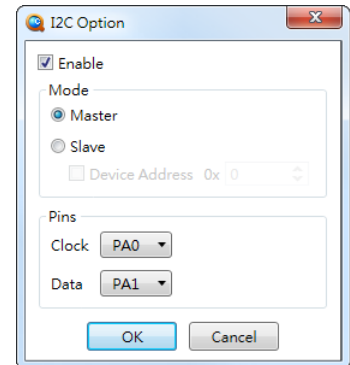
Note: Clock / Data pin must use the same port.

Ex.

I2C_Mode = Master

I2C_Timeout_Check = 30ms

I2C_Speed = Standard



3.1.13.2 NX1

Mode: Q-Code supports I2C, both Master / Slave modes. Q-Code supports Standard (100kHz) / Fast (400kHz) / Fast+ (1MHz).

Timeout Check: When acting as Master, the time slave does not response will be treated as timeout. Specify 0 is treated as never timeout.

Device Address: Address when acting as Slave. Accepts 0 ~ 127.

Device Address Bit: Address bit when acting as Slave, supports 7 / 10 bits.

Clock: Clock (SCL) pin.

Data: Data (SDA) pin.

Speed: Transmit speed, supports Standard / Fast / Fast+. Default is Fast.

Note: When using NX1 as slave, master must supports Clock Stretching, or it may not work correctly.

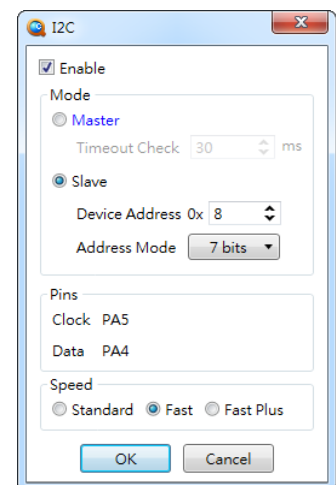
Ex.

I2C_Mode = Slave

I2C_Address = 8

I2C_Address_Bit = 7

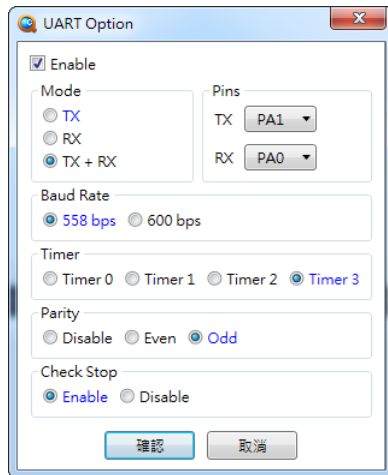
I2C_Speed = Standard



3.1.14 UART

Q-Code supports UART(Universal Asynchronous Receiver / Transmitter) communication protocol.

3.1.14.1 NY5+



Mode: Accepts TX / RX / TX+RX.

Ex. UART_Mode = RX

Pins: TX and RX pin of UART.

Ex. UART_TX = PA.1

Ex. UART_RX = PA.0

Baud Rate: Baud Rate of UART, accepts 558bps / 600bps.

Ex. UART_BaudRate = 558bps

Timer: The timer used as Baud Rate generator.

Ex. UART_Timer_Channel = 1

Parity: Whether to perform parity check, accepts Disable / Even / Odd.

Ex. UART_Parity = Odd

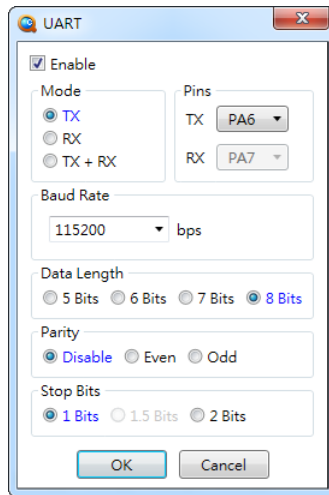
Check Stop: Whether to check Stop Bit.

Ex. UART_Stop_Bit = 2

Note:

1. TX / RX pin must use the same port.
2. When UART is enabled, specified Timer is occupied, and Voice playback or Timer function using same Timer could not work.

3.1.14.2 NX1



Mode: Accepts TX / RX / TX+RX.

Ex. `UART_Mode` = RX

Ex. `UART_Mode` = TX

Ex. `UART_Mode` = TX + RX

Pins: TX and RX pin of UART.

Note:

1. NX1 EF series TX / RX pin must use the same port.
2. NX1 EF series TX also supports PD.0 if PD.0 is set as GPIO.
3. NX1 EF series RX also supports PD.1 if PD.1 is set as GPIO.

Ex. `UART_TX` = PD.0

Ex. `UART_RX` = PA.7

Baud Rate: Baud Rate of UART.

Ex. `UART_BaudRate` = 115200bps

Timer: The timer used as Baud Rate generator.

Ex. `UART_Bit` = 7

Parity: Whether to perform parity check, accepts Disable / Even / Odd.

Ex. `UART_Parity` = Odd

Check Stop: Whether to check Stop Bit.

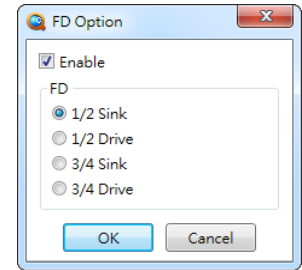
Ex. `UART_Stop_Bit` = 2

3.1.15 FD

3.1.15.1 NY4 / NY5

Flash with Dynamic means flashing according to the volume. It can be selected to flash with larger than 1/2 or 3/4 of the volume, and the output mode can be configured to be Drive or Sink modes. (D for Drive, S for Sink. Default value is Disable)

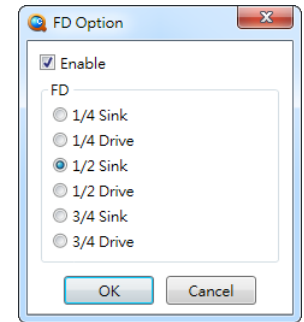
Ex. FD = 1/2D



3.1.15.2 NY5+ / NY6 / NY7 / NX1

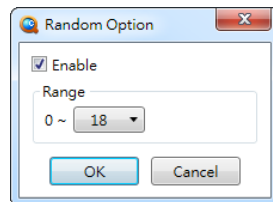
Flash with Dynamic means flashing according to the volume. It can be selected to flash with larger than 1/4, 1/2 or 3/4 of the volume, and the output mode can be configured to be Drive or Sink modes. (D for Drive, S for Sink. Default value is Disable)

Ex. FD = 1/2D



3.1.16 Random

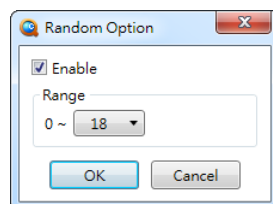
3.1.16.1 NY4 / NY5 / NY5+ / NY6 / NY7



Random means the range of randomly generated numbers (default Disable). The value of the randomly generated number can be from 0 to 255.

Ex. Random = 122

3.1.16.2 NX1

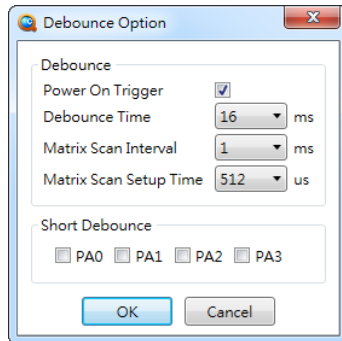


Random means the range of randomly generated numbers (default Disable). The value of the randomly generated number can be from 0 to 65535.

Ex. Random = 822

3.1.17 Debounce

3.1.17.1 NY4 / NY5



Power On Trigger : When user holds the button and power on, system will execute the key path immediately after “PowerOn” path is done (Default value is Enable).

Note: If this function is activated, the PowerOn path must set input state, otherwise this function has no effect, because there is no target path.

Ex. PowerOnTrigger = Disable ; Disable Power On Trigger function.

Debounce Time : User can select the debounce time of input signal within the range of 1 to 1000ms (The default value is 16ms).

Note: Due to Q-Code processes the time through polling, the actual time may have deviation depending on the load at that time.

Ex. Debounce = 16 ms ; Total debounce time for all the keys is 16ms.

Matrix Scankey Interval : User can select the scankey interval of matrix key, the range is 0 ~ 4ms (Default=1ms).

Note: Due to Q-Code processes the time through polling, the actual time may have deviation depending on the load at that time.

Ex. ScankeyInterval = 4ms

Matrix Scan Setup Time : User can select the output setup time of matrix. For example, when user sets the time as 256us, to avoid input pins are at unstable state, Q-Code will delay 256us to access the status of input pins after output pins are set as low level. support 0 / 512us / 1024us / 1536us. (Default = 512us)

Note: Due to Q-Code processes time through polling, the actual time will have deviation that base on the loading.

Ex. Matrix_Scan_SetupTime = 256us ; Set the matrix scan setup time as 256us.

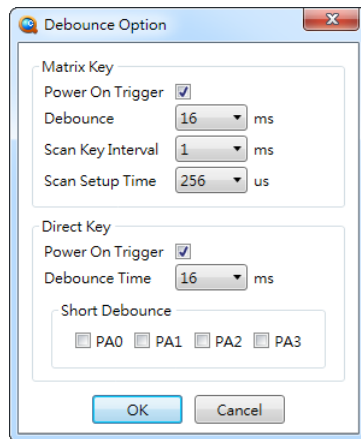
Short Debounce : User can select the short debounce time within the range of 0 to 1ms.

Note:

1. **Short Debounce option is only available for Matrix Key.**
2. **Due to Q-Code processes time via polling, the actual time will have deviation. For more precise debounce time or different debounce time, user needs to program assembly code.**

Ex. ShortDebounce = PA.0, PA.1, PA.2 ; **PA.0, PA.1 and PA.2 are Short Debounce functions.**

3.1.17.2 NY5+ / NY6 / NY7



Power On Trigger : When user holds the button and power on, system will execute the key path immediately after “PowerOn” path is done (Default is Enable). Allows user enabling / disabling Power On Trigger function of Direct Key and Matrix Key separately.

Note: *If this function is activated, the PowerOn path must set input state, otherwise this function has no effect, because there is no target path.*

Ex. Direct_PowerOnTrigger = Disable ; **Disable Power On Trigger function of Direct Key.**

Matrix_PowerOnTrigger = Disable ; **Disable Power On Trigger function of Matrix Key.**

Debounce Time : User can select the debounce time of input signal within the range of 1 to 1000ms (The default value is 16ms). Allows user setting the different debounce time of Direct Key and Matrix Key via the Direct_Debounce and Matrix_Debounce command.

Note: *Actually Q-Code uses polling approach to determine debounce time. The actual debounce time must be greater than the setting. For more precise debounce time or different debounce time, user needs to program assembly code.*

Ex. Direct_Debounce = 16 ms ; **Set Debounce time of Direct Key as 16ms.**

Matrix_Debounce = 16 ms ; **Set Debounce time of Matrix Key as 16ms.**

Scankey Interval : User can select the scankey interval of matrix key, the range is 0 ~ 4ms. (Default is 1ms).

Note: Due to Q-Code processes the time through polling, the actual time may have deviation depending on the load at that time.

Ex. ScankeyInterval = 4ms

Scan Setup Time : User can select the output setup time of matrix. For example, when user sets the time as 256us, to avoid input pins are at unstable state, Q-Code will delay 256us to access the status of input pins after output pins are set as low level. Support 0 / 256us / 512us / 768us / 1024us. (Default is 256us)

Note: Due to Q-Code processes time through polling, the actual time will have deviation that base on the loading.

Ex. Matrix_Scan_SetupTime = 256us ; Set the matrix scan setup time as 256us.

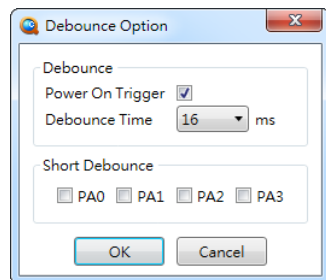
Short Debounce : User can select the short debounce time within the range of 0 to 1ms.

Note:

1. Short Debounce option is only available for Matrix Key.
2. Due to Q-Code processes the time through polling, the actual time may have deviation depending on the load at that time.

Ex. ShortDebounce = PA.0, PA.1, PA.2

3.1.17.3 NY9T



Power On Trigger : When user holds the button and power on, system will execute the key path immediately after "PowerOn" path is done (default Enable).

Note:

1. If this function is activated, the PowerOn path must set input state, otherwise this function has no effect, because there is no target path.
2. This function is only available for normal keys, Touch-Keys doesn't support the function.

Ex. PowerOnTrigger = Disable ; Disable Power On Trigger function of Direct Key

Debounce Time : User can select the debounce time of input signal within the range of 1 to 1000ms (Default is 16ms).

Note: Due to Q-Code processes time via polling, the actual time will have deviation. For more precise debounce time or different debounce time, user needs to program assembly code.

Ex. Debounce = 16 ms ; **Total debounce time for all the keys is 16ms.**

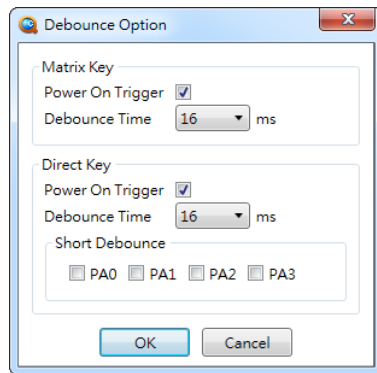
Short Debounce : User can select the short debounce time within the range of 0 to 1ms.

Note:

1. Short Debounce option is only available for Matrix Key.
2. Due to Q-Code processes time via polling, the actual time will have deviation. For more precise debounce time or different debounce time, user needs to program assembly code.

Ex. ShortDebounce = PA.0, PA.1, PA.2 ; **PA.0, PA.1 and PA.2 are Short Debounce functions.**

3.1.17.4 NX1



Power On Trigger : When user holds the button and power on, system will execute the key path immediately after “PowerOn” path is done (Default value is Enable). Allows user enabling / disabling Power On Trigger function of Direct Key and Matrix Key separately.

Note: If this function is activated, the PowerOn path must set input state, otherwise this function has no effect, because there is no target path.

Ex. Direct_PowerOnTrigger = Disable ; **Disable Power On Trigger function of Direct Key.**

Ex. Matrix_PowerOnTrigger = Disable ; **Disable Power On Trigger function of Matrix Key.**

Debounce Time : User can select the debounce time of input signal within the range of 1 to 1000ms (default 16ms). Allows user setting the different debounce time of Direct Key and Matrix Key via the Direct_Debounce and Matrix_Debounce command.

Note: Due to Q-Code processes the time through polling, the actual time may have deviation depending on the load at that time.

Ex. Direct_Debounce = 16 ms ; **Set Debounce time of Direct Key as 16ms.**

Ex. Matrix_Debounce = 16 ms ; **Set Debounce time of Matrix Key as 16ms.**

Short Debounce : User can select the short debounce time within the range of 0 to 1ms.

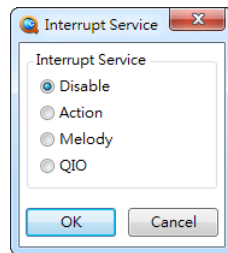
Note:

1. **Short Debounce option is only available for Matrix Key.**
2. **Due to Q-Code processes the time through polling, the actual time may have deviation depending on the load at that time.**

Ex. ShortDebounce = PA.0, PA.1, PA.2 ; **PA.0, PA.1 and PA.2 are Short Debounce functions.**

3.1.18 Interrupt Service

3.1.18.1 NY5



Q-Code provides different interrupt services for three diverse functions, they are Action, QIO and Melody. For higher accuracy requirements, user can change the function to interrupt.

Note:

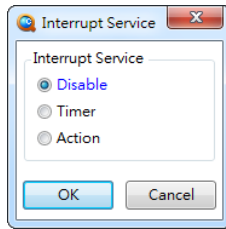
1. **NY5+ / NY6 / NY7 always executes the interrupt service to achieve higher accuracy requirements of all functions.**
2. **Interrupt service only supports a function at one time, and it cannot be changed after setting.**
3. **When interrupt service function is activated, the system will automatically enabled by default, It is not necessary that user activate INT_ON command.**
4. **Using interrupt service will take up more RAM.**
5. **For NY5, ADSR enables interrupt by default, PCT mode disable interrupt by default, and the others features disable interrupt by default.**
6. **Use time interrupt command and Interrupt service together will lead to unpredictable results.**

Ex. InterruptService = Action ; **Using interrupt for Action function.**

Ex. InterruptService = QIO ; **Using interrupt for QIO function.**

Ex. InterruptService = Melody ; **Using interrupt for Melody function.**

3.1.18.2 NY9T



Q-Code provides different interrupt services for two diverse functions, they are Action and Timer. For higher accuracy requirements, user can change the function to interrupt.

Note:

1. *Interrupt service only supports a function at one time, and it cannot be changed after setting.*
2. *When interrupt service function is activated, the system will automatically enabled by default, It is not necessary that user activate INT_ON command.*
3. *Using interrupt service will take up more RAM.*
4. *Use time interrupt command and Interrupt service together will lead to unpredictable results.*
5. *The interrupt service of Action would occupy more RAM and activate the Timer interrupt service.*

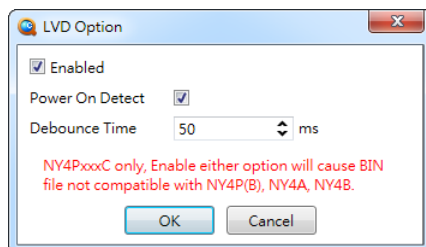
Ex. InterruptService = Action

; Using interrupt for Action function.

3.1.19 LVD

User can monitor voltage by IC LVD to avoid system error caused by the unstable voltage. Associated path is executed when system voltage is change between voltage boundaries.

3.1.19.1 NY4PxxxC

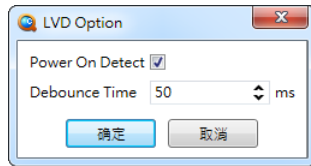


Enabled : whether to detect system voltage when power on.

Power On Detect : whether to detect system voltage when power on.

Debounce : Change of voltage is treated as legitimate only when voltage is kept for a period of time.

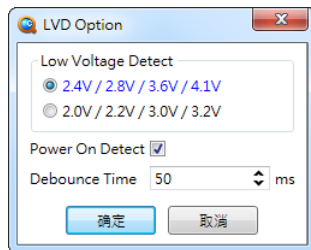
3.1.19.2 NY5+ / NY6B / NY6C / NX1



Power On Detect : whether to detect system voltage when power on.

Debounce : Change of voltage is treated as legitimate only when voltage is kept for a period of time.

3.1.19.3 NY6P025A



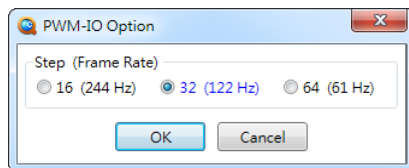
There are 2 sets of voltages supported in NY6P025A.

Power On Detect : whether to detect system voltage when power on.

Debounce : Change of voltage is treated as legitimate only when voltage is kept for a period of time.

3.1.20 PWM-IO

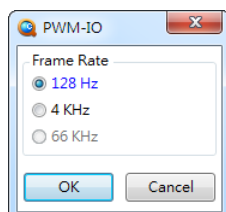
3.1.20.1 NY4 / NY5 / NY6 / NY7



PWMIO_Step: Set the step of PWM-IO output. There are three steps of 16 / 32 / 64, which corresponds to 244Hz / 122Hz / 61Hz frame rate.

Ex. PWMIO_Step = 32 ;Set PWM output step to 32.

3.1.20.2 NY9T



Frame Rate: The refresh rate of PWM-IO signals. (NY9T001A default is 4KHz, other NY9T IC default is 128Hz)

Ex. PWMIO_FrameRate = 128Hz ; PWM-IO Frame Rate as 128Hz.

Ex. PWMIO_FrameRate = 4KHz ; PWM-IO Frame Rate as 4KHz .

Note:

1. The higher refresh rate, the LED will less likely to glitter.
2. When NY9T001A sets FrameRate=66KHz, PWM-IO only provides one PWM-IO output. User can set the output as one of PE.0 / PE.1 / PE.2 pins, and the rest of pins only can be set as normal IO. (Only NY9T001A provides the 66K option)
3. When NY9T001A / NY9T008A / NY9T016A sets FrameRate=4KHz, the PWM-IO only can use the pin from PE.0 / PE.1 / PE.2, and the rest of pins only can be set as normal IO. (NY9T004A provides 128Hz option only)

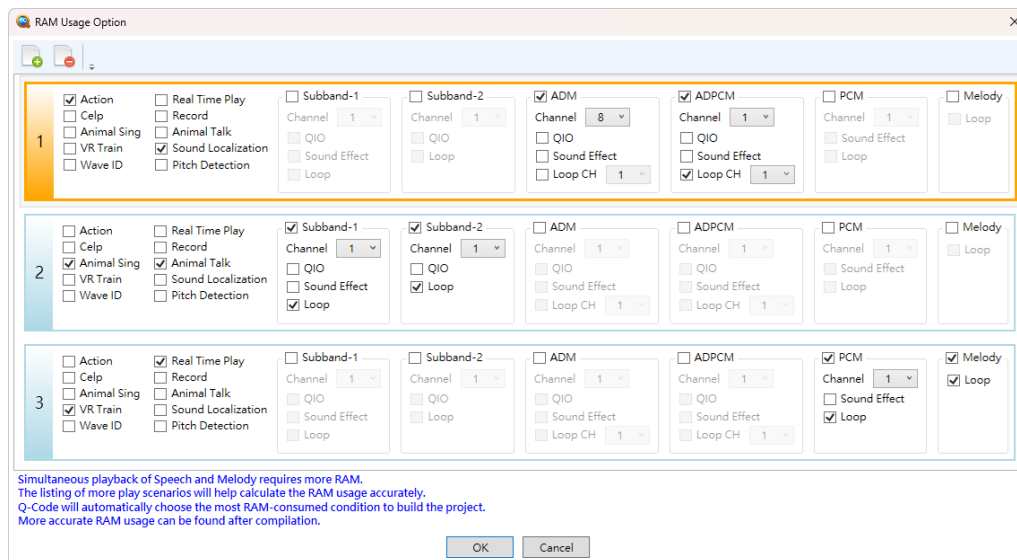
The PWM frame rate supported by NY9T series and the available pins:

	NY9T001A		NY9T004A	NY9T008A		NY9T016A	
Frame Rate	4KHz	66KHz	128Hz	128Hz	4KHz	128Hz	4KHz
Pin	PE.0 ~ PE.2	one of PE.0 ~ PE.2 pin only	PE	PE~PF	PE.0 ~ PE.2	PE~PF	PE.0 ~ PE.2

3.1.21 RAM Usage

3.1.21.1 NX1

Specify the condition of using a variety of functions simultaneously. According to this setting, the memory space occupied during the execution period can be estimated. Users have to set the functional combination, when two channel are used simultaneously. Each function combination represents the operation that will be executed at the same time



Simultaneous playback of Speech and Melody requires more RAM.
The listing of more play scenarios will help calculate the RAM usage accurately.
Q-Code will automatically choose the most RAM-consumed condition to build the project.
More accurate RAM usage can be found after compilation.

Action: Enable Action output function.

Celp: Play the voice of Celp algorithm.

Animal Sing: Enable the Animal Sing function.

VR Train: Enable the Voice tag training function.

WaveID: Enable the WaveID.

Real Time Play: Enable the real time playback (Bullhorn) function.

Record: Enable the Record function.

Animal Talk: Enable the Animal Talks function.

Sound Localization: Enable the Sound Localization function.

Pitch Detection: Enable the Pitch Detection function.

Subband-1: Play the voice of Subband-1 algorithm, up to 2 channels.

Subband-2: Play the voice of Subband-2 algorithm, up to 2 channels

ADPCM: Play the voice of ADPCM algorithm, up to 8 channels.

ADM: Play the voice of ADM algorithm, up to 8 channels.

PCM: Play the voice of PCM, up to 3 channels.

Melody: Play the MIDI.

QIO: QIO output of the corresponding compression algorithm..

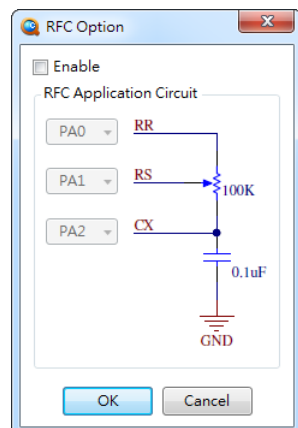
Sound Effect: Enable the sound effect of the corresponding compression algorithm.

Loop: Enable the sound loop of the corresponding algorithm / MIDI. ADM / ADPCM can support multiple channel loops.

Ex. RAM_Usage = ADMx8+ADPCM+ADPCM_Loop+Action+SoundLoc,
Subband+Subband_Loop+SBC2+SBC2_Loop+AnimalTalk+AnimalSing,

3.1.22 RFC

3.1.22.1 NY5+ / NY6 / NY7



The RFC module needs an external capacitor and potentiometer to detect the position of the potentiometer knob by the time constant of charging the capacitor, and the resolution is 16 levels. In a nutshell, the potentiometer's knob is divided into 16 parts. Users can read the current position of

the knob through the API and perform different controls, such as the volume or the speed of the rhythm.

Note: The recommended capacitor and potentiometer value is 0.1uF and 100KΩ, users have to maintain the same charging time constant as possible if they need to adjust; for example, when the resistance value of the potentiometer is reduced to a half, the capacitor will need to be doubled, and so on.

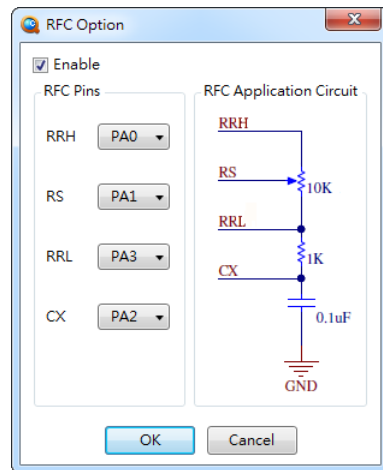
Ex.

RFC_RR = PA.0

RFC_RS = PA.1

RFC_CX = PA.2

3.1.22.2 NYX1



The RFC module needs an external capacitor and potentiometer to detect the position of the potentiometer knob by the time constant of charging the capacitor, and the resolution is 16 levels. In a nutshell, the potentiometer's knob is divided into 16 parts. Users can read the current position of the knob through the API and perform different controls, such as the volume or the speed of the rhythm.

Note: The recommended capacitor and potentiometer value is 0.1uF and 10KΩ, users have to maintain the same charging time constant as possible if they need to adjust; for example, when the resistance value of the potentiometer is reduced to a half, the capacitor will need to be doubled, and so on.

Ex.

RFC_RR = PA.0

RFC_RS = PA.1

RFC_CX = PA.2

RFC_RRL = PA.3

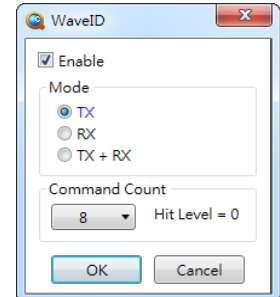
3.1.23 WaveID

3.1.23.1 NX1

The Wave ID sends ID by a speaker and receives ID through microphone. The frequency transmitted by the speaker is a frequency band that is not sensitive to human ears (16K ~ 20KHz). In application, it can be output with the general voice by the speaker and the corresponding voice is played by Ch0 only.

Mode: Select TX or RX.

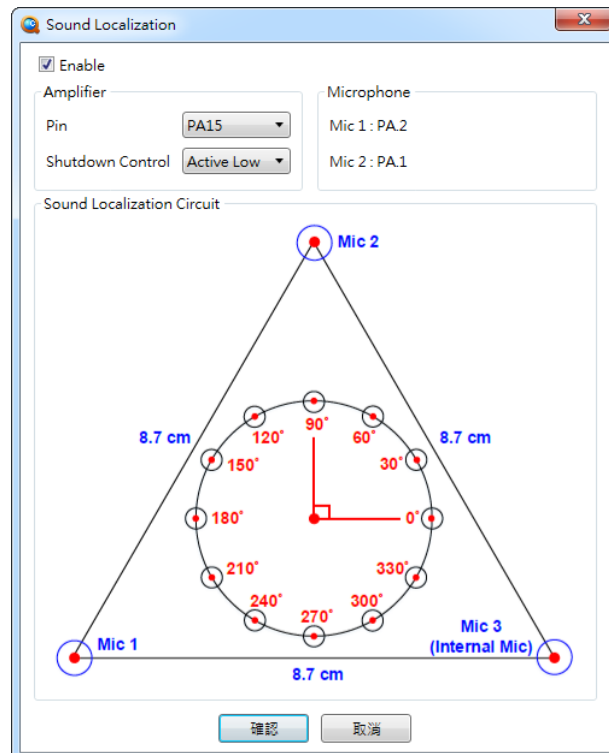
Command Count: Select the count of sent ID, the maximum is 38. The more IDs, the lower hit rate.



ID	Hit Rate
<= 12	95%
<= 19	90%
<= 30	85%
<= 38	80%

3.1.24 Sound Localization

3.1.24.1 NX1 OTP



Amplifier Pin: Pin connected to external amplifier.

Amplifier Shutdown Control: Control type of external amplifier (Default is Active Low).

Ex.

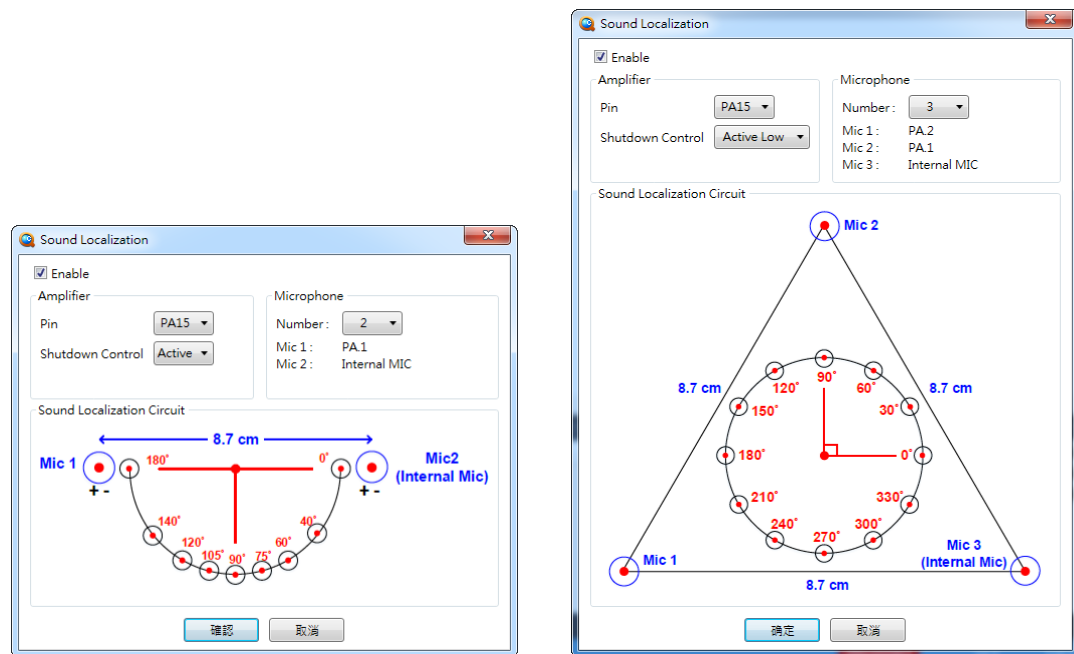
SoundLoc = Enable

SoundLoc_Amp_Pin = PA.15

SoundLoc_Amp_Shutdown_Control = Active_Low

3.1.24.2 NX1 EF

This function can choose to use 2 or 3 microphone, and the received signal is amplified by an external amplifier. The angle of sound source can be calculated in corporate with built-in mic.



Amplifier Pin: Pin connected to external amplifier.

Amplifier Shutdown Control: Control type of external amplifier (Default is Active Low).

Microphone Number : Number of microphones.

Note: 2 microphones need to occupy PA.1, 3 microphones need to occupy PA.1, PA.2

Ex.

SoundLoc = Enable

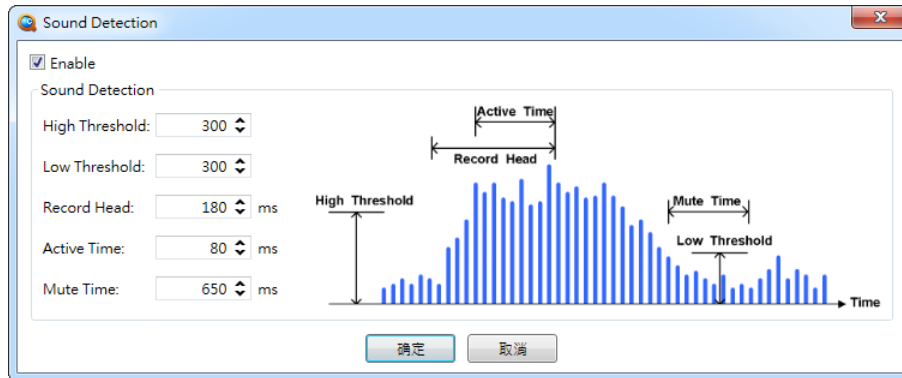
SoundLoc_Amp_Pin = PA.15

SoundLoc_Amp_Shutdown_Control = Active_Low

SoundLoc_MIC = 2

3.1.25 Sound Detection

3.1.25.1 NX1



High Threshold: Detect the threshold value of voice. If the value is higher than the threshold and exceeds Active Time, it will be considered a valid speech.

Low Threshold: Detect the threshold of nonverbal. If the value is lower than the threshold and exceeds Mute Time, it will be considered as background sound then stop recording.

Active Time: Detect the duration of voice. If the duration is longer than High Threshold and exceeds the detection time, the voice will be considered a valid speech.

Mute Time: Detect the duration of nonverbal. If the speech duration is shorter than Low Threshold and exceeds the detection time, it will be considered as background sound then stop recording.

Ex. SoundDetect = Enable

SoundDetect_High_Threshold = 300

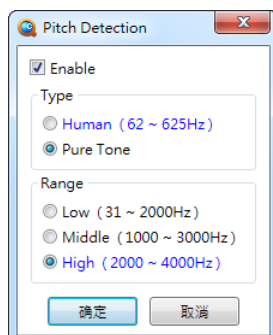
SoundDetect_Low_Threshold = 300

SoundDetect_Active_Time = 80ms

SoundDetect_Mute_Time = 650ms

3.1.26 Picth Detection

3.1.26.1 NX1



Type: Set to detect Human voice or Pure Tone.

Range: Set the detect range of Pure Tone.

Ex. `PitchDetect` = Enable

`PitchDetect_Range` = Low

3.1.27 Maximum Single Note

3.1.27.1 NY5+ / NY6 / NY7

User can select the maximum number of single note played simultaneously, the ranges is 1~8 (default 4).

Note:

- 1. NY5+ have a maximum of 4 channels, only the channel without playing single note or voice can be used to play melody.*
- 2. NY6 have a maximum of 6 channels, only the channel without playing single note or voice can be used to play melody.*
- 3. NY7 have a maximum of 8 channels, only the channel without playing single note or voice can be used to play melody.*

Ex. `Melody_MaxSingleNote` = 4

; Set the Maximum Single Note as 4.

3.1.28 Melody_OKON_BgNote

3.1.28.1 NY5+ / NY6 / NY7

When using One-Key-One-Note, only MIDI channel 1 is played by default. With this option enabled, other MIDI channels will be played too.

Ex. `Melody_OKON_BgNote` = Enable

; Enable OKON BgNote function.

3.1.29 Variable Compatible

3.1.29.1 NX1

The variable compatible provides the compatibility of 4-bit variable operations for NX1 series. If user needs to use the 4-bit data format for NX1 series, please set the minimum length of variables as 8-bit. If user needs 4-bit variables, please enable the variable compatible. Q-Code will generate variables such as R0, R1, X0, X0H, X0L to make the command of operation consist with NY4 / 5 / 6 / 7 / 9T.

Ex. `Variable-Compatible` = Enable

; Enable the variable compatible.

3.1.30 Realtime Play

3.1.30.1 NX1

The realtime play (loud speaker) function.

Sampling Rate : Select the sampling rate.

Ex. RT_Sampling_Rate = 8000

Suppress Howling : Enable Suppress Howling or not.

Ex. RT_SuppressHowling = Enable

Echo Delay Time: Select the length of echo delay.

Ex. RT_Echo_DelayTime = 200ms

Echo Delay Repeat Times : Select the repeat times of echo.

Ex. RT_ECHO_Repeat = 2

Echo Delay Quality : Select the quality of echo.

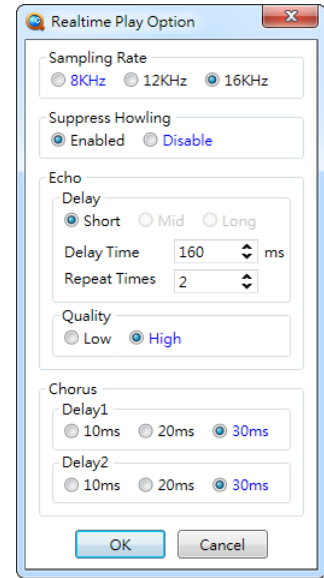
Ex. RT_ECHO_Repeat = 2

Chorus Delay1: Delay time between chorus track 1 and track 2, accepts 10 / 20 / 30ms.

Ex. RT_Chorus_Delay1 = 30ms

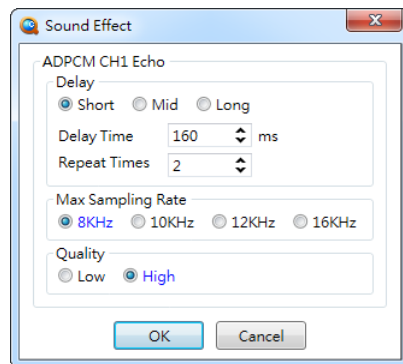
Chorus Delay2: Delay time between chorus track 2 and track 3, accepts 10 / 20 / 30ms.

Ex. RT_Chorus_Delay2 = 30ms



3.1.31 Sound Effect

3.1.31.1 NX1



ADPCM CH1 Echo Delay Time : User define the length of ADPCM CH1 echo delay.

Ex. SFX_ADPCM_Echo_DelayTime = 200ms

ADPCM CH1 Echo Repeat Times : User define the repeat times of ADPCM CH1 echo.

Ex. SFX_ADPCM_Echo_Repeat = 4

ADPCM CH1 Echo Max Sampling Rate: Select the ADPCM CH1 echo max sampling rate.

Ex. SFX_ADPCM_Echo_Max_SR = 8000

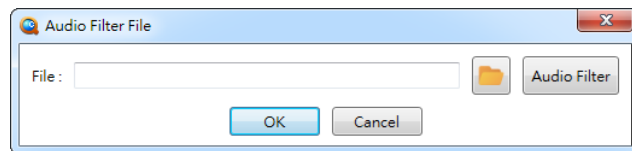
ADPCM CH1 Echo Advance Quality: Select advance quality ADPCM CH1 echo.

Ex. SFX_ADPCM_Echo_Quality = High

3.1.32 Audio Filter

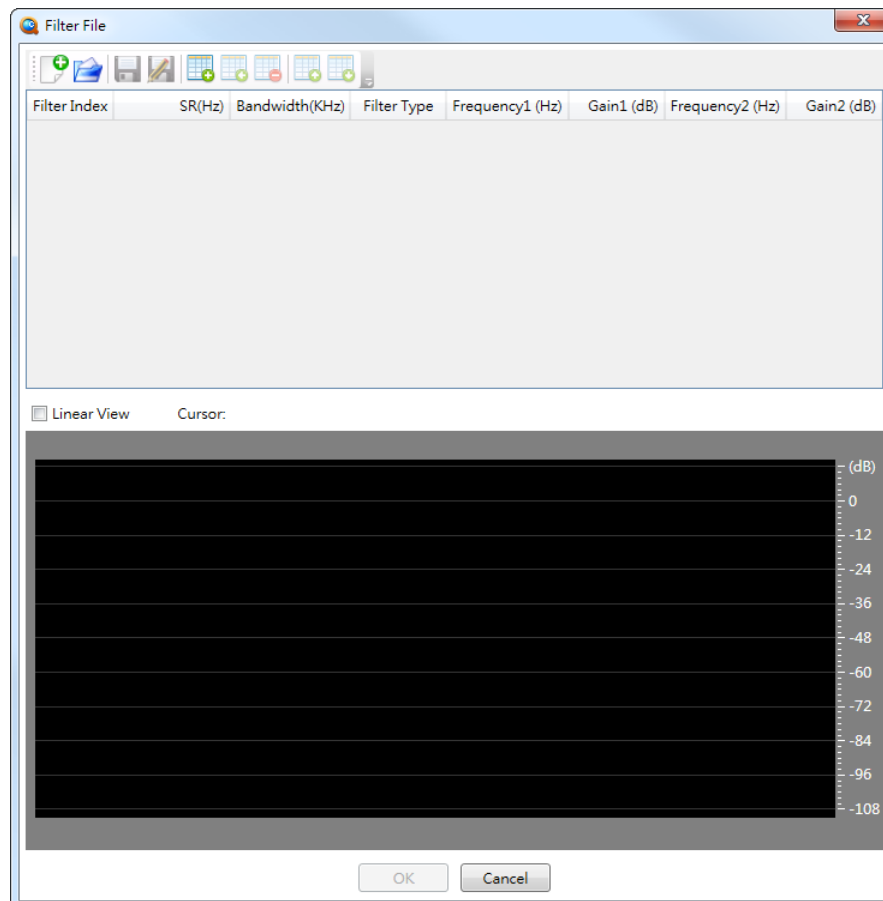
3.1.32.1 NX1 OTP / NX1 EF

The setting window is shown below. User can directly select the edited .fnx file, or call the audio filter to generate .fnx files. Click “Clear” button to undo settings.



After clicking the “Audio Filter” button, the edit window is shown below. It can set multiple filters, user can set the sample rate (Hz), bandwidth (KHz) and the filter type of each filter, the supported types are LPF, HPF, BPF, and EQ. After selecting the filter type, input the required frequency, range and gain value to complete the setting. The saved file extension is .fnx, which is convenient for repeated use.

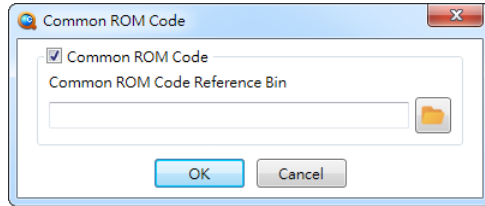
Ex. Custom_Filter = Filter.fnx



3.1.33 Common ROM Code

3.1.33.1 NX1

Common ROM Code is used for codes having the same structure or function, but different contents. The replacement content is placed in SPI Flash, and you can have different effects by simply replacing SPI Flash. In order to maintain the main ROM content consistent, the following screen will pop up for setting.



Common ROM Code Reference Bin is to select the bin file created before, Q-Code will automatically compare the ROM data in the current bin file to ensure that the ROM data is completely consistent.

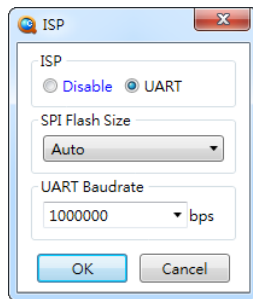
The basic development of Common ROM application process is as follows:

1. Enable Common ROM on the first development.
2. For subsequent development programs, Common ROM Code needs to be enabled and the first generated bin file be assigned as the reference file.

According to the above operation, Q-Code will ensure that the ROM data produced is consistent, otherwise an error will be issued.

3.1.34 ISP

3.1.34.1 NX1 EF



Set the ISP type. Currently only UART is supported, and UART communication uses the PD.0 / 1 pins. For more detailed ISP operation and instructions, please refer to the appendix of the NYISP user manual..

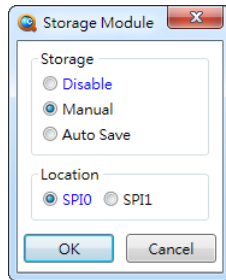
Ex. ISP = UART

ISP_UART_Baudrate = 1000000bps

ISP_FlashSize = 256Mbit

3.1.35 Storage Modulal

3.1.35.1 NX1OTP / NX1 EF



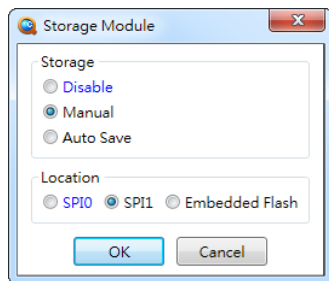
Storage : Set storage mode.

Ex. Storage = Disable

Locaiton : Set storage location.

Ex. Storage_Location = SPI0

3.1.35.2 NX1OTP / NX1 EF



Storage : Set storage mode.

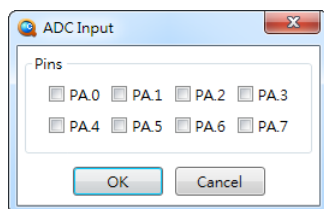
Ex. Storage = Disable

Locaiton : Set storage location.

Ex. Storage_Location = EF

3.1.36 ADC Input

3.1.36.1 NX1



Set Analog signal input pins of Analog-to-Digital Converter.

Ex. ADC_Input_Pins = PA.0, PA.1, PA.2

3.2 Voice File

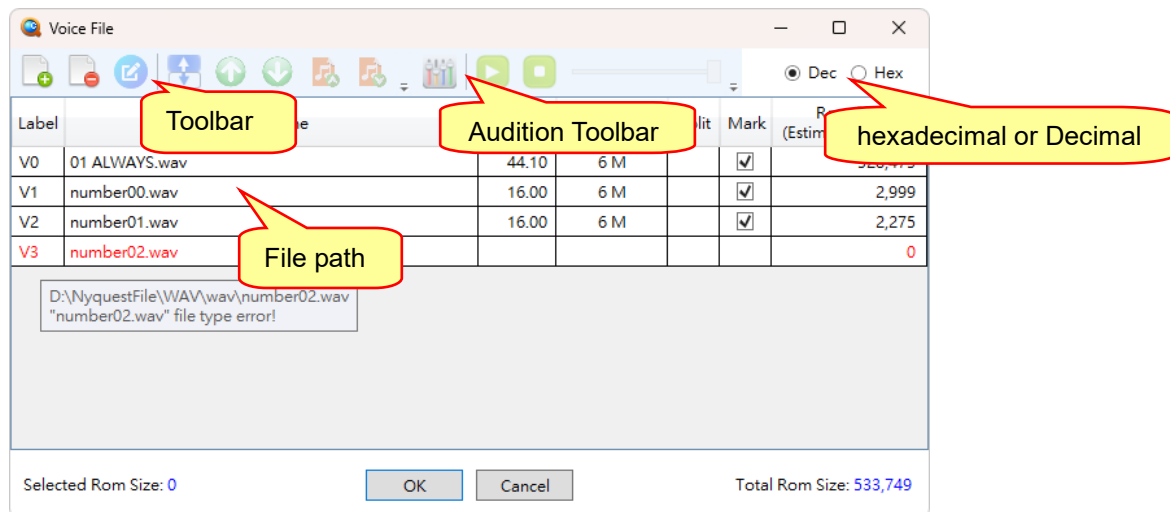
Usually the file extension of voice file for playback is .wav or the compiled .v4x / .v5x / .v5px / .v6x / .v7x / .vnx file, .nyw file which is generated via *Q-Sound* and previous .wsk file. The voice file with *Quick-IO* (QIO signal or Wave Mark data) feature has the file extension of .nyq. If the file imported is .nyq file, the QIO or Wave Mark feature will be activated during playback.

By single clicking the “Add File” on the “Voice File” section menu, the “Voice File” can be opened and the voice source can be inserted to the program. For user to remove one or more voice sources, just select the voice source file to be removed and then click “Delete Files” to remove the selected file.

Note:

1. User can change voice file label.
2. If added file path is not absolute, Q-Code will treat it as relative path to .qc path.

The “Voice File” dialog box will appear by clicking “Add File” and the required voice source can be added in this dialog box. Voice File dialog for different series has different columns.



Add File : Add new voices to list.

Delete File : Delete selected files.

Quickly Renamer : Tool for quickly renaming multiple Labels.

Invert : Invert selection, click this button will switch selected and un-selected files.

Move UP : move selected rows upward.

Move Down: move selected rows downward.

Move File UP : move selected files upward.

Move File Down: move selected files downward.

Q-Sound : This is a software tool for enhancing sound quality and splitting voice file. The user-friendly interface makes it easy to save ROM by dividing wave files, and what user have to do is edit markers and save it.

Demo : User can listen to selected voice file. Drag the indicator to search the progress of playback.

Hex/Dec : Switch ROM Size display as hexadecimal or Decimal.

Selected ROM size : The total estimated ROM Size of selected voice files.

Total ROM Size : The total estimated ROM Size of all voice files.

Note:

1. The built-in Q-Sound of Q-Code accepts .nyw files split by external Q-Sound.
2. All inserted marks inside Q-Sound will only be recorded after the file is saved.
3. If user adds encoded .v5px / .v6x / .v7x / .vnx file or .v4x / .v5x file encoded by Voice_Encoder 2.20 or newer version, Q-Code will use the sample rate of the original sources. For example: PlayV(Ch0, \$V0).
4. If voice file is .v4x / .v5x encoded by Voice_Encoder 2.20 or older version, the sampling rate of original voice source must be typed in manually. Ex. PlayV(Ch0, \$V0, 6K).

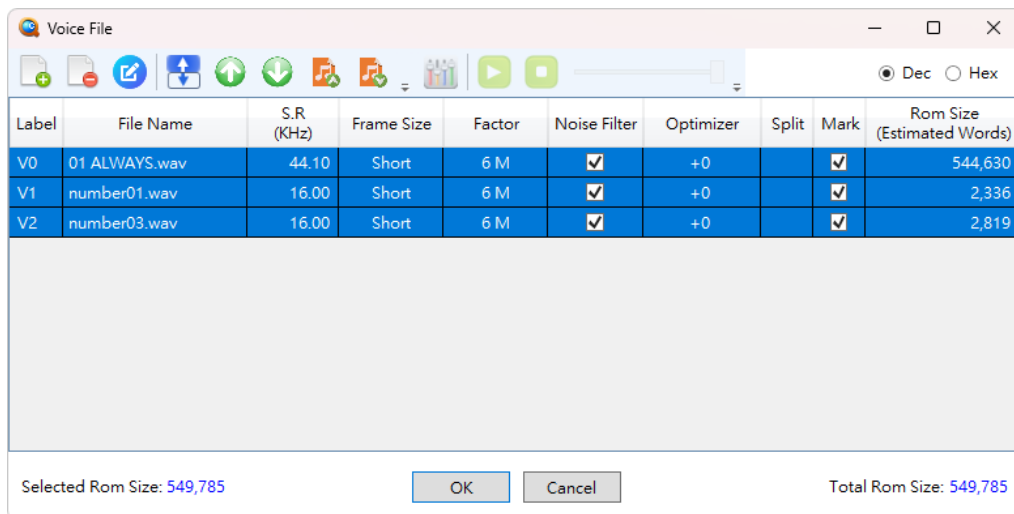
Ex.

[Voice File]

V01 = C:\MyQ-CodeProjects\Prj1\voice1.wav /7

V02 = Prj1\voice2.wav /PCM

3.2.1 NY4



Label	File Name	S.R (KHz)	Frame Size	Factor	Noise Filter	Optimizer	Split	Mark	Rom Size (Estimated Words)
V0	01 ALWAYS.wav	44.10	Short	6 M	☒	+0	☐	☒	544,630
V1	number01.wav	16.00	Short	6 M	☒	+0	☐	☒	2,336
V2	number03.wav	16.00	Short	6 M	☒	+0	☐	☒	2,819

Selected Rom Size: 549,785 OK Cancel Total Rom Size: 549,785

S.R (Sample Rate) : Sampling rate of voice file

Frame Size : Amount of samples in an encoding frame: Short or Long. Double clicking on this column can prompt a drop-down menu for frame size setting.

1. Short Frame Size: The sound quality is better but file size is larger.
2. Long Frame Size: The sound quality is normal but file size is smaller.

Factor : Factor affects the sound quality of voice file. Provides 12 kinds of compression factor and PCM (non-compressed). PCM means the highest quality. Double clicking on this column can prompt a drop-down menu for factor selection. The smaller the factor value, the higher the

compressing ratio, and vice versa. Meanwhile, the smaller the factor value, the smaller the ROM Size. PCM format has the best sound quality, but occupies the largest ROM Size.

Noise Filter : User can choose whether or not to enable noise filter function to lower background noise.

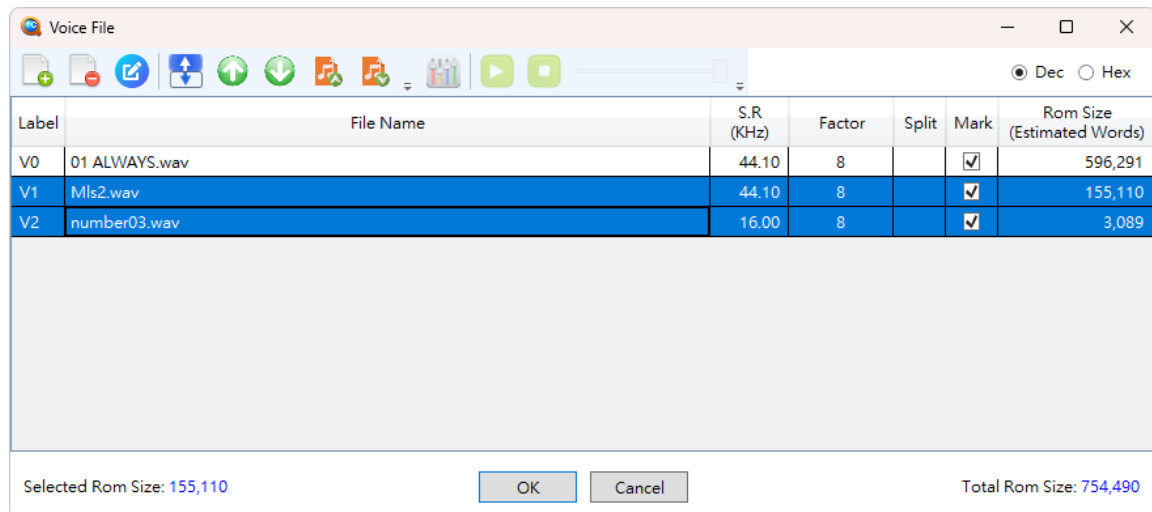
Optimizer : Optimize the sound according to the setting value. Sound optimization function can adjust the voice to enhance contrast (sharp), or make the sound smoother.

Split : User can choose whether or not to split voice file according to inserted marks in voice file, this option is available via the *Q-Sound* function of *Q-Code*.

Mark : Enable wave mark function of .wav file.

Estimated ROM Size : The estimated ROM Size information.

3.2.2 NY5 / NY5+



Label	File Name	S.R (KHz)	Factor	Split	Mark	Rom Size (Estimated Words)
V0	01 ALWAYS.wav	44.10	8		☒	596,291
V1	Mls2.wav	44.10	8		☒	155,110
V2	number03.wav	16.00	8		☒	3,089

Selected Rom Size: 155,110 OK Cancel Total Rom Size: 754,490

S.R (Sample Rate) : Sampling rate of voice file.

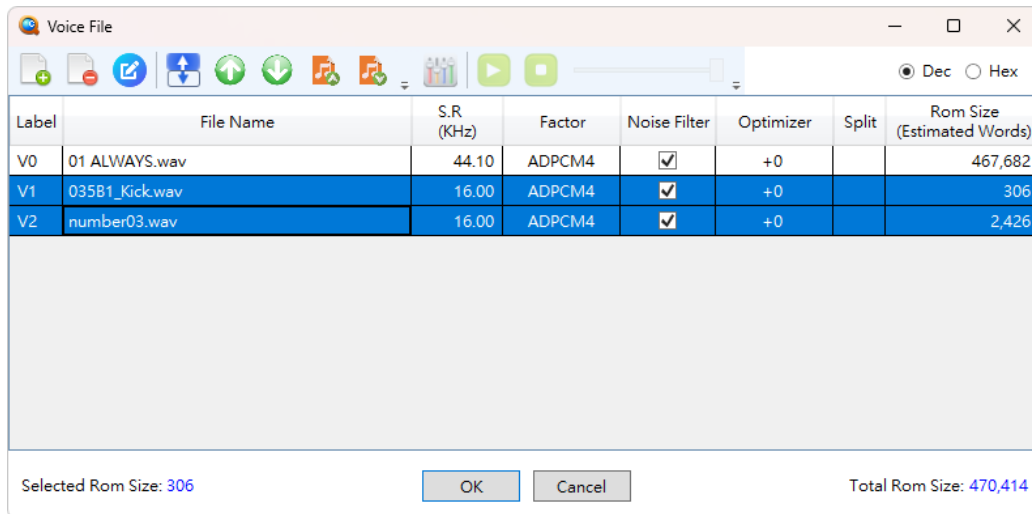
Factor : Factor affects the sound quality of voice file. Provides 12 kinds of compression factor and PCM (non-compressed). PCM means the highest quality. Double clicking on this column can prompt a drop-down menu for factor selection. The smaller the factor value, the higher the compressing ratio, and vice versa. Meanwhile, the smaller the factor value, the smaller the ROM Size. PCM format has the best sound quality, but occupies the largest ROM Size (default value is "6").

Split : User can choose whether or not to split voice file according to inserted marks in voice file, this option is available via the *Q-Sound* function of *Q-Code*.

Mark : Enable wave mark function of .wav file.

Estimated ROM Size : The estimated ROM Size information.

3.2.3 NY6



Voice File dialog box showing a table of voice files and their properties. The table has columns: Label, File Name, S.R (KHz), Factor, Noise Filter, Optimizer, Split, and Rom Size (Estimated Words). The files listed are V0 (01 ALWAYS.wav), V1 (035B1_Kick.wav), and V2 (number03.wav). The selected Rom Size is 306 and the Total Rom Size is 470,414.

Label	File Name	S.R (KHz)	Factor	Noise Filter	Optimizer	Split	Rom Size (Estimated Words)
V0	01 ALWAYS.wav	44.10	ADPCM4	☒	+0		467,682
V1	035B1_Kick.wav	16.00	ADPCM4	☒	+0		306
V2	number03.wav	16.00	ADPCM4	☒	+0		2,426

Selected Rom Size: 306 OK Cancel Total Rom Size: 470,414

S.R (Sample Rate) : Sampling rate of voice file.

Factor : Factor affects the sound quality of voice file. Use ADPCM4 as default. User also can use ADPCM5 or PCM (non-compressed) mode. PCM format has the best sound quality, but occupies the largest ROM Size.

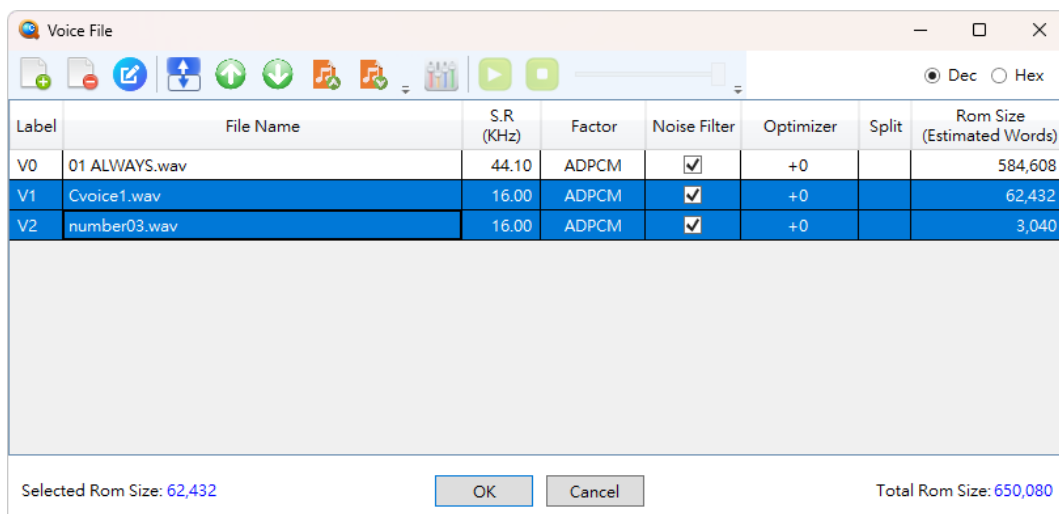
Noise Filter : User can choose whether or not to enable noise filter function to lower background noise.

Optimizer : Optimize the sound according to the setting value. Sound optimization function can adjust the voice to enhance contrast (sharp), or make the sound smoother.

Split : User can choose whether or not to split voice file according to inserted marks in voice file, this option is available via the *Q-Sound* function of *Q-Code*.

Estimated ROM Size : The estimated ROM Size information.

3.2.4 NY7



Voice File dialog box showing a table of voice files and their properties. The table has columns: Label, File Name, S.R (KHz), Factor, Noise Filter, Optimizer, Split, and Rom Size (Estimated Words). The files listed are V0 (01 ALWAYS.wav), V1 (Cvoice1.wav), and V2 (number03.wav). The selected Rom Size is 62,432 and the Total Rom Size is 650,080.

Label	File Name	S.R (KHz)	Factor	Noise Filter	Optimizer	Split	Rom Size (Estimated Words)
V0	01 ALWAYS.wav	44.10	ADPCM	☒	+0		584,608
V1	Cvoice1.wav	16.00	ADPCM	☒	+0		62,432
V2	number03.wav	16.00	ADPCM	☒	+0		3,040

Selected Rom Size: 62,432 OK Cancel Total Rom Size: 650,080

S.R (Sample Rate) : Sampling rate of voice file.

Factor : Factor affects the sound quality of voice file. Use DPCM as default. User also can use PCM (non-compressed) mode. PCM format has the best sound quality, but occupies the largest ROM Size.

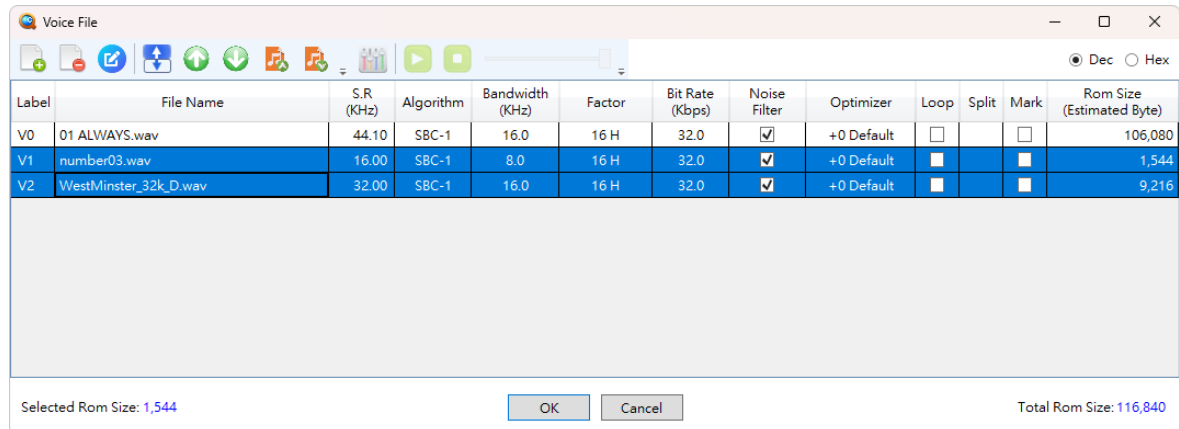
Noise Filter : User can choose whether or not to enable noise filter function to lower background noise.

Optimizer : Optimize the sound according to the setting value. Sound optimization function can adjust the voice to enhance contrast (sharp), or make the sound smoother.

Split : User can choose whether or not to split voice file according to inserted marks in voice file, this option is available via the *Q-Sound* function of *Q-Code*.

Estimated ROM Size : The estimated ROM Size information.

3.2.5 NX1



Label	File Name	S.R (KHz)	Algorithm	Bandwidth (KHz)	Factor	Bit Rate (Kbps)	Noise Filter	Optimizer	Loop	Split	Mark	Rom Size (Estimated Byte)
V0	01 ALWAYS.wav	44.10	SBC-1	16.0	16 H	32.0	☒	+0 Default	☐	☐	☐	106,080
V1	number03.wav	16.00	SBC-1	8.0	16 H	32.0	☒	+0 Default	☐	☐	☐	1,544
V2	WestMinster_32k_D.wav	32.00	SBC-1	16.0	16 H	32.0	☒	+0 Default	☐	☐	☐	9,216

Selected Rom Size: 1,544 OK Cancel Total Rom Size: 116,840

S.R (Sample Rate) : Sampling rate of voice file.

Algorithm : Provides PCM, ADPCM and high compression formats—SBC-1 / SBC-2 / CELP. Sub-Band Coding is a kind of wave coding that based on signal spectrum, it breaks a signal into a number of different frequency bands, and encodes each one independently via ADPCM, then compounds as raw signals. NX1 series can complete Sub-Band Coding and achieve lower bit rate to save ROM size. The higher sample rate and wider bandwidth can represent high frequency range perfectly. SBC-2 reduces the capacity of ROM and RAM, and CPU loading but the sound quality will be slightly worse than SBC-1. CELP a speech coding algorithm applies for human sound which uses low Bit Rate compression.

Bandwidth : The unit is KHz.

Factor : Use dynamic compression algorithm, based on the selection of the algorithm has a different compression rate. SBC-1 / SBC-2 has 19 level parameters (-2~16), ADPCM4 and ADPCM5 have 12 level parameters (1~12). CELP is a fixed compression ratio. User can also adjust the compression rate by the Factor parameters after select algorithm:

The smaller the factor value, the higher the compressing ratio, and vice versa. Meanwhile, the smaller the factor value, the smaller the ROM Size. Users can select the appropriate algorithm according to the needs of the product, and then adjust the weight of the Factor to achieve the optimization of memory according to the requirements of the quality of sound.

Bit Rate : The bit rate means the quantity of bit per unit time while playing the compressed voice. It is equivalent to the consumption of bandwidth while playing voice. With a higher bit rate, it can perform better voice quality. Its unit is Kbps.

Noise Filter : User can choose whether or not to enable noise filter function to lower background noise.

Optimizer : Optimize the sound according to the setting value. Sound optimization function can adjust the voice to enhance contrast (sharp), or make the sound smoother.

Loop : Tick to enable the loop function and will cause occupying more ROM size.

Split : User can choose whether or not to split voice file according to inserted marks in voice file, this option is available via the *Q-Sound* function of *Q-Code*.

Mark : Enable wave mark function of .wav file.

Estimated ROM Size : The estimated ROM Size information.

3.2.6 How to Play Files Processed by Q-Sound in Q-Code

Simple methods are provided by *Q-Code* allowing user to use voice source with splitting function enabled:

Step1: Add the voice file to be played to the dialog box of "Voice File" and check the "Split" feature.

Step2: Open *Q-Sound* and configure the voice source to be split.

Step3: Issue the PlayV command in *Q-Code*.

Ex.

[Voice File]

V01 = C: \MyQ-CodeProjects\Prj1\voice1.nyw /s /1

; Parameter "/s" is attached to the voice file indicating the use of "Split" feature. ("/1" means factor is 1)

[Path]

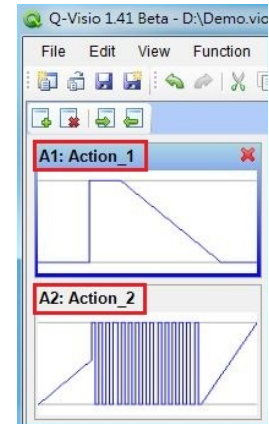
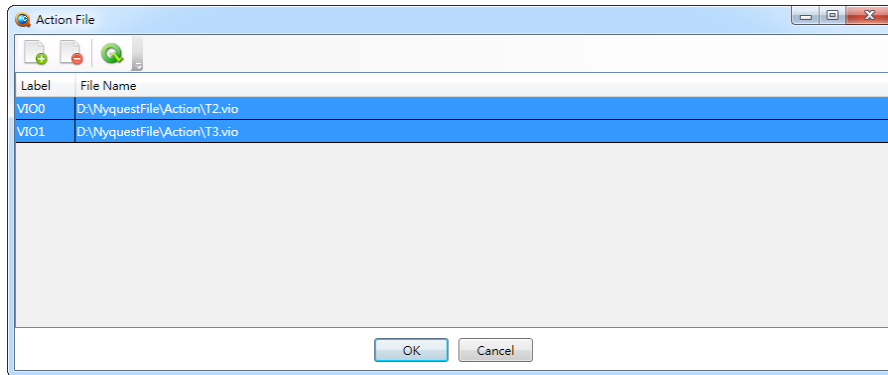
PowerOn: PlayV(ch1, \$V01)

; For playing voice file of Label \$V01" only general PlayV command is needed. System will automatically expand the split file, and there is no extra actions needed to be taken by user.

3.3 Action File

3.3.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1

Action file consists of signal files used for playing, and the file extension of action file is .vio. Please see the Q-Visio User Manual for details.



Ex.

[Action File]

VIO0 = C:\Project.vio ; Add action file.

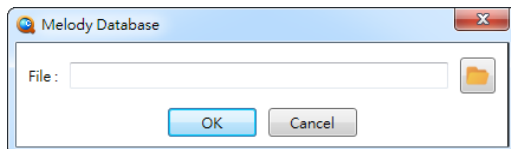
[Path]

PowerOn: PlayA(PB.0, CH1, \$A1)

; Play action, output Label "A1" signal to PB.0, right after power on.

3.4 Melody Database

To play melody, melody database is necessary for timbre / envelope / MIDI data.



Ex.

[Melody Database]

C:\Project.md2

; m0 = mla_a.mid, 2 channel

; m1 = song14_1ch.mid, 1 channel

3.4.1 NY5

Melody database is .md2 file generated by Q-MIDI.

Note: User must compose sound library by themselves using Q-MIDI to generate an .idb file. Then Q-MIDI is used to link the melodies and sound library to generate a .md2 sound file, so that these sound files can be added to Q-Code program. (For details, please refer to user manuals of Q-MIDI).

3.4.2 NY5+ / NY6 / NY7 / NX1

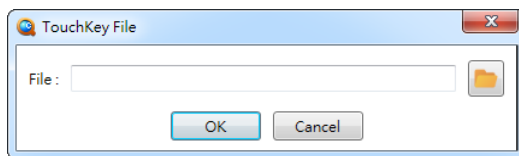
Melody database is .qmd file generated by Q-MIDI.

Note: User must compose sound library by themselves using Q-MIDI to generate an .idb2 file. Then Q-MIDI is used to link the timbre and Instrument Database to generate a .qmd sound file, so that these sound files can be added to Q-Code program. (For details, please refer to Q-MIDI user manuals).

3.5 Touch-Key File

3.5.1 NY9T

Touch-Key file consists of sensitivity setting files, user can change files for applying different sensitivity. The filename extension is .t9x.



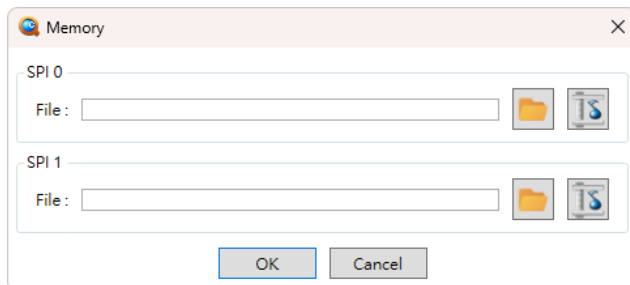
Ex.

[TouchKey File]

C:\TouchKey.t9x ; Add the Touch-Key sensitivity setting file.

3.6 Memory

3.6.1 NX1 OTP



Resources can be placed on SPI Flash, SPI0 / SPI1 are available.

To place resources on SPI Flash, add .spiprj generated by SPI_Encoder. _SPI.bin is generated after build process, which can be downloaded by Q-Writer or Q-Code.

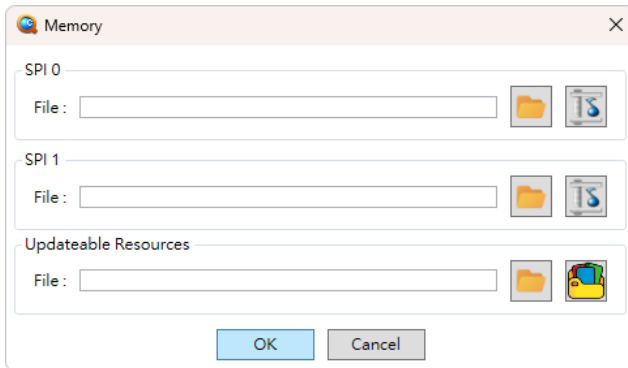
Ex.

[Memory]

SPI0_File = C:\Users\Desktop\SPI0.spiprj

SPI1_File = C:\Users\Desktop\SPI1.spiprj

3.6.2 NX1 EF



Resources can be placed on SPI Flash, SPI0 / SPI1 are available. Besides, also supports Updateable Resources.

To place resources on SPI Flash, add .spiprj generated by *SPI_Encoder*. _SPI.bin is generated after build process, which can be downloaded by *Q-Writer* or *Q-Code*.

Ex.

[Memory]

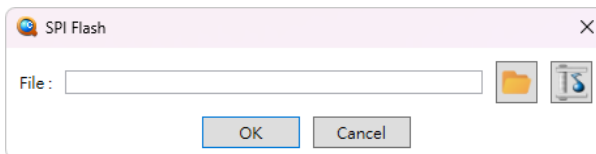
SPI0_File = C:\Users\Desktop\SPI0.spiprj

SPI1_File = C:\Users\Desktop\SPI1.spiprj

UDR_File = C:\Users\Desktop\SPI1.spiprj

3.7 SPI Flash

3.7.1 NY6



The project of *SPI_Encoder*, with extension .spiprj, can be added to **[SPI Flash]**. After *Q-Code* build, the content of SPI Flash will be generated as a _SPI.bin file, which can be downloaded via *Q-Writer* or *Q-Code*.

About the *SPI_Encoder* editing, please refer to *SPI_Encoder* user manual.

Ex.

[SPI Flash]

C:\Users\Desktop\SPI.spiprj

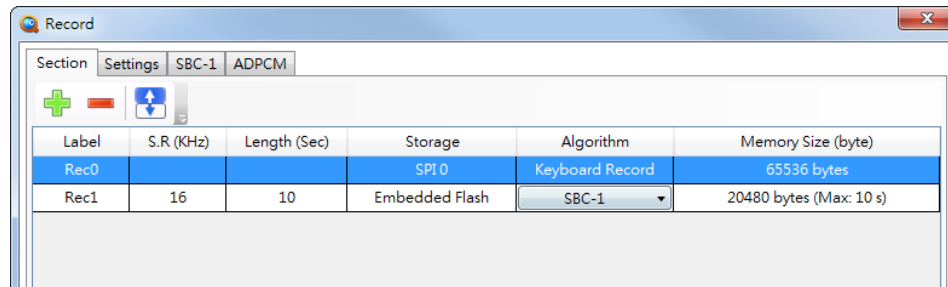
3.8 Record

3.8.1 NX1

NX1 provides this function to perform recording. If there any record blocks using SPI Flash as storage, recorded data will be written to SPI Flash. To save recorded data, spaces must be reserved on SPI Flash. Thus no matter add SPI Flash or not, _SPI.bin file will be generated after build.

3.8.1.1 Section

Record blocks are defined in **[Record]**. Recording / Erasing operation works on record block.



Length: Time of voice recording, The size of record section if determined by Recording Algorithm and specified Length. The size of KRecord block is fixed to 64k bytes.

Storage : Where to save recorded data. Accepts SPI0 / SPI1 / Embedded Flash. KRecord block only supports SPI0.

Algorithm : SBC-1 / ADPCM is for voice recording, and Keyboard Record is for Keyboard recording.

Memory Size : 1 / ADPCM is for voice recording, and Keyboard Record is for Keyboard recording.

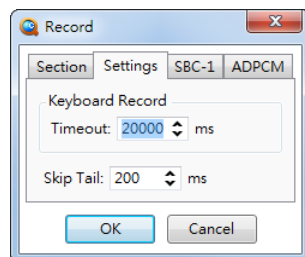
Ex.

[Record]

Rec0 = [Algorithm: KRecord, Storage: SPI0, Size: 64kb]

Rec1 = [Algorithm: SBC-1, Storage: EF, Size: 10s]

3.8.1.2 Settings



Skip Tail: Skip tail data of recorded voice. This can be used to avoid button clicking sound being recorded.

Keyboard Record Timeout: End the keyboard recording when not receiving InstNoteOn / InstNoteOff for a period of time.

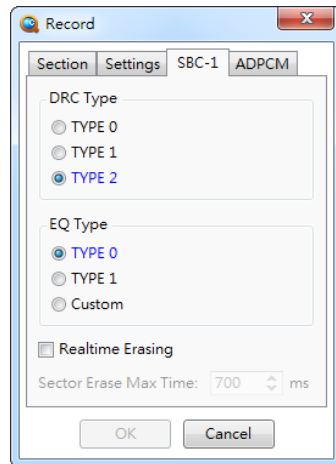
Ex.

[Record]

KRecord_Timeout = 20000ms

Skip_Tail = 200ms

3.8.1.3 SBC-1



DRC Type: For record playback, TYPE 0 does not make any change, which can reduce loading. TYPE 1 decrease the gap between the loudest and smallest volume, making the listening feel thicker and louder. TYPE 2 optimizes the low volume part to make it more clear (Default is TYPE 2).

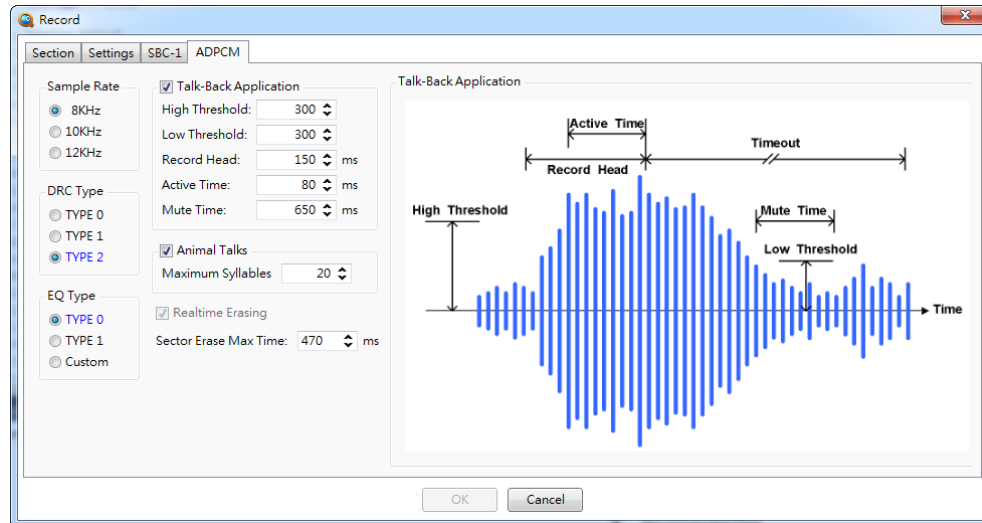
EQ Type: EQ effect for record playback. TYPE 0 will sharp the sound characteristics which makes sound more bright, suitable for music type, TYPE 1 enhances the bass, makes the sound thicker and makes human voice more prominent. Customize uses Audio Filter to process audio (Default is TYPE 0).

Realtime Erasing: If realtime erasing is not enabled, user has to erase record block data by EraseR / EraseRS commands before recording. With realtime erasing enabled, Q-Code will automatically erase record block on SPI Flash on demand. Realtime Erasing cost more RAM.

Sector Erase Max Time: When using realtime erasing, erasing process will terminate immediately after erasing timeout is occurred. Its strongly recommended to set suitable value according to SPI Flash spec.

Note: SBC-1 algorithm only supports 16k sampling rate.

3.8.1.4 ADPCM



Sample Rate: Set the Sample Rate option of ADPCM, supports 8KHz / 10KHz / 12KHz.

DRC Type: Same as DRC Type of SBC-1.

EQ Type: Same as EQ Type of SBC-1.

Talk-Back Application: Enable the Talk-Back function.

High Threshold: Detect the threshold value of voice. If the value is higher than the threshold and exceeds Active Time, it will be considered a valid speech.

Low Threshold: Detect the threshold of nonverbal. If the value is lower than the threshold and exceeds Mute Time, it will be considered as background sound then stop recording.

Record Head: The starting speech of the recording. When the valid speech is detected, the actual starting point of recording will extend forward the time of Record Head.

Active Time: Detect the duration of voice. If the duration is longer than High Threshold and exceeds the detection time, the voice will be considered a valid speech.

Mute Time: Detect the duration of nonverbal. If the speech duration is shorter than Low Threshold and exceeds the detection time, it will be considered as background sound then stop recording.

Animal Talks: Enable Animal Talks.

Maximum Syllable: Maximum number of syllables.

Realtime Erasing: Same as Realtime Erasing of SBC-1.

Sector Erase Max Time: Same as Sector Erase Max Time of SBC-1.

Note:

1. Record function requires SPI_Encoder 1.53 or new version.
2. Recording and playing audio (voice or melody) cannot be used at the same time.
3. When Q-Code is erasing record block on SPI Flash, program on XIP will be paused about

*30 ~ 100ms (Realtime erasing is enabled) or 100 ~ 200ms (Realtime erasing is disabled).
Pause time may vary according to SPI Flash.*

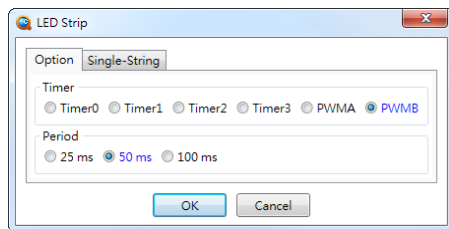
- 4. When record block is erasing, SPI Flash operations do not work.*
- 5. When playing content on SPI Flash (voice / melody / action ...), erasing record block will cause playing not work correctly.*
- 6. Keyboard recording function is not available if Realtime Erasing is enabled.*

3.9 LED Strip

LED Strip can be used to output to led strip composed by multiple RGB LED. LED strip data are introduced by .csv generated by Vixen Lights. RGB LEDs need support dedicated communication protocol.

3.9.1 Option

3.9.1.1 NX1



Timer: Timer source, which are exclusive to other applications.

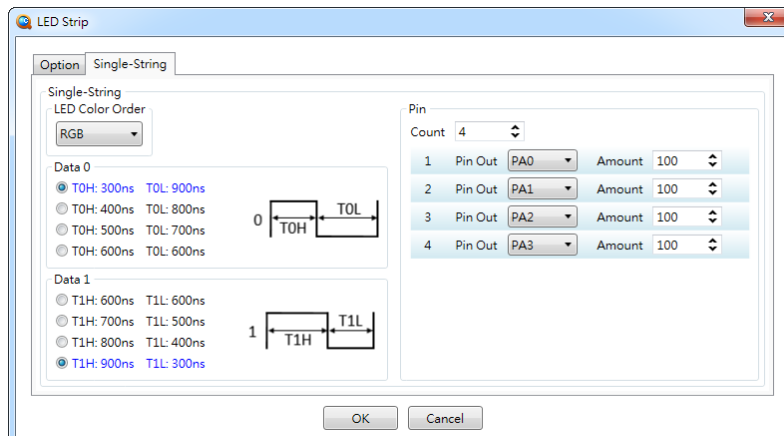
Ex. Timer = PWMA

Period: Update period of LED strip.

Ex. Period = 100ms

3.9.2 Single String

3.9.2.1 NX1



LED Color Order: RGB ordering of LED on strip.

Ex. Single_LED_Order = RGB

LED Data 0: Communication format of LED strip Data 0.

Ex. Single_Data0 = H300_L900

LDE Data 1: Communication format of LED strip Data 1.

Ex. Single_Data1 = H900_L300

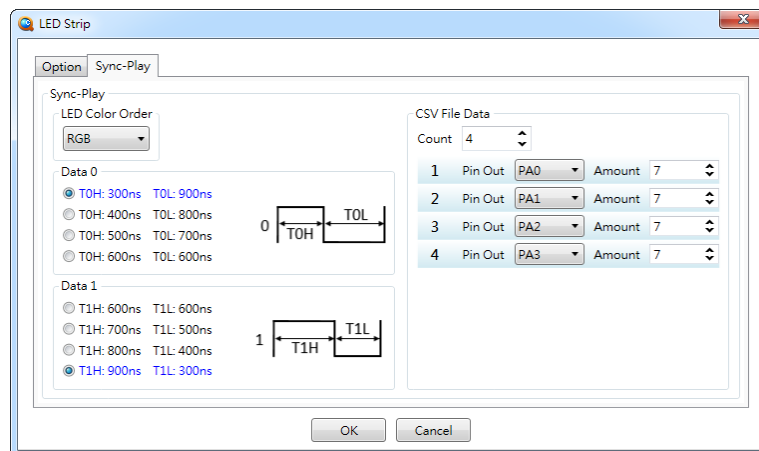
Pin: Select pins to be used to output Single-String. Also set the LED count on strip attached on specified pin. Total number of LED cannot exceed 399 if Period is 25ms, 799 if Period is 50ms, 1599 if Period is 100ms. And in addition, count of output pin effects max number of LED.

Ex. Single_LedCount = [PA.0:100]

Note: The Reset Time of LED Strip must exceed 60us.

3.9.3 Sync-Play

3.9.3.1 NX1



LED Color Order: RGB ordering of LED on strip.

Ex. Sync_LED_Order = RGB

LED Data 0: Communication format of LED strip Data 0.

Ex. Sync_Data0 = H300_L900

LDE Data 1: Communication format of LED strip Data 1.

Ex. Sync_Data1 = H900_L300

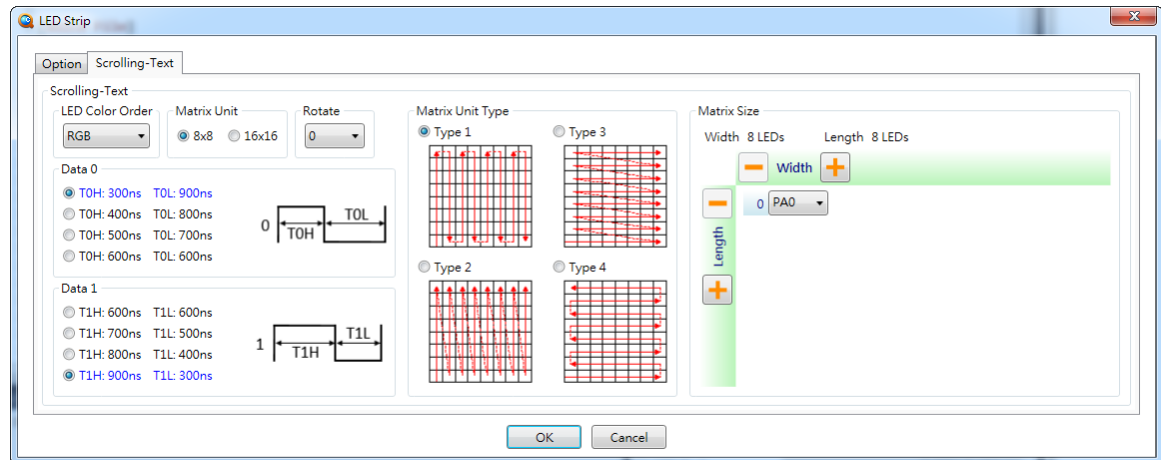
CSV File Data: Select pins to be used to output Sync-Play pattern. Total number of LED cannot exceed 300 for each pin and must be the same Port.

Ex. Sync_LEDCount = [PA.0:7]

Note: The Reset Time of LED Strip must exceed 60us

3.9.4 Scrolling-Text

3.9.4.1 NX1



LED Color Order: RGB ordering of LED on strip.

Matrix Unit: Select Marix Unit.

Rotate: Set Marix Unit rotation angle.

LED Data 0: Communication format of LED strip Data 0.

LDE Data 1: Communication format of LED strip Data 1.

Matrxi Unit Type: Select Marix Unit Type.

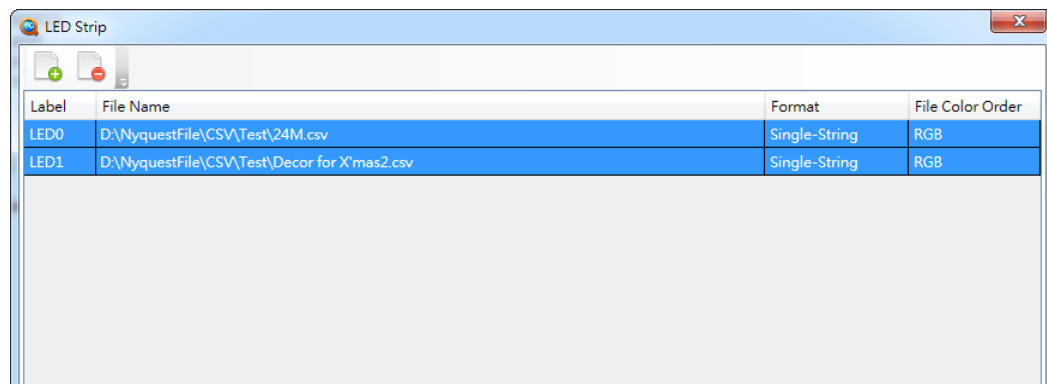
Matrix Size: Select matrix size and pins to be used to output signal to unit. Total number of LED cannot exceed 300 both in width and height. And must use consecutive pins in the same Port.

Note: The Reset Time of LED Strip must exceed 60us

3.9.5 Add File

3.9.5.1 NX1

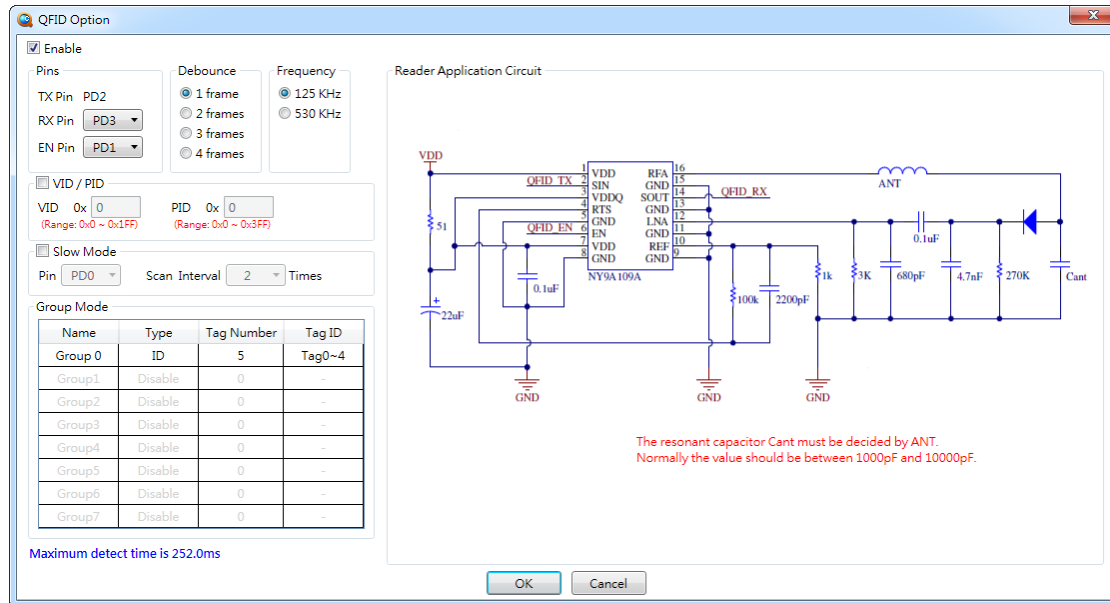
Use Add File to introduce .csv and .ledstr files. File Color Order field can be used to change the RGB order.



3.10 QFID

The QFID function is to provide RFID application development in Q-Code. With application circuit and associated Tag provided by Nyquest, user can easily develop RFID applications.

3.10.1 NY5+ / NY7 / NX1



TX Pin: Pin for generating QFID carrier.

- Supports PA.0, PB.0, PC.0, PD.0, PE.0 in NY5+.
- TX pin is fixed to PB.2 of NY7A, PD.2 of NY7B, PF.2 of NY7C.
- TX pin is fixed to PA.5 of NX1 OTP.
- Supports PA.1 / PA.2 / PC.0 / PC.1 in NX1 EF.

RX Pin: Pin to receive data from Tag. RX pin should be connected to the SOUT pin of NY9A109A

En Pin: Pin to enable the function of NY9A109A. Q-Code will control NY9A109A automatically, EN pin should be connected to the EN pin of NY9A109A.

Debounce: Set the debounce time when QFID is scanning. NY5+ / NY7 supports 1 / 2 / 3 / 4 frames, both for Detect / Leave of tag. In NX1, Detect / Leave debounce time can be set separately, supports 1 ~ 50ms.

Frequency: QFID frequency has two options: 125KHz and 530KHz. When the user chooses 530KHz, although needs to add a NY8A051D, the tag can be made as PCB antenna. After setting project, An additional CarrierGen-530KHz.bin for programming to NY8A051D will be automatically generated by Q-Code.

Timer: Select system timer used for QFID in NX1 series.

Slow Pin: The rest time of Slow Mode is counted by RC charging / discharging of specified I/O. To enable Slow Mode, it is necessary to connect a 100KΩ resistor and 0.1uF capacitor in parallel to the specified I/O.

Scan Interval: QFID Slow Mode will scan for tags once and rest several times, the rest times is determined by the value of Scan Interval. For example, when Scan Interval is selected as 2, it means to scan once and rest twice. If the scan time is 200ms, the rest time is about 400ms. The scan time is determined by tag number. The user can also change the resistance and capacitance to adjust the Interval unit time. The calculated unit time after the change is as follows.

$$\text{Interval}_{(\text{new})} = (\text{Interval}_{(\text{original})} \times R \times C) / 0.01$$

Note: The rest time is an approximate value and may deviate.

QFID_GroupX_Mode: QFID mode. ID is for identifying Tag only. ID+Input is to identify Tag and read the I/O state on Tag.

QFID_GroupX_Tags: The available numbers of tag. 1 to 16 tags are applicable if QFID_GroupX_Mode is ID, and the corresponding tag IDs are Tag0 ~ Tag15. 1 to 8 tags are applicable if QFID_GroupX_Mode is ID+input, and the corresponding tag IDs are Tag0 ~ Tag7.

QFID_VID: Set the Vender ID of product, the range is 0x0~0x1FF. If the Vender ID of Tag is different from the reader's setting, the Tag won't be recognized.

QFID_PID: Set the Project ID of product, the range is 0x0~0x3FF. If the Project ID of Tag is different from the reader's setting, the Tag won't be recognized.

QFID_VID / QFID_PID is used to avoid the interference between different products' Tags. The VID / PID of Tag and reader must be exactly the same in order to be recognized. User can also disable the VID / PID function.

Note:

1. NX1 does not support Slow Pin and Scankey Interval options.
2. NY5+ / NY7 only supports QFID_Group0.

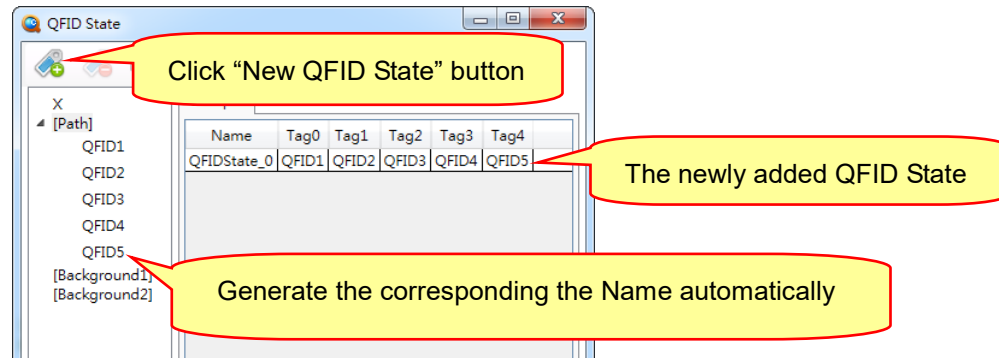
User has to define the corresponding trigger state in **[QFID]** section. The entry in QFID state specifies the action to perform when Tag is detected by reader or left reader. The entry number cannot exceed the number of QFID_GroupX_Tags, and the entries have to be separated by blank (TAB or Space).

The allowed entries are as follows.

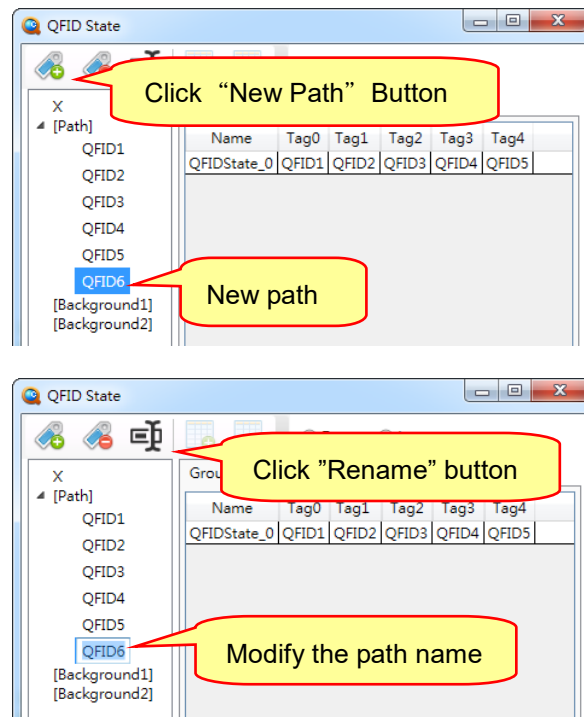
Format	Tag Trigger Type	Descriptions
X	Ignore	No action.
Path1	Detect	When Tag is detected by reader, executing Path1.
/Path2	Leave	When Tag leaves reader, executing Path2.
Path1/Path2	Detect & Leave	When Tag is detected, executing Path1. When Tag is left, executing Path2.

Step: QFID → QFID State → Add State Name → Add PathName → OK, the operations are below.

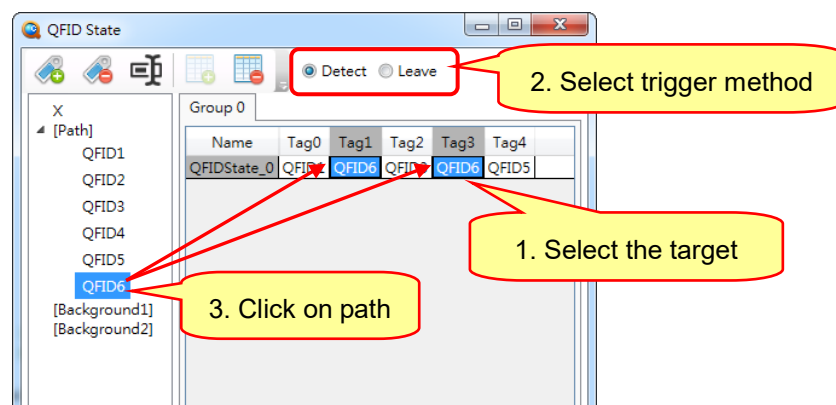
Step1: Add QFID State.



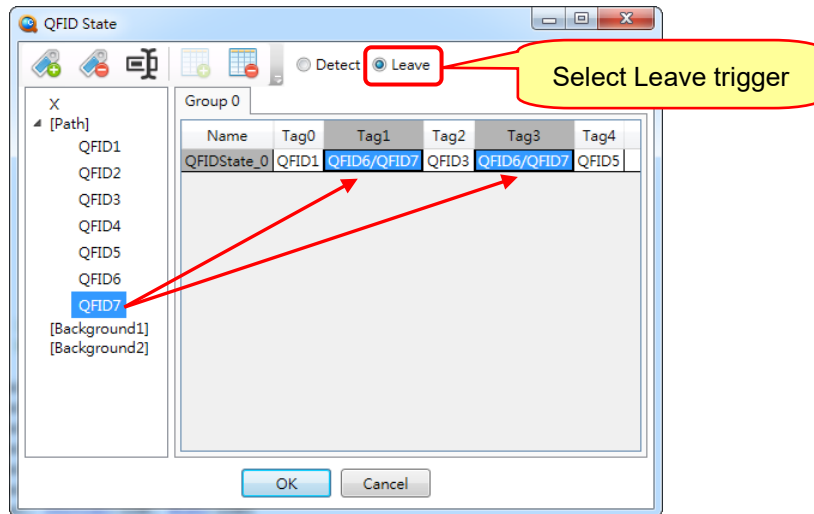
Step2: Add path or modify the path name.



Step3: Modify the corresponding path. Although Q-Code fills the entries by default, user can modify the corresponding path if necessary.



If user wants to set the entry when Tag is left, just select the Leave button on the 2nd action as the picture shown below.



Ex.

[QFID]

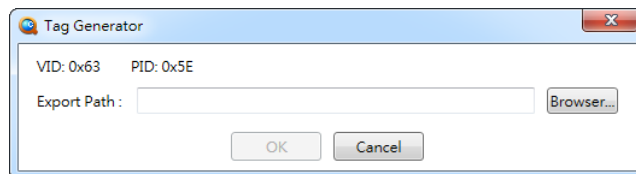
;===== State =====

QFIDGroup0_0: QFID1 QFID2 QFID3 QFID4 QFID5

QFIDGroup0_1: QFID1/QFID3 QFID2/QFID3 QFID3/QFID3 QFID4/QFID3 QFID5/QFID3

The QFID function can only coordinate with the .bin file of Tag generated by Q-Code. Q-Code provides Tag Generator tool to generate the Tag .bin file. User can program the file to Tag IC through Q-Writer.

Usage: Tool→QFID Tag Generator



VID: The corresponding QFID Vender ID.

PID: The corresponding QFID Project ID.

Export Path: Set the location of the generated .bin file.

After setting, click the OK button, the Tag .bin file and circuit diagram applied by Tag will be generated in the path that specified by user.

Out put File and Circuit:

{VID}_{PID}_{TagID}.bin

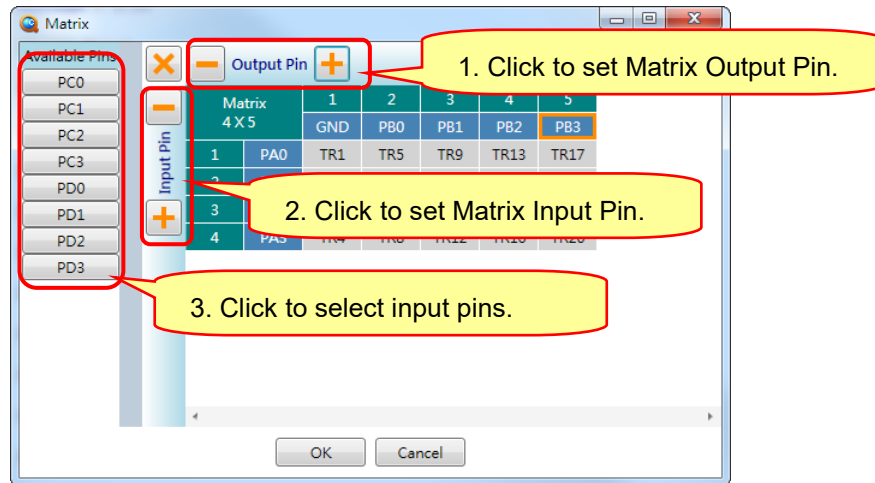
Tag_Schematic_ID.jpg.

3.11 I/O

3.11.1 Matrix Key

3.11.1.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NX1

Click “Matrix” in the I/O section and click +/- buttons on the popped out window to configure “Matrix”.
Click the Available Pins buttons to set pins. The actual operation is shown in the following figure:

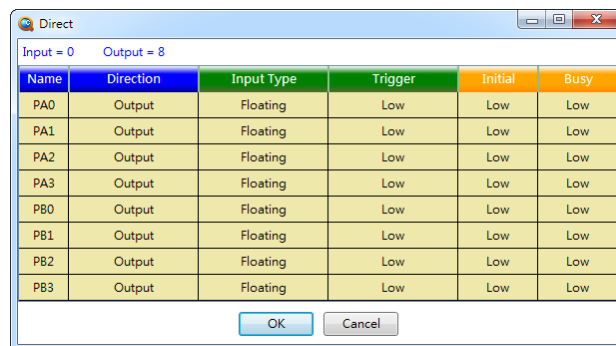


Ex.

Matrix = [PA.0, PA.1] * [GND, PB.0, PA.2] ; Using GND as a pin of matrix.

3.11.2 Direct Key

3.11.2.1 NY4



Direction : Set the output/input direction when initializing I/O.

- Input : Set as input type when initializing,
- Output : Set as output type when initializing.

Input Type: Set the pull-high resistor of normal I/O pin as input.

- Pull-High: Input mode with Pull-High Resistor.
- Floating: Input mode with Floating.

Trigger: Set the trigger level of input pin.

- Low: Low Trigger.

- High: High Trigger.

Initial: Set the initial output voltage.

- High: Set the initial output type as Output High.
- Low: Set the initial output type as Output Low.

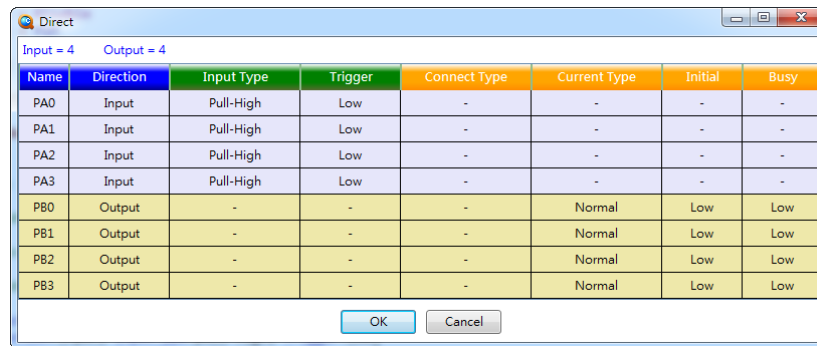
Busy: Set the output of Action signals.

- Low: Set Action as Active Low (Sink Output).
- High: Action as Active High (Drive Output).

Ex.

PA.0 = [**Direction**:Input, **InputType**:Pull-high, **Trigger**:Low, **Initial**:Low, **Busy**:High]

3.11.2.2 NY5



Name	Direction	Input Type	Trigger	Connect Type	Current Type	Initial	Busy
PA0	Input	Pull-High	Low	-	-	-	-
PA1	Input	Pull-High	Low	-	-	-	-
PA2	Input	Pull-High	Low	-	-	-	-
PA3	Input	Pull-High	Low	-	-	-	-
PB0	Output	-	-	-	Normal	Low	Low
PB1	Output	-	-	-	Normal	Low	Low
PB2	Output	-	-	-	Normal	Low	Low
PB3	Output	-	-	-	Normal	Low	Low

Direction : Set the output/input direction when initializing I/O.

- Input: Set as input type when initializing.
- Output: Set as output type when initializing.

Input Type: Set the pull-high resistor of normal I/O pin as input.

- Pull-High: Input mode with Pull-High Resistor.
- Floating: Input mode with Floating.
- IO Pull-High: I/O mode with Pull-High resistor.
- IO Open-Drain: I/O Input mode without Pull-High resistor and with Open Drain.

Trigger: Set the trigger level of input pin.

- Low: Low Trigger.
- High: High Trigger.

Connect Type: Set the driving mode of external components.

- Sink: Set drive mode of external components as Active Low.
- Drive: Set drive mode of external components as Active High.

Current Type: Set the output of current.

- Normal: Normal current output.
- Large: Large current.

Initial: Set the initial output voltage.

- High: Set the initial output type as Output High.
- Low: Set the initial output type as Output Low.

Busy: Set the output of Action signals.

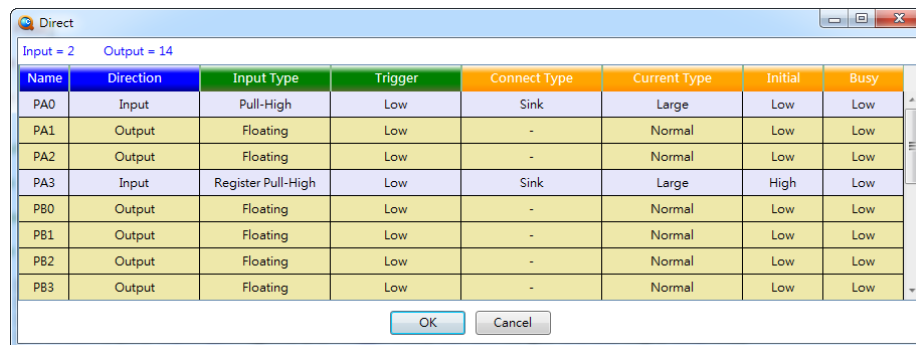
- Low: Set Action as Active Low (Sink Output).
- High: Action as Active High (Drive Output).

Note: I/O pin must use IO_Pull-High to use as input pin. If I/O pin is as input pin and not switching to output mode, the system won't enter sleep mode as pin is pressed. To use I/O pin as output, it must be set as Normal current output and Initial Low, and only supports Sink output.

Ex.

PA.0 = [Direction:Input, InputType:Pull-high, Trigger:Low]

3.11.2.3 NY5+



Name	Direction	Input Type	Trigger	Connect Type	Current Type	Initial	Busy
PA0	Input	Pull-High	Low	Sink	Large	Low	Low
PA1	Output	Floating	Low	-	Normal	Low	Low
PA2	Output	Floating	Low	-	Normal	Low	Low
PA3	Input	Register Pull-High	Low	Sink	Large	High	Low
PB0	Output	Floating	Low	-	Normal	Low	Low
PB1	Output	Floating	Low	-	Normal	Low	Low
PB2	Output	Floating	Low	-	Normal	Low	Low
PB3	Output	Floating	Low	-	Normal	Low	Low

Direction : Set the output/input direction when initializing I/O.

- Input: Set as input type when initializing.
- Output: Set as output type when initializing.

Input Type: Set the pull-high resistor of normal I/O pin as input.

- Pull-High: Input mode with Pull-High Resistor.
- Floating: Input mode with Floating.
- Register Pull-High: Pull-High Resistor Control. After setting this mode, Pull-High Resistor can be switched on or off by the IO Register.

Ex.

[Path]

TR1: PA=0xF ; Open Pull-High resistor of PA.0, PA.1.

TR2: PA=0x0 ; Close Pull-High resistor of PA.0, PA.1.

Trigger: Set the trigger level of input pin.

- Low: Low Trigger.
- High: High Trigger.

Note: In **InputType: Register_PH** and **Trigger: Low** mode, IC can only be awaked by low-level signal.

Connect Type: Set the driving mode of external components.

- Sink: Set drive mode of external components as Active Low.
- Drive: Set drive mode of external components as Active High.

Current Type: Set the output of current.

- Normal: Normal current output.
- Large: Large current.

Initial: Set the initial output voltage.

- High: Set the initial output type as Output High.
- Low: Set the initial output type as Output Low.

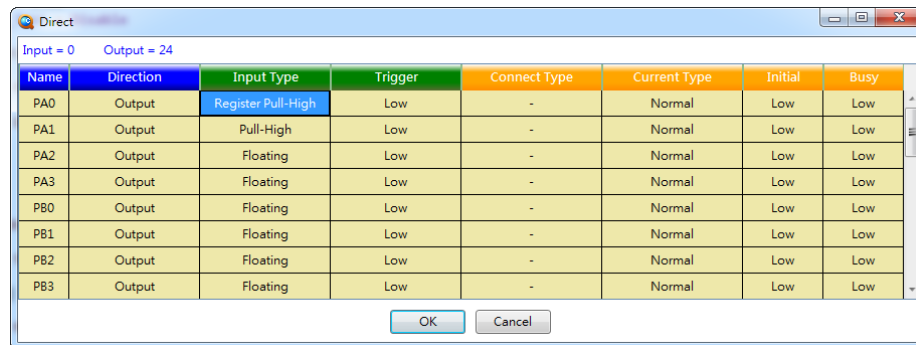
Busy: Set the output of Action signals.

- Low: Set Action as Active Low (Sink Output).
- High: Action as Active High (Drive Output).

Ex.

PA.1 = [**Direction:**Output, **InputType:**Floating, **Trigger:**Low, **Current:**Normal, **Initial:**Low, **Busy:**High]

3.11.2.4 NY6 / NY7



Name	Direction	Input Type	Trigger	Connect Type	Current Type	Initial	Busy
PA0	Output	Register Pull-High	Low	-	Normal	Low	Low
PA1	Output	Pull-High	Low	-	Normal	Low	Low
PA2	Output	Floating	Low	-	Normal	Low	Low
PA3	Output	Floating	Low	-	Normal	Low	Low
PB0	Output	Floating	Low	-	Normal	Low	Low
PB1	Output	Floating	Low	-	Normal	Low	Low
PB2	Output	Floating	Low	-	Normal	Low	Low
PB3	Output	Floating	Low	-	Normal	Low	Low

Direction : Set the output/input direction when initializing I/O.

- Input: Set as input type when initializing.
- Output: Set as output type when initializing.

Input Type: Set the pull-high resistor of normal I/O pin as input.

- Pull-High: Input mode with Pull-High Resistor.
- Floating: Input mode with Floating.
- Register Pull-High: Pull-High Resistor Control. After setting this mode, Pull-High Resistor can be switched on or off by the IO Register.

Ex.

[Path]

TR1: PA=0xF ; Open Pull-High resistor of PA.0, PA.1.

TR2: PA=0x0 ; Close Pull-High resistor of PA.0, PA.1.

Trigger: Set the trigger level of input pin.

- Low: Low Trigger.
- High: High Trigger.

Note: In InputType: Register_PH and Trigger: Low mode, IC can only be awaked by low-level signal.

Connect Type: Set the driving mode of external components.

- Sink: Set drive mode of external components as Active Low.
- Drive: Set drive mode of external components as Active High.

Current Type: Set the output of current.

- Normal: Normal current output.
- Large: Large current.

Initial: Set the initial output voltage.

- High: Set the initial output type as Output High.
- Low: Set the initial output type as Output Low.

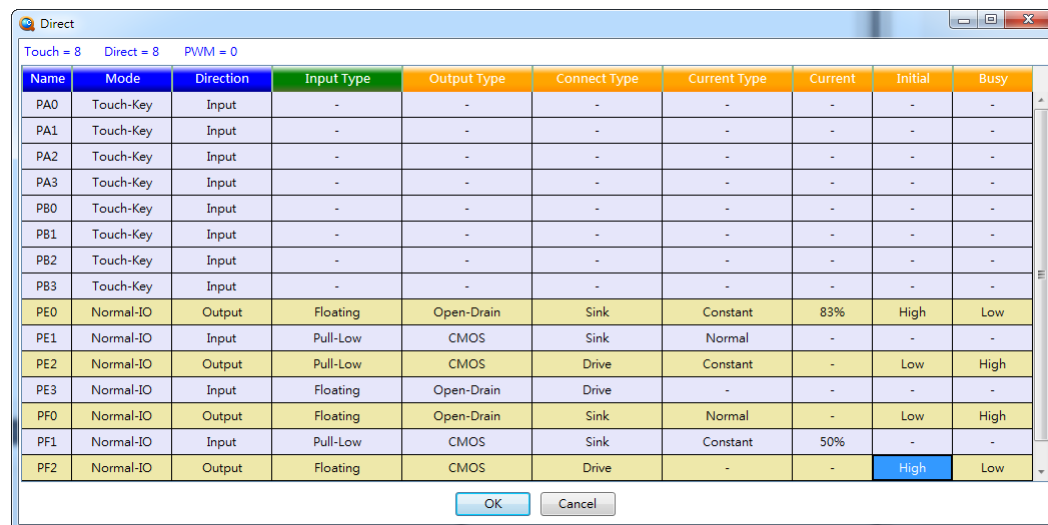
Busy: Set the output of Action signals.

- Low: Set Action as Active Low (Sink Output).
- High: Action as Active High (Drive Output).

Ex.

PA.1 = [Direction:Output, InputType:Large, Trigger:Low, Current:Normal, Initial:Low, Busy:High]

3.11.2.5 NY9T



Name	Mode	Direction	Input Type	Output Type	Connect Type	Current Type	Current	Initial	Busy
PA0	Touch-Key	Input	-	-	-	-	-	-	-
PA1	Touch-Key	Input	-	-	-	-	-	-	-
PA2	Touch-Key	Input	-	-	-	-	-	-	-
PA3	Touch-Key	Input	-	-	-	-	-	-	-
PB0	Touch-Key	Input	-	-	-	-	-	-	-
PB1	Touch-Key	Input	-	-	-	-	-	-	-
PB2	Touch-Key	Input	-	-	-	-	-	-	-
PB3	Touch-Key	Input	-	-	-	-	-	-	-
PE0	Normal-IO	Output	Floating	Open-Drain	Sink	Constant	83%	High	Low
PE1	Normal-IO	Input	Pull-Low	CMOS	Sink	Normal	-	-	-
PE2	Normal-IO	Output	Pull-Low	CMOS	Drive	Constant	-	Low	High
PE3	Normal-IO	Input	Floating	Open-Drain	Drive	-	-	-	-
PF0	Normal-IO	Output	Floating	Open-Drain	Sink	Normal	-	Low	High
PF1	Normal-IO	Input	Pull-Low	CMOS	Sink	Constant	50%	-	-
PF2	Normal-IO	Output	Floating	CMOS	Drive	-	-	High	Low

Mode : Set the I/O mode.

- Normal-IO: Set the pin as normal I/O.

- Touch: Touch-Key mode. Direction is fixed as Input.
- PWM: Set the pin as PWM output

Direction : Set the output/input direction when initializing I/O.

- Input: Set as input type when initializing.
- Output: Set as output type when initializing.

Note: *PA.0 ~ PD.3 are Touch-Keys that share pin with normal I/O, the PE.0 ~ PF.3 keys are normal I/O that share pin with PWM-I/O output. Pins that are set as Touch-Key or PWM-I/O cannot change direction in program. Whereas Normal I/O pins can change direction in program.*

Input Type: Set the pull-high resistor of normal I/O pin as input.

- Pull-Low: Input mode with Pull-Low Resistor.
- Floating: Input mode with Floating.

Output Type: Set the output type of normal I/O pin as output.

- CMOS: The normal CMOS I/O, Q-Code provides this option when user sets output type as Normal-I/O mode.
- Open-Drain: Open-Drain without pull-high resistor I/O mode, Q-Code provides this option when user sets output type as Normal-I/O mode.

Connect Type: Set the driving mode of external components.

- Sink: Set drive mode of external components as Active Low.
- Drive: Set drive mode of external components as Active High.

Current Type: Set the output of current.

- Normal: Normal current output.
- Large: Large current.
- Constant: Constant current output.

Current: Set the sink current option. Q-Code provides 4 options that only support in Constant and Large mode.

- 33%: Output 33% current.
- 50%: Output 50% current.
- 83%: Output 83% current.
- 100%: Output 100% current.

Initial: Set the initial output voltage.

- High: Set the initial output type as Output High.
- Low: Set the initial output type as Output Low.

Busy: Set the output of Action signals.

- Low: Set Action as Active Low (Sink Output).
- High: Action as Active High (Drive Output).

The table of NY9T pins corresponding with output current is shown below.

	NY9T001A	NY9T004A	NY9T008A	NY9T016A
Constant Drive	PE.1 ~ PE.2	Not Available	PE.2	Not Available
Constant Sink	PE.0 ~ PE.2	PE.0 ~ PE.3	PE.0 ~ PF.3	PE.0 ~ PF.3

Ex.

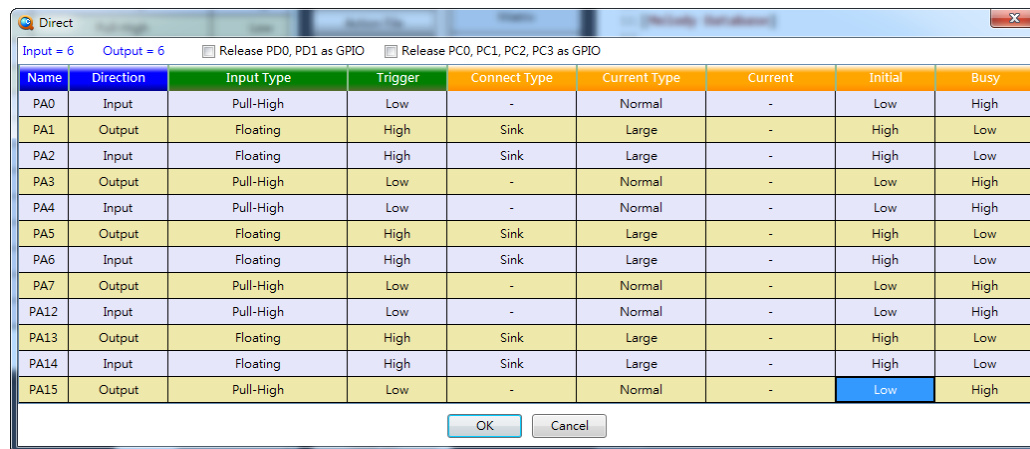
PA.0 = [Mode:Touch]

PB.0 = [Mode:Touch]

PE.0 = [Direction:Output, InputType:Pull-Low, OutputType:cmos, Connect:Sink,
Current:constant_100%, Initial:Low, Busy:High]

PE.1 = [Direction:Input, InputType:Pull-Low, OutputType:cmos, Connect:Sink, Current:Normal,
Initial:Low, Busy:High]

3.11.2.6 NX1



Name	Direction	Input Type	Trigger	Connect Type	Current Type	Current	Initial	Busy
PA0	Input	Pull-High	Low	-	Normal	-	Low	High
PA1	Output	Floating	High	Sink	Large	-	High	Low
PA2	Input	Floating	High	Sink	Large	-	High	Low
PA3	Output	Pull-High	Low	-	Normal	-	Low	High
PA4	Input	Pull-High	Low	-	Normal	-	Low	High
PA5	Output	Floating	High	Sink	Large	-	High	Low
PA6	Input	Floating	High	Sink	Large	-	High	Low
PA7	Output	Pull-High	Low	-	Normal	-	Low	High
PA12	Input	Pull-High	Low	-	Normal	-	Low	High
PA13	Output	Floating	High	Sink	Large	-	High	Low
PA14	Input	Floating	High	Sink	Large	-	High	Low
PA15	Output	Pull-High	Low	-	Normal	-	Low	High

Direction : Set the output/input direction when initializing I/O.

- Input: Set as input type when initializing.
- Output: Set as output type when initializing.

Input Type: Set the pull-high resistor of normal I/O pin as input.

- Pull-High: Input mode with Pull-High Resistor.
- Floating: Input mode with Floating.

Trigger: Set the trigger level of input pin.

- Low: Low Trigger.
- High: High Trigger.

Connect Type: Set the driving mode of external components.

- Sink: Set drive mode of external components as Active Low.
- Drive: Set drive mode of external components as Active High.

Current Type: Set the output of current.

- Normal: Normal current output.

- Large: Large current.
- Constant: Constant current output.

Current: Set the sink current option. Q-Code provides 4 options that only support in Constant and Large mode.

- 33%: Output 33% current.
- 50%: Output 50% current.
- 83%: Output 83% current.
- 100%: Output 100% current.

Initial: Set the initial output voltage.

- High: Set the initial output type as Output High.
- Low: Set the initial output type as Output Low.

Busy: Set the output of Action signals.

- Low: Set Action as Active Low (Sink Output).
- High: Action as Active High (Drive Output).

Ex.

PA.0 = [Direction:Input, InputType:Pull-high, Trigger:Low, Current:Normal, Initial:High, Busy:Low]

PA.1 = [Direction:Output, InputType:Floating, Trigger:High, Current:Large, Initial:Low, Busy:High]

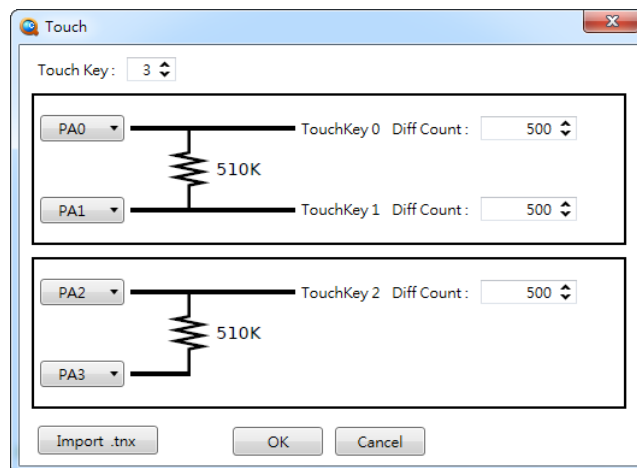
PA.2 = [Direction:Input, InputType: Pull-high, Trigger:High, Current:Large, Initial:Low, Busy:High]

PA.3 = [Direction:Input, InputType: Pull-high, Trigger:High, Current:Large, Initial:Low, Busy:High]

3.11.3 Touch Key

3.11.3.1 NX1

User can set the pins and the count of Touch Key and in the window shown below. The settings of Touch Key that supported by NX1 is different from NY9T.



Touch Key: Set the count of Touch Key.

Import .tnx: Import .tnx file.

Note: Each Touch needs 2 pins and 2 Touch Keys.

Ex.

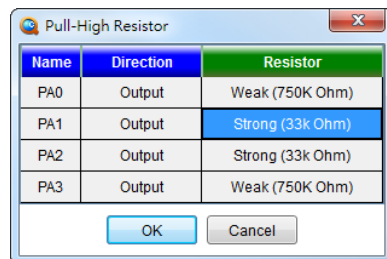
TouchKey = 2

TouchGroup = [PIN0:PA.2, PIN1:PA.3, Diff0:500] ; **TouchKey0**

3.11.4 Push-High Resistor

3.11.4.1 NY4

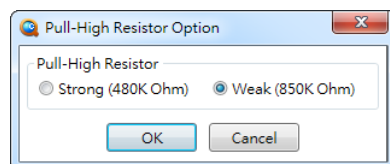
In NY4, each pins can set pull-up resistor. Weak is 750k Ohm pull-high resistor, Strong is 33k Ohm pull-high resistor.



Ex. Input_Resistor = [PA.0:weak, PA.1:weak, PA.2:weak, PA.3:weak]

3.11.4.2 NY5

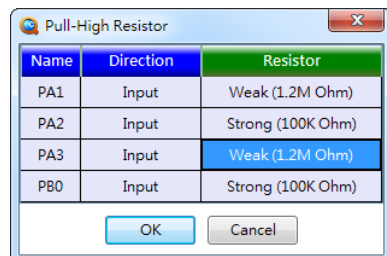
NY5 can only use the global pull-up resistor option to set pull-high resistor. Weak is input pull-high resistor with 850k Ohm pull-high resistor, Strong is input pull-high resistor with 480k Ohm pull-high resistor.



Ex. Input_Resistor = strong

3.11.4.3 NY5+

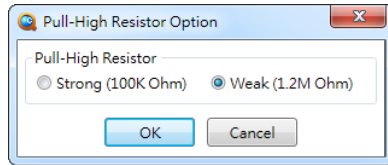
In NY5+, each pins can set pull-up resistor. Weak is Weak pull-high resistor 1.2M Ohm, Strong is Strong pull-high resistor (100k Ohm).



Ex. Input_Resistor = [PA.0:weak, PA.1:Strong, PA.2:weak, PA.3:weak]

3.11.4.4 NY6

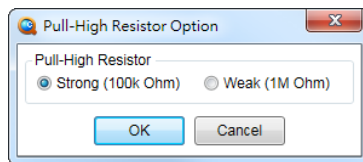
NY6 can only use the global pull-up resistor option to set pull-high resistor. Weak is input pull-high resistor with 1.2M Ohm pull-high resistor. Strong is input pull-high resistor with 100k Ohm pull-high resistor.



Ex. Input_Resistor = strong

3.11.4.5 NY7

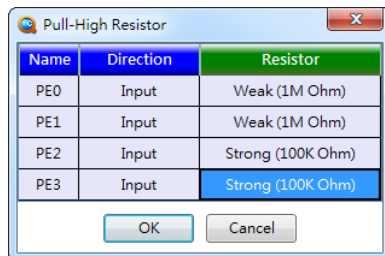
NY7 can only use the global pull-up resistor option to set pull-high resistor. Weak is input pull-high resistor with 1M Ohm pull-high resistor. Strong is input pull-high resistor with 100k Ohm pull-high resistor.



Ex. Input_Resistor = strong

3.11.4.6 NY9T

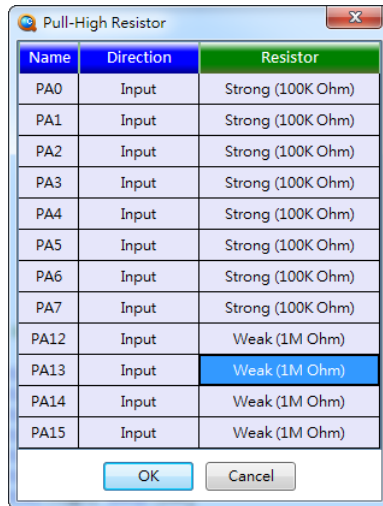
For NY9T, each pin can set pull-up resistor. Weak is 1M Ohm pull-high resistor, Strong is 100k Ohm pull-high resistor.



Ex. Input_Resistor = [PE.0:weak, PE.1:weak, PE.2:weak, PE.3:weak]

3.11.4.7 NX1

NX1, each pin can set pull-up resistor, but pin 0 ~ 7 of the same port must be the same, same as pin 8 ~ 15. Weak is 1M Ohm pull-high resistor, Strong is 100k Ohm pull-high resistor.

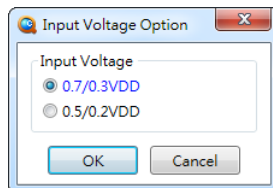


Ex. Input_Resistor = [PA.0:Strong, PA.1:Strong, PA.2:Strong, PA.3:Strong,
PA.4:Strong, PA.5:Strong, PA.6:Strong, PA.7:Strong,
PA.12:weak, PA.13:weak, PA.14:weak, PA.15:weak]

3.11.5 Input Voltage

3.11.5.1 NX1

Select the input voltage.



Ex. Input_Voltage = VIH7_VIL3 ; High Potential 0.7VDD, Low Potential 0.3VDD

3.12 Input State

3.12.1.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1

User can define corresponding key configuration in **[Input State]** section with up to 255 input states. The key configuration means the reactions while the corresponding input keys are pressed or released. The trigger number must be identical to the number of input pins, and the triggered events must be separated by TAB or Space.

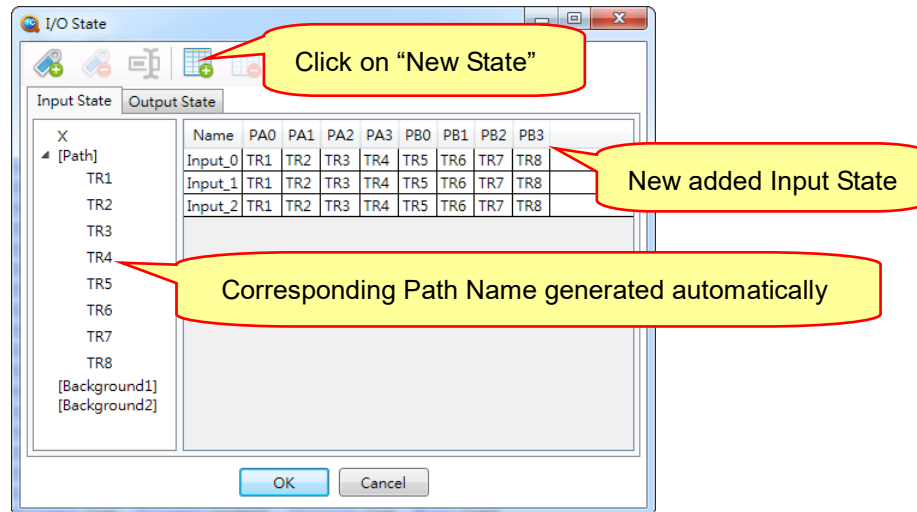
The status of the input pin can be in one of the following forms:

Format	Trigger	Description
X	Ignore	No action.
Path1	Falling edge	Execute Path1 as input pin changed from high to low electric potential.

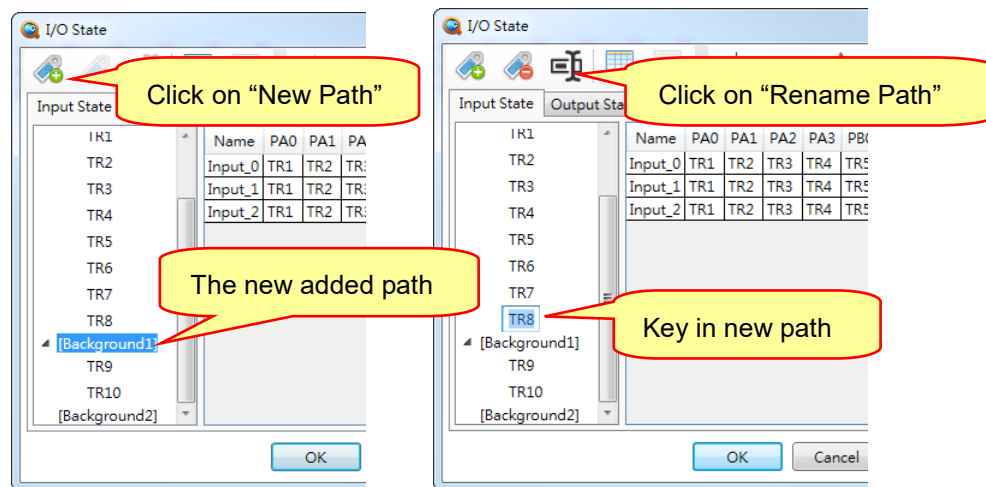
/Path2	Rising edge	Execute Path2 as input pin changed from low to high electric potential.
Path1	Falling edge	Execute Path1 as input pin changed from high to low electric potential.
/Path2	& Rising edge	Execute Path2 as input pin changed from low to high electric potential.

The actual operating steps are shown below.

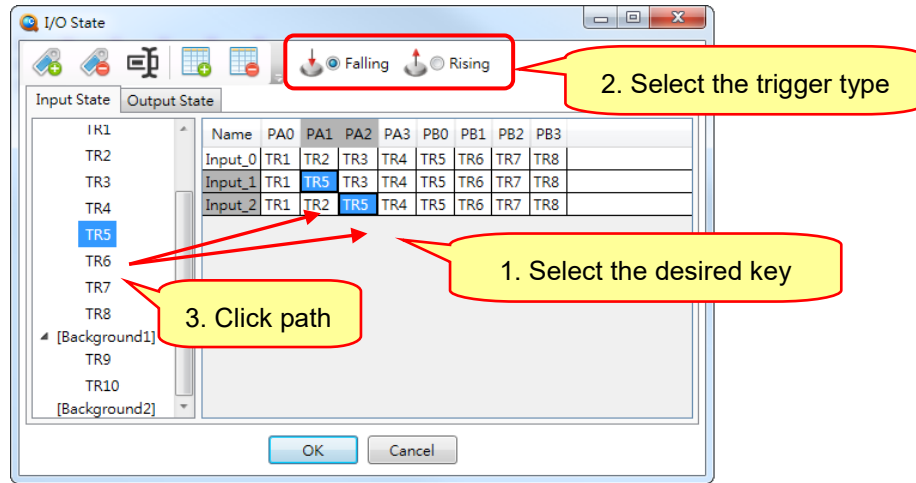
Step 1: Click to add **Input State**.



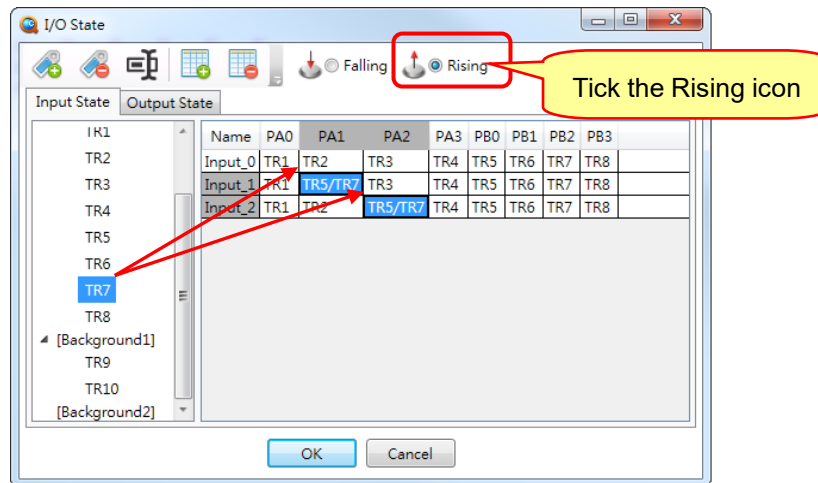
Step 2: Add/modify the name of path as shown in the following figure.



Step 3: Write the "Path Name" into the input state. Since system has pre-assigned path name, user might want to change the path name according to their needs.



If user wants to select trigger type as Rising, just tick the Rising icon after selecting the desired key.



Ex.

[Input State]

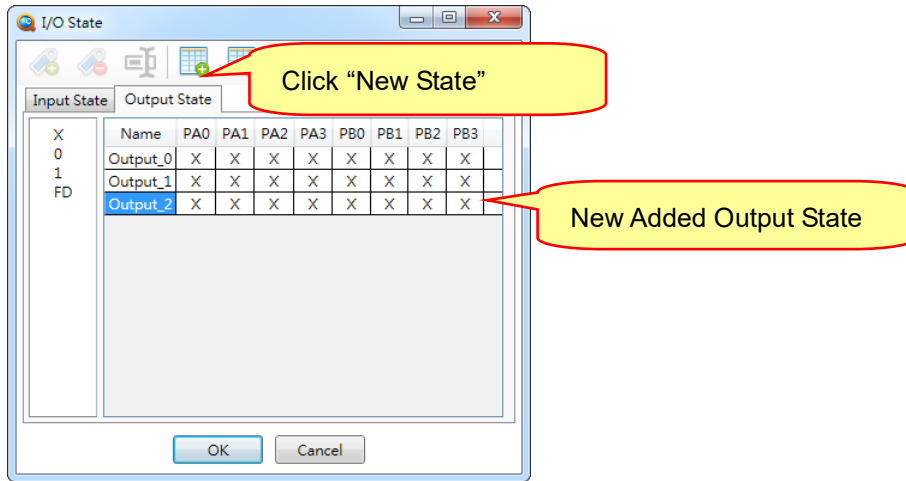
Input_0: TR1R X /TR3F TR4R/TR4F
Input_1: TR1R X X /TR3F TR4R/TR4F
Input_2: TR1R X TR3F / X TR4R/TR4F

3.13 Output State

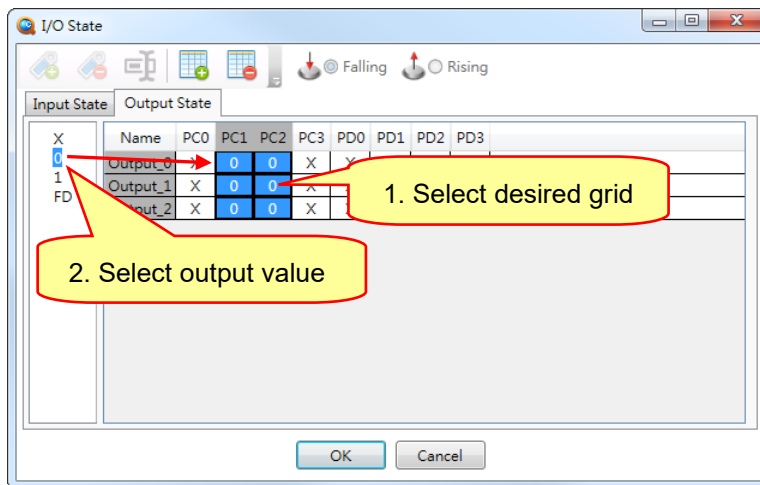
User can define output state in **[Output State]** section with up to 255 output states. The output states defined in **[Output State]** can be called by **[Path]** and **[Background]**. Each output state consists of one output state name, one “:” and designated output value. The output value means the output signal for each output pin when the output state is called. The number of output value must be identical to the number of output pins, and the output values must be separated by TAB or space.

The actual operating steps are shown below.

Step 1: Add Output State.



Step 2: Modify output value.



Ex.

[Output State]

OState0: 0 1 0 FD
OState1: 1 0 0

3.13.1 NY4 / NY5

The followings are six possible values for output state definition:

Output Value	Description
X	Indicating the pin maintains the original output signal.
0	Indicating the pin is with low electric potential output.
1	Indicating the pin is with high electric potential output.
A	Indicating the pin is with Action output (This is function is only available for playing entire .vio files).
FD	Indicating the output will change along with volume (Flash with Dynamic).

Output Value	Description
Q	Indicating each pin will follow the variation of the QIO signal of Quick-IO.

3.13.2 NY5+

The followings are six possible values for output state definition:

Output Value	Description
X	Indicating the pin maintains the original output signal.
0	Indicating the pin is with low electric potential output.
1	Indicating the pin is with high electric potential output.
A	Indicating the pin is with Action output (This is function is onlyavailable for playing entire .vio files).
P	Indicating the pin will output PWM-IO (Only takes effect for PlayPWM(\$MPIOx)).
FD	Indicating the output will change along with volume (Flash with Dynamic).
Q	Indicating each pin will follow the variation of the QIO signal of Quick-IO.

3.13.3 NY6 / NY7

The followings are six possible values for output state definition:

Output Value	Description
X	Indicating the pin maintains the original output signal.
0	Indicating the pin is with low electric potential output.
1	Indicating the pin is with high electric potential output.
A	Indicating the pin is with Action output (This is function is onlyavailable for playing entire .vio files.)
FD	Indicating the output will change along with volume (Flash with Dynamic.)

3.13.4 NX1

The followings are six possible values for output state definition:

Output Value	Description
X	Indicating the pin maintains the original output signal.
0	Indicating the pin is with low electric potential output.
1	Indicating the pin is with high electric potential output.
FD	Indicating the output will change along with volume (Flash with Dynamic.)
Q	Indicating each pin will follow the variation of the QIO signal of Quick-IO.

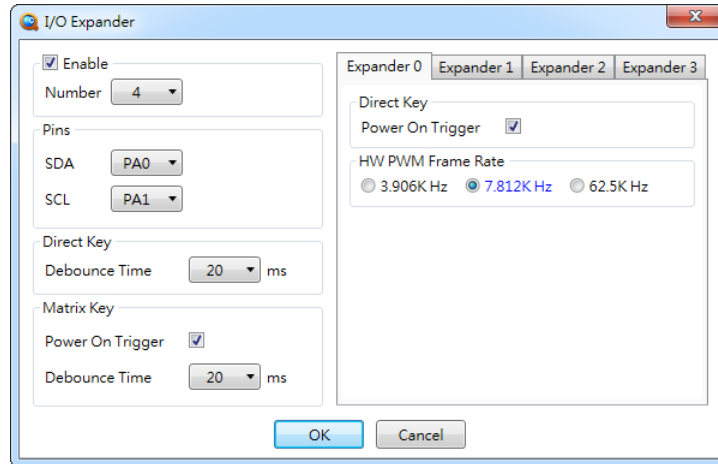
3.14 I/O Expander

3.14.1 I/O Expander

Q-Code supports I/O Expander function. When more IC pins are required, user use I/O Expander to fulfill their needs.

3.14.1.1 NY5+ / NX1

Q-Code supports up to 4 I/O Expander.



Number: I/O Expander number to use.

SDA pin: SDA pin used to connect to I/O Expander.

SCL pin: SCL pin used to connect to I/O Expander.

Debounce Time: Debounce time for I/O Expander Direct pins and Matrix keys. Supported time range is 1~1000ms.

Expander Power On Trigger: When user holds the button associated to I/O Expander and power on, system will execute the key path immediately after "PowerOn" path is done. User can set the option for Matrix keys and each I/O Expander Direct pins. (Default value is Enable)

Expander HW PWM Frame Rate: Frame rate of I/O Expander pin when used as hardware PWM output (7.812kHz by default).

Ex.

`IO_Expander_SDA = PA.0`

`IO_Expander_SCL = PA.1`

`IO_Expander0 = Enable`

`IO_Expander0_PowerOnTrigger = Enable`

`IO_Expander0_HW_PWM_FrameRate = 7812`

`IO_Expander_Direct_Debounce = 20ms`

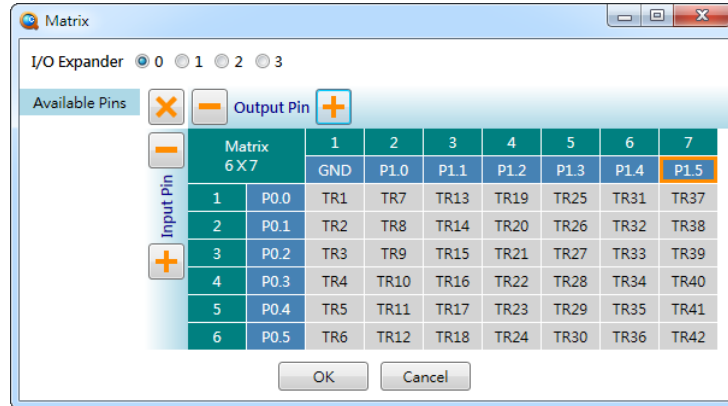
`IO_Expander_Matrix_Debounce = 20ms`

`IO_Expander_Matrix_PowerOnTrigger = Enable`

3.14.2 Matrix

Settings of I/O Expander Marix keys. Q-Code supports up to 1 I/O Expander with Matrix keys.

3.14.2.1 NY5+ / NX1



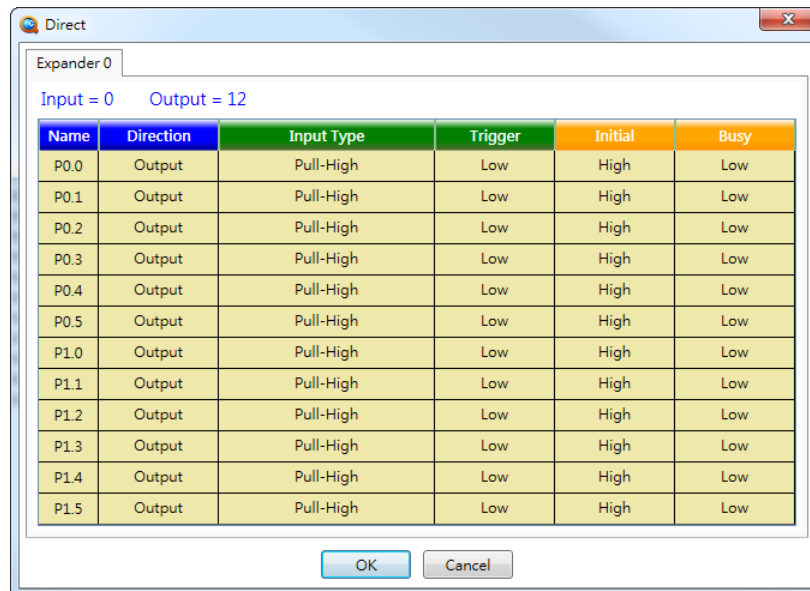
Ex.

IO_Expander0_Matrix = [EXP0_P0.0, EXP0_P0.1, EXP0_P0.2, EXP0_P0.3, EXP0_P0.4, EXP0_P0.5]*[GND, EXP0_P1.0, EXP0_P1.1, EXP0_P1.2, EXP0_P1.3, EXP0_P1.4, EXP0_P1.5]

3.14.3 Direct

Settings of every pin for each I/O Expander.

3.14.3.1 NY5+ / NX1



Direction: Set the output / input direction when initializing I/O.

Input Type: Set the pull-high or pull-low resistor of normal I/O Expander input pin.

- Pull-High: Input mode with Pull-High Resistor.

- Pull-Low: Input mode with Pull-Low Resistor.
- Floating: Input mode with Floating.

Trigger: Set the trigger level of input pin.

- Low: Low Trigger.
- High: High Trigger.

Initial: Set the initial output voltage.

- Low: Set the initial output type as Output Low.
- High: Set the initial output type as Output High.

Busy: Set the output of Action signals.

- High: Action as Active High (Drive Output).
- Low: Action as Active Low (Sink Output).

Ex.

EXP0_P0.0 = [Direction:Output, InputType:pull-high, Trigger:low, Initial:high, Busy:low]

EXP0_P0.1 = [Direction:Output, InputType:pull-high, Trigger:low, Initial:high, Busy:low]

EXP1_P0.0 = [Direction:Output, InputType:pull-high, Trigger:low, Initial:high, Busy:low]

EXP1_P0.1 = [Direction:Output, InputType:pull-high, Trigger:low, Initial:high, Busy:low]

3.14.4 Input State

After enabling I/O Expander function, user can define corresponding key configuration of each I/O Expander in **[Input State Exp0] / [Input State Exp1] / [Input State Exp2] / [Input State Exp3]** section.

The key configuration means the reactions while the corresponding input keys are pressed or released. The trigger number must be identical to the number of input pins, and the triggered events must be separated by TAB or Space.

3.14.4.1 NY5+ / NX1

The way to use Q-Code to create / edit I/O Expander Input States is the same as Input State. There can be up to 255 input states for each I/O Expander in NY5+, but NX1 has no limitation. The status of the input pin can be in one of the following forms:

Format	Trigger	Description
X	Ignore	No action.
Path1	Falling edge	Execute Path1 as input pin changed from high to low electric potential.
/Path2	Rising edge	Execute Path2 as input pin changed from low to high electric potential.
Path1 /Path2	Falling edge & Rising edge	Execute Path1 as input pin changed from high to low electric potential. Execute Path2 as input pin changed from low to high electric potential.

3.14.5 Output State

After enabling I/O Expander function, user can define output state for each I/O Expander in **[Output State Exp0]** / **[Output State Exp1]** / **[Output State Exp2]** / **[Output State Exp3]** section.

The output states defined in **[Output State Expn]** can be called by **[Path]** and **[Background]**. Each output state consists of one output state name, one “: ” and designated output value. The output value means the output signal for each output pin when the output state is called. The number of output value must be identical to the number of output pins, and the output values must be separated by TAB or space.

3.14.5.1 NY5+ / NX1

The followings are six possible values for output state definition:

Output Value	Description
X	Indicating the pin maintains the original output signal.
0	Indicating the pin is with low electric potential output.
1	Indicating the pin is with high electric potential output.

3.15 QIO

3.15.1 Configure

3.15.1.1 NY5+

PB HW PWM Frame Rate : The refresh rate of port B Hardware PWM signals.

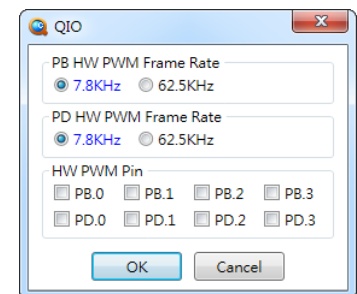
PD HW PWM Frame Rate : The refresh rate of port D Hardware PWM signals.

HW PWM Pin : Select the pins to output Hardware PWM signals.

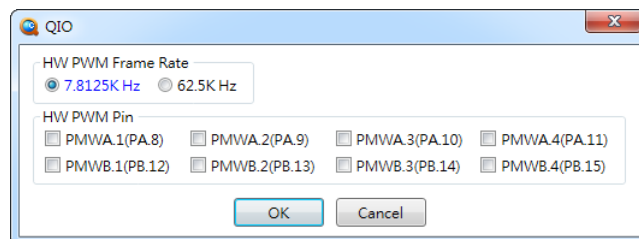
Ex. **HW_PWM_Pins** = PB.1, PD.1

HW_PWM_PB_FrameRate = 7.8KHz

HW_PWM_PD_FrameRate = 7.8KHz



3.15.1.2 NX1



HW PWM Frame Rate : The refresh rate of Hardware PWM signals.

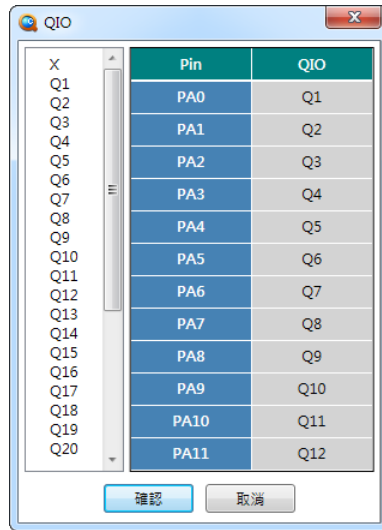
HW PWM Pin : Select the pins to output Hardware PWM signals.

Ex. HW_PWM_Pins = PA.9, PA.10

HW_PWM_Framerate = 7812

3.15.2 QIO Table

The mapping between QIO signal and pin can be configured in this section. Only 1 set of mapping is supported. Once mapping is configured, its applied to all QIO output.



Ex.

[QIO]

MQIO = [PA.0: Q1, PA.1: Q2, PA.2:Q3, PA.3: Q4]

3.15.2.1 NY4

If [QIO] is not configured, the mapping between QIO signal and pin is as following:

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
NY4A	PA.0	PA.1	PA.2	PA.3				
NY4B	PA.0	PA.1	PA.2	PA.3	PB.0	PB.1	PB.2	PB.3

3.15.2.2 NY5

If [QIO] is not configured, the mapping between QIO signal and pin is as following:

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
PA.0	PA.1	PA.2	PA.3	PB.0	PB.1	PB.2	PB.3
Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
PC.0	PC.1	PC.2	PC.3	PD.0	PD.1	PD.2	PD.3
Q17	Q18	Q19	Q20	Q21	Q22	Q23	Q24
PE.0	PE.1	PE.2	PE.3	PF.0	PF.1	PF.2	PF.3

3.15.2.3 NY5+

If **[QIO]** is not configured, the mapping between QIO signal and pin is as following:

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
PA.0	PA.1	PA.2	PA.3	PB.0	PB.1	PB.2	PB.3
Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
PC.0	PC.1	PC.2	PC.3	PD.0	PD.1	PD.2	PD.3

3.15.2.4 NX1

If **[QIO]** is not configured, the mapping between QIO signal and pin is as following:

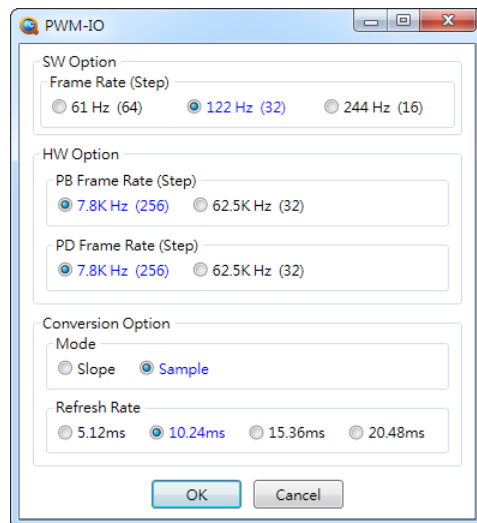
Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
PA.0	PA.1	PA.2	PA.3	PA.4	PA.5	PA.6	PA.7
Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
PA.8	PA.9	PA.10	PA.11	PA.12	PA.13	PA.14	PA.15
Q17	Q18	Q19	Q20	Q21	Q22	Q23	Q24
PB.6	PB.7	PB.8	PB.9	PB.10	PB.11	PB.12	PB.13
Q25	Q26	Q27	Q28	Q29	Q30	Q31	Q32
PB.14	PB.15	PC.0	PC.1	PC.2	PC.3	PC.4	PC.5

3.16 PWM-IO

PWM-IO is used to output action signal through PWM-IO.

3.16.1 Configure

3.16.1.1 NY5+



SW Frame Rate (Step): With higher frame rate, the output will be less flickering, but less output level and higher loading. With lower frame rate, the output will have more

output level and lower loading, but may have more flickering. This option only applies when using PWMOut / PWMOutS and output pin is not PB / PD.

HW PB Frame Rate (Step) : With higher Frame Rate, the output will be less flickering, but less output level. With lower Frame Rate, the output have more output level, but may have more flickering. These options only apply when using PlayPWM / PlayPWMS / PWMOut / PWMOutS and PB as output pins.

HW PD Frame Rate (Step) : With higher Frame Rate, the output will be less flickering, but less output level. With lower Frame Rate, the output have more output level, but may have more flickering. These options only apply when using PlayPWM / PlayPWMS / PWMOut / PWMOutS and PD as output pins.

Conversion Option : It is an option during data conversion, only applicable to PlayPWM / PlayPWMS.

Mode: Uses the slope method for data conversion, and uses the Sample method of aligning sampling points for data conversion. When using Sample mode, more ROM is consumed, and when using Slope mode, more RAM is consumed.

Refresh Rate: The sampling interval of the data. The smaller the interval, the larger the amount of data.

Ex.

SW_FrameRate = 122Hz

PB_FrameRate = 7.8KHz

PD_FrameRate = 7.8KHz

Conversion_Mode = Sample

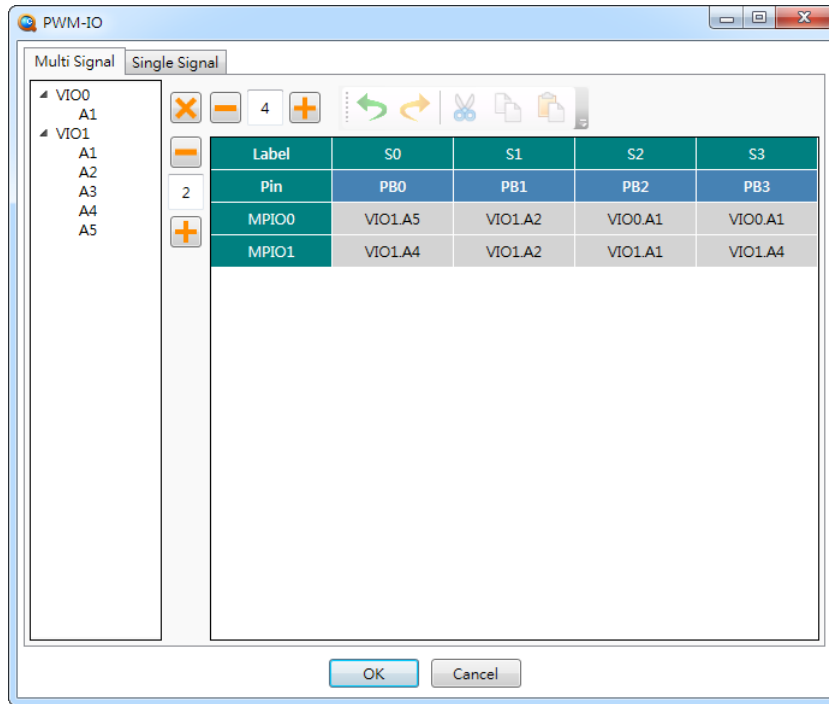
Refresh_Rate = 10.24ms

Conversion_Mode = Sample

3.16.2 Multi Signal

3.16.2.1 NY5+

User can set the output configuration of multi-signal. It will output to the corresponding pin at the same time during playback. At least 2 pins are required, and the number of pins can be adjusted by +/- . When playing Multi Signal through PlayPWM command, the signal will be output from the pin according to the setting.



Ex.

[PWM-IO]

Multi_Pins = [PB.0, PB.1, PB.2, PB.3]

MPIO0 = [VIO0.A1, VIO0.A1, X, X]

3.16.3 Single Signal

3.16.3.1 NY5+

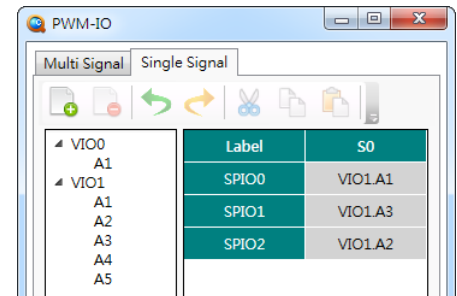
Output config of single signal can be set in Single Signal dialog. Single Signal does not associate with pin, Px.n parameter is needed to use with PlayPWM.

Ex.

[PWM-IO]

SPIO0 = [VIO0.A1]

SPIO1 = [VIO1.A1]



3.17 Action

Used to configure action output. Configs defined can be used by PlayA / PlayAS.

3.17.1 Configure

3.17.1.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T

Frame Rate: With higher frame rate, the less flickering, but available output level is less and higher system loading. With lower frame rate, the more output level, but more flickering.

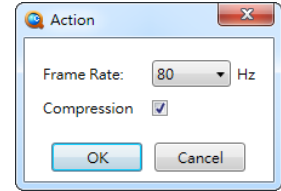
Compression: Action data compression, its Enable by default.

Ex.

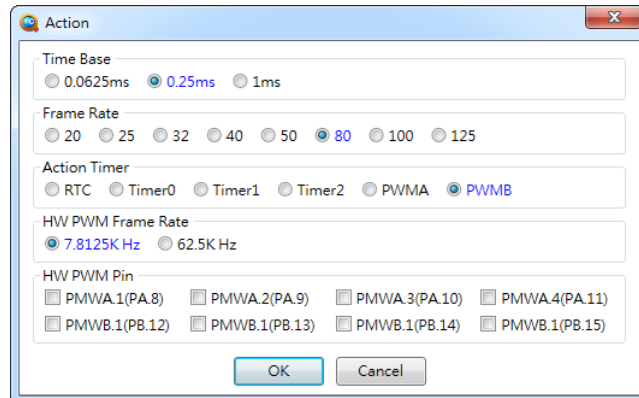
[Action]

Compression = Enable

FrameRate = 80



3.17.1.2 NX1 OTP / NX1 EF



Time Base : The least unit of measurement of time, its unit is ms.

Frame Rate : Frame Rate is the number of signals that are displayed per second, the unit is Hz.

Action Timer : Set the Timer used by Action.

HW PWM Frame Rate : HW PWM signal are displayed per second.

HW PWM Pin : Set the HW PWM pins.

Ex.

[Action]

FrameRate = 80

TimeBase = 0.25ms

Timer = PWMB

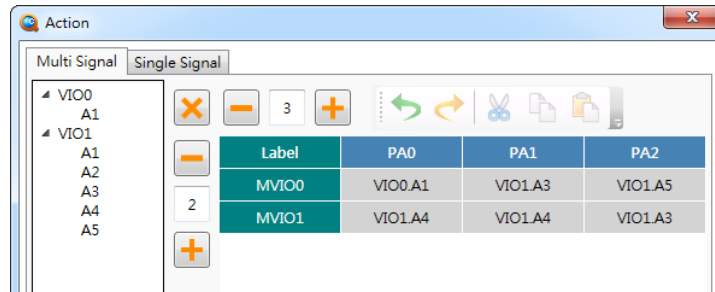
HW_PWM_FrameRate = 7812

HW_PWM_Pins = PA.8, PA.9

3.17.2 Multi Signal

3.17.2.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T

Multiple signal config. When multiple signal config is playing, multiple signal will be output to specified pin simultaneously. Multiple signal config must contain at least 2 pins, can be adjusted by +/- on the top of dialog.



Ex.

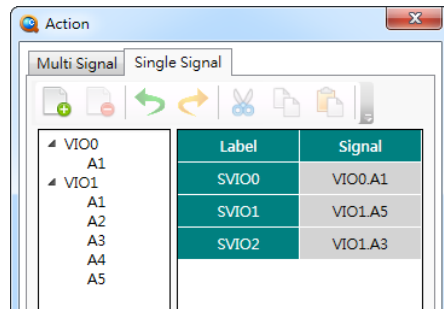
[Action]

Multi_Pins = [PB.0, PB.1, PB.2, PB.3]

MPIO0 = [VIO0.A1, VIO0.A1, X, X]

3.17.3 Single Signal

3.17.3.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T



Single Signal config does associate to output pin. When used with PlayA, output pin is required.

Ex.

[Action]

SVIO0 = [VIO0.A1]

SVIO1 = [VIO1.A1]

3.18 VR

VR stands for Voice Recognition. User can include the .cvr file generated by *Cyberon CSpotter Modeling Tool*, and set the VR state to specify the executing path of voice command.

The .cvr file can be stored on IC ROM or external SPI Flash, which is specified by VR File option. User can only choose either Rom or SPI Flash to store file.

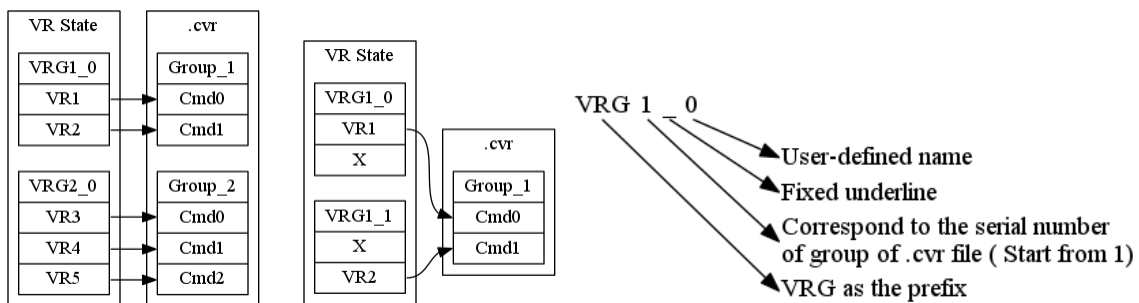
In addition to creating voice commands offline, The Voice Tag provides our users to train voice commands online during IC operation. The Voice Password provides online voice training commands and records the voiceprint features, the trained voice command needs similar voice to complete recognition.

There are two ways to set voice command status and its corresponding path: VR State and VRGC state

(VR Group Combo state). *Cyberon CSpotter Modeling Tool* project can have multiple groups. VR State specifies to enable one of these groups. There can have multiple VR states, the path corresponded from every voice command, and only one VR Group can be enabled at the same time. Each VR State can be assigned to different VR Group. VRGC state specifies to enable multiple VR groups (up to three groups) at the same time, for example, if a VRGC state specifies to enable two VR groups, all voice commands in these two groups will be identified, and each command in VRGC has a corresponding path.

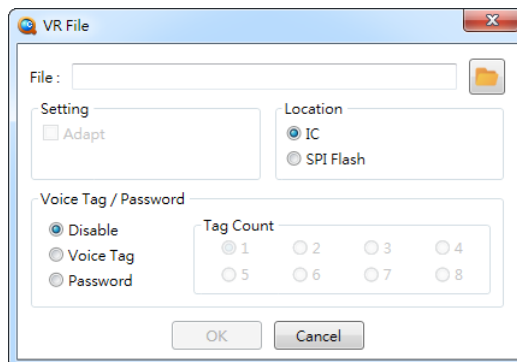
The name of VR State has to correspond to the command group of the .cvr file, for example, the 1 from VRG1_0 means the first voice command group, the 2 of VRG2_0 means the second group. The path name is default as VR1, VR2...sequentially. If user wants to close some voice commands, just put an X on the corresponding path of VR State which means no action. The name of VR State has to start with VRG plus one number to represent the serial number of command group, and the following “_” underline is to separate the user-defined name.

When executing the VoiceTag / Password mode, user can use tge voice command group VTG, the name of corresponding VR State is VTG_0, VTG_1, etc. The VTG can also be applied to the VRGC state.



3.18.1 VR File

3.18.1.1 NX1



File: The filepath of .cvr file.

Location: If the VR file is placed in external SPI Flash, the content of the .spiprj file will be overwritten. (If the .spiprj file does not exist, user needs to choose a .spiprj file). If the VR file is placed in IC ROM, there will have commands automatically generated in qc as

below.

Adapt: Tick the Adapt function or not.

Voice Tag / Password: Select to use Voice Tag or Password.

Tag Count: Set the tag count of voice command which corresponds to the used Voice Tag / Password. The Voice Tag can set up to 8 tags, Password up to 1 tag.

Ex.

[VR]

VR File = C:\test.cvr

VR_Application = VoiceTag

VT_TagCount = 6

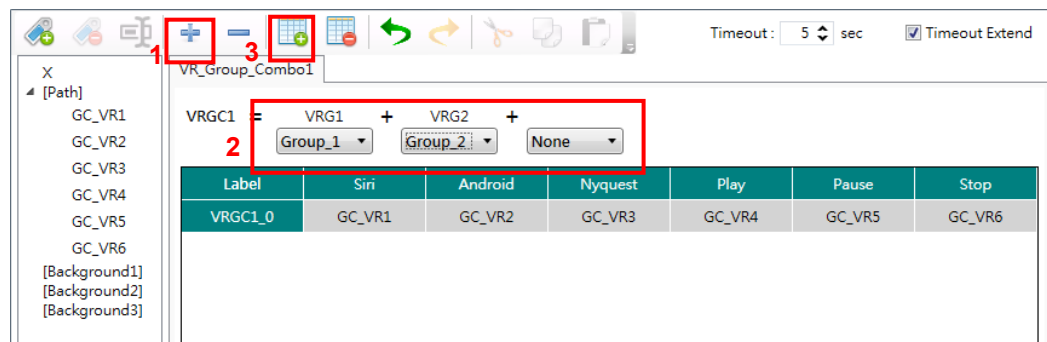
3.18.2 VR Combo

3.18.2.1 NX1

VRGC state is a combination of multiple VR groups. User must select the VR groups in VRGC, and specify a path corresponding to each voice command of all the groups. No matter the voice command belongs to which group, once identified, the corresponding path will be executed immediately.

How to edit VRGC State

1. Add a new state, Q-Code_NX1 will generate a new VR Group Combo named VRGC1 automatically.
 2. Choose the VR groups (up to three groups) to be recognized at the same time.
 3. Then add a new VRGC state, Q-Code_NX1 will generate a state named VRG1_0 automatically.
- In the following example, there are three voice commands in VRG1, and their paths are G1_VR1, G1_VR2 and G1_VR3 separately. The VRG2 has three voice commands and their paths are G2_VR1, G2_VR2 and G3_VR3 separately.



Note: The .cvr file must have more than two groups to use with VRGC State.

VRGC Timeout is the timeout handling when it fails to get response from subsequent VR group. the following illustrates for 2 VR Groups Combo and 3 VR Groups Combo situations.

2 Groups Combo: When the command of the 1st VR Group is recognized successfully, it will start counting for timeout until the VRGC_Timeout_CLR command is called. If the counting time is up, and VRGC_Timeout_CLR is still not called, jump to timeout path.

3 Groups Combo: When the command of the 1st VR Group is recognized successfully, it will start counting for timeout. If the command of the 2nd VR Group is recognized successfully, it will restart counting until VRGC_Timeout_CLR is called. If the command of the 1st VR Group fails, but the 2nd VR Group is recognized, the counting is ended.

Timeout Extend is to extend the timeout, and only extend one time. If the sound from mic exceeds the VAD threshold before VRGC timeout is reached, it will be automatically extended once.

Ex.

VRGC1 = VRG1 + VRG2 ; **VRGC1 combines VRG1 and VRG2.**

VRGC1_0: G1VR1 G1VR2 G2VR1 G2VR2

; The path of two voice commands in VRG1.; The path of two voice commands in VRG2.

VRGC state provides the timeout function. In the above example, if the voice command in VRG1 (the 1st VR Group) is recognized successfully but the voice command in VRG2 (the 2nd VR Group) is not recognized in limited time, VRGC_Timeout will be executed automatically. If the command of the 2nd VR Group is recognized successfully first, the Timeout condition won't be established. The VRGC_Timeout path is established only when the command of the 1st VR Group is recognized successfully first. The unit of VRGC_Timeout is second (s), the longest length of time can be set to 60 seconds. When Timeout Extend is enabled, and the commands of VRG1 are recognized successfully, the duration of timeout will be extended automatically if the volume exceed the VAD threshold.

Ex. VRGC_Timeout = 5s ; **Specify the timeout to 5 seconds.**

Ex. VR_Timeout_Extend = Enable ; **Enable the automatic extension of timeout function.**

The VRGC state can also be used to set the recognized order of VR groups. The actual voice command can be triggered only after a specific voice command (or trigger word) is recognized. For example: "Hi Siri, what time is it?". "Hi Siri" is the voice command of VRG1, after it is recognized, the voice command of VRG2, "what time is it ?", can only be accepted. When writing program, user can use a variable flag to log whether VRG1 has been recognized. Please refer to the following example.

Ex.

[VR]

VRGC_Timeout = 5s ; **Specify the timeout to 5 seconds.**

VRGC1 = VRG1 + VRG2 ; **VRGC1 combines VRG1 and VRG2.**

; VRG1 is the group of preceding voice commands

; VRG2 is the group of succeeding voice commands

VRGC2 = VRG1 + VRG3 ; **VRGC2** combines **VRG1** and **VRG3**.

VRGC1_0: G1_VR1 G1_VR2 ; Define the paths of all voice commands in the state of **VRG1**.
G2_VR1 G2_VR2 ; The paths corresponding to group 2.

[Variable]

VAR8: flag

[Path]

PowerOn: VRGC1_0 ; Enable **VRGC1_0** state.

G1_VR1: flag = 1 ; The preceding command appears and set flag.

G1_VR2: flag = 1 ; The preceding command appears and set flag.

G2_VR1: flag = 1 ? G2_VR1_True ; If there is a preceding command, the path will be executed.

G2_VR2: flag = 1 ? G2_VR2_True ; If there is a preceding command, the path will be executed.

VRGC_Timeout: flag = 0

; If there is a preceding command but no succeeding command, the **VRGC_Timeout** path will be executed for user to reset the state when time's up.

G2_VR1_True: flag = 0, VRGC_Timeout_CLR, PlayV(ch0, \$V1)

; Play voice when the voice command is recognized.

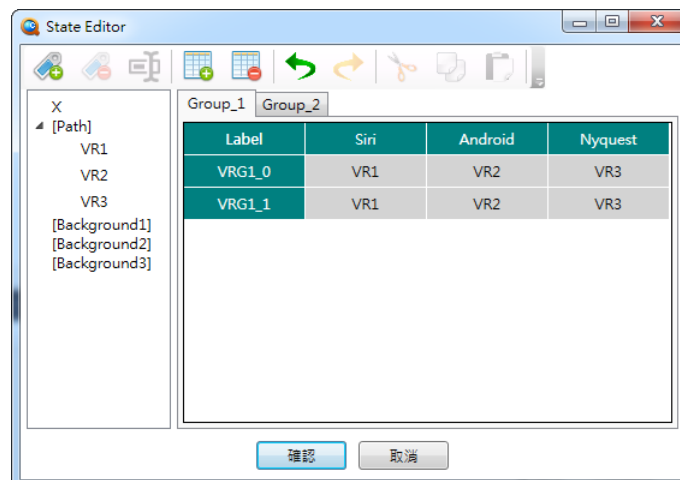
G2_VR2_True: flag = 0, VRGC_Timeout_CLR, PlayV(ch0, \$V1)

; Play voice when the voice command is recognized.

3.18.3 VR State

3.18.3.1 NX1 OTP / NX1 EF

Add a new state to VRG1, Q-Code_NX1 will generate a state named VRG1_0 automatically. As the following example, the three voice commands in VRG1 correspond to the VR1, VR2 and VR3 path separately.



Ex.

[VR]

VRG1_0: VR1 VR2 VR3

VRG2_0: VR1 VR2 VR3

[Path]

PowerOn: VRG1_0, VR_On

VR1: PlayV(ch0, \$V0)

VR2: PlayV(ch0, \$V1)

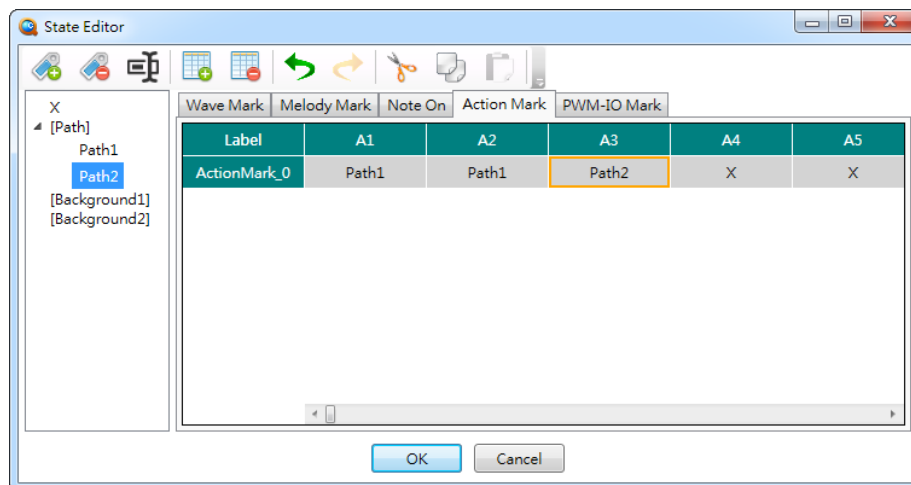
VR3: PlayV(ch0, \$V2)

3.19 Action Mark

3.19.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1

User can insert marks anywhere in the action file to manage the background actions while using Q-Visio for editing files. When Q-Code accesses the marks, it will execute the corresponding path of Action Mark state automatically. User can define the Action Mark states in [Action Mark] section.

There can be as many as 255 such marks from A1 to A255. After Action Marks are defined, users just have to insert corresponding Action Mark path and .vio file into Q-Code to activate Action Mark feature. User can define the setting of corresponding states up to maximum 255.



Note:

1. Switching Action Mark state is similar to switching Input State.
2. The mark numbers must be A1~A255.

Ex. Use Q-Visio to insert A1 ~ A4 marks.

[Action Mark]

; A1 A2 A3 A4
ActionMark1: AM1 AM2 AM3 AM4

[Path]

PowerOn: ActionMark1, PlayA(PA.0, Ch1, \$A0)

[Background1]

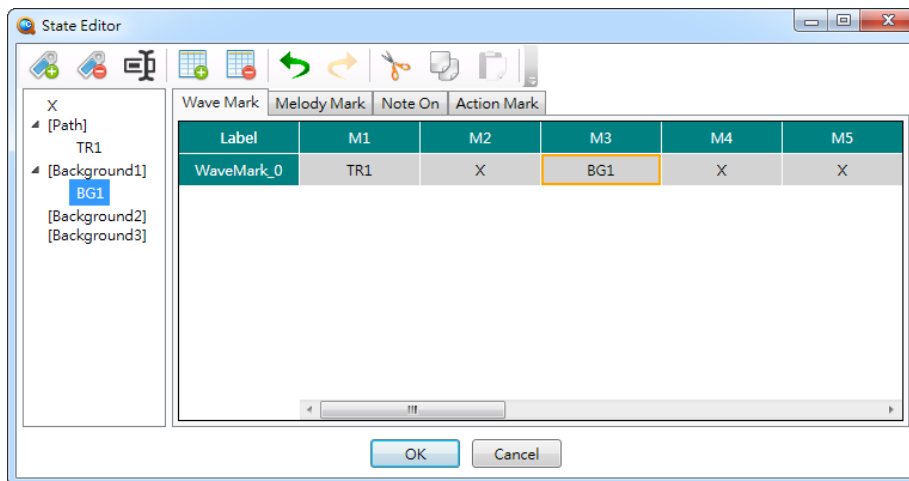
AM1: PB=0x1 ; System will execute AM1 when action mark 1 is read.
AM2: PB=0x2 ; System will execute AM2 when action mark 2 is read.
AM3: PB=0x3 ; System will execute AM3 when action mark 3 is read.
AM4: PB=0x0 ; System will execute AM4 when action mark 4 is read.

3.20 Wave Mark

3.20.1 NY4 / NY5 / NY5+ / NX1

This feature is for user to add the background path action according to the BGP mark in *Quick-IO* file during the playback of voice file. User can insert Wave Mark anywhere in the voice file to manage the background actions while using *Quick-IO* for editing files. User can define the Wave Mark status in Wave Mark section.

The mark number is, for example, M1; there can be as many as 255 such numbers from M1 to M255. After Wave Marks are defined, users just insert corresponding Mark using *Quick-IO* to activate Wave Mark feature. User can define the setting of corresponding states up to maximum 255.



Note:

1. To switch Wave Mark state is similar to Input State.
2. The mark numbers must be M1 ~ M255.

Ex. Use *Quick-IO* to insert M1 ~ M4 marks.

[Wave Mark]

; **M1** **M2** **M3** **M4**
WaveMark_0: WM1 WM2 WM3 WM4

[Path]

PowerOn: WaveMark_0, PlayV(\$V0)

[Background1]

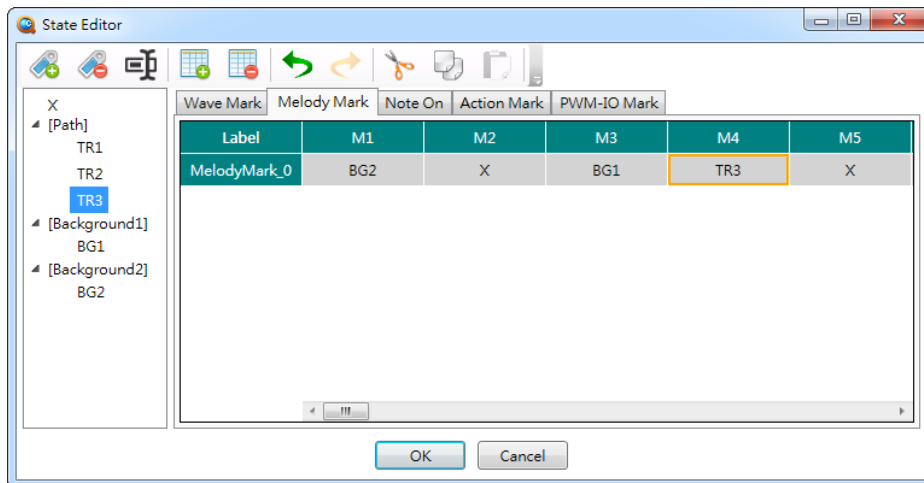
WM1: PB=0x1 ; System will execute WM1 when wave mark is read and the number is 1.
WM2: PB=0x2 ; System will execute WM2 when wave mark is read and the number is 2.
WM3: PB=0x3 ; System will execute WM3 when wave mark is read and the number is 3.
WM4: PB=0x0 ; System will execute WM4 when wave mark is read and the number is 4.

3.21 Melody Mark

3.21.1 NY5 / NY5+ / NY6 / NY7 / NX1

User can insert Background control code to any Note to control the output pin state or execute other operations while using *Cakewalk* to edit MIDI music files. During the playback of Melody file, the motor rotation or LED flashing actions will be executed according to the corresponding mark in the melody file.

The mark number of background path, for example, M1, is corresponding to the control code inserted by *Cakewalk* or *Q-MIDI*; there can be as many as 255 numbers (from M1 to M255) on the Melody Mark page with up to 15 background paths attached to each mark number. After Melody Mark are defined, user just has to insert corresponding background control codes into MIDI files to activate Melody Mark paths. User can define the setting of corresponding states up to maximum 255 sets.



Note:

1. To switch Melody Mark state is similar to Input State.
2. The mark numbers must be M1 ~ M255.

Ex. Insert M1 ~ M4 in *CakeWalk* / *Q-MIDI*.

[Melody Mark]

; M1 M2 M3 M4

NewMelodyMark1: MM1 MM2 MM3 MM4

; System will execute MM1 when melody mark is read and the number is 1.

; System will execute MM2 when melody mark is read and the number is 2.

; System will execute MM3 when melody mark is read and the number is 3.

; System will execute MM4 when melody mark is read and the number is 4.

[Path]

PowerOn: NewMelodyMark1, PlayM(\$M0)

[Background1]

MM1: PB=0x1

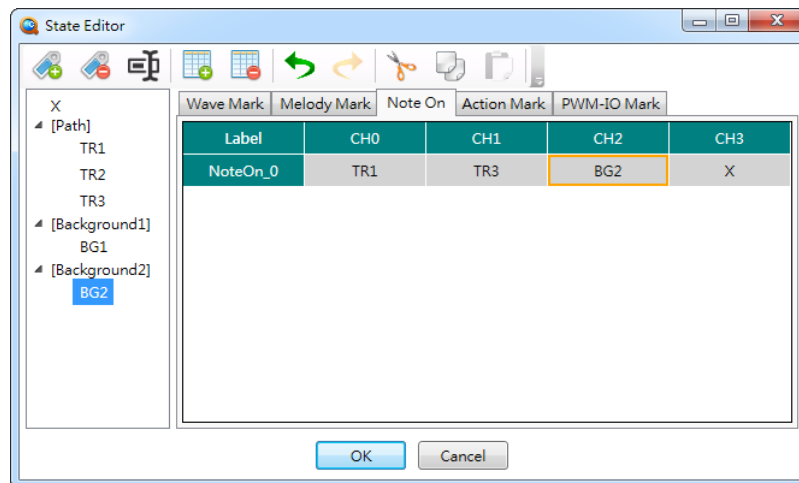
MM2: PB=0x2

MM3: PB=0x3

3.22 Note On

3.22.1 NY5 / NY5+ / NY6 / NY7 / NX1

When playing Melody files, users do not need to insert code for each MIDI channel with *Q-MIDI* to save ROM Size. [Note On] section can be used to directly execute corresponding steps after reading Notes in each MIDI channel. User can define the setting of corresponding states up to maximum 255.



Note:

1. To switch Note On states is similar to Input State.
2. NY5 / NY5+: Ch0 ~ 3, NY6: Ch1 ~ 6, NY7 / NX1: Ch1 ~ 16.
3. Direct execution of play instruction in [Note On] section might interrupt current playback.
4. There will be no action for the rest note.

Ex.

[Note On]

```

;          CH1          CH2          CH3          CH4
NoteOn1:  BG1          X            X            X
    
```


NoteOn2: X BG2 X X

[Path]

TR1: PlayM(\$M0) ; Play melody.

TR2: SWITCH(R0)=[NO1, NO2], R0=0, TR2 ; Choose Note On state according to the value of R0.

NO1: NoteOn1, R0=1 ; Set Note On state to NoteOn1.

NO2: NoteOn2, R0=0 ; Set Note On state to NoteOn2.

[Background1]

BG1:PB=[X X X 1], DELAY(0.08), PB=[X X X 0]

[Background2]

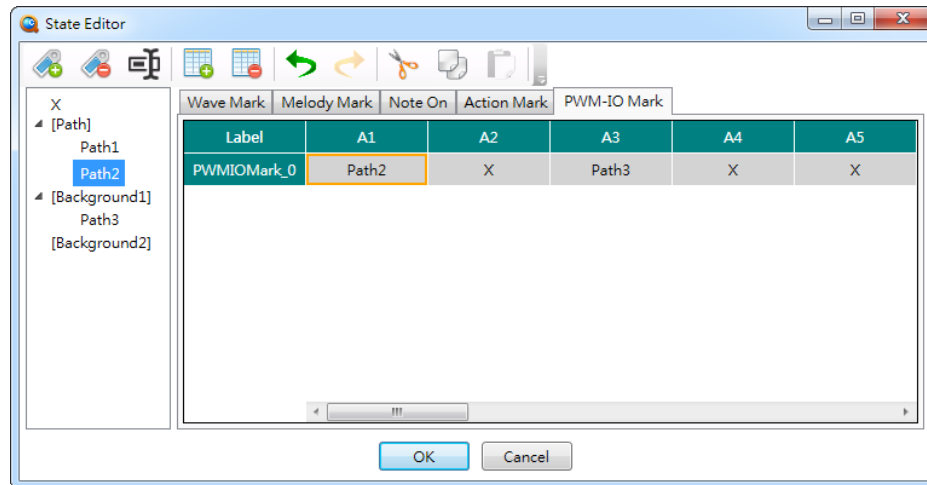
BG2:PB=[X X 1 X], DELAY(0.08), PB=[X X 0 X]

3.23 PWM-IO Mark

3.23.1 NY5+

User can insert marks anywhere in the action file to manage the background actions while using Q-Visio for editing files. When Q-Code uses PlayPWM and reaches the marks, it will execute the corresponding path of PWM-IO Mark state automatically. User can define the PWM-IO Mark states in PWM-IO Mark section.

There can be as many as 255 such marks from A1 to A255. User can define the setting of corresponding states up to maximum 255.



Note:

1. Switching PWM-IO Mark state is similar to switching Input State.
2. The mark numbers must be A1~A255.

Ex. Use Q-Visio to insert A1 ~ A4 marks.

[PWM-IO]

SPIO0 = [VIO0.A1]

[PWM-IO Mark]

```

;           A1      A2      A3      A4
PWMIOMark_0: PM1      PM2      PM3      PM4

```

[Path]

PowerOn: PWMIOMark_0, PlayPWM(PA.0, Ch1, \$SPIO0)

[Background1]

```

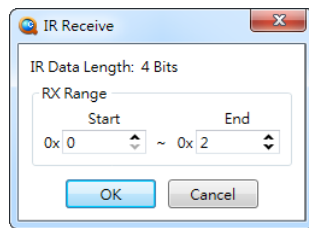
PM1: PB=0x1           ; System will execute PM1 when PWM-IO mark 1 is read.
PM2: PB=0x2           ; System will execute PM2 when PWM-IO mark 2 is read.
PM3: PB=0x3           ; System will execute PM3 when PWM-IO mark 3 is read.
PM4: PB=0x0           ; System will execute PM4 when PWM-IO mark 4 is read.

```

3.24 IR Receive

3.24.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NX1

First, select required IR Mode (IR transmission or reception features, code length and carrier frequency in the Option section, and then compose "IR Receive" command in IR Receive section.



User can only select IR data length in [Option] section, and the window will show the selected data length.

If 4-bit coding is selected for IR data, the range of RX code is "0x0~0xF".

If 8-bit coding is selected for IR data, the range of RX code is "0x00~0xFF".

If 12-bit coding is selected for IR data, the range of RX code is "0x000~0xFFF".

If 16-bit coding is selected for IR data, the range of RX code is "0x0000~0xFFFF".

Ex.

[IR Receive]

Code [0x0]: PlayV(CH0, \$V0)

Code [0x1]: PlayV(CH1, \$V1)

Code [0x2]: PlayV(CH0, \$V2)

Code [0x3]: PlayV(CH1, \$V3)

Note: For IR transmission, select "Enable IR TX mode" in "Option" menu and send "TX=data" command in "Path" section.

3.25 Symbol

3.25.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1

Symbol section is for defining “Constant” or “Variable” used in conditional statement and system command. Please try to use “Symbol” in the program to enhance the readability and flexibility of this program.

Ex.

[Symbol]

On = 5

Off = 15

“Variable” can be set stand for Ri. User can customize names for access the original names of user RAMs equivalently (Ri / Xi).

Ex. Counter is 8-bit variable, CounterH is the high nibble of Counter and CounterL is the low nibble of Counter.

[Symbol]

Counter = X0

CounterL = R0

CounterH = R1

3.26 Variable

3.26.1 NX1

Variable section is used to define the Variable name and data size.

The available data types are as below table:

Type	Size	Range
VAR8	8-bit (1 byte)	0 ~ 0xFF
VAR16	16-bit (2 byte)	0 ~ 0xFFFF
VAR32	32-bit (4 byte)	0 ~ 0xFFFFFFFF

Variable name starts with A-Z, and the following characters can be A-Z, 0-9 and “_”. The variable name must not conflict with reserved word. The variables used in any sections must be declared in [Variable] section, and also the data size must be defined.

Ex. Buf0, Buf1 are 8-bit variables, Buf2 is 16-bit variable.

[Variable]

VAR8: Buf0, Buf1

VAR16: Buf2

3.27 Storage Variable

3.27.1 NX1

The variables defined in this section can be stored in SPI Flash or Embedded Flash using the Storage command. After the IC is powered off and on again, the system will automatically reload and can be used to save the system state.

Note: If Common ROM Code is enabled, the program of second and subsequent versions cannot change storage variables (amount, initial value).

Ex.

[Option]

Storage = Auto ;The system will automatically save variables to SPI0 before entering sleep.

Storage_Location = SPI0

[Storage Variable]

Var8: var0

[Path]

PowerOn: SPIPlay(ch0, var0) ; Automatically play the next song after power on.

TR1: SPIPlay(ch0, var0++)

3.28 Table

3.28.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1

Table can be used for data reference which is defined in **[Table]** section by two-dimensional matrix. For NY4 / NY5 / NY5+ / NY6 / NY9T, each number can be any integer from 0x000 to 0x3FF. For NY7, each number can be any integer from 0x000 to 0xFFF. For NX1, each number can be any integer from 0 to 0xFFFFFFFF (32-bit). However, for 4-bit command, the matrix cannot exceed the 16x16 array, and for 8-bit command the arrangement cannot exceed the 256x256 array. Decimal and hexadecimal values can both be accepted in Q-Code.

Ex.

[Table]

Trans:

```
{
    [0x1, 0x3, 0x153, 0x17, 0x9],
    [0x4, 0x5, 0x3F6, 0x8, 0x10],
    [0x0F, 0x1C, 0x2, 0x3, 0xF4]
}
```

[Path]

P1: R0=2, R1=1, TableL(Trans,R0,R1,R2) ; **R2 = 0x6.**

P2: R0=2, R1=1, TableM(Trans,R0,R1,R2) ; R2 = 0xF.

P3: R0=2, R1=1, TableH(Trans,R0,R1,R2) ; R2 = 0x3.

Table command: Table { Return_Type } (Table_Name, Col_Index, Row_Index, Return_Reg)

ReturnType: {L, M, H} ; L: obtain data[3:0], M: obtain data[7:4], H: obtain data[11:8].

Table_Name: {name of Table}

Col_Index: {Constant or RAM} ; Constant: 0 ~ 15, RAM: User RAM.

Row_Index: {Constant or RAM} ; Constant: 0 ~ 15, RAM: User RAM.

Return_Reg: {RAM}

Note:

1. The table matrix must be filled, or Q-Code will automatically fill empty address with 0.
2. "0" will be read if address exceeds the size of array.

3.29 ASM

3.29.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T

Sometimes during the development of certain project, the accuracy and efficiency of timing control of Q-Code might not meet user's demand because it is a simple high level language. In this case, instead of re-composing the whole program with assembly language, user can consider directly composing certain subroutines in ASM section to meet the demand. The subroutines composed with assembly language in ASM section can be directly called in Path, Background1 and Background2 sections.

Due to the fact that these assembly subroutines are using different syntax from Q-Code, it is required to program the name of the assembly subroutine, "{", "}" and each instruction of assembly language at an independent line.

Syntax:

[ASM]

Name

```
{
    ...
    assembly code
    ...
}
```

Ex.

[ASM]

ABC

```
{
    mvla    5
```

```

.....
mvam pa
}

[Path]
P1: ABC ; Call Assembly.

```

Note: Because Q-Code dynamically allocates User RAM, when inserting ASM in Q-Code, users have to check which Page is RAM defined to. Users need to switch RAM page in Assembly code.

3.30 C-Code

3.30.1 NX1

C-Code section allows users to write any legal C code. The code written in this section will directly copy to C Compiler for compiling. Q-Code only parses the function definition. The functions defined in C-Code section can be called directly by code in [Path].

C language is case sensitivity, but Q-Code is not case sensitivity. If there are two functions with the same name, but cases are different, Q-Code will always find the first C function in the [Path] section. It is recommended that do not define the same function names with different cases in C-Code Section. C-Code section can read / access the defined variables of 5.4.2 section directly. When reading / accessing variables, please use lowercase letters.

Type	Size	Range
VAR8	8-bit (1-byte)	0 ~ 0xFF
VAR16	16-bit (2-byte)	0 ~ 0xFFFF
VAR32	32-bit (4-byte)	0 ~ 0xFFFFFFFF

The name of available variable starts with A-Z letter, and the second character can start with A-Z, 0-9, and the bottom line, and cannot conflict with reserved words. The variables used in NX1 are declared in this paragraph and their data size is defined.

Ex. Set Buf0、Buf1 as 8-bit variable, Buf2 as 16-bit variable.

[Variable]

VAR8: Buf0, Buf1

VAR16: Buf2

Please refer to 5.4.2 C-Code for examples.

3.31 Macro

3.31.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1

For maintaining a neat program, user could write codes in the Macro section, rather than copy the same program to everywhere in need. Programs in the Macro section could be applied in Path, Background1, Background2 and Background3 sections.

Each macro consists of a path name, a “:”, and several steps.

Note: Commands in [Macro] section will be expanded when it is applied each time, and it does take ROM size.

As the following example, programs in the Macro section can be called in Path, Background1 or Background2 sections, and it can be called at different sections simultaneously. Besides, it can be called by the “Switch” command.

Ex.

[Macro]

MACRO1: PlayV(Ch1, \$V2), DELAY(1)

MACRO2: PlayV(Ch1, \$V3), DELAY(1)

MACRO3: PlayV(Ch1, \$V4), DELAY(1)

MACRO4: PB=1

[Path]

PowerOn: KEY1, R0=0

TR1: SWITCH(R0)=[SW1, SW2, SW3, SW4] ; Value of RAM R0 to decide jump “Label”

SW1: R0=1, MACRO1 ; Call Macro1.

SW2: R0=2, BG1 ; Call BG1.

SW3: R0=3, BG2 ; Call BG2.

SW4: R0=0, PlayV(Ch1, \$V1) ; Execute PlayV.

TR2: VOICE? MACRO4, PB=0 ; If Voice is playing, execute Macro4 and the next command.

[Background1]

BG1: MACRO2 ; Call Marco2.

[Background2]

BG2: MACRO3 ; Call Macro3.

3.32 Sentence

For maintaining a neat program, user could write codes in the Sentence section, rather than copy the same program to everywhere in need. Programs in the Sentence section could be applied in Path, Background1, Background2 and Background3 sections.

3.32.1 NY4 / NY5

Each sentence consists of a path name, a “: ”, and several steps.

Ex.

[Sentence]

S1: PlayV(CH2,\$V0), DELAY(0.1), PlayV(CH2,\$V1), DELAY(0.2), PlayV(CH2,\$V2)

S2: PlayV(CH2,\$V3), PlayV(CH2,\$V4), PlayV(CH2,\$V5)

S3: PlayV(CH2,\$V6), DELAY(0.1), PlayV(CH2,\$V7)

[Path]

PowerOn: KEY1

TR1: PlayS(\$S1) ; Play Sentence “S1”.

[Background1]

BG1: PlayS(\$S1), PlayS(\$S3) ; Play Sentence “S1” and “S3”.

[Background2]

BG3: PlayS(\$S1), PlayS(\$S2) ; Play Sentence “S1” and “S2”.

3.32.2 NY5+ / NY6 / NY7 / NX1

Each sentence consists of a path name, a “: ”, and several steps.

Note: The function of Sentence is aimed for combining resource files (Melody, Speech, Action,etc.) as one single sentence. Therefore, the Sentence section only allows the Play and Delay commands, other commands cannot be used in the Sentence section.

Ex.

[Sentence]

S1: PlayV(CH2,\$V0), DELAY(0.1), PlayV(CH2,\$V1), DELAY(0.2), PlayV(CH2,\$V2)

S2: PlayV(CH2,\$V3), PlayV(CH2,\$V4), PlayV(CH2,\$V5)

S3: PlayV(CH2,\$V6), DELAY(0.1), PlayV(CH2,\$V7)

[Path]

PowerOn: KEY1

TR1: PlayS(\$S1) ; Play Sentence “S1”.

[Background1]

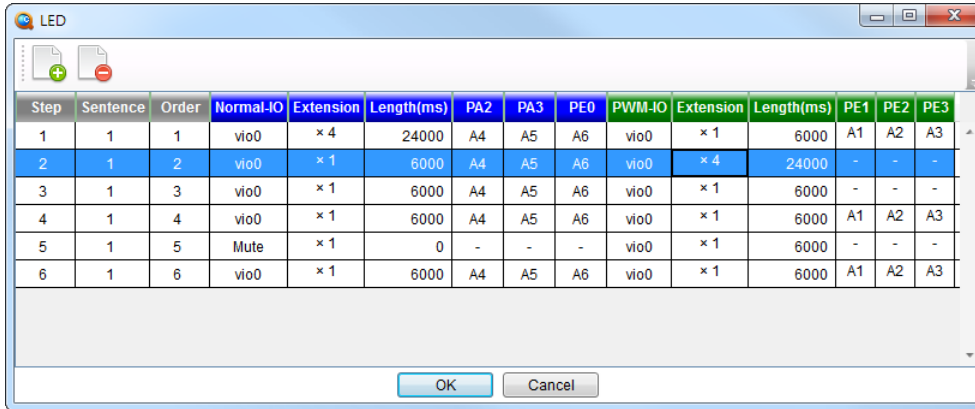
BG1: PlayS(\$S1), PlayS(\$S3) ; Play Sentence “S1” and “S3”.

[Background2]

BG3: PlayS(\$S1), PlayS(\$S2) ; Play Sentence “S1” and “S2”.

3.32.3 NY9T

Q-Code provides LED Sentence editor to help editing Action / PWM output sentence.



ContextMenu

Funciton	Description	Hot Key
New Step	Add a new step.	+
Insert Step	Inset a new step.	Insert
Delete Step	Delete Selected Steps.	—, Delete
Delete Sentence	Delete Selected Sentences.	-
Remove All	Remove all Sentences.	-

Ex.

[Sentence]

L1: PlayPWMS(@000,\$VIO0,1), PA=[A5 A4 0 0], PE=[0 0 0 A6], PlayA(Ch1,\$VIO0,4), WaitPN

L2: PlayPWMS(@000,\$VIO0,4), PA=[A5 A4 0 0], PE=[0 0 0 A6], PlayA(Ch1,\$VIO0,1), WaitPN

L3: PA=[A5 A4 0 0], PE=[0 0 0 A6], PlayA(Ch1,\$VIO0,1)

L4: PlayPWMS(@111,\$VIO0,1), PA=[A5 A4 0 0], PE=[0 0 0 A6], PlayA(Ch1,\$VIO0,1), WaitPN,
Delay(300ms), PlayPWMS(@111,\$VIO0,1), PA=[A5 A4 0 0], PE=[0 0 0 A6],
PlayA(Ch1,\$VIO0,1), WaitPN

[Path]

PowerOn: KEY1, PlayS(\$L1), L4

; Play Sentence "L1" and "L4"

TR1: BG1

; Play BG1

TR2: BG2

; Play BG2

[Background1]

BG1: PlayS(\$L2), PlayS(\$L3), L4

; Play Sentence "L2", Sentence "L3" and "L4"

[Background2]

BG2: PlayS(\$L3), PlayS(\$L2), L4

; Play Sentence "L3", Sentence "L2" and "L4"

3.33 Subroutine

3.33.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1

For maintaining a neat program, user can consider directly composing certain Q-Code subroutines in Subroutine section to meet this demand instead of composing same programs portion in different paragraphs. The Q-Code subroutines composed in Subroutine section can be directly used in Path, Background1, Background2 and Background3 sections.

Each subroutine consists of one path name, a “.” and one or more steps.

Note: For NY4 / NY5, the subroutine called in another subroutine will be expanded in the caller subroutine.

Ex.

[Subroutine]

SUB1: PA=0xF, DELAY(0.5), PA=0x0, DELAY(0.5), PlayV(Ch0,\$V1)

SUB2: PA=0xF, DELAY(0.5), PA=0x0, DELAY(0.5), PlayV(Ch1,\$V1)

[Path]

PowerOn: BG(BG1,BG2), SUB1 ; Play Subroutine “SUB1”.

[Background1]

BG1:SUB2 ; Play Subroutine “SUB2”.

[Background2]

BG2:SUB2 ; PlaySubroutine “SUB2”.

3.34 QIO Custom

3.34.1 NY4 / NY5 / NY5+

In QIO Custom section, user can decode QIO / Wave Mark. When using this feature, user must use assembly language to compile QIO decoding program, the original decoding program will be replaced.

Note:

1. When using QIO Custom, it will apply to the entire project. User must control their own QIO process.
2. For more information about QIO format and Q-Code structure, please contact Nyquest Technology for assistance.

3.35 Path

3.35.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1

Path section lists the execution flows in the Q-Code program sheet, each execution flow is named as a “PATH”. There are 2 kinds of path, one is the Foreground Path to play voice or melody mostly, and the other is the Background Path to cooperate with the Foreground Path for background output.

Each Foreground Path consists of the path name and Steps. The path name is a symbol which leads a line prior to a colon ":". The paths can be linked to some trigger event by other sections or direct called by another path.

Note: While system executing each in Path section, if there are no PlayV, PlayS, Delay, Stop and StopV command to stop current playing, after executing other commands, program will go back to play or delay command address to execute next command. In Q-Code system, it's not necessary to add extra path for return to next command.

Ex.

[Input State]

KEY1: TR1R TR2R

[Path]

PowerOn: KEY1

TR1R: PlayV(Ch1, \$V0), Delay(1)

; When PlayV of TR be executed, then will delay for 1 second.

TR2R: R0=5

; Execute R0=5. If system is executing PlayV or Delay command, it will return after R0=5 is executed.

3.36 Background1

3.36.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1

The "Background1" is for defining the first background path. Each background path consists of the name of path, a ":", and one or more steps. The execution of a path usually results from external trigger events in "Foreground Path" or the direct call from other background paths.

Ex.

[Path]

P0: R0=0, PlayV(Ch0, Indian, 8K) & [BG1, X], PlayV(Ch0, ABC, 7K) & [BG3, X]

[Background1]

BG1: output1, delay(200ms), output2, delay(200ms), BG1

BG3: output3, delay(200ms), output4, delay(200ms), BG3

3.37 Background2

3.37.1 NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1

The "Background2" is for defining the second background path.

Ex.

[Path]

P0: R0=0, PlayV(Indian, 8K) & [X, BG2]

P1: BG(X, BG2)

[Background2]

BG2: PlayV(Ch0, \$V0, 12k)

3.38 Background3

3.38.1 NX1

The “Background3” is for defining the third background path.

Ex.

[Path]

P0: R0=0, PlayV(Indian, 8K) & [X, X, BG3]

P1: BG(X, X, BG3)

[Background3]

BG3: PlayV(Ch0, \$V0, 12k)

3.39 Background4

3.39.1 NX1

The “Background4” is for defining the fourth background path.

Ex.

[Path]

P0: R0=0, PlayV(Indian, 8K) & [X, X, X, BG4]

P1: BG(X, X, X, BG4)

[Background4]

BG4: PlayV(Ch0, \$V0, 12k)

3.40 Background5

3.40.1 NX1

The “Background5” is for defining the fifth background path.

Ex.

[Path]

P0: R0=0, PlayV(Indian, 8K) & [X, X, X, X, BG5]

P1: BG(X, X, X, X, BG5)

[Background5]

BG5: PlayV(Ch0, \$V0, 12k)

3.41 Interrupt

3.41.1 NX1

It is the same as Path section. Interrupt paths and associated paths should be placed in Interrupt section.

In NX1, interrupt handler must be placed in ROM. But Q-Code will automatically put paths on SPI Flash when XIP or Common ROM Code is enabled, which may cause problem.

Ex.

; Error when interrupt occurred.

[Option]

XIP = Speed_Optimize ; Enable XIP.

SPI0_Code_Access_Mode = Dual

[Path]

PowerOn: TMR_On(TMR0, 256us) ; Enable TMR0 interrupt.

Int_TMR0: Var0=1?Path1

Path1: ... ; Q-Code places Path1 to SPI Flash.

; Error may occurred when TMR0 interrupt is triggered and execute to Path1.

To prevent error, interrupt path and associated paths should be placed in **[Interrupt]** section, Q-Code will be forced to place them on ROM. Correct code is as following:

Ex.

[Option]

XIP = Speed_Optimize ; Enable XIP.

SPI0_Code_Access_Mode = Dual

[Path]

PowerOn: TMR_On(TMR0, 256us) ; Enable TMR0 interrupt.

[Interrupt]

Int_TMR0: Var0=1?Path1

Path1: ...

Interrupt paths are as following:

- INT_TMR0
- INT_TMR1
- INT_TMR2
- INT_TMR3
- INT_PWMA

- INT_PWMB
- RTC_2Hz
- RTC_64Hz
- RTC_1024Hz
- RTC_4096Hz
- RTC_16384Hz

4 Q-Code Special Paths

4.1 General Paths

4.1.1 PowerOn

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

What to be noticed is that the path name of “PowerOn” must appear in [Path] section and as the initial statement for the beginning of Q-Code program execution. “PowerOn” path is the first statement executed in [Path] section after the power of IC is turned on.

Note: PowerOn is the path reserved by the system. User is not allowed to set another path with the same name.

Ex.

[Path]

PowerOn: R0=0, R1=0, R2=0, R3=0xF

P0: R0=0, PlayV(Ch1, \$Indian,8K) & [BG1, x]

P1: PlayV(Ch1, \$Star,6K), PA=1, Delay(0.5), PA=0

P2: R0++, R0=2 ? P1, P0

P3: R1=R2, R3=R1|R2, PlayV(Ch1, \$ABC,7K)

4.1.2 Before_PowerOn

[NX1]

The path “Before_PowerOn” can only be applied in [Path] section. “Before_PowerOn” path is executed before “PowerOn”. When XIP(Execute in place) function is enabled, “PowerOn” path program will be stored in SPI Flash, while the “Before_PowerOn” path will be stored in the IC's internal ROM. If the SPI Flash chip selected by the user needs to be initialized with a special command, the SPI Flash command can be written in “Before_PowerOn”.

Once the end of the execution of “Before_PowerOn”, the system will execute “PowerOn” automatically. Users don't have to jump to “PowerOn” at the end of “Before_PowerOn”.

Ex. Initialize the SPI Flash before the system starts.

[Path]

PowerOn:

Before_PowerOn: SPI_WRSR(0x31,0x2)

Ex. Check the status of PB.3 before starting the system and initialize the SPI Flash.

[Path]

PowerOn:

Before_PowerOn: PB.3=0?PowerOn, SPI_WRSR(0x31,0x2)

4.1.3 Main

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The path “**Main**” can only be applied in [Path] section. When “Main” is applied, system will execute the path every time after the main loop of system is finished.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. Due to system may execute other functions simultaneously and in different ways of usages, the execution time would vary.
3. User cannot execute any Play or Delay commands in this path, but conditional jump is allowed.

Ex.

[Path]

PowerOn:

Main: !PA ; PA outputs invert after main loop is finished.

4.1.4 Sleep

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The path “**Sleep**” only can be applied in [Path] section. When “Sleep” applied, system will execute the path once before IC falls into sleep.

Ex.

[Path]

PowerOn: PB=0xF, DELAY(1) ; After power on, PB outputs 0xF for 1 second

SLEEP: PB=0 ; Set PB=0 before entering sleep mode.

4.1.5 WakeUp

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The path “**WakeUp**” only can be applied in [Path] section. When “WakeUp” applied, it represents system will execute the path once after IC being woken up.

Note: With I/O Expander, it requires at least 1ms to operate on I/O Expander after IC is woken up.

Ex.

[Path]

PowerOn: KEY1 ; After power on, execute KEY1 and enter sleep.

WakeUp: PB=0xF ; Every time IC is waked up, set PB = 0xF.

4.2 Timer Paths

4.2.1 512us

[NY7]

The path “512us” only can be applied in [Path] section. When “512us” is applied, system will execute the path every 512us.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. It may not be exact 512us since system may be busy on other functions.
3. Please refer [Unusable Commands for Special Path](#) for commands that cannot be used.

Ex.

[Path]

PowerOn:

512us: !PA ; PA output inverts every 512us.

4.2.2 1ms

[NY5+ / NY6 / NY7 / NY9T / NX1]

The path “1ms” can only be applied in [Path] section. When “1ms” is applied, system will execute the path every 1ms.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. It may not be exact 1ms since system may be busy on other functions.
3. Please refer [Unusable Commands for Special Path](#) for commands that cannot be used.
4. In NY9T, after executing 1ms path, must execute 1ms_RET command. Otherwise, there will be unexpected system error.
5. In NY9T, if TimeBase=4ms, 1ms path will not be executed.

Ex.

[Path]

PowerOn:

1ms: !PA ; PA output inverts every 1ms.

4.2.3 2ms

[NY5+ / NY6 / NY7 / NX1]

The path “2ms” only can be applied in [Path] section. When “2ms” applied, system will execute the path every 2 ms.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. It may not be exact 2ms since system may be busy on other functions.
3. Please refer [Unusable Commands for Special Path](#) for commands that cannot be used.

Ex.**[Path]****PowerOn:****2ms: !PA** ; PA output inverted every 2ms.**4.2.4 4ms****[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]**

The path “**4ms**” can only be applied in **[Path]** section. When “4ms” is applied, system will execute the path every 4ms.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. It may not be exact 4ms since system may be busy on other functions.
3. Please refer [Unusable Commands for Special Path](#) for commands that cannot be used.
4. In NY9T, after executing 4ms path, must execute 4ms_RET command. Otherwise, there will be unexpected system error.

Ex.**[Path]****PowerOn:****4ms: !PA** ; Invert PA output every 4ms.**4.2.5 8ms****[NY5+ / NY6 / NY7 / NX1]**

The path “**8ms**” only can be applied in **[Path]** section. When “8ms” applied, system will execute the path every 8ms.

Note:

1. Do not use too many commands in this path, or it will lower the performance of system.
2. It may not be exact 8ms since system may be busy on other functions.
3. Please refer [Unusable Commands for Special Path](#) for commands that cannot be used.

Ex.**[Path]****PowerOn:**

8ms: !PA

; PA output inverts every 8ms.

4.2.6 500ms

[NY9T / NX1]

The path “500ms” only can be applied in [Path] section. When “500ms” is applied, system will execute the path every 500ms.

Ex.

[Path]

PowerOn:

500ms: R0++, R0=2?Timer_1sec

; R0 counts every 500ms, 2 times means 1 sec.

Timer_1sec: R0=0, User Program.....

Note:

1. In NY9T, user can acquire the better power saving by using the internal 16KHz RC oscillator when applying long duration counting. There are frequency deviations in the 16KHz RC oscillator, so please don't use it if an accurate timing is required.
2. Please [Unusable Commands for Special Path](#) refer for commands that cannot be used.

4.2.7 1sec

[NY9T]

The path “1sec” only can be applied in [Path] section. When “1sec” is applied, system will execute the path every 1 second.

Ex.

[Path]

PowerOn:

1sec: X0++, X0=60?Timer_1min

; X0 counts every 1sec, 60 times means 1 minute.

Timer_1min: X0=0, User Program.....

Note:

1. In NY9T, user can acquire the better power saving by using the internal 16KHz RC oscillator when applying long duration counting. There are frequency deviations in the 16KHz RC oscillator, so please don't use it if an accurate timing is required.
2. Please [Unusable Commands for Special Path](#) refer for commands that cannot be used.

4.2.8 4sec

[NY9T]

The path “**4sec**” only can be applied in [**Path**] section. When “4sec” applied, system will execute the path every 4 seconds.

Ex.

[**Path**]

PowerOn:

4sec: R0++, R0=15?Timer_1min ; R0 counts every 4sec, 15 times means 1 minute.

Timer_1min: R0=0, User Program.....

Note:

1. *NY4 / NY5 / NY5+ / NY6 / NY7 / NX1 do not support the option.*
2. *In NY9T, user can acquire the better power saving by using the internal 16KHz RC oscillator when applying long duration counting. There are frequency deviations in the 16KHz RC oscillator, so please don't use it if an accurate timing is required.*
3. *Please refer [Unusable Commands for Special Path](#) for commands that cannot be used.*

4.3 Interrupt Paths

4.3.1 Interrupt

[NY5 / NY5+ / NY7]

The “**Interrupt**” only can be applied in [**Path**] section. When “Interrupt” is applied, it represents that system will execute the path once each time IC is interrupted.

Note:

1. *There is only one level stack for interrupt function, PlayA / PlayAS, PlayM / PlayMS and QIO of Q-Code language are implemented with interrupt function. Therefore, commands in the interrupt path will be executed when running PlayM and PlayA.*
2. *For NY5, after executing PlayA / PlayAS, PlayM / PlayMS and QIO command, the interrupt interval will be modified.*
3. *Please refer [Unusable Commands for Special Path](#) for commands that cannot be used.*

Ex.

[**Path**]

PowerOn: KEY1, INT_ON, INT=1.024

Interrupt: !PA ; Interrupts every 1ms. And PA output is inverted when interrupted.

4.3.2 Int_256us

[NY6]

The path “**Int_256us**” only can be applied in [**Path**] section. When Int_256us applied, system will

execute this path when 256us interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. Please refer [Unusable Commands for Special Path](#) for commands that cannot be used.

Ex.

[Path]

PowerOn:

Int_256us: !PA ; PA output inverts when 256 us interrupts.

4.3.3 Int_512us

[NY6]

The path “Int_512us” only can be applied in [Path] section. When Int_512us applied, system will execute this path when 512us interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. Please refer [Unusable Commands for Special Path](#) for commands that cannot be used.

Ex.

[Path]

PowerOn:

Int_512us: !PA ; PA output inverts when 512us interrupts.

4.3.4 Int_1ms

[NY6]

The path “Int_1ms” only can be applied in [Path] section. When Int_1ms applied, system will execute this path when 1ms interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. Please refer [Unusable Commands for Special Path](#) for commands that cannot be used.

Ex.

[Path]

PowerOn:

Int_1ms: !PA ; PA output inverts when 1ms interrupts.

4.3.5 Int_16ms

[NY6]

The path “Int_16ms” only can be applied in [Path] section. When Int_16ms applied, system will execute this path when 16ms interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. Please refer [Unusable Commands for Special Path](#) for commands that cannot be used.

Ex.

[Path]

PowerOn:

Int_16ms: !PA ; PA output inverts when 16ms interrupts.

4.3.6 Int_CMP

[NY6B / NY6C]

The path “Int_CMP” only can be applied in [Path] section. When Int_CMP applied, system will execute this path when comparator interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. User cannot execute any Play or Delay commands in this path, but conditional jump is allowed.

Ex.

[Path]

PowerOn:

Int_CMP: !PA ; PA output inverts when CMP interrupts.

4.3.7 Int_TMR

[NY6]

The path “Int_TMR” only can be applied in [Path] section. When Int_TMR applied (user can set the timer interval via program), system will execute this path when timer interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. User cannot execute any Play or Delay commands in this path, but conditional jump is allowed.

Ex.

[Path]

PowerOn:

Int_TMR: !PA ; PA output inverts when TMR interrupts.

4.3.8 Int_TMR0

[NY5+ / NX1]

In NY5+, the path “Int_TMR0” only can be applied in [Path] section. In NX1, “Int_TMR0” only can be applied in [Interrupt] section. When Int_TMR0 applied (user can set the timer interval via program), system will execute this path when Timer0 interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. User cannot execute any Play or Delay commands in this path, but conditional jump is allowed.
3. In NY5+, supported timer range: 128us ~ 256us.
4. In NX1 OTP, timer range depends on High Clock Frequency:
 - 64 ~ 5461us when High Clock Frequency is 12MHz.
 - 64 ~ 4096us when High Clock Frequency is 16MHz.
 - 64 ~ 2730us when High Clock Frequency is 24MHz.
 - 64 ~ 2048us when High Clock Frequency is 32MHz.
5. In NX1 EF, supported timer range: 64 ~ 1365us.

Ex.

[Path]

PowerOn: TMR_On(TMR0, 128us)

Int_TMR0: !PA ; In every 128us, PA output inverts when TMR0
; interrupts.

4.3.9 Int_TMR1

[NY5+ / NX1]

In NY5+, the path “Int_TMR1” only can be applied in [Path] section. In NX1, “Int_TMR1” only can be applied in [Interrupt] section. When Int_TMR1 applied (user can set the timer interval via program), system will execute this path when Timer1 interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. User cannot execute any Play or Delay commands in this path, but conditional jump is allowed.
3. In NY5+, supported timer range: 128us ~ 256us.

4. In NX1 OTP, timer range depends on High Clock Frequency:

- 64 ~ 5461us when High Clock Frequency is 12MHz.
- 64 ~ 4096us when High Clock Frequency is 16MHz.
- 64 ~ 2730us when High Clock Frequency is 24MHz.
- 64 ~ 2048us when High Clock Frequency is 32MHz.

5. In NX1 EF, supported timer range: 64 ~ 1365us.

Ex.

[Path]

PowerOn: TMR_On(TMR1, 128us)

Int_TMR1: !PA ; In every 128us, PA output inverts when TMR1
; interrupts.

4.3.10 Int_TMR2

[NY5+ / NX1]

In NY5+, the path “Int_TMR2” only can be applied in [Path] section. In NX1, “Int_TMR2” only can be applied in [Interrupt] section. When Int_TMR2 applied (user can set the timer interval via program), system will execute this path when Timer2 interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. User cannot execute any Play or Delay commands in this path, but conditional jump is allowed.
3. In NY5+, supported timer range: 128us ~ 256us.
4. In NX1 OTP, timer range depends on High Clock Frequency:
 - 64 ~ 5461us when High Clock Frequency is 12MHz.
 - 64 ~ 4096us when High Clock Frequency is 16MHz.
 - 64 ~ 2730us when High Clock Frequency is 24MHz.
 - 64 ~ 2048us when High Clock Frequency is 32MHz.
5. In NX1 EF, supported timer range: 64 ~ 1365us.

Ex.

[Path]

PowerOn: TMR_On(TMR2, 128us)

Int_TMR2: !PA ; In every 128us, PA output inverts when TMR2
; interrupts.

4.3.11 Int_TMR3

[NY5+ / NX1]

In NY5+, the path “**Int_TMR3**” only can be applied in **[Path]** section. In NX1, “**Int_TMR3**” only can be applied in **[Interrupt]** section. When Int_TMR3 applied (user can set the timer interval via program), system will execute this path when Timer3 interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. User cannot execute any Play or Delay commands in this path, but conditional jump is allowed.
3. In NY5+, supported timer range: 128us ~ 256us.
4. In NX1 OTP, timer range depends on High Clock Frequency:
 - 64 ~ 5461us when High Clock Frequency is 12MHz.
 - 64 ~ 4096us when High Clock Frequency is 16MHz.
 - 64 ~ 2730us when High Clock Frequency is 24MHz.
 - 64 ~ 2048us when High Clock Frequency is 32MHz.
5. In NX1 EF, supported timer range: 64 ~ 1365us.

Ex.

[Path]

PowerOn: TMR_On(TMR3, 128us)

Int_TMR3: !PA ; In every 128us, PA output inverts when TMR3
; interrupts.

4.3.12 Int_PWMA

[NX1]

The path “**Int_PWMA**” only can be applied in **[Interrupt]** section. When Int_PWMA applied (user can set the timer interval via program), system will execute this path when PWMA Timer interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. User cannot execute any Play or Delay commands in this path, but conditional jump is allowed.
3. In NX1 OTP, timer range depends on High Clock Frequency:
 - 64 ~ 5461us when High Clock Frequency is 12MHz.
 - 64 ~ 4096us when High Clock Frequency is 16MHz.
 - 64 ~ 2730us when High Clock Frequency is 24MHz.
 - 64 ~ 2048us when High Clock Frequency is 32MHz.
4. In NX1 EF, supported timer range: 64 ~ 1365us.

Ex.

[Path]

TR0: TMR_On(TMR3, 128us)

[Interrupt]

Int_PWMA: !PA ; In every 128us, PA output inverts when PWMA
; Timer interrupts.

4.3.13 Int_PWMB

[NX1]

The path “Int_PWMB” only can be applied in [Interrupt] section. When Int_PWMB applied (user can set the timer interval via program), system will execute this path when PWMB Timer interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. User cannot execute any Play or Delay commands in this path, but conditional jump is allowed.
3. In NX1 OTP, timer range depends on High Clock Frequency:
 - 64 ~ 5461us when High Clock Frequency is 12MHz.
 - 64 ~ 4096us when High Clock Frequency is 16MHz.
 - 64 ~ 2730us when High Clock Frequency is 24MHz.
 - 64 ~ 2048us when High Clock Frequency is 32MHz.
4. In NX1 EF, supported timer range: 64 ~ 1365us.

Ex.

[Path]

TR0: TMR_On(PWMB, 128us)

[Interrupt]

Int_PWMB: !PA.3 ; In every 128us, PA.3 output inverts when PWMB
; Timer interrupts.

4.3.14 RTC Interrupt (RTC_2Hz / RTC_64Hz / RTC_1024Hz / RTC_4096Hz / RTC_16384Hz)

[NX1]

“RTC_2Hz / RTC_64Hz / RTC_1024Hz / RTC_4096Hz / RTC_16384Hz” only can be applied in [Interrupt] section. System will execute these paths when RTC interrupt occurred.

Note:

1. Do not use too many commands in this path, or it may lower the performance of system.
2. User cannot execute any Play or Delay commands in this path, but conditional jump is allowed.
3. RTC interrupt is activated when system initialized, there is no need to switch manually.

4. *RTC_4096Hz is only supported by NX11P22A / NX12P44A / NX13P44A. And RTC_16384Hz is supported by other NX1 bodies.*

Ex.

[Interrupt]

RTC_2Hz: !PA.6 ; Triggered twice every second. PA.6 output is inverted when
; RTC_2Hz interrupt occurred.

4.4 LVD Paths

4.4.1 LVD_mVn / LVD_Max

[NY4PxxxC / NY5+ / NY6P025A / NY6B / NY6C / NX1]

The path “LVD_mVn” / “LVD_Max” only can be applied in [Path] section. When VDD matches supported range, system will execute path once.

Supported VDD range and system path is as following:

VDD Range	NY4PxxxC / NY5+	NY6P025A LVD1	NY6P025A LVD2	NY6B / NY6C	NX1 OTP	NX1 EF
<2.0V	LVD_2V0	LVD_2V4	LVD_2V0	LVD_2V4	LVD_2V2	LVD_2V0
2.0V ~ 2.2V	LVD_2V2		LVD_2V2			LVD_2V2
2.2V ~ 2.4V	LVD_2V4		LVD_3V0		LVD_2V7	LVD_2V4
2.4V ~ 2.6V	LVD_2V8	LVD_2V8		LVD_2V6		LVD_2V8
2.6V ~ 2.7V				LVD_3V2	LVD_3V2	
2.6V ~ 2.8V	LVD_3V6	LVD_Max	LVD_3V6			LVD_Max
2.8V ~ 3.0V				LVD_3V0	LVD_Max	
3.0V ~ 3.2V			LVD_3V3	LVD_3V2	LVD_Max	
3.2V ~ 3.3V	LVD_3V6	LVD_Max	LVD_Max	LVD_3V4	LVD_3V4	
3.3V ~ 3.4V				LVD_Max	LVD_Max	
3.4V ~ 3.6V	LVD_Max	LVD_Max	LVD_Max	LVD_3V6	LVD_3V6	
3.6V ~ 4.1V				LVD_4V1	LVD_Max	LVD_Max
>4.1V				LVD_Max	LVD_Max	

Note:

1. *Execution of PowerOn may be interrupted if this is path is triggered while power is on.*
2. *in NY5+, for example, when LVD_2V8 and LVD_2V4 are present, LVD_2V4 covers the changes between <2.0V ~ 2.4V.*

4.4.2 LVD

[NY5+ / NY6B / NY6C]

The path “**LVD**” only can be applied in **[Path]** section. When VDD changes among different LVD levels, system will execute this path once.

Ex.

[Path]

PowerOn:

LVD: PA=1

; PA output 1 when voltage changes among different LVD levels.

4.5 Communication Paths

4.5.1 IR_RX

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

The path “**IR_RX**” can only be applied in **[Path]** section. When “IR_RX” is applied, it represents that system will execute the path once after IR receives data. User can access the data of IR in this path.

Ex.

[Path]

IR_RX: [R1, R0] = IR_RX

; Store received IR data to R0 and R1.

4.5.2 SC_RX

[NY4 / NY5 / NY5+ / NY6 / NY7]

The path “**SC_RX**” can only be applied in **[Path]** section. When “SC_RX” is applied, it represents that system will execute the path once after SC receives data. User can access the data of SC in this path.

Ex.

[Path]

SC_RX: [R1, R0] = SC_RX

; Store received SC data to R0 and R1.

4.5.3 I2C_RX

[NY5+ / NX1]

The path “**I2C_RX**” can only be applied in **[Path]** section. When “I2C_RX” is applied, the system will execute the path once after I2C data is received. User can access the received I2C data in this path.

Ex.

[Path]

I2C_RX: [R1, R0] = I2C_RX

; Store received I2C data to R0 and R1.

4.5.4 I2C_TX_Ok

[NY5+ / NX1]

The path “I2C_TX_Ok” can only be applied in [Path] section. The system will execute this path once everytime I2C data is transmitted. User can check whether ACK is responded.

Ex.

[Path]

I2C_TX_Ok: I2C_Ack?Path1 ; I2C data is transmitted, check whether ACK is responded.

Path1: I2C_TX(0x5A) ; transmit 0x5A is ACK is responded.

4.5.5 I2C_Error

[NX1]

The path “I2C_Error” can only be applied in [Path] section. The system will execute this path once I2C data transmit failed due to Clock Stretching timeout or Bus arbitration, etc.

Ex.

[Path]

I2C_Error: I2C_Reset ; I2C transmit fail, reset I2C.

4.5.6 UART_RX

[NY5+]

The path “UART_RX” can only be applied in [Path] section. When “UART_RX” is applied, the system will execute the path once after UART data is received. User can access the received UART data in this path.

Ex.

[Path]

UART_RX: [R1, R0] = UART_RX ; Store received UART data to R0 and R1.

4.5.7 WaveID_RX

[NX1]

The path “WaveID_RX” can only be applied in [Path] section. When WaveID_RX is applied, it represents that system will execute this path once after WaveID receives data. User can access the data of WaveID in this path.

Ex.

[Path]

WaveID_RX: R0= WaveID_RX ; Store received WaveID data to R0.

4.5.8 SL_RX

[NX1]

The path “**SL_RX**” can only be applied in [Path] section. When Sound Localization is applied, it represents that system will execute this path once after Sound Localization receives data. User can access the data of Sound Localization in this path.

Ex.

[Path]

SL_RX: R0= SL_RX ; Save detected angle to R0 after receiving data.

4.6 Touchkey Paths

4.6.1 Enforce_Calibrate

[NY9T]

“**Enforce_Calibrate**” path name can only appear in [Path] section. Every time after Enforce_Calibrate is done, the path will be executed once.

Note:

1. *Enforce_Calibrate path will be executed only when SW_Reset is disabled.*
2. *If it needs to define a different way of reset due to different applications, user may disable SW_Reset option, and manually reset the system by program in Enforce_Calibrate Path.*

Ex. There is no need to keep previous setting or mode, user may use SW_Reset command to reset the system.

[Path]

PowerOn: KEY1 ; Enter sleep after Power On.
TR1R: PE.0=1 ; PE.0 output high when Touch-Key is pressed.
TR1F: PE.0=0 ; PE.0 output low when Touch-Key is released.
Enforce_Calibrate: SW_Reset ; After every Enforce_Calibrate, do a software reset.

Ex. It needs to retain setting or mode, and keep the previous status after Enforce_Calibrate is done.

[Path]

PowerOn: KEY1 ; Enter sleep after Power On.
TR1R: PE.0=1 ; PE.0 output high when Touch-Key is pressed.
TR1F: PE.0=0 ; PE.0 output low when Touch-Key is released.
TR2R: R0++ ; switch mode.
Enforce_Calibrate: TouchKey_CLR ; After every Enforce_Calibrate, clear Touch-Key Status

Ex. It needs to retain setting or mode, and clear IO status after Enforce_Calibrate is done. If Touch-Key status does not clear after Enforce_Calibrate is done, it will automatically execute Touch-Key release

paths sequentially.

[Path]

PowerOn: KEY1 ; Enter sleep after Power On.
 TR1R: PE.0=1 ; PE.0 output high when Touch-Key is pressed.
 TR1F: PE.0=0 ; PE.0 output low when Touch-Key is released.
 TR2R: R0++ ; switch mode.
 Enforce_Calibrate: PE=0 ; After every Enforce_Calibrate, clear IO status.

Ex. It needs to retain setting or mode, and clear IO and Touch-Key status after Enforce_Calibrate is done, without executing Touch-Key release paths.

[Path]

PowerOn: KEY1 ; Enter sleep after Power On.
 TR1R: PE.0=1 ; PE.0 output high when Touch-Key is pressed.
 TR1F: PE.0=0 ; PE.0 output low when Touch-Key is released.
 TR2R: R0++ ; switch mode.
 Enforce_Calibrate: PE=0, TouchKey_CLR ; After every Enforce_Calibrate, clear IO and Touch-Key status.

4.7 SPI Flash Paths

4.7.1 SPI_EraseEnd

[NY6B / NY6C / NY7 / NX1]

“SPI_EraseEnd” path name can only appear in [Path] section. Every time after SPI flash erasing is done, the SPI_EraseEnd path will be executed once so that user knows when erase is completed in order to perform subsequent actions. Erase commands include SPI_CE, SPI_SE and SPI_BE °

Ex.

[Path]

Erase: SPI_CE, R0=1 ; Erase all memory and set R0 to 1.
 SPI_EraseEnd: R0=0 ; After erase is completed, clear R0 to 0.

4.7.2 SPI_EraseTimeout

[NX1]

The path “SPI_EraseTimeout” can only be applied in [Path] section. SPI_EraseTimeout path will be executed once when SPI Flash is erasing and wait exceed the erase time set in Option->SPI Flash, which timeout is occurred. This path is used to detect whether SPI Flash is working correctly and conditional jump is not allowed in this path.

Ex.

[Path]

TR1: EraseR(\$Rec0) ; Erase Rec0 recording block on SPI Flash.
 SPI_EraseTimeout: PA.0=1 ; Timeout is occurred, set PA.0 to 1.

4.8 Voice Recognition Paths

4.8.1 VRGC_Timeout

[NX1]

The path “**VRGC_Timeout**” can only be applied in [Path] section. If setting VR combo group VRGC1=VRG1+VRG2, and the command in VRG1 is successfully recognized but the VRG2 command is not recognized within a given time, the VRGC_Timeout path will be automatically executed.

Ex.

[Path]

VRGC_Timeout: PlayV(ch0, \$V0) ; When the voice command is not recognized, the user is prompted to speak again.

4.8.2 VR_Unknown

[NX1]

The path “**VR_Unknown**” can only be applied in [Path] section. When voice recognition detects a sound, but command is not successfully recognized, the path is executed once.

Note: VAD must be enabled before using this function.

Ex.

[Path]

VR_Unknown: PlayV(ch0, \$V0) ; When the voice command is not clear, the user is prompted to speak again.

4.8.3 VT_BeforeRecord

[NX1]

The path “**VT_BeforeRecord**” can only be applied in [Path] section. Voice Tag recording will execute several times before training. When “**VT_BeforeRecord**” is applied, it will be executed before Voice Tag recording is performed.

Note:

- 1. The path cannot call other path.*
- 2. This path cannot execute delay commands, except PlayV / PlayA / SpiPlay / SDelay.*
- 3. When this path is executing, other paths will be paused.*

Ex.

[Path]

VT_BeforeRecord: SpiPlay(ch0, 0) ; Call this path before Voice Tag records

4.8.4 VT_BeforeTraining

[NX1]

The path “**VT_BeforeTraining**” can only be applied in **[Path]** section. After Voice Tag recording is completed, and before training, this path will be executed.

Note:

1. *The path cannot call other path.*
2. *This path cannot execute delay commands, except PlayV / PlayA / SpiPlay / SDelay .*
3. *When this path is executing, other paths will be paused.*

Ex.

[Path]

VT_BeforeTraining: SpiPlay(ch0, 0) ; Call this path before Voice Tag records

4.8.5 VT_AddTagFail

[NX1]

The path “**VT_AddTagFail**” can only be applied in **[Path]** section. When the training of Voice Tag fails, it will call this path.

Ex.

[Path]

VT_AddTagFail: SpiPlay(ch0, 0) ; Call this path before Voice Tag records

4.9 Record Paths

4.9.1 KRecord_Timeout

[NX1]

The path “**KRecord_Timeout**” can only be applied in **[Path]** section. When the recording of keyboard exceeds time, but the InstNoteON / InstNoteOff command doesn't execute, it will call this path.

Ex.

[Path]

KRecord_Timeout: PlayKS(\$Rec0, 0) ;When the keyboard recording timeout occurred, using Rec0 to play the recording.

4.10 Sound Detect Paths

4.10.1 Sound_Detected

[NX1]

The path “**Sound_Detect**” can only be applied in **[Path]** section. When sound is detected, this path will be executed.

Ex.

[Option]

SoundDetect = Enable
 SoundDetect_High_Threshold = 300
 SoundDetect_Low_Threshold = 300
 SoundDetect_Active_Time = 80ms
 SoundDetect_Mute_Time = 650ms

[Path]

TR1: SoundDetect_On ; Active sound detection.
 Sound_Detect: PA.0 = 1 ; When sound is detected, set PA.0 to 1.
 Sound_Mute: PA.0 = 0 ; After sound is not detected for more than 150ms,
 ; set PA.0 to 0.

4.10.2 Sound_Muted

[NX1]

The path “**Sound_Mute**” can only be applied in **[Path]** section. Once sound is not detected any more, this path will be executed.

4.11 Pitch Detect Paths

4.11.1 Pitch_Detected

[NX1]

The path “**Pitch_Detected**” can only be applied in **[Path]** section, Once pitch is detected, this path will be executed.

Ex.

[Option]

PitchDetect = Tone ; Pure-Tone detection.
 PitchDetect_Range = Middle ; Detection range: 1000 ~ 3000Hz.

[Variable]

Var16: Pitch_Value

[Path]

TR1: PitchDetect_On ; Activate pitch detection.
 TR2: PitchDetect_Off ; Deactivate pitch detection.
 Pitch_Detected: ReadPitch(Pitch_Value) ; Pure tone in 1000 ~ 3000Hz is detected,
 ; save detected pitch to variable Pitch_Value ◦

4.12 I/O Expander Path

4.12.1 IoExp_CommFail

[NY5+ / NX1]

The path “IoExp_CommFail” can only be applied in [Path] section. When communication with I/O Expander failed, this path will be executed.

Ex.

[Path]

IoExp_CommFail: R0 = IoExp_ErrId

; When communication with I/O Expander failed, get failed I/O Expander ID to R0.

5 Q-Code Command

Q-Code provides 4-bit commands and 8-bit commands. The RAM used by 4-bit commands is **Ri**, and the RAM used by 8-bit commands is **Xi**.

Xi is composed by two 4-bit RAM, the corresponding method is shown below.

$$X(i) = [X(i)L, X(i)H] = [R(i*2), R(i*2+1)]$$

Ex. X10 = [R20, R21], where R20 / X10L is the low-nibble of X10, and R21 / X10H is the high-nibble of X10.

Please refer to [Corresponding table of Ri and Xi](#) for the complete table.

Note:

1. **NY9T001A/004A do not provide 8-bit multiplication and division.**
2. **NX1 do not predefine Ri / Xi variables, user should declare needed variables in [Variable] section.**

5.1 Arithmetic Logic Command

Arithmetic Logic Command			
Var1 = Var2	Var1 = Var2 + Var3	Var1 = Var2 - Var3	Var1 = Var2 * Var3
Var1 = Var2 / Var3	Var1 = Var2 % Var3	Var++	Var--
Var1 = Var2 & Var3	Var1 = Var2 Var3	Var1 = Var2 ^ Var3	Var1 = Var2 << Var3
Var1 = Var2 >> Var3	!Var	Var=RandomL	Var=RandomH
Var=Random	-	-	-

5.1.1 Variable Operation

Q-Code supports the assignments and operations of variables. The variable can be:

- 1-bit operand: Ri.n / Xi.n / Pin
- 4-bit operand: Ri / Port
- 8-bit operand: Xi
- Variables defined in **[Variable]**
- System variables

Supported variable operations are listed below.

No.	Variables	Note	Example
1	Var1 = Var2	Assignment operation	R0 = 0x4, R0 = R1, R0 = RandomL, R0 = RandomH
2	Var1 = Var2 + Var3	Addition	R2 = R0 + 5, R2 = 5 + R0

No.	Variables	Note	Example
3	Var1 = Var2 – Var3	Subtraction	R2 = R0 – 3, R2 = 5 – R0
4	Var1 = Var2 * Var3 [Var1, Var2] = Var3 * Var4	Multiplication NX1 does not support second form.	X1 = R0 * R1, X1 = R0 * 5 [R0, R1] = R2 * 15
5	Var1 = Var2 / Var3 [Var1, Var2] = Var3 / Var4	Division In second form, the quotient of division operation is saved in Var1, and the remainder is saved in Var2.	R2 = R0 / 5, X2 = X1 / R0, [R2, R1] = R0 / 5
6	Var1 = Var2 % Var3	Remainder operation	R2 = R1 % 15
7	Var++	Increment operation	R0++
8	Var--	Decrement operation	PA--
9	Var1 = Var2 & Var3	“AND” operation	R2 = R0 & R1
10	Var1 = Var2 Var3	“OR” operation	R2 = R0 R1
11	Var1 = Var2 ^ Var3	“XOR” operation	R2 = R0 ^ R1
12	Var1 = Var2 << Var3	Left shift	R2 = R0 << 2, R2 = R0 << R1
13	Var1 = Var2 >> Var3	Right shift	R2 = R0 >> 2, R2 = R0 >> R1
14	!Var	Invert operation	!R0, !R0.3

Note:

1. Supported system variables are listed below.

Variables	Note	Availability	Example
Vol	Volume	NY5 / NY5+ / NY6 / NY7 / NX1	Vol = R1
V_Chx_Vol	The channel volume of voice	NY5+ / NY6 / NY7	V_Ch0_Vol = X0
M_Chx_Vol	The individual channel volume of Melody	NY5+ / NY6 / NY7 / NX1	M_Ch10_Vol = R0
SPIVol	The channel volume of SPIPlay	NY6B / NY6C / NY7	SPIVol = R0
Px	IO port	All	PA = R0 ^ 0x9
PxM	Input / Output mode of IO port	NY4 / NY5+ / NY6 / NY7 / NY9T / NX1	PAM = PFM 0x1
PxM.n	Input / Output mode of IO pin	NY4 / NY5+ / NY6 / NY7 / NY9T / NX1	PAM.0 = PFM.0
RandomL	Low nibble of random number	All	R0 = RandomL ^

Variables	Note	Availability	Example
	Read only, do not use at the left side of “=”		0x5
RandomH	Hight nibble of random number Read only, do not use at the left side of “=”	All	R0 = RandomH 0x9
Random	Random number Read only, do not use at the left side of “=”	All	X0 = Random ^ 0x33

2. When 1-bit / 4-bit / 8-bit variables are used in variable operation, assigning the variable or operation result to variable with less bit, the overflow bits would be truncated

Ex. X0 = 0xFA, R3 = X0 ; R3 = 0xA

5.1.2 Var=RandomL

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Generate random number, and save the low-nibble to Var. Random number range is determined by option Random.

5.1.3 Var=RandomH

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Generate random number, and save the high-nibble to Var. Random number range is determined by option Random.

5.1.4 Var=Random

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Generate random number, and save to Var. Random number range is 0 ~ Random Option if Min / Max is not specified. If Min / Max is specified, range of generated random number is Min ~ Max – 1.

Var=Random

Var=Random({Min, }Max)

Var: Variable used to store generated random number.

Min: Minimum value of generated random number. 0 if omitted.

Max: Accepts max 0xFFFFFFFF.

Notes:

- 1. NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T do not support Min / Max parameters.**
- 2. In NX1, generated random number is not restricted by Random option if Min / Max is specified.**

5.2 Flow Control Command

Flow Control Command					
Var = Var ? Path	Var != Var ? Path	Var >= Var ? Path	Var <= Var ? Path	Var > Var ? Path	
Var < Var ? Path	Px = data?Path	Px != data?Path	Px.n = 0?Path	Px.n != 0?Path	
Px.n = 1?Path	Px.n != 1?Path	Vol = n?Path	Vol !=n ?Path	-	
MixCtrl = data?Path		MixCtrl != data ?Path		RandomL=data?Path	RandomH=data?Path
RandomL!=data?Path		RandomH!=data?Path		Random=data?Path	Random!=data?Path
Px[1 X 0 X]?Path	ChUsed(Ch)?Path	Voice(Ch)?Path	PaueV(Ch)?Path	Melody?Path	
PauseM?Path	Delay(n)?Path	PauseD(n)?Path	Action(Ch)?Path	PauseA(Ch)?Path	
PWMIO(n)?Path	PausePWM?Path	HoldPWM?Path	SPIPlay?Path	SPIPause?Path	
Pause(n)?Path	Record?Path	KRecord?Path	Recorded?Path	PlayK?Path	
EraseR?Path	VR_VAD?Path	LEDStr?Path	LEDSync?Path	LEDText?Path	
Animaltalks_Record?Path		Animaltalks_Play?Path			-
Animalsing_Record?Path		Animalsings_Play?Path			Calibrate?Path
IoExp_Exists?Path		CheckSum?Path	SPI_CheckSum?Path		
!Px[1 X 0 X]?Path	!ChUsed?Path	!Voice?Path	!PauseV?Path	!Melody?Path	
!PauseM?Path	!Delay?Path	!PauseD?Path	!Action?Path	!PauseA?Path	
!PWMIO?Path	!PausePWM?Path	!HoldPWM?Path	!SPIPlay?Path	!SPIPause?Path	
!Pause?Path	!Record?Path	!KRecord?Path	!Recorded?Path	!PlayK?Path	
!EraseR?Path	!VR_VAD?Path	!LEDStr?Path	!LEDSync?Path	!LEDText?Path	
!Animaltalks_Record?Path		!Animaltalks_Play?Path			-
!Animalsings_Record?Path		!Animalsings_Play?Path			!Calibrate?Path
!IoExp_Exists?Path		!CheckSum?Path	!SPI_CheckSum?Path		
If-Else	-	-	-	-	
Switch(Ri) = [Path0, Path1, Path2...Path15]		Switch(Px) = [Path0, Path1, Path2...Path15]			
Switch(RandomL) = [Path0, Path1...Path15]		Switch(RandomH) = [Path0, Path1...Path15]			
Switch(Px[d x d x]) = [Path0, Path1, Path2....Path15]					
Switch(Xi)=[Path0, Path1, Path2...Path255]			Switch(Random)=[Path0, Path1,Path2...Path255]		
While	Do-While	For	-	-	

Syntax:

R op T ? True_Branch { : False_Branch }

R: RAM or variable.

Op: Operator, see table below.

T: R or constant.

?: Symbol of judgment.

True_Branch { : False_Branch }: can be the name of path or ASM.

{ }: false clause inside the curly brackets can be omitted.

Conditional jump instruction is to decide whether to execute True_Branch or False_Branch according to conditional True or False value. If the False_Branch is neglected and the conditional value is False, it will jump to the next step of current path.

In addition to using a common path name, the branch path can also be used with a series of instruction sets that are wrapped in braces. Instruction set will be automatically launched as anonymous path. The example below on the left and right sides are equivalent.

P1: R0=1? P2, P3 P2: PlayV(ch0, \$V0), P4 P3: PlayV(ch0, \$V1), P4 P4: PlayV(ch0, \$V2)	P1: R0=1? {PlayV(ch0, \$V0) } : {PlayV(ch0, \$V1)}, PlayV(ch0, \$V2)

5.2.1 Comparison

Operator	Definition	Example
Var1 = Var2	Equal to	R0 = 2 ?TRUE, X0 = R1 ?TRUE, R0.0 = 1 ? True
Var != Var	Not equal to	R0 != R1?TRUE ; R0 != 3 ?TRUE ; PA.0 = 1 ?TRUE
Var >= Var	Greater than or equal to	Vol >= R1?TRUE ; R0 >= 3 ?TRUE
Var > Var	Greater than	Random > 128 ?TRUE ; R0 > 5 ?TRUE
Var <= Var	Less than or equal to	R0 <= R1?TRUE ; R0 <= 5 ?TRUE
Var < Var	Less than	R0 < R1 ?TRUE ; R0 = 7, R0 < 5 ?TRUE

Q-Code supports the comparison of variables, the variables could be:

- 1-bit operator: Ri.n / Xi.n / Px.n
- 4-bit operator: Ri / Px
- 8-bit operator: Xi
- Variables defined in **[Variable]**
- System variables

Note: The available system variables are listed below.

Name	Descriptions	Availability	Example
Vol	Volume	NY5 / NY5+ / NY6 / NY7 / NX1	Vol = R1
V_Chx_Vol	The channel volume of voice	NY5+ / NY6 / NY7	V_Ch0_Vol = X0
M_Chx_Vol	The individual channel volume of Melody	NY5+ / NY6 / NY7 / NX1	M_Ch10_Vol = R0

Name	Descriptions	Availability	Example
SPIVol	The channel volume of SPIPlay	NY6B / NY6C / NY7	SPIVol = R0
Px	IO Port	All	PA = R0 ^ 0x9
PxM	Input / Output mode of IO port	NY4 / NY5+ / NY6 / NY7 / NY9T / NX1	PAM = PFM 0x1
PxM.n	Input / Output mode of IO pin	NY4 / NY5+ / NY6 / NY7 / NY9T / NX1	PAM.0 = PFM.0
RandomL	Low nibble of random number Read only, do not use at the left side of "="	All	R0 = RandomL ^ 0x5
RandomH	Hight nibble of random number Read only, do not use at the left side of "="	All	R0 = RandomH 0x9
Random	Random number Read only, do not use at the left side of "="	All	X0 = Random ^ 0x33

5.2.2 Px = data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path when the value of Px is equal to data.

Ex. PA = 3? Path ; Jump to Path if the value of PA is 3.

5.2.3 Px != data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path when the value of Px is not equal to data.

Ex. PA != 3? Path ; Jump to Path if the value of PA is not 3.

5.2.4 Px.n = 0?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path when the value of the nth pin of Px is 0.

Ex. PA.0 = 0? Path ; Jump to Path if the value of PA.0 is 0.

5.2.5 Px.n != 0?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path when the value of the nth pin of Px is not 0.

Ex. PA.0 != 0? Path ; Jump to Path if the value of PA.0 is not 0.

5.2.6 Px.n = 1?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path when the value of the n^{th} pin of Px is 1.

Ex. PA.0 = 1? Path ; Jump to Path if the value of PA.0 is 1.

5.2.7 Px.n != 1?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path when the value of the n^{th} pin of Px is not 1.

Ex. PA.0 != 1? Path ; Jump to Path if the value of PA.0 is not 1

5.2.8 Vol = n?Path

[NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path if currently volume equals to n. n=0~15.

Ex.
Vol=5?True ; Jump to “True” if volume is at level 5.

5.2.9 Vol != n?Path

[NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path if currently volume does not equal to n. n=0~15.

Ex.
Vol!=5?True ; Jump to “True” if volume is not at level 5.

5.2.10 MixCtrl = data?Path

[NY5]

Jump to Path if MixCtrl value is equal to data.

Ex.
MixCtrl=0x2?Set_MixCtrl ; Jump to “Set_MixCtrl” if MixCtrl value is equal to 0x2.
Set_MixCtrl: MixCtrl(0,0,4,0) ; Volume on channel 2 is 100%.

5.2.11 MixCtrl != data?Path

[NY5]

Jump to Path if MixCtrl value is not equal to data.

Ex.

MixCtrl!=0xE?Set_MixCtrl ; Jump to “Set_MixCtrl” if MixCtrl value is not equal to 0xE.
Set_MixCtrl: MixCtrl(1,1,2) ; 1/4 volume on Ch0/Ch1 respectively, and 2/4 volume on Ch2.

5.2.12 RandomL = data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The random number is generated by random number generator, and it can be set on [Option] page. RandomH provides high-nibble and RandomL provides low-nibble.

Jump to Path if the low-nibble of random number generator is equal to data. data=0~15.

Ex.

RandomL=0xA?True ; Jump to “True” if the low-nibble of Random is equal to 0xA.

5.2.13 RandomH = data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The random number is generated by random number generator, and it can be set on [Option] page. RandomH provides high-nibble and RandomL provides low-nibble.

Jump to Path if the high-nibble of random number generator is equal to data. Data=0~15.

Ex.

RandomH=0x3?True ; Jump to “True” if the high-nibble of Random is equal
; to 0x3.

5.2.14 RandomL != data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The random number is generated by random number generator, and it can be set on [Option] page. RandomH provides high-nibble and RandomL provides low-nibble.

Jump to Path if the low-nibble of random number generator is not equal to data. Data=0~15.

Ex.

RandomL!=0xA?True ; Jump to “True” if the low-nibble of Random is not
; equal to 0xA.

5.2.15 RandomH != data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The random number is generated by random number generator, and it can be set on [Option] page. RandomH provides high-nibble and RandomL provides low-nibble.

Jump to Path if the high-nibble of random number generator is not equal to data. Data=0~15.

Ex.

RandomH!=0x3?True ; Jump to “True” if the high-nibble of Random is not
; equal to 0x3.

5.2.16 Random = data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The random number is generated by random number generator, and it can be set on **[Option]** page.

Jump to Path if the number generated by random number generator equals to data.

Random = Data?Path

Random(Max) = Data?Path

Random(Min, Max) = Data?Path

Data: number.

- NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T support 0 ~ 255.
- NX1 supports 0 ~ 0xFFFFFFFF. If Min / Max is specified, Data range is Min ~ Max - 1.

Min: Minimum value of generated random number. 0 if omitted.

Max: Accepts max 0xFFFFFFFF.

Note:

1. **NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T do not support Min / Max parameters.**
2. **In NX1, generated random number is not restricted by Random option if Min / Max is specified.**

Ex. Random=0x3A

Random=0x3A?True ; Jump to “True” if obtained value is equal to 0x3A.

Ex. NX1

Random(0x10, 0x1000)=0x100?True ; Jump to “True” if obtained value is equal to 0x100.

5.2.17 Random != data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The random number is generated by random number generator, and it can be set on **[Option]** page.

Jump to Path if the number generated by random number generator is not equal to data.

Random != Data?Path

Random(Max) != Data?Path

Random(Min, Max) != Data?Path

Data: number.

- NY4 / NY5 / NY6 / NY7 / NY9T support 0 ~ 255.

- NX1 supports 0 ~ 0xFFFFFFFFE. If Min / Max is specified, Data accepts Min ~ Max - 1.

Min: Minimum value of generated random number. 0 if omitted.

Max: Accepts max 0xFFFFFFFFF.

Note:

1. *NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T do not support Min / Max parameters.*
2. *In NX1, generated random number is not restricted by Random option if Min / Max is specified.*

Ex. Random=0x2A

Random!=0x3A?True ; Jump to “True” if obtained value is not equal to 0x3A.

Ex. NX1

Random(0x10, 0x1000)!=0x100?True ; Jump to “True” if obtained value is not equal to 0x100.

5.2.18 Px[1 X 0 X]?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

User can use this command for reading designated IO Pin of Port, and jump to the corresponding path.

Ex. Read PB.1 and PB.3.

TR1: PB[1 X 0 X]?True ; Jump to “True” if PB.3=1 and PB.1=0
True: PlayV(Ch0, \$V0)

5.2.19 ChUsed(Ch)?Path

[NY5+ / NY6 / NY7 / NX1]

Jump to Path if Voice or Melody is playing in the specified channel.

Ch: channel. If Ch is not designated, it is regarded as all channels.

- NY5+ supports Ch= Ch0 ~ Ch3.
- NY6 supports Ch= Ch0 ~ Ch5.
- NY7 supports Ch= Ch0 ~ Ch7.
- NX1 supports Ch= Ch0 ~ Ch7.

Ex.

TR1: PlayV(ch4,\$V0)

TR2: PlayM(\$M0)

TR3: ChUsed(3)?Channel_Playing ; Jump to “Channel_Playing” if channel 3 is in use.

5.2.20 Voice(Ch)?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path if voice is being played on the designated voice channel.

Ch: voice channel. If Ch is not designated, it is regarded as all voice channels.

- NY4 does not support Ch parameter.
- NY5 / NY5+, Ch= Ch0 ~ Ch3.
- NY6, Ch= Ch0 ~ Ch5.
- NY7, Ch= Ch0 ~ Ch7.
- NX1, Ch= Ch0 ~ Ch7.

Note: This command is available only when program uses PlayV or PlayVS.

Ex.

PowerOn: InputState

TR1: PlayV(ch0,\$V0), PlayV(ch1,\$V1), PlayV(ch2,\$V2), PlayV(ch3, \$V3)

TR2: Voice?Voice_Playing ; Jump to “Voice_Playing” if any voice is playing.

TR3: Voice(0)?Ch0_Playing ; Jump to “CH0_Playing” if voice is playing on channel 0.

TR4: Voice(1)?Ch1_Playing ; Jump to “CH1_Playing” if voice is playing on channel1.

TR5: Voice(ch2)?Ch2_Playing ; Jump to “CH2_Playing” if voice is playing on channel 2.

TR6: Voice(ch3)?Ch3_Playing ; Jump to “CH3_Playing” if voice is playing on channel 3.

5.2.21 PauseV(Ch)?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path if voice is been paused on the designated voice channel.

Ch: voice channel. If Ch is not designated, it is regarded as all voice channels.

- NY4 does not support Ch parameter.
- NY5 / NY5+, Ch= Ch0 ~ Ch3.
- NY6, Ch= Ch0 ~ Ch5.
- NY7, Ch= Ch0 ~ Ch7.
- NX1, Ch= Ch0 ~ Ch7.

Note: This command is available only when program uses PlayV, PlayVS or PauseV.

Ex. Any playing voice is paused.

PowerOn: InputState

TR1: PlayV(ch0,\$V0), PlayV(ch1,\$V1), PlayV(ch2,\$V2), PlayV(ch3, \$V3)

TR2: PauseV?VoicePause ; Jump to “VoicePause” when any playing voice is paused.

TR3: PauseV(0)?Ch0_Pause ; Jump to “CH0_Pause” when playing voice on channel 0 is paused.

TR4: PauseV(1)?Ch1_Pause ; Jump to “CH1_Pause” when playing voice on

TR5: PauseV(ch2)?Ch2_Pause ; channel 1 is paused.
; Jump to CH2_Pause” when playing voice on
channel 2 is paused.

TR6: PauseV(ch3)?Ch3_Pause ; Jump to “CH3_Pause” when playing voice on
channel 3 is paused.

5.2.22 Melody?Path

[NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path if melody is being playing.

Ex. Melody is playing.

Melody?True ; Jump to “True” if melody is playing.

5.2.23 PauseM?Path

[NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path if PauseM feature is currently being executed.

Ex. The playing of melody is being paused.

PauseM?True ; Jump to “True” when melody is paused.

5.2.24 Delay(n)?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path if executing the foreground delay feature.

n: 0 = foreground, 1 = background1, 2 = background2 and 3 = background3. If n is not designated, it is regarded as all delays.

Ex. **Delay(0)?True** ; Jump to “True” when foreground delay is executing.

Ex. **Delay?True** ; Jump to “True” when any delay is executing.

5.2.25 PauseD(n)?Path

[NY9T]

Jump to Path if the designated foreground or background is executing pause feature.

n: 0 = foreground, 1 = background1, 2 = background2. If n is not designated, it is regarded as all pauses.

Ex. **PauseD(0)?True_Path** ; Jump to “True_Path” when executing pause in foreground.

Ex. **PauseD?True_Path** ; Jump to “True_Path” when executing pause in all foreground and background.

5.2.26 Action(Ch)?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path if specified action channel is being played.

Ch: action channel, If not designated, it is regarded as all action channels.

- NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T support Ch1 ~ Ch8.
- NX1 supports Ch1 ~ Ch32.

Note: This command is available only when program uses PlayA or PlayAS.

Ex. Check if action channel 1 is being executing.

Action(1)?StopOut ; Jump to “StopOut” when Action channel 1 is being played.

Action?StopOut ; Jump to “StopOut” when any action channel is being played.

5.2.27 PauseA(Ch)?Path

[NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path if specified action channel is executing PauseA feature.

Ch: action channel. If not designated, it is regarded as all action channels.

- NY5+ / NY6 / NY7 / NY9T support Ch1 ~ Ch8.
- NX1 supports Ch1 ~ Ch32.

Note: This command is available only when program uses PlayA or PlayAS.

Ex. Any playing action is paused.

PowerOn: InputState

TR1: PlayA(PC.0,ch1,\$A1), PlayA(PC.0,ch2,\$A1), PlayA(PC.0,ch3,\$A1), PlayA(PC.0,ch4,\$A1)

TR2: PauseA?ActionPause ; Jump to “ActionPause” when any playing action is paused.

TR3: PauseA(1)?Ch1_Pause ; Jump to “CH1_Pause” when playing action on channel 1 is paused.

TR4: PauseA(2)?Ch2_Pause ; Jump to “CH2_Pause” when playing action on channel 2 is paused.

TR5: PauseA(ch3)?Ch3_Pause ; Jump to “CH3_Pause” when playing action on channel 3 is paused.

TR6: PauseA(ch4)?Ch4_Pause ; Jump to “CH4_Pause” when playing action on channel 4 is paused.

5.2.28 PWMIO?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Jump to Path if the designated foreground or background is executing PWM-IO.

PWMIO?Path

PWMIO(n)?Path

PWMIO(Pin)?Path

PWMIO(Ch)?Path

PWMIO(PIO)?Path

n: 0 = foreground, 1 = background1 and 2 = background2. If n is not designated, it is regards as all PlayPWM / PlayPWMS / PWMOut.

Pin: check for PWM-IO on specified pin.

- **Px.n:** IC pin.
- **EXPx_Py.n:** I/O Expander pin.

Ch: PWM-IO channel by PlayPWM / PlayPWMS.

PIO: All PlayPWM / PlayPWMS channels.

Note:

1. **NY4 / NY5 / NY5+ / NY6 / NY7 / NX1 do not support parameter n.**
2. **NY4 / NY5 / NY6 / NY7 / NY9T do not support parameter Pin.**
3. **NY4 / NY5 / NY6 / NY7 / NY9T / NX1 do not support parameter Ch / PIO.**

Ex. PWMIO is playing.

PWMIO?True ; Jump to “True” if PWMIO is playing in all foreground and background.

PWMIO(0)?True_Path ; Jump to “True_Path” when PWM is playing in foreground.

PWMIO(PB.0)?True_Path ; Jump to “True_Path” when PWM-IO is executing on PB.0.

PWMIO(EXP0_P1.2)?True_Path ; Jump to “True_Path” when PWM-IO is executing on P1.2 pin of I/O Expander 0.

5.2.29 PausePWM?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Jump to Path if PausePWM feature is currently being executed.

PausePWM?Path

PausePWM(n)?Path

PausePWM(Pin)?Path

PausePWM(Ch)?Path

PausePWM(PIO)?Path

n: 0 = foreground, 1 = background1 and 2 = background2. If n is not designated, it is regards as all PlayPWM / PlayPWMS / PWMOut.

Pin: check whether PWM-IO on specified pin is paused.

- **Px.n:** IC pin.
- **EXPx_Py.n:** I/O Expander pin.

Ch: PWM-IO channel by PlayPWM / PlayPWMS.

PIO: All PlayPWM / PlayPWMS channels.

Note:

1. **NY4 / NY5 / NY5+ / NY6 / NY7 / NX1 do not support parameter n.**
2. **NY4 / NY5 / NY6 / NY7 / NY9T do not support parameter Pin.**
3. **NY4 / NY5 / NY6 / NY7 / NY9T / NX1 do not support parameter Ch / PIO.**

Ex. The playing of PWMIO is being paused.

PausePWM?True_Path ; Jump to "True" when PWMIO is paused.

5.2.30 HoldPWM(n)?Path

[NY9T]

Jump to Path if the designated foreground or background is executing HoldPWM function.

n: 0=foreground, 1=background1 and 2=background2. If n is not designated, it is regards all HoldPWM.

Ex. HoldPWM(0)?True_Path ; Jump to "True_Path" when executing pause in PlayPWM / PlayPWMS in foreground.

Ex. HoldPWM?True_Path ; Jump to "True_Path" when executing pause in PlayPWM / PlayPWMS of all foreground and background.

5.2.31 SPIPlay(Ch)?Path

[NX1]

Jump to Path is specified audio channel is playing SPI data.

Ch: Audio channel. If omitted, means all audio channels and action channels by SPIPlay.

- NX1 supports Ch0 ~ Ch3 ◦

Note: This command is only available when SPIPlay / SPIPlayS is used.

Ex.

PowerOn: InputState

TR1: SPIPlay(ch0, 0), SPIPlay(ch1, 1), SPIPlay(ch2, 2), SPIPlay(ch3, 3)

TR2: SPIPlay?Voice_Playing ; Jump to "Voice_Playing" if SPIPlay is executing.

TR3: SPIPlay(ch0)?Ch0_Playing ; Jump to "CH0_Playing" when SPIPlay is executing on channel 0.

TR4: SPIPlay(ch1)?Ch1_Playing ; Jump to "CH1_Playing" when SPIPlay is executing on channel 1.

TR5: SPIPlay(ch2)?Ch2_Playing ; Jump to "CH2_Playing" when SPIPlay is executing on channel 2.

TR6: SPIPlay(ch3)?Ch3_Playing ; Jump to "CH3_Playing" when SPIPlay is executing on

channel 3.

5.2.32 SPIPause(Ch)?Path

[NX1]

Jump to Path if specified channel is paused by SPIPause.

Ch: Audio channel. If omitted, means all audio channels and action channels by SPIPlay.

- NX1 supports Ch0 ~ Ch3 ◦

Note: This command is only available when SPIPlay / SPIPlayS / SPIPause is used.

Ex.

PowerOn: InputState

TR1: SPIPlay(ch0,0), SPIPlay(ch1,1), SPIPlay(ch2,2), SPIPlay(ch3, 3)

TR2: SPIPause?VoicePause ; Jump to "VoicePause" if SPIPause is taking effect on any channel.

TR3: SPIPause(ch0)?Ch0_Pause ; Jump to "CH0_Pause" is SPIPause is taking effect on channel 0.

TR4: SPIPause(ch1)?Ch1_Pause ; Jump to "CH1_Pause" is SPIPause is taking effect on channel 1.

TR5: SPIPause(ch2)?Ch2_Pause ; Jump to "CH2_Pause" is SPIPause is taking effect on channel 2.

TR6: SPIPause(ch3)?Ch3_Pause ; Jump to "CH3_Pause" is SPIPause is taking effect on channel 3.

5.2.33 Pause(n)?Path

[NY9T]

Jump to Path if the designated foreground or background is executing play or delay.

n: 0=foreground, 1=background1, 2=background2 and 3=background3. If n is not designated, it is regards all channels.

Ex. Pause(0)?True_Path ; Jump to "True_Path" when executing pause in foreground.

Ex. Pause?True_Path ; Jump to "True_Path" when executing pause in all foreground and background.

5.2.34 Record?Path

[NX1]

Jump to Path when the Record function is executing.

Ex.

Record?True_Path ; Jump to "True_Path" when recording.

5.2.35 KRecord?Path

[NX1]

Jump to Path when the KRecord function is executing.

Note: When KRecord / KRecordS is executed, it must execute InstNoteOn or InstNoteOff once to start the Keyboard Record function.

Ex.

KRecord?True_Path ; Jump to "True_Path" when recording.

5.2.36 Recorded(\$Rec)?Path

[NX1]

Jump to Path when specified Record block contains recorded data.

Ex.

Recorded(\$Rec0)?True_Path ; Jump to "True_Path" if \$Rec0 contains recorded data.

5.2.37 PlayK?Path

[NX1]

Jump to Path when the playback of keyboard is executing.

Ex.

PlayK?True_Path ; Jump to "True_Path" when recording.

5.2.38 EraseR?Path

[NX1]

Jump to Path when the Record Erasing function is executing.

Ex.

EraseR?True_Path ; Jump to "True_Path" when Record section is erasing

5.2.39 VR_VAD?Path

[NX1]

Jump to Path if the volume exceeds VAD threshold.

Ex.

VR_VAD?True_Path ; Jump to “True_Path” if volume exceeds VAD threshold.

5.2.40 LEDStr?Path

[NX1]

Jump to Path if the Single-String LED strip playback is executing.

LEDStr?Path

LEDStr(Px.n)?Path

Px.n: Specify which pin to check, check all pins if omitted.

Ex.

Path1: LEDStr?True_Path ; Jump to “True_Path” if the LED strip playback is playing.

5.2.41 LEDSync?Path

[NX1]

Jump to Path if the Sync-Play LED strip playback is executing.

LEDSync?Path

Ex.

Path1: LEDSync?True_Path ; Jump to “True_Path” if the LED strip playback is playing.

5.2.42 LEDText?Path

[NX1]

Jump to Path if the Scrolling-Text LED strip playback is executing.

LEDStr?Path

Ex.

Path1: LEDText?True_Path ; Jump to “True_Path” if the LED strip playback is playing.

5.2.43 Animaltalks_Record?Path

[NX1]

Jump to Path if animaltalks recording is executing.

Animaltalks_Record?Path

Ex.

Path1: Animaltalks_Record?True_Path ; Jump to “True_Path” if animaltalks recoding is executing.

5.2.44 Animaltalks_Play?Path**[NX1]**

Jump to Path if animaltalks playback is executing.

Animaltalks_Play?Path**Ex.**

Path1: Animaltalks_Play?True_Path ; Jump to “True_Path” if animaltalks playback is executing.

5.2.45 Animalsings_Record?Path**[NX1]**

Jump to Path if animalsings recording is executing.

Animalsings_Record?Path**Ex.**

Path1: Animalsings_Record?True_Path ; Jump to “True_Path” if animalsings recoding is executing.

5.2.46 Animalsings_Play?Path**[NX1]**

Jump to Path if animalsings playback is executing.

Animalsings_Play?Path**Ex.**

Path1: Animalsings_Play?True_Path ; Jump to “True_Path” if animalsings playback is executing.

5.2.47 Calibrate?Path**[NY9T]**

Jump to Path when the Calibrate function is executing.

Ex.

Calibrate?True_Path ; Jump to “True_Path” when AutoJudge_Calibrate or Enforce_Calibrate is executing.

5.2.48 IoExp_Exists?Path**[NY5+ / NX1]**

Check whether I/O Expander with specified ExpID exists, jump to “Path” if exists.

IoExp_Exists(ExpId)?Path

ExpId: I/O Expander ID.

5.2.49 CheckSum?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This feature is for quick inspection on whether or not IC ROM DATA is damaged. CheckSum?Path command will automatically check the correctness of ROM DATA and will jump to “Path” if ROM DATA is correct.

Note:

- 1. While checksum command is being executed, all current activities will be stopped. It will take some time to execute Checksum function, and the system will not response before the completion of execution.*
- 2. In NX1 EF , Checksum?Path will not check whole part of ROM space, that's because that UDR is defined as updateable data, not included for checking.*

Ex.

TEST_ROUTINE: Checksum?OK, PlayV(Ch1,Fail.wav) ; Jump to “OK” if checksum is correct.
Otherwise execute the next command.

OK: PlayV(Ch1, OK.wav)

5.2.50 SPI_CheckSum?Path

[NX1]

This feature is for quick inspection on whether or not SPI Flash data is damaged. SPI_CheckSum?Path command will automatically check the correctness of SPI Flash data and will jump to “Path” if SPI Flash data is correct.

Note: *While SPI CheckSum command is being executed, all current activities will be stopped. It will take some time to execute SPI CheckSum function, and the system will not response before the completion of execution.*

Ex.

TEST_ROUTINE: SPI_CheckSum?OK, PlayV(Ch1, \$Fail) ; Jump to “OK” if SPI checksum
; is correct. Otherwise execute
; the next command.

OK: PlayV(Ch1, \$OK)

5.2.51 !Px[1 X 0 X]?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

User can use this command for reading designated IO Pin of Port, and jump to the corresponding path.

Ex. Read PB.1 and PB.3.

TR1: !PB[1 X 0 X]?False ; Jump to “False” if PB.3!=1 or PB.1!=0
False: PlayV(Ch0, \$V0)

5.2.52 !ChUsed(Ch)?Path

[NY5+ / NY6 / NY7 / NX1]

Jump to Path if Voice or Melody is not playing in the specified channel.

Ch: channel. If Ch is not designated, it is regarded as all channels.

- NY5+ supports 0 ~ 3 or Ch0 ~ Ch3.
- NY6 supports 0 ~ 5 or Ch0 ~ Ch5.
- NY7 supports 0 ~ 7 or Ch0 ~ Ch7.
- NX1 supports 0 ~ 7 or Ch0 ~ Ch7.

Ex.

TR1: PlayV(ch4,\$V0)

TR2: PlayM(\$M0)

TR3: !ChUsed(3)?Channel_NotPlaying ; Jump to “Channel_NotPlaying” if Ch3 is not in use.

5.2.53 !Voice(Ch)?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path if voice is not being played on the designated voice channel.

Ch: voice channel. If Ch is not designated, it is regarded as all voice channels.

- NY4 does not support Ch parameter.
- NY5 / NY5+, Ch=0 ~ 3 or Ch0 ~ Ch3.
- NY6, Ch=0 ~ 5 or Ch0 ~ Ch5.
- NY7, Ch=0 ~ 7 or Ch0 ~ Ch7.
- NX1, Ch=0 ~ 7 or Ch0 ~ Ch7.

Note: This command is available only when program uses PlayV or PlayVS.

Ex.

PowerOn: InputState

TR1: PlayV(ch0,\$V0), PlayV(ch1,\$V1), PlayV(ch2,\$V2), PlayV(ch3, \$V3)

TR2: !Voice?Voice_NotPlaying ; Jump to “Voice_NotPlaying” if any voice is not playing.

TR3: !Voice(0)?Ch0_NotPlaying ; Jump to “Ch0_NotPlaying” if voice is not playing on Ch 0.

TR4: !Voice(1)?Ch1_NotPlaying ; Jump to “Ch1_NotPlaying” if voice is not playing on Ch 1.

TR5: !Voice(ch2)?Ch2_NotPlaying ; Jump to “Ch2_NotPlaying” if voice is not playing on Ch 2.

TR6: !Voice(ch3)?Ch3_NotPlaying ; Jump to “Ch3_NotPlaying” if voice is not playing on Ch 3.

5.2.54 !PauseV(Ch)?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path if voice is not been paused on the designated voice channel.

Ch: voice channel. If Ch is not designated, it is regarded as all voice channels.

- NY4 does not support Ch parameter.
- NY5 / NY5+, Ch=0 ~ 3 or Ch0 ~ Ch3.
- NY6, Ch=0 ~ 5 or Ch0 ~ Ch5.
- NY7, Ch=0 ~ 7 or Ch0 ~ Ch7.
- NX1, Ch=0 ~ 7 or Ch0 ~ Ch7.

Note: This command is available only when program uses PlayV, PlayVS or PauseV.

Ex. Any playing voice is paused.

PowerOn: InputState

TR1: PlayV(Ch0, \$V0), PlayV(Ch1, \$V1), PlayV(Ch2, \$V2), PlayV(Ch3, \$V3)

TR2: !PauseV?VoiceNotPause ; Jump to "VoicePause" when any playing voice is not paused.

TR3: !PauseV(0)?Ch0_NotPause ; Jump to "Ch0_NotPause" when playing voice on channel 0 is not paused.

TR4: !PauseV(1)?Ch1_NotPause ; Jump to "Ch1_NotPause" when playing voice on channel 1 is not paused.

TR5: !PauseV(ch2)?Ch2_NotPause ; Jump to Ch2_NotPause" when playing voice on channel 2 is not paused.

TR6: !PauseV(ch3)?Ch3_NotPause ; Jump to "Ch3_NotPause" when playing voice on channel 3 is not paused.

5.2.55 !Melody?Path

[NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path if melody is not being playing.

Ex. Melody is not playing.

!Melody?False_Path ; Jump to "False_Path" if melody is not playing.

5.2.56 !PauseM?Path

[NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path if PauseM is currently not being executed.

Ex. The playing of melody is being paused.

!PauseM?False_Path ; Jump to "False_Path" if melody is not paused.

5.2.57 !Delay(n)?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path if delay is not executing.

n: 0 = foreground, 1 = background1, 2 = background2 and 3 = background3. If n is not designated, it is regarded as all delays.

Ex. !Delay(0)?False_Path ; Jump to “False_Path” when foreground delay is executing.

Ex. !Delay?False_Path ; Jump to “False_Path” when any delay is executing.

5.2.58 !PauseD(n)?Path

[NY9T]

Jump to Path if the designated foreground or background is not executing PauseD.

n: 0 = foreground, 1 = background1, 2 = background2. If n is not designated, it is regarded as all pauses.

Ex. !PauseD(0)?False_Path ; Jump to “False_Path” if PauseD is not executing in foreground.

Ex. !PauseD?False_Path ; Jump to “False_Path” if PauseD is not executing in all foreground and background.

5.2.59 !Action(Ch)?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path if specified action channel is not being played.

Ch: action channel, If not designated, it is regarded as all action channels.

- NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T support 1 ~ 8 or Ch1 ~ Ch8.
- NX1 supports 1 ~ 20 or Ch1 ~ Ch20.

Note: This command is available only when program uses PlayA or PlayAS.

Ex. Check if action channel 1 is being executing.

!Action(1)?StartOut ; Jump to “StartOut” when Action channel 1 is not being played.

!Action?StartOut ; Jump to “StartOut” when any action channel is not being played.

5.2.60 !PauseA(Ch)?Path

[NY5+ / NY6 / NY7 / NY9T / NX1]

Jump to Path if specified action channel is not executing PauseA.

Ch: action channel. If not designated, it is regarded as all action channels.

- NY5+ / NY6 / NY7 / NY9T support 1 ~ 8 or Ch1 ~ Ch8.
- NX1 supports 1 ~ 20 or Ch1 ~ Ch20.

Note: This command is available only when program uses PlayA or PlayAS.

Ex. Any playing action is paused.

PowerOn: InputState

TR1: PlayA(PC.0,ch1,\$A1), PlayA(PC.0,ch2,\$A1), PlayA(PC.0,ch3,\$A1), PlayA(PC.0,ch4,\$A1)

TR2: !PauseA?ActionNotPause ; Jump to "ActionNotPause" when any playing action is not paused.

TR3: !PauseA(1)?Ch1_NotPause ; Jump to "Ch1_NotPause" when playing action on channel 1 is not paused.

TR4: !PauseA(2)?Ch2_NotPause ; Jump to "Ch2_NotPause" when playing action on channel 2 is not paused.

TR5: !PauseA(ch3)?Ch3_NotPause ; Jump to "Ch3_NotPause" when playing action on channel 3 is not paused.

TR6: !PauseA(ch4)?Ch4_NotPause ; Jump to "Ch4_NotPause" when playing action on channel 4 is not paused.

5.2.61 !PWMIO?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Jump to Path if the designated foreground or background is not executing PWM-IO.

PWMIO?Path

PWMIO(n)?Path

PWMIO(Pin)?Path

PWMIO(Ch)?Path

PWMIO(PIO)?Path

n: 0 = foreground, 1 = background1 and 2 = background2. If n is not designated, it is regards as all PlayPWM / PlayPWMS / PWMOut.

Pin: check for PWM-IO on specified pin.

- **Px.n:** IC pin.
- **EXPx_Py.n:** I/O Expander pin.

Ch: PWM-IO channel by PlayPWM / PlayPWMS.

PIO: All PlayPWM / PlayPWMS channels.

Note:

1. NY4 / NY5 / NY5+ / NY6 / NY7 / NX1 do not support parameter n.
2. NY4 / NY5 / NY6 / NY7 / NY9T do not support parameter Pin.
3. NY4 / NY5 / NY6 / NY7 / NY9T / NX1 do not support parameter Ch / PIO.

Ex. PWMIO is playing.

!PWMIO?False_Path ; Jump to "False_Path" if PWMIO is not playing in all

foreground and background.

!PWMIO(0)?False_Path ; Jump to “False_Path” when PWM is not playing in foreground.

!PWMIO(PB.0)?False_Path ; Jump to “False_Path” when PWM-IO is not executing on PB.0.

!PWMIO(EXP0_P1.2)?False_Path ; Jump to “False_Path” when PWM-IO is not executing on P1.2 pin of I/O Expander 0.

5.2.62 !PausePWM?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Jump to Path if PausePWM is currently not being executed.

!PausePWM?Path

!PausePWM(n)?Path

!PausePWM(Pin)?Path

!PausePWM(Ch)?Path

!PausePWM(PIO)?Path

n: 0 = foreground, 1 = background1 and 2 = background2. If n is not designated, it is regards as all PlayPWM / PlayPWMS / PWMOOut.

Pin: check whether PWM-IO on specified pin is paused.

- **Px.n:** IC pin.
- **EXPx_Py.n:** I/O Expander pin.

Ch: PWM-IO channel by PlayPWM / PlayPWMS.

PIO: All PlayPWM / PlayPWMS channels.

Note:

1. NY4 / NY5 / NY5+ / NY6 / NY7 / NX1 do not support parameter n.
2. NY4 / NY5 / NY6 / NY7 / NY9T do not support parameter Pin.
3. NY4 / NY5 / NY6 / NY7 / NY9T / NX1 do not support parameter Ch / PIO.

Ex. The playing of PWMIO is being paused.

!PausePWM?False_Path ; Jump to “False” when PWMIO is not being paused.

5.2.63 !HoldPWM(n)?Path

[NY9T]

Jump to Path if the designated foreground or background is not executing HoldPWM.

n: 0=foreground, 1=background1 and 2=background2. If n is not designated, it is regards all HoldPWM.

Ex. !HoldPWM(0)?False_Path ; Jump to “False_Path” when PlayPWM / PlayPWMS in foreground is not paused.

Ex. !HoldPWM?False_Path ; Jump to "False_Path" when no PlayPWM / PlayPWMS of all foreground and background is paused.

5.2.64 !SPIPlay(Ch)?Path

[NX1]

Jump to Path is specified audio channel is not playing SPI data.

Ch: Audio channel. If omitted, means all audio channels and action channels by SPIPlay.

- NX1 supports Ch0 ~ Ch3 ◦

Note: This command is only available when SPIPlay / SPIPlayS is used.

Ex.

PowerOn: InputState

TR1: SPIPlay(ch0, 0), SPIPlay(ch1, 1), SPIPlay(ch2, 2), SPIPlay(ch3, 3)

TR2: !SPIPlay?Voice_NotPlaying ; Jump to "Voice_NotPlaying" if no SPIPlay is executing.

TR3: !SPIPlay(ch0)?Ch0_NotPlaying ; Jump to "Ch0_NotPlaying" when no SPIPlay is executing on channel 0.

TR4: !SPIPlay(ch1)?Ch1_NotPlaying ; Jump to "Ch1_NotPlaying" when no SPIPlay is executing on channel 1.

TR5: !SPIPlay(ch2)?Ch2_NotPlaying ; Jump to "Ch2_NotPlaying" when no SPIPlay is executing on channel 2.

TR6: !SPIPlay(ch3)?Ch3_NotPlaying ; Jump to "Ch3_NotPlaying" when no SPIPlay is executing on channel 3.

5.2.65 !SPIPause(Ch)?Path

[NX1]

Jump to Path if specified channel is not paused by SPIPause.

Ch: Audio channel. If omitted, means all audio channels and action channels by SPIPlay.

- NX1 supports Ch0 ~ Ch3 ◦

Note: This command is only available when SPIPlay / SPIPlayS / SPIPause is used.

Ex.

PowerOn: InputState

TR1: SPIPlay(ch0,0), SPIPlay(ch1,1), SPIPlay(ch2,2), SPIPlay(ch3, 3)

TR2: !SPIPause?VoiceNotPause ; Jump to "VoiceNotPause" if SPIPause is not taking effect on any channel.

TR3: !SPIPause(ch0)?Ch0_NotPause ; Jump to "Ch0_NotPause" is SPIPause is not taking effect on channel 0.

TR4: !SPIPause(ch1)?Ch1_NotPause ; Jump to "Ch1_Pause" is SPIPause is not taking effect on channel 1.

TR5: !SPIPause(ch2)?Ch2_NotPause ; Jump to "Ch2_Pause" is SPIPause is not taking effect on channel 2.

TR6: !SPIPause(ch3)?Ch3_NotPause ; Jump to "Ch3_Pause" is SPIPause is not taking effect on channel 3.

5.2.66 !Pause(n)?Path

[NY9T]

Jump to Path if the designated foreground or background is not executing Pause.

n: 0=foreground, 1=background1, 2=background2 and 3=background3. If n is not designated, it is regards all channels.

Ex. !Pause(0)?False_Path ; Jump to "False_Path" if Pause is not executing in foreground.

Ex. !Pause?False_Path ; Jump to "False_Path" if Pause is not executing in all foreground and background.

5.2.67 !Record?Path

[NX1]

Jump to Path when the Record function is not executing.

Ex.

!Record?False_Path ; Jump to "False_Path" when recording is not executing.

5.2.68 !KRecord?Path

[NX1]

Jump to Path when the KRecord function is not executing.

Note: When KRecord / KRecordS is executed, it must execute InstNoteOn or InstNoteOff once to start the Keyboard Record function.

Ex.

!KRecord?False_Path ; Jump to "False_Path" if not recording.

5.2.69 !Recorded(\$Rec)?Path

[NX1]

Jump to Path when specified Record block does not contain recorded data.

Ex.

!Recorded(\$Rec0)?False_Path ; Jump to "False_Path" if \$Rec0 does not contain recorded data.

5.2.70 !PlayK?Path

[NX1]

Jump to Path when the playback of keyboard is not executing.

Ex.

!PlayK?False_Path ; Jump to "False_Path" when keyboard recording is not being played.

5.2.71 !EraseR?Path

[NX1]

Jump to Path when the Record Erasing function is not executing.

Ex.

!EraseR?False_Path ; Jump to "False_Path" when Record section is not erasing

5.2.72 !VR_VAD?Path

[NX1]

Jump to Path if the volume does not exceed VAD threshold.

Ex.

!VR_VAD?False_Path ; Jump to "False_Path" if volume does not exceed VAD threshold.

5.2.73 !LEDStr?Path

[NX1]

Jump to Path if the Single-String playback of LED strip is not executing.

!LEDStr?Path

!LEDStr(Px.n)?Path

Px.n: Specify which pin to check, check all pins if omitted.

Ex.

Path1: !LEDStr?False_Path ; Jump to "False_Path" if the LED strip playback is not playing.

5.2.74 !LEDSync?Path

[NX1]

Jump to Path if the Sync-Play LED strip playback is not executing.

!LEDSync?Path

Ex.

Path1: !LEDSync?False_Path ; Jump to “False_Path” if Sync-Play of LED strip is not playing.

5.2.75 !LEDText?Path

[NX1]

Jump to Path if the Scrolling-Text LED strip playback is not executing.

!LEDStr?Path

Ex.

Path1: !LEDText?False_Path ; Jump to “False_Path” if LED strip is not playing.

5.2.76 !Animaltalks_Record?Path

[NX1]

Jump to Path if animaltalks recording is not executing.

!Animaltalks_Record?Path

Ex.

Path1: !Animaltalks_Record?False_Path ; Jump to “False_Path” if animaltalks recoding is not executing.

5.2.77 !Animaltalks_Play?Path

[NX1]

Jump to Path if animaltalks playback is not executing.

!Animaltalks_Play?Path

Ex.

Path1: !Animaltalks_Play?False_Path ; Jump to “False_Path” if animaltalks playback is not executing.

5.2.78 !Animalsings_Record?Path

[NX1]

Jump to Path if animalsings recording is not executing.

!Animalsings_Record?Path

Ex.

Path1: !Animalsings_Record?False_Path ; Jump to “False_Path” if animalsings recoding is not executing.

5.2.79 !Animalsings_Play?Path

[NX1]

Jump to Path if animalsings playback is not executing.

!Animalsings_Play?Path

Ex.

Path1: !Animalsings_Play?False_Path ; Jump to “False_Path” if animalsings playback is not executing.

5.2.80 !Calibrate?Path

[NY9T]

Jump to Path when the Calibrate function is not executing.

Ex.

!Calibrate?False_Path ; Jump to “False_Path” when AutoJudge_Calibrate or Enforce_Calibrate is not executing.

5.2.81 !IoExp_Exists?Path

[NY5+ / NX1]

Check whether I/O Expander with specified ExpID exists, jump to “Path” if not exists. If ExpID is not specified, all I/O Expander will be checked.

IoExp_Exists?Path

IoExp_Exists(ExpId)?Path

ExpId: I/O Expander ID.

5.2.82 !Checksum?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This feature is for quick inspection on whether or not IC ROM DATA is damaged. CheckSum?Path command will automatically check the correctness of ROM DATA and will jump to “Path” if ROM DATA is incorrect.

Note:

1. While checksum command is being executed, all current activities will be stopped. It will take some time to execute Checksum function, and the system will not response before the

completion of execution.

2. In NX1 EF , !Checksum?Path will not check whole part of ROM space, that's because that UDR is defined as updateable data, not included for checking.

Ex.

TEST_ROUTINE: !Checksum?NotOK, PlayV(Ch1,Pass.wav); Jump to "NotOK" if checksum is incorrect. Otherwise execute the next command.

NotOK: PlayV(Ch1, Fail.wav)

5.2.83 !SPI_CheckSum?Path

[NX1]

This feature is for quick inspection on whether or not SPI Flash data is damaged. SPI_CheckSum?Path command will automatically check the correctness of SPI Flash data and will jump to "Path" if SPI Flash data is incorrect.

Note: While SPI CheckSum command is being executed, all current activities will be stopped. It will take some time to execute SPI CheckSum function, and the system will not response before the completion of execution.

Ex.

TEST_ROUTINE: !SPI_CheckSum?NotOK, PlayV(Ch1, \$Pass); Jump to "NotOK" if SPI ; checksum is incorrect. Otherwise ; execute the next command.

NotOK: PlayV(Ch1, \$Fail)

5.2.84 If-Else

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

If-Else is an alternative to Condition?Path.

Voice?{ PA.0 = 0 }:{ PA.0 = 1 }

is equivalent to following code:

```
If(Voice)
{
    PA.0 = 0
}
Else
{
    PA.0 = 1
}
```

Codes inside If block is executed if Condition is true, or codes inside Else block is executed. Codes behind If-Else statement is executed afterward.

If(Condition) { ... }

If(Condition) { ... } Else { ... }

Condition: Condition.

Ex. Random Number Lottery

[Option]

Random=15

[Voice File]

V1 = win.wav

V2 = draw.wav

V3 = lose.wav

[Path]

P1: R0 = RandomL,

```
    If(R0 > 7)
    {
        If(R0 > 13)
        {
            PlayV(Ch0, $v1)
        }
        Else
        {
            PlayV(Ch0, $v0)
        }
    }
    Else
    {
        PlayV(Ch0, $v3)
    }
```

5.2.85 Switch(Ri) = [Path0, Path1, Path2,... Path15]

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Switch command will switch from current path to corresponding path according to the value of Ri. This path can be a foreground path, a background path, or "X" (indicating no action). If all the paths in switch statement are "X", this switch statement can be skipped. In other words, there is no need of this switch statement. If values are not defined in switch statement, next command after switch will be executed.

Jump to Path0 if the content of Ri is 0, and jump to Path1 if it is 1, and so on.

Ex. R0=3, Switch(R0)=[Path1, Path2, X, Path3], Path4 ; R0=3 so jump to Path3.

Ex. R0=4, Switch(R0)=[Path1, Path2, X, Path3], Path4 ; R0 exceeds the range defined by switch command, so the next command will be executed.

5.2.86 Switch(Px) = [Path0, Path1, Path2,... Path15]

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Similar to Switch(Ri), this switch command will switch from current path to corresponding path according to the value of Px. Jump to Path0 if the content of Px is 0, and jump to Path1 if it is 1, and so on.

Ex. PB =3, Switch(PB)=[Path1, Path2, X, Path3], Path4 ; PB = 3 so jump to Path3.

Ex. PB =4, Switch(PB)=[Path1, Path2, X, Path3], Path4 ; PB exceeds the range defined by switch command, so next command will be executed.

5.2.87 Switch(RandomL) = [Path0, Path1, Path2,... Path15]

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Switch command will switch from current path to corresponding path according to the value of RandomL. This path can be a foreground path, a background path, or the "X" (indicating no action). If this command is undefined in [Path], Q-Code will execute the next command.

Ex. Switch(RandomL)=[Path1, Path2, X, Path3], Path4 ; According the RandomL value, switching to the corresponding path.

5.2.88 Switch(RandomH) = [Path0, Path1, Path2,... Path15]

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Switch command will switch from current path to corresponding path according to the value of RandomH. This path can be a foreground path, a background path, or the "X" (indicating no action). If this command is undefined in [Path], Q-Code will execute the next command.

Ex. Switch(RandomH)=[Path1, Path2, X, Path3], Path4 ; According to RandomH value, switch to the corresponding path.

5.2.89 Switch(Xi) = [Path0, Path1, Path2,..Path255]

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Similar to the 4-bit Switch(Ri), this switch command will switch from current path to corresponding path

according to the value of Xi. This path can be a foreground path, a background path, or the "X" (indicating no action). Jump to Path0 if the content of Xi is 0, and jump to Path1 if it is 1, and so on.

Ex. X0=3, Switch(X0)=[Path1, Path2, X, Path3], Path4 ; X0=3 so jump to Path3.
Ex. X0=4, Switch(X0)=[Path1, Path2, X, Path3], Path4 ; X0=4 exceeds the range defined by switch, so switch command is skipped. The next command will be executed.

5.2.90 Switch(Random) = [Path0, Path1, Path2,... Path255]

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Switch command will switch from current path to corresponding path according to the value of Random. This path can be a foreground path, a background path, or the "X" (indicating no action).

Ex. Random=3, Switch(Random)=[Path1, Path2, X, Path3], Path4 ; Random=3, so jump th Path3.
Ex. Random=4, Switch(Random)=[Path1, Path2, X, Path3], Path4 ; Random=4, exceeds the range defined by Switch, so switch command is skipped. The next command will be executed.

5.2.91 Switch(Px[d x d x]) = [Path0, Path1, Path2,... Path15]

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

User can use this command for reading designated IO Pin of Port and combine the IO status as a value, then jump to the corresponding path based on the value. Switch command will switch from current path to corresponding path according to the value of d. This path can be a foreground path, a background path, or "X" (indicating no action). This is aimed for achieving the function of bonding option.

Ex. Read PB.1 and PB.3, the two bits can have 4 combinations.

TR1: Switch(PB[d x d x]) = [P0, P1, P2, P3] ; Using PB.1 and PB.3 to combine 4 outcomes.
P0: PlayV(Ch0, \$V0) ; When PB.1 and PB.3 are 0, execute P0 path.
P1: PlayV(Ch0, \$V2) ; When PB.1 is 1 and PB.3 is 0, execute P1 path.
P2: PlayV(Ch0, \$V8) ; When PB.1 is 0 and PB.3 is 1, execute P2 path.
P3: PlayV(Ch0, \$V10) ; When PB.1 and PB.3 are 1, execute P3 path.

5.2.92 While

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Execute codes between {...} until Condition becomes false.

Condition is evaluated before the codes in {...} is executed. Therefore the body of the loop will be executed many times or none.

While is executed as following:

Condition is evaluated. If expression is not true, the While terminates and proceed to steps following.

The codes in {...} is executed.

Continue to Step 1.

While can be terminated if path or conditional jump commands is used within {...}.

While(Condition) { ... }

Condition : Condition. Expression used in conditional jump commands can be used here also.

5.2.93 Do-While

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The Do-While statement allows executing codes in {...} until Condition becomes true.

Condition is evaluated after the codes in {...} is executed. Therefore, the body of the loop is always executed at least once.

Do-While is executed as following:

1. The codes in {...} is executed.
2. Next, Condition is evaluated. If expression is not true, the Do-While terminates and proceed to steps following. If Condition is true, the process is repeated, beginning with step 1.

Do-While can be terminated if path or conditional jump commands is used within {...}.

Do { ... } While(Condition)

Condition: Condition. Expression used in conditional jump commands can be used here also.

Ex. Save sum of 1 to 9 to X0.

[Path]

P1: X0 = 0, X1 = 1,

Do

{

X0 = X0 + X1,

X1++

} While(X1 < 10)

5.2.94 For

For loop.

Before entering for loop, Var will be set as Start, and increased or decreased during execution of codes between {...}.

When variable range is specified as Start..End, ending value will be included, while Start..<End, ending value is excluded.

1. 1..10 generates result as 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
2. 1..<10 generates result as 1, 2, 3, 4, 5, 6, 7, 8, 9.
3. 1..10 step 3 generates result as 1, 4, 7, 10.
4. 1..<10 step 3 generates result as 1, 4, 7.
5. 1..10 step -1 generates empty result.

For(Var in [Start..End]) { ... }

For(Var in [Start..<End]) { ... }

For(Var in [Start..End] Step Increment) { ... }

For(Var in [Start..<End] Step Increment) { ... }

Var: Var will be increased or decreased according to Increment during execution of for loop.

Start: Initial value of Var.

End: Ending value of Var.

Increment: Increment is used to increase / decrease value of Var during execution of for loop. Minus number is acceptable. Default is 1 if omitted.

Ex. NY5 / NY5+ / NY6 / NY7 / NX1

[Voice File]

V0 = 00.wav

V1 = 01.wav

V2 = 02.wav

V3 = 03.wav

[Path]

TR1:

For(r0 in [3..1] Step -1)

; Play voice V1 ~ V3 in reverse order, with 0.5s spacing.

```
{
    PlayV(Ch0, r0),
    Delay(0.5s)
}
```

5.3 I/O Command

4-bit I/O Command				
<u>Ri = Px</u>	<u>Ri = PxM</u>	<u>Ri.n = Px.n</u>	<u>PxM = data</u>	<u>PxM = Ri</u>
<u>PxM.n = 0</u>	<u>PxM.n = 1</u>	<u>Px = data</u>	<u>Px = Ri</u>	<u>Px = Py</u>
<u>!Px</u>	<u>!Px.n</u>	<u>Px.n = 0</u>	<u>Px.n = 1</u>	<u>Px.n = Ri.n</u>
<u>Px.n = Py.n</u>	<u>Px = Py + Ri</u>	<u>Px = Py – Ri</u>	<u>Px = Py Ri</u>	<u>Px = Py ^ Ri</u>
<u>Px = Py & Ri</u>	<u>Ri = Px + Ri</u>	<u>Ri = Px – Ri</u>	<u>Ri = Px Ri</u>	<u>Ri = Px ^ Ri</u>
<u>Ri = Px & Ri</u>	<u>Px = Py + data</u>	<u>Px = Py – data</u>	<u>Px = Py data</u>	<u>Px = Py ^ data</u>
<u>Px = Py & data</u>	<u>Ri = Px + data</u>	<u>Ri = Px – data</u>	<u>Ri = Px data</u>	<u>Ri = Px ^ data</u>
<u>Ri = Px & data</u>	<u>Px = [1 x 0 FD An]</u>	<u>Px.n= 1KHz(time)</u>	-	-
8-bit I/O Command				
<u>XiL = Px</u>	<u>XiH = Px</u>	<u>XiL.n = Px.n</u>	<u>XiH.n = Px.n</u>	<u>Xi.n = Px.n</u>
<u>Xi = [Px, Py]</u>	<u>Px = XiL</u>	<u>Px = XiH</u>	<u>Px.n = XiL.n</u>	<u>Px.n = XiH.n</u>
<u>Px.n = Xi.n</u>	<u>[Px, Py] = Xi</u>	-	-	-

Operations to I/O port / pin. Available value of Px is as follow:

- **NY4A:** PA.
- **NY4B:** PA ~ PB.
- **NY5A:** PA ~ PB.
- **NY5B:** PA ~ PD.
- **NY5C:** PA ~ PE.
- **NY5C450x / NY5C520x / NY5C640x / NY5C720x:** PA ~ PF.
- **NY5Q026A:** PA.
- **NY5Q020A / NY5Q040A:** PA ~ PB.
- **NY5Q046A / NY5Q080A / NY5Q160A:** PA ~ PC.
- **NY5Q060A / NY5Q092A / NY5Q172A:** PA ~ PD.
- **NY5Q342A:** PA ~ PE.
- **NY6A:** PA ~ PB.
- **NY6B:** PA ~ PD.
- **NY6C:** PA ~ PF.
- **NY7A:** PA ~ PB.
- **NY7B:** PA ~ PD.
- **NY7C:** PA ~ PF.
- **NY9T001A:** PE.

- **NY9T004A:** PA, PE.
- **NY9T008A:** PA, PB, PE, PF.
- **NY9T016A:** PA ~ PF.
- **NX1:** PA ~ PC.

5.3.1 Ri = Px

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Keep value of Px in Ri.

Ex. PB= 0x3, R0=PB ; R0=0x3

5.3.2 Ri = PxM

[NY4 / NY5+ / NY6 / NY7 / NY9T / NX1]

Keep value of Port Input/Output Mode in Ri.

Ex. PBM= 0x3, R0=PBM ; R0=0x3

5.3.3 Ri.n = Px.n

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Assign the value of nth pin of Px to the nth bit of Ri.

Ex. PB=0x3, R0=0x0, R0=PB ; R0=0x3

5.3.4 PxM = data

[NY4 / NY5+ / NY6 / NY7 / NY9T / NX1]

Set the input and output mode of I/O Port with data.

**Ex. PBM=0x5 ; PB.0 as input, PB.1 as output,
; PB.2 as input, PB.3 as output.**

5.3.5 PxM = Ri

[NY4 / NY5+ / NY6 / NY7 / NY9T / NX1]

Set Input and Output mode of Port with Ri.

Ex. R0=0x3, PBM=R0 ; PBM=0x3

5.3.6 PxM.n = 0

[NY4 / NY5+ / NY6 / NY7 / NY9T / NX1]

Set the n^{th} pin of Port to output mode.

Ex. `PBM.0=0` ; `PB.0` as output.

5.3.7 `PxM.n = 1`

`[ NY4 / NY5+ / NY6 / NY7 / NY9T / NX1 ]`

Set the n^{th} pin of Port to input mode.

Ex. `PBM.0=1` ; `PB.0` as input.

5.3.8 `Px = data`

`[ NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1 ]`

Assign data to Px as output value.

Ex. `PB=0x5` ; `PB=0x5`

5.3.9 `Px = Ri`

`[ NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1 ]`

Assign Ri to Px as output value.

Ex. `R0=0x3, PB=R0` ; `PB=0x3`

5.3.10 `Px = Py`

`[ NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1 ]`

Assign the value of Py to Px as output value.

Ex. `PB=0x3, PA=PB` ; `PA=0x3`

5.3.11 `!Px`

`[ NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1 ]`

Invert to value of Px.

Ex. `PB=0x3, !PB` ; `PB=0xC`

5.3.12 `!Px.n`

`[ NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1 ]`

Invert the value of n^{th} pin of Px.

Ex. `PB=0x3, !PB.0` ; `PB=0x2`

5.3.13 Px.n = 0

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Set the value of nth pin of Px to be 0.

Ex. PB=0xD, PB.3=0 ; PB=0x5

5.3.14 Px.n = 1

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Set the value of nth pin of Px to be 1.

Ex. PB=0x5, PB.3=1 ; PB=0xD

5.3.15 Px.n = Ri.n

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Assign the nth bit of Ri to the nth pin of Px.

Ex. PB=0x0, R0=0x5, PB.2=R0.0 ; PB=0x4

5.3.16 Px.n = Py.n

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Assign the value of the nth pin of Py to the nth pin of Px.

Ex. PB=0x0, PA=0x5, PB.2=PA.0 ; PB=0x4

5.3.17 Px = Py + Ri

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Sum up the values of Py and Ri, and then assign output value to Px.

Ex. PB=0x3, R0=0x4, PB=PB+R0 ; PB=0x7

5.3.18 Px = Py – Ri

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Subtract Ri from Py, and then assign output value to Px.

Ex. PB=0x3, R0=0x2, PB=PB-R0 ; PB=0x1

5.3.19 $Px = Py \mid Ri$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Py OR Ri, and then assign output value to Px.

Ex. PB=0x3, R0=0x4, PB=PB | R0 ; PB=0x7

5.3.20 $Px = Py \wedge Ri$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Py XOR Ri, and then assign output value to Px.

Ex. PB=0x3, R0=0x4, PB=PB^R0 ; PB=0x7

5.3.21 $Px = Py \& Ri$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Py AND Ri, and then assign output value to Px.

Ex. PB=0x3, R0=0x4, PB=PB&R0 ; PB=0x0

5.3.22 $Ri = Px + Rj$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Sum up the values of Px and Rj and keep it in Ri.

Ex. R0=0x3, PB=0x4, R1=PB+R0 ; R1=0x7

5.3.23 $Ri = Px - Rj$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Subtract Rj from Px and keep the result in Ri.

Ex. R0=0x3, PB=0x4, R1=PB-R0 ; R1=0x1

5.3.24 $Ri = Px \mid Rj$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Px OR Rj and keep the result in Ri.

Ex. PB=0x3, R0=0x4, R1=PB|R0 ; R1=0x7

5.3.25 $R_i = P_x \wedge R_j$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

P_x XOR R_j and keep the result in R_i .

Ex. PB=0x3, R0=0x4, R1=PB^R0 ; R1=0x7

5.3.26 $R_i = P_x \& R_j$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

P_x AND R_j and keep the result in R_i .

Ex. PB=0x3, R0=0x4, R1=PB&R0 ; R1=0x0

5.3.27 $P_x = P_y + \text{data}$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Sum up the values of P_y and data, and then assign output value to P_x .

Ex. PB=0x3, PB=PB+0x4 ; PB=0x7

5.3.28 $P_x = P_y - \text{data}$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Subtract data from P_y and keep the result in P_x .

Ex. PB=0x3, PB=PB-0x2 ; PB=0x1

5.3.29 $P_x = P_y | \text{data}$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

P_y OR data and keep the result in P_x .

Ex. PB=0x3, PB=PB | 0x4 ; PB=0x7

5.3.30 $P_x = P_y \wedge \text{data}$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

P_y XOR data and keep the result in P_x .

Ex. PB=0x3, PB=PB^0x4 ; PB=0x7

5.3.31 $P_x = P_y \& \text{data}$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Py AND data, and then assign output value to Px.

Ex. PB=0x3, PB=PB&0x4 ; PB=0x0

5.3.32 Ri = Px + data

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Sum up the values of Px and data and keep it in Ri.

Ex. PB=0x3, R1=PB+0x4 ; R1=0x7

5.3.33 Ri = Px – data

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Subtract data from Px and keep the result in Ri.

Ex. PB=0x3, PB=PB-0x2 ; R1=0x1

5.3.34 Ri = Px | data

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Px OR data and keep the result in Ri.

Ex. PB=0x3, R1=PB | 0x4 ; R1=0x7

5.3.35 Ri = Px ^ data

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Px XOR data and keep the result in Ri.

Ex. PB=0x3, R1=PB^0x4 ; R1=0x7

5.3.36 Ri = Px & data

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Px AND data and keep the result in Ri.

Ex. PB=0x3, R1=PB&0x4 ; R1=0x0

5.3.37 Px = [1 X 0 FD A P Q]

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command is used to control output of Px. Accepted output types are as follow:

	Description	NY4	NY5	NY5+	NY6	NY7	NY9T	NX1
X	Indicating the pin maintains the original output signal.	✓	✓	✓	✓	✓	✓	✓
0	Indicating the pin is with low electric potential output.	✓	✓	✓	✓	✓	✓	✓
1	Indicating the pin is with high electric potential output.	✓	✓	✓	✓	✓	✓	✓
A	Indicating the pin is with Action output (This function is only available for playing entire .vio file).	✓	✓	✓	✓	✓	✓	✗
P	Indicating the pin will output PWM-IO (Only takes effect for PlayPWM(\$MPIOx)).	✗	✗	✓	✗	✗	✗	✗
FD	Indicating the output will change along with volume (Flash with Dynamic).	✓	✓	✓	✓	✓	✗	✓
Q	Indicating each pin will follow the variation of the QIO signal of <i>Quick-IO</i> .	✓	✓	✓	✗	✗	✗	✓

Ex. Set PB=[1 1 1 1] to let the output of PB be 15.

Ex. Set PB=[0 1 0 1] to let the output of PB be 5.

Ex. Set PB=[X 1 0 1] to change the value of the three LSB bits to 5, and keep the MSB value.

Ex. Set PB=[FD X X X] to let the output of PB.3 flash dynamically with volume.

Ex. Set PB=[Q X X X] to let the output of PB.3 accord with the Quick-IO signal.

Ex. Set IO command to play the whole .vio.

[Action File]

VIO0 = Action1.vio

; Action Label: A1 ~ A8.

VIO1 = Action2.vio

; Action Label: A1 ~ A10.

[Action]

Compression = Enable

FrameRate = 80

Multi_Pins = [PA.0, PA.1, PA.2, PA.3, PB.0, PB.1, PB.2, PB.3]

MVIO0 = [VIO0.A1, VIO0.A2, VIO0.A3, VIO0.A4, VIO0.A5, VIO0.A6, VIO0.A7, VIO0.A8]

; MVIO0 uses PA.0 ~ PB.3 to output A1 ~ A8 signals of VIO0.

MVIO1 = [VIO1.A3, VIO1.A4, VIO1.A5, VIO1.A6, VIO1.A7, VIO1.A8, VIO1.A9, VIO1.A10]

; MVIO1 uses PA.0 ~ PB.3 to output A3 ~ A8 signals of VIO1.

[Path]

PowerOn: PA=[A A A A], PB=[A A A A]

; Set PA and PB to output action.

Path1: PlayA(Ch4, \$MVIO0) ; Output MVIO0 on action channel 4.

Path2: PlayA(Ch4, \$MVIO1) ; Output MVIO1 on action channel 4.

5.3.38 [Px.n ...] = [1 0 X Q]

[NX1]

This command is used to control output of each specified pins. Accepted output types are as follow:

参数	説明
X	Indicating the pin maintains the original output signal.
0	Indicating the pin is with low electric potential output.
1	Indicating the pin is with high electric potential output.
FD	Indicating the output will change along with volume (Flash with Dynamic).
Q	Indicating each pin will follow the variation of the QIO signal of Quick-IO.

5.3.39 Px.n = 1KHz(time)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Px.n outputs square wave at 1kHz for seconds.

Time: the range is 0~15, in the unit of second, If Time=0, it outputs in loop.

Note: While this command is being executed, the system will not response to actions until this command execution is finished.

Ex. PB.3=1KHz(5) or PB.3=1KHz(5s) ; PB.3 outputs square wave at 1kHz for 5 seconds.

5.3.40 XiL = Px

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Keep the value of Px in the low-nibble of Xi.

Ex. PB=0x5, X0L = PB ; X0L = 0x5

5.3.41 XiH = Px

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Keep the value of Px in the high-nibble of Xi.

Ex. PB=0x5, X0H = PB ; X0H = 0x5

5.3.42 $XiL.n = Px.n$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Assign the n^{th} pin value of Px to the n^{th} bit of XiL.

Ex. PB=0x5, X0=0x0, X0L.2 = PB.0 ; X0L = 0x4

5.3.43 $XiH.n = Px.n$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Assign the n^{th} pin value of Px to the n^{th} bit of XiH.

Ex. PB=0x5, X0=0x0, X0H.2 = PB.0 ; XiH = 0x4

5.3.44 $Xi.n = Px.n$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Assign the n^{th} pin value of Px to the n^{th} bit of Xi.

Ex. PB=0x5, X0=0x0, X0.2 = PB.0 ; X0 = 0x4

5.3.45 $Xi = [Px, Py]$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Keep the value of Px in the high-nibble of Xi, and keep the value of Py to the low-nibble of Xi.

Ex. X0 = [PB, PA] ; X0H = PB, X0L = PA

5.3.46 $Px = XiL$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Assign the low-nibble of Xi to be the output value of Px.

Ex. X0L=0x3, PB=X0L ; PB = 0x3

5.3.47 $Px = XiH$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Assign the high-nibble of Xi to be the output value of Px.

Ex. X0H=0x3, PB=X0H ; PB = 0x3

5.3.48 $Px.n = XiL.n$

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Assign the n^{th} bit of XiL to the n^{th} bit of Px.

Ex. PB=0x0, X0=0x5, PB.2=X0L.0 ; PB = 0x4

5.3.49 Px.n = XiH.n

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Assign the n^{th} bit of XiH to the n^{th} pin of Px.

Ex. PB=0x0, X0=0x50, PB.2=X0H.0 ; PB = 0x4

5.3.50 Px.n = Xi.n

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Assign the n^{th} bit of Xi to the n^{th} pin of Px.

Ex. PB=0x0, X0=0x5, PB.2=X0.0 ; PB = 0x4

5.3.51 [Px, Py] = Xi

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Assign the output value of Px to be the high-nibble of Xi; Assign the output value of Py to be the low-nibble of Xi.

Ex. X0H=0x3, X0L=0x1, [PB, PA]=X0 ; PB = 0x3, PA = 0x1

5.4 Path Command

Path Command				
ASM	C-Code	BG	BreakFG	StopFG
StopBG	StopBG1	StopBG2	StopBG3	StopBG4
StopBG5	Subroutine	Label(Pathname)	Macro	1ms_RET
4ms_RET	-	-	-	-

5.4.1 ASM

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

User can insert assembly language module defined in [ASM] in Q-Code program.

Ex. Use ASM to control IC.

[ASM]

LED

```
{
mvla    b'0101'
mpg     pa_page
mvam    pa_state
mvam    pa
}
```

[Path]

TR1: LED ; Call assembly.

5.4.2 C-Code

[NX1]

User can insert C module after the label of **[C-Code]** in Q-Code.

Ex. Uses C programming language to control IC.

[C-Code]

```
//define USER_DEBUG_MODE 1
#ifdef USER_DEBUG_MODE
void debug_print(int data)
{
    // do something
}
#else
#define debug_print(x)
#endif
```

[Path]

PowerOn: debug_print(0)

If the XIP (Execute in place) function is enabled, the C-Code can also specify that some programs are stored in the XIP section. And add XIP attribute to the function prototype declaration.

Ex. Specify that the function is stored in the XIP section.

[C-Code]

```
void func1(void) __XIP;
void func1(void)
{
    // do something
}
```

In addition to using the __XIP attribute directly, user can also know whether the XIP (Execute in place)

function is enabled by the pre-defined `_SPI_XIP` macro. If the XIP function is enabled, the `_SPI_XIP` macro will be 1, otherwise will be 0. It is recommended to use the following example when programming C-Code to refer to the `_SPI_XIP` macro and to define the XIP attribute separately. Therefore, when the XIP (Execute in place) function state of `Q-Code_NX1` changes, the C-Code does not require additional fixes. When users define macros, please avoid starting with the underline and avoid conflicts with the `Q-Code_NX1` Firmware kernel function.

Ex. Reference the `_SPI_XIP` macros.

[C-Code]

```
#if _SPI_XIP == 1
#  define XIP __XIP
#else
#  define XIP
#endif
void func1(void) XIP;
void func1(void)
{
    // do something
}
```

5.4.3 BG

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command is for user to execute command in background. *Q-Code* provides 3 threads of background, and user can start a background thread in another one.

BG(BG1)

BG(BG1, BG2)

BG(BG1, BG2, BG3)

BG(BG1, BG2, BG3, BG4)

BG(BG1, BG2, BG3, BG4, BG5)

BG1: Background 1 path.

- Path name defined in **[Background1]**.
- **X**: does not change the current status of path.
- **OFF**: stop background 1 path.

BG2: Background 2 path.

- Path name defined in **[Background2]**.
- **X**: does not change the current status of path.
- **OFF**: stop background 2 path.

BG3: Background 3 path.

- Path name defined in [**Background3**].
- **X**: does not change the current status of path.
- **OFF**: stop background 3 path.

BG4: Background 4 path.

- Path name defined in [**Background4**].
- **X**: does not change the current status of path.
- **OFF**: stop background 4 path.

BG5: Background 5 path.

- Path name defined in [**Background5**].
- **X**: does not change the current status of path.
- **OFF**: stop background 5 path.

Note:

1. **NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T only support BG1 / BG2.**
2. **NX1 supports BG1 / BG2 / BG3 / BG4 / BG5.**
3. **BG1 can only call the paths defined in [Background1], and vice versa.**

Ex. Use background command to achieve 3Hz flashing.

[**Path**]

PowerOn: BG(OUT1,OUT2), Delay(1), Stop ; execute next command after calling background1 and background2 path.

[**Background1**]

OUT1: PB=[X X 1 1], Delay (0.166), PB=[X X 0 0], Delay (0.166), OUT1

[**Background2**]

OUT2: PB=[0 0 X X], Delay (0.166), PB=[1 1 X X], Delay (0.166), OUT2

5.4.4 BreakFG

[**NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1**]

This command is for user to immediately stop current PlayV, PlayM or Delay commands in foreground, and execute the next command.

Ex. V0 will stop playing after 1 second, and immediately execute StopV command.

[**Path**]

TR1: PlayV(Ch1, \$V0)&[BG1], StopV

[**Background1**]

BG1: Delay(1), BreakFG

5.4.5 StopFG

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Stop currently executing steps in foreground.

Ex. Stop steps executing in foreground using StopFG command.

[Path]

PowerOn:

TR1: BG1, Label(Pathname), PB=[X X 1 1], Delay (0.166), PB=[X X 0 0], Delay (0.166), Pathname

[Background1]

BG1: Delay(10), StopFG ; Stop executing steps in foreground after delay 10 seconds in background1.

5.4.6 StopBG

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Stop currently executing background threads (background1 thread, background2 thread and background3 thread).

Ex. Stop steps executing in background using StopBG command.

[Path]

PowerOn: BG(OUT1,OUT2)

TR1: StopBG

[Background1]

OUT1: PB=[X X 1 1], Delay (0.166), PB=[X X 0 0], Delay (0.166), OUT1

[Background2]

OUT2: PB=[0 0 X X], Delay (0.166), PB=[1 1 X X], Delay (0.166), OUT2

5.4.7 StopBG1

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Stop currently executing step in background1.

Ex. Stop executing background1 step using StopBG1 command.

[Path]

PowerOn: BG(OUT1,OUT2)

TR1: StopBG1

[Background1]

OUT1: PB=[X X 1 1], Delay (0.166), PB=[X X 0 0], Delay (0.166), OUT1

[Background2]

OUT2: PB=[0 0 X X], Delay (0.166), PB=[1 1 X X], Delay (0.166), OUT2

5.4.8 StopBG2

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Stop currently executing step in background2.

Ex. Stop executing background2 step using StopBg2 command.

[Path]

PowerOn: BG(OUT1,OUT2)

TR1: StopBG2

[Background1]

OUT1: PB=[X X 1 1], Delay(0.166), PB=[X X 0 0], Delay(0.166), OUT1

[Background2]

OUT2: PB=[0 0 X X], Delay(0.166), PB=[1 1 X X], Delay(0.166), OUT2

5.4.9 StopBG3

[NX1]

Stop currently executing step in background3.

Ex. Stop executing background3 step by using StopBg3 command, the actions of background1 and Background2 continue.

[Path]

PowerOn: BG(OUT1,OUT2,OUT3)

TR1: StopBG3

[Background1]

OUT1: PB=[X X 1 1], Delay(0.166), PB=[X X 0 0], Delay(0.166), OUT1

[Background2]

OUT2: PB=[0 0 X X], Delay(0.333), PB=[1 1 X X], Delay(0.333), OUT2

[Background3]

OUT3: PB=[0 0 X X X X], Delay(0.5), PB=[1 1 X X X X], Delay(0.5), OUT3

5.4.10 StopBG4

[NX1]

Stop currently executing step in background4.

5.4.11 StopBG5

[NX1]

Stop currently executing step in background5.

5.4.12 Subroutine

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

User can name and define a snippet of Q-Code command in **[Subroutine]**. A named snippet is a subroutine. User can place subroutine name in Q-Code program. Q-Code compiler will treat subroutine name as function call at compiling time.

Ex. Use subroutine name when defining a path.

[Subroutine]

SUB:PA=0xF, DELAY(0.5), PA=0x0, DELAY(0.5), PlayV(Ch1, \$V1)

[Path]

TR1:SUB, PlayV(Ch1, \$V2) ; After executing "SUB", execute PlayV(Ch1, \$V2).

5.4.13 Label(Pathname)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

User could use Label instead of ": " to set the path name for reducing the program segment in program.

Ex.

TR1: PlayV(Ch1, \$V0), Label(Pathname), PlayV(Ch1, \$V1) , Pathname

; After playing V0, play \$V1 repeatedly.

The path above is equal to the following:

TR1: PlayV(Ch1, \$V0), Label_Loop ; Play V0, and jump to "Label_Loop".

Label_Loop : PlayV(Ch1, \$V1) , Label_Loop ; Play \$V1 repeatedly.

5.4.14 Macro

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

User can name and define a snippet of Q-Code command in **[Macro]**. A named snippet of commands is a macro. User can place macro name in Q-Code program. Q-Code compiler will expand the content of macro whenever macro name is met at compiling time.

Ex. Use macro name when defining a path.

[Macro]

MAC: PA=0xF, Delay(0.166)

[Path]

TR1: MAC, PlayV(Ch1, \$V1) ; After executing "MAC" macro, execute PlayV(\$V1).

5.4.15 1ms_RET**[NY9T]**

Return to 1ms path.

Note:

1. After executing 1ms path, must execute 1ms_RET command. Otherwise, there will be unexpected system error.
2. If TimeBase=4ms, 1ms path will not be executed.

Ex.**[Option]**

TimeBase = 1ms

[Path]

PowerOn: Delay(10), PA=0

1ms: !PA, 1MS_RET ; After executing "!PA", return to the main program when executing 1MS_RET.

5.4.16 4ms_RET**[NY9T]**

Return to 4ms path.

Note: After executing 4ms path, must execute 4ms_RET command. Otherwise, there will be unexpected system error.

Ex.**[Option]**

TimeBase = 1ms or 4ms

[Path]

PowerOn: Delay(10), PA=0

4ms: !PA, 4MS_RET ; After executing "!PA", return to the main program when executing 4MS_RET.

5.5 Voice Command

Voice Command				
PlayV	PlayVS	Voicename	WaitVN(Ch)	PauseV(Ch)
ResumeV(Ch)	StopV(Ch)	FreqCH=nK	V_Chx_Freq=nK	V_Chx_Vol = n
V_Chx_Vol = Xi	Xi = V_Chx_Vol	SBC_Loop_On	SBC_Loop_Off	ADPCM_Loop_On
ADPCM_Loop_Off	ADPCM_UpSampling		ADM_Loop_On	ADM_Loop_Off
ADM_UpSampling	PCM_Loop_On	PCM_Loop_Off	ReadFileCountV	-

5.5.1 PlayV / PlayVS

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

This command provides user with simple way to play voice files.

The format of command is shown as the followings:

PlayV{Ch,} **Voice**{, **PlaySpeed**}

PlayV{Ch,} **Voice**{, **Storage**}

PlayVS{Ch,} **Voice**{, **PlaySpeed**}

PlayVS{Ch,} **Voice**{, **Storage**}

Voicename{@CH}

PlayV: execute next command after finishing playing voice file.

PlayVS: execute next command after starting playing voice file.

Ch: assign voice channel number for playing voice (0 represents the first channel).

- NY4 series does not support Ch parameter.
- NY5 / NY5+ supports the voice channel Ch0 ~ Ch3 or 0 ~ 3.
- NY6 supports the voice channel Ch0 ~ Ch5 or 0 ~ 5.
- NY7 supports the voice channel Ch0 ~ Ch7 or 0 ~ 7.
- NX1 supports the voice channel Ch0 ~ Ch7 or 0 ~ 7.

Voice: Voice file to be played.

- Voice file label.
- Combination of Ri (supports 1 ~ 4 Ri) or Xi (supports 1 ~ 2 Xi). Indicate the nth voice file to be played. If Ri / Xi with “++”, the value of Ri / Xi will increase by 1 after playback.
- Recording section defined in **[Record]**.

PlaySpeed: the play speed to play specified voice file. If omitted, voice file will be played as the sample rate of file. PlaySpeed accepts frequency or Xi, range 4k ~ 44.1k.

- NY4 / 5 assign playback speed via Ri / Xi command. For more detail, please refer to [Sample Rate Table](#) for playback speed.
- NY5+ / NY6 / NY7 use the parameter of Ri / Xi as the playback speed, R0 = 0xA, the playback

speed is 10k.

- NX1 does not support PlaySpeed parameter.

Storage: Location of voice file to be played. Internal Storage if omitted.

- NY4 / NY5 / NY5+ / NY6 / NY7 do not support Storage parameter.
- NX1 supports SPI0 / SPI1 / UDR.

Note: For NY4 / NY5, using Voicename{*n} commands, the system will not have default sampling rate and will play voice according to the sampling rate of previous voice. Therefore, if the current voice has different sampling rate from the previous voice, it may lead to inappropriate sampling rate for current voice.

Ex. Specify voice to be played.

P1: PlayV(Ch1,\$V1,6k)/PlayVS(Ch1,\$V1,6k) ; Play voice by voice file label.

P2: PlayV(ch0, X1:X0, 10k), PB.3=1 ; Execute PB.3=1 after playing the voice file specified by register value in voice channel 0, and the playback frequency is 10k (If X1=0x01, X0=0xF2, then play no.V498 voice file).

P3: PlayV(ch0, R3:R2:R1:R0, 10k), PB.3=1 ; Execute PB.3=1 after playing the voice file specified by register value in voice channel 0, and the playback frequency is 10k (If R3=0x0, R2=0x3, R1=0xF, R0=0x1, then play no.V1009 voice file).

P4: PlayV(ch0, X1:X0++,10k), PB.3=1 ; Execute PB.3=1 after playing the voice file specified by register value in voice channel 0, and the playback frequency is 10k (If X1=0x01, X0=0xF2, then play no.V498 voice file, X0 automatically increases 1, X0=0xF3).

P5: PlayVS(ch0, X1:X0, 10k), PB.3=1 ; Play the voice file specified by register value in voice channel 0, the playback frequency is 10k, and simultaneously execute PB.3=1 (If X1=0x01, X0=0xF2, then play no.V498 voice file).

P6: PlayVS(ch0, R3:R2:R1:R0, 10k), PB.3=1 ; Play the specified voice file by Register content value in voice channel 0, the playback frequency is 10k, and simultaneously execute PB.3=1. (If R3=0x0, R2=0x3, R1=0xF, R0=0x1, then play no.V1009 voice file.)

P7: PlayVS(ch0, X1:X0++,10k), PB.3=1 ; Play the specified voice file by Register content value in voice channel 0, the playback frequency is 10k, and simultaneously execute PB.3=1. (If X1=0x01, X0=0xF2, then play no.V498 voice file, X0 automatically increases 1, X0=0xF3.)

Ex. Specify play speed.

P1: PlayV(Ch1, \$V0,6.35k) ; Play voice with play speed 6.35kHz.
P2: PlayV(Ch1, \$V0) / PlayVS(Ch1, \$V1) ; Use voice file sample rate as play speed.
P3: PlayV(Ch1, \$V0, R1:R0) ; Use R1 and R0 to specify play speed.
P4: PlayV(Ch1, \$V0 ,X0) / PlayVS(Ch1, \$V0,X0) ; Use Xi to specify play speed.

Ex. Repeat.

P1: PlayV(Ch1, \$V0) *3 ; Play voice file for 3 times.

Ex. Execute background path while playing voice.

P1: PlayV(Ch1, \$V0)&[BG1,BG2] / PlayVS(Ch1, \$V1)&[BG1,BG2]
; Execute BG1 and BG2 while playing voice.
P2: PlayV(Ch1, \$V0)&[BG1,X] / PlayVS(Ch1, \$V0)&[BG1,X]
; Execute BG1 while playing voice, but keep background path not changed.
P3: PlayV(Ch1, \$V0)&[OFF,X] / PlayVS(Ch1, \$V0)&[OFF,X]
; Stop Background1 while playing voice, and keep background2 not changed.

Ex. Play voice and output QIO according to [QIO].

[Voice File]

V1 = Qio.nyq ; Add voice file.

[Output State]

Output_0: Q Q Q Q Q Q Q Q ; Use PA and PB as QIO output.

[QIO]

MQIO = [PA.0:Q9, PA.1:Q10, PA.2:Q11, PA.3:Q12, PB.0:Q13, PB.1:Q14, PB.2:Q15, PB.3:Q16]
; PA and PB outputs Q9 ~ Q16.

[Path]

Path1: Output_0, PlayV(Ch0, \$V1) ; Play voice V1 and output QIO signal according to MQIO.

Ex. Play voice file on SPI Flash.

[Variable]

Var8: R0

[Memory]

SPI0_File = Spi.spiprj

[Path]

PowerOn: R0 = 0

Path1: PlayV(Ch0, R0++, SPI0) ; Play nth voice file on SPI Flash, which n is specified by R0.

5.5.1.1 PlayV / PlayVS with Table

Ex.

[Table]

Tab:

```
{
[0x001, 0x003, 0x005, 0x007, 0x009],
[0x004, 0x005, 0x3F6, 0x008, 0x010],
[0x000, 0x001, 0x002, 0x003, 0x0F4]
}
```

[Path]

TR1: TableL(Tab,1,1,R0), PlayV(ch0, R0), PB.3=1

; Store the specified address value by Table to R0, execute PB.3=1 after playing voice file specified by R0 in voice channel 0 (R0=0x5, plays no.V5 voice file).

TR2: TableL(Tab,1,1,X0), PlayV(ch0, X0), PB.3=1

; Store the specified address value by Table to X0, execute PB.3=1 after playing voice file specified by X0 in voice channel 0 (X0=0x05, play no.V5 voice file).

TR3: TableL(Tab,1,1,X0), PlayV(ch0, \$V0, X0), PB.3=1

; Store the specified address value by Table to X0, execute PB.3=1 after playing voice file specified by V0 in voice channel 0. (X0=0x05, the playback frequency is 5k).

TR4: TableL(Tab,1,1,R0), PlayVS(ch0, R0), PB.3=1

; Store the specified address value by Table to R0, play the voice file specified by R0 in voice channel 0, and simultaneously execute PB.3=1 (R0=0x5, play no.V5 voice file).

TR5: TableL(Tab,1,1,X0), PlayVS(ch0, X0), PB.3=1

; Store the specified address value by Table to X0, play voice file specified by X0 in voice channel 0, and simultaneously execute PB.3=1 (X0=0x05, play no.V5 voice file).

TR6: TableL(Tab,1,1,X0), PlayVS(ch0, \$V0, X0), PB.3=1

; Store the specified address value by Table to X0, play no.V0 voice file

in voice channel 0, the playback frequency is X0, and simultaneously execute PB.3=1. (X0=0x05, the playback frequency is 5k).

5.5.2 WaitVN(Ch)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

If voice file is being played, it will execute next command after finishing playing Voice file. Otherwise, it will immediately execute the next command.

Ch: voice channel to wait, wait voice playing in all voice channels if omitted.

- NY4 does not support Ch parameter.
- NY5 / NY5+ support the voice channel Ch0 ~ Ch3.
- NY6 supports the voice channel Ch0 ~ Ch5.
- NY7 supports the voice channel Ch0 ~ Ch7.
- NX1 supports the audio channel Ch0 ~ Ch3.

Note: PlayV = PlayVS + WaitVN.

Ex.

[Path]

TR1: PlayV(CH1,\$V0)

TR2: PlayVS(CH1, \$V0), WaitVN(1) ; The result is the same as TR1.

5.5.3 PauseV(Ch)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

This command can be used to pause Voice file which currently being played in designated channel.

Ch: voice channel to pause, pause all voice playing if omitted.

- NY4 does not support Ch parameter.
- NY5 / NY5+ support the voice channel Ch0 ~ Ch3.
- NY6 supports the voice channel Ch0 ~ Ch5.
- NY7 supports the voice channel Ch0 ~ Ch7.
- NX1 supports the audio channel Ch0 ~ Ch3.

Ex.

PowerOn: InputState

TR1: PlayV(Ch0,\$V0), PlayV(Ch1,\$V1), PlayV(Ch2,\$V2), PlayV(Ch3, \$V3)

TR2: PauseV ; Pause all voice playing.

TR3: ResumeV ; Resume all voice playing.

TR4: PauseV(0) ; Pause voice playing in voice channel 0.

TR5: ResumeV(0)	; Resume voice playing in voice channel 0.
TR6: PauseV(1)	; Pause voice playing in voice channel 1.
TR7: ResumeV(1)	; Resume voice playing in voice channel 1.
TR8: PauseV(2)	; Pause voice playing in voice channel 2.
TR9: ResumeV(2)	; Resume voice playing in voice channel 2.
TR10: PauseV(3)	; Pause voice playing in voice channel 3.
TR11: ResumeV(3)	; Resume voice playing in voice channel 3.

5.5.4 ResumeV(Ch)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

This command can be used to resume the paused playback of Voice file.

Ch: voice channel to resume, resume all voice playing if omitted.

- NY4 does not support Ch parameter.
- NY5 / NY5+ support the voice channel Ch0 ~ Ch3.
- NY6 supports the voice channel Ch0 ~ Ch5.
- NY7 supports the voice channel Ch0 ~ Ch7.
- NX1 supports the audio channel Ch0 ~ Ch3.

This command has no effect for the following conditions:

1. The playback of voice data is finished.
2. StopV or Stop command has been executed.

Ex.

PowerOn: InputState

TR1: PlayV(ch0,\$V0), PlayV(ch1,\$V1), PlayV(ch2,\$V2), PlayV(ch3, \$V3)

TR2: PauseV	; Pause all voice playing.
TR3: ResumeV	; Resume all voice playing.
TR4: PauseV(0)	; Pause voice playing in voice channel 0.
TR5: ResumeV(0)	; Resume voice playing in voice channel 0.
TR6: PauseV(1)	; Puase voice playing in voice channel 1.
TR7: ResumeV(1)	; Resume voice playing in voice channel 1.
TR8: PauseV(2)	; Pause voice playing in voice channel 2.
TR9: ResumeV(2)	; Resume voice playing in voice channel 2.
TR10: PauseV(3)	; Pause voice playing in voice channel 3.
TR11: ResumeV(3)	; Resume voice playing in voice channel 3.

5.5.5 StopV(Ch)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

The command is to stop voice playing.

Ch: voice channel to stop, stop all voice playing if omitted.

- NY4 does not support Ch parameter.
- NY5 / NY5+ support the voice channel Ch0 ~ Ch3.
- NY6 supports the voice channel Ch0 ~ Ch5.
- NY7 supports the voice channel Ch0 ~ Ch7.
- NX1 supports the audio channel Ch0 ~ Ch3.

Ex.

PowerOn: InputState

TR1: PlayV(ch0, \$V0), PlayV(ch1, \$V1), PlayV(ch2, \$V2), PlayV(ch3, \$V3)

TR3: StopV(0) ; Stop voice file playing in channel 0.

TR4: StopV(1) ; Stop voice file playing in channel 1.

TR5: StopV(2) ; Stop voice file playing in channel 2.

TR6: StopV(3) ; Stop voice file playing in channel 3.

TR7: StopV ; Stop voice file playing in all channels.

5.5.6 FreqCH = nK

[NY4 / NY5]

This command can only be used during direct voice playback with voice label or filename, and it cannot be used with PlayV command. To use this command, the playback speed command Freq=nK must be issued once before play command, and then the following play command will use the designated playback speed. (n=4K~44.1K)

CH: Specify the voice channel.

- NY4 does not support CH parameter.
- NY5 / NY5+ support the voice channel Ch0 ~ Ch3.

n: playback speed. n=4K~44.1K (Refer to [Sample Rate Table](#)).

Ex. Freq=10k, \$V0, \$V0

5.5.7 96V_Chx_Freq = nK

[NY5+]

Change sampling rate of playing voice on specified channel. Sampling rate accepts 4K~44.1K. This change will not be retained for the next voice playback.

Chx: Voice channel.

- NY5+ supports Ch0 ~ Ch3.

Note: This command has no effect when voice is not playing on specified channel.

Ex. V_Ch0_Freq = 10K ; Change sample rate on voice channel 0 to 10K.

5.5.8 V_Chx_Vol = n / V_Chx_Vol = Xi

[NY5+ / NY6 / NY7]

Modify the channel volume of the playing voice. The range of channel volume is 0~255, which can be immediate value or controlled by Xi. n=0 means the voice channel is mute, while n=255 means keep the voice channel volume as maximum.

Chx: Specify the voice channel.

- NY5+ supports Ch0 ~ Ch3.
- NY6 supports Ch0 ~ Ch5.
- NY7 supports Ch0 ~ Ch7.

Note: This command is not valid when the specified voice channel is not playing voice.

Ex. V_Ch0_Vol = 200 ; Set the volume of Ch0 as 200, 78% to the original.

Ex. V_Ch1_Vol = X0 ; Set the volume of Ch1 by the value of X0.

5.5.9 Xi = V_Chx_Vol

[NY5+ / NY6 / NY7]

Access the channel volume of speech being played.

Chx: Specify the voice channel.

- NY5+ supports Ch0 ~ Ch3.
- NY6 supports Ch0 ~ Ch5.
- NY7 supports Ch0 ~ Ch7.

Ex. X0 = V_Ch0_Vol ; Read the volume of voice channel 0 and save the result to X0.

5.5.10 SBC_Loop_On

[NX1]

Turn on the SBC loop.

Note:

1. When turns on the loop function, SBC-1 and SBC-2 only provides a single channel for loop.
2. This command is deprecated since Q-Code 8.20, its recommended to replace with [Audio Loop On](#).

Ex.

PowerOn: InputState

TR1: PlayV(ch0, \$V0) ; Playback of SBC voice file.

TR2: SBC_Loop_On ; Turn on the loop of playback.

TR3: SBC_Loop_Off ; Turn off the loop of playback.

5.5.11 SBC_Loop_Off

[NX1]

When turns off the loop function, the voice will keep playing untill reaching to an end.

Note: This command is deprecated since Q-Code 8.20, its recommended to replace with [Audio Loop Off](#).

5.5.12 ADPCM_Loop_On

[NX1]

The playback of ADPCM loops on.

Note:

- 1. When using the loop fuction, it only provides one channel to play.*
- 2. This command is deprecated since Q-Code 8.20, its recommended to replace with [Audio Loop On](#).*

Ex.

PowerOn: InputState

TR1: PlayV(ch0, \$V0) ; Play ADPCM audio file.

TR2: ADPCM_Loop_On ; Turn on the loop.

TR3: ADPCM_Loop_Off ; Turn off the loop.

5.5.13 ADPCM_Loop_Off

[NX1]

Stop the loop of ADPCM playback. After turning off, the playback of the audio file still be played to the end.

Note: This command is deprecated since Q-Code 8.20, its recommended to replace with [Audio Loop Off](#).

5.5.14 ADPCM_UpSampling

[NX1]

Chage upsampling factor for ADPCM playback.

ADPCM_UpSampling(Factor)

Factor: Upsampling factor, accepts 1 ~ 8.

Note:

- 1. Upsampling only apply to single channel.*
- 2. Playback sample rate after upsampling should not exceed 64kHz.*

Ex.

PowerOn: ADPCM_UpSampling(2) ; Use ADPCM Upsampling with factor 2.
TR1: PlayV(ch0, \$V0) ; ADPCM playback.

5.5.15 ADM_Loop_On

[NX1]

The playback of ADM loops on.

Note:

1. When using the loop function, it only provides one channel to play.
2. This command is deprecated since Q-Code 8.20, its recommended to replace with [Audio Loop On](#).

Ex.

PowerOn: InputState
TR1: PlayV(ch0, \$V0) ; Play ADM audio file.
TR2: ADM_Loop_On ; Turn on the loop.
TR3: ADM_Loop_Off ; Turn off the loop.

5.5.16 ADM_Loop_Off

[NX1]

Stop the loop of ADM playback. After turning off, the playback of the audio file still be played to the end.

Note: This command is deprecated since Q-Code 8.20, its recommended to replace with [Audio Loop Off](#).

5.5.17 ADM_UpSampling

[NX1]

Change upsampling factor for ADM playback.

ADM_UpSampling(Factor)

Factor: Upsampling factor, accepts 1 ~ 8.

Note:

1. Upsampling only apply to single channel.
2. Playback sample rate after upsampling should not exceed 64kHz.

Ex.

PowerOn: ADM_UpSampling(2) ; Use ADM Upsampling with factor 2.
TR1: PlayV(ch0, \$V0) ; ADM playback.

5.5.18 PCM_Loop_On

[NX1]

The playback of PCM loops on.

Note:

1. When using the loop function, it only provides one channel to play.
2. This command is deprecated since Q-Code 8.20, its recommended to replace with [Audio_Loop_On](#).

Ex.

PowerOn: InputState

TR1: PlayV(ch0, \$V0) ; Play PCM audio file.

TR2: PCM_Loop_On ; Turn on the loop.

TR3: PCM_Loop_Off ; Turn off the loop.

5.5.19 PCM_Loop_Off

[NX1]

Stop the loop of PCM playback. After turning off, the playback of the audio file still be played to the end.

Note: This command is deprecated since Q-Code 8.20, its recommended to replace with [Audio_Loop_Off](#).

5.5.20 ReadFileCountV

[NX1]

Read voice file count.

ReadFileCountV(Ri)

ReadFileCountV(Ri, Storage)

Ri: Result voice file count, range 0~65535.

Storage: Specify storage of voice files. Accepts UDR / SPI0 / SPI1. Voice count in **[Voice File]** if omitted.

Ex.

[Variable]

Var16: voice_max=0, voice_idx=0

[Path]

PowerOn: ReadFileCountV(voice_max) ; Read voice file count.

TR1: voice_idx++, ; voice_idx+1 ◦

voice_idx>=voice_max?{voice_idx=0}, ; index exceed boundary, reset to 0.

PlayV(ch0, voice_idx)

; Play voice which index is voice_idx.

5.6 Record Command

Sentence Command				
Record	RecordS	WaitRN	StopR	EraseR
EraseRS	WaitEN	-	-	-

5.6.1 Record / RecordS

[NX1]

This command is to record voice and write to record block on SPI Flash, defined in **[Record]**.

Recorded voice can be played by PlayV command.

The format of command is shown as the followings:

Record(Label {,Time})

RecordS (Label {, Time})

Record: execute next command after finishing playing voice file.

RecordS: execute next command after starting playing voice file.

Label: Record block defined in **[Record]**.

Time: Recording time, record block length is used if omitted.

Ex.

[Path]

TR1: Record(\$Rec0) **; Execute recording.**

TR2: PlayV(Ch0, \$Rec0) **; Play recorded voice.**

5.6.2 WaitRN

[NX1]

If record function is executing, it will execute next command after recording ends. Otherwise, it will immediately execute the next command.

Note: Record = RecordS + WaitRN.

Ex.

[Path]

TR1: Record(\$Rec0)

TR2: RecordS(\$Rec0), WaitRN **; The result is the same as TR1.**

5.6.3 StopR

[NX1]

The command is to stop recording.

Ex.

PowerOn: InputState

TR1: Record(\$Rec0)

TR2: StopR ; Stop recording.

5.6.4 EraseR / EraseRS

[NX1]

This command is to erase record block defined in **[Record]**.

The format of command is as following:

EraseR(Label) { & [BG1 {,BG2 {,BG3}}] }

EraseRS (Label) { & [BG1 {,BG2 {,BG3}}] }

EraseR: execute next command after record erasing ends.

EraseRS: execute next command after starting record erasing.

Label: record block defined in **[Record]**.

Ex.

EraseR(\$Rec0)

5.6.5 WaitEN

[NX1]

If Record Erasing is executing, it will execute next command after Record Erasing ends. Otherwise, it will immediately execute the next command.

Note:

1. **EraseR = EraseRS + WaitRN.**

2. **This command is not available if Realtime Erasing is enabled.**

Ex.

[Path]

TR1: EraseR(\$Rec0)

TR2: EraseRS(\$Rec0), WaitEN ; The result is the same as TR1.

5.7 Sentence Command

Sentence Command				
PlayS	WaitSN(n)	PauseS(n)	ResumeS(n)	StopS(n)

5.7.1 PlayS(Parameter)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command enables the program to play a sequence of voice combination.

The format of command is shown as the followings:

PlayS(Parameter)

PlayS: execute next command after sentence play ends.

Parameter: label of the sentence.

Ex. PlayS(\$S1)

Note: In NY5+ / NY6 / NY7 / NX1, sentence is designed for combining resource files playback (Melody, Speech, Action, etc) as single sentence. Therefore, the Sentence section only allows the Play commands (PlayV / PlayM / PlayA) and Delay commands, other commands cannot be used in the Sentence section.

Ex.

[Sentence]

S1: PlayV(CH2,\$V0), DELAY(0.1), PlayV(CH2,\$V1), DELAY(0.2), PlayV(CH2,\$V2)

[Path]

PowerOn: KEY1

TR1: PlayS(\$S1)

[Background1]

BG1: PlayS(\$S1)

[Background2]

BG3: PlayS(\$S1)

5.7.2 WaitSN(n)

[NY6 / NY7 / NX1]

If a sentence is being played, it will execute next command after finishing playing sentences. If there is no playing sentence, it will immediately execute the next command.

n: 0=foreground, 1=background1, 2=background2, 3=background3. If n is not designated, the system will execute the next command after all sentences finish playing.

Ex.

[Path]

TR1: PlayS(\$S1)

TR2: WaitSN(0), PB.0=1 ; Set PB.0 as 1 after sentence finishing playing.

5.7.3 PauseS(n)

[NY6 / NY7 / NX1]

This command can be used to pause the sentence which currently being played in designated channel.

n: 0=foreground, 1=background1, 2=background2, 3=background3. If n is not designated, the system will pause all playing sentences.

Ex.

PowerOn: InputState

TR1: PlayS(\$S1)

TR2: PauseS ; Pause all the playing sentences.

TR3: ResumeS ; Resume all sentences.

TR4: PauseS(0) ; Pause the playing sentence of foreground.

TR5: ResumeS(0) ; Resume the sentence of foreground.

5.7.4 ResumeS(n)

[NY6 / NY7 / NX1]

This command can be used to resume the designated sentence.

n: 0=foreground, 1=background1, 2=background2, 3=background3. If n is not designated, it will resume all the sentences.

In the following situation, the command is not available:

1. The sentence has finished playing.
2. User had executed the StopS or Stop command.

Ex.

PowerOn: InputState

TR1: PlayS(\$S1)

TR2: PauseS ; Pause all the playing sentences.

TR3: ResumeS ; Resume all the sentences.

TR4: PauseS(0) ; Pause the playing sentence of foreground.

TR5: ResumeS(0) ; Resume the sentences of foreground.

5.7.5 StopS(n)

[NY6 / NY7 / NX1]

This command can be used to stop the playback of sentences. The parameter n can be designated to the sentences of foreground or background.

n: 0=foreground, 1=background1, 2=background2, 3=background3. If n is not designated, it will stop all the playing sentences.

Ex.

PowerOn: InputState

TR1: PlayS(\$S1)

TR2: StopS

; Stop all the playing sentences.

TR3: StopS(0)

; Stop the sentence of foreground.

5.8 SPI Play Command

SPI Play Command				
SPIPlay	SPIPlayS	SPIWaitN	SPIStop	SPIPause
SPIResume	SPIVol = n	SPIVol = Ri	Ri = SPIVol	-
SPIGetIndex(Result, SPIGroup)	-	-	-	-

Note: Please make sure SPI Flash is in readable / writable state before executing any commands on SPI Flash. According to different SPI Flash, the stable time after power-up is also different, ranging from about 200us to 15ms. The capacitance on the PCB also affects the time that SPI Flash reaches the operable voltage after power-up, as well as the discharge time after power-off. Therefore, if the SPI Flash is powered on after power-off, the status of SPI Flash should be checked, or there should be an adequate delay.

5.8.1 SPIPlay / SPIPlayS

[NY6B / NY6C / NY7 / NX1]

This command is provided for user to play data stored on SPI Flash. There are 2 SPI channels, but cannot use both at the same time. The format of command is as following:

SPIPlay ({Ch,} Index {++} {,SPIGroup})

SPIPlayS ({Ch,} Index {++} {,SPIGroup})

SPIPlay ({Ch,} Index {++}, Px.n)

SPIPlayS ({Ch,} Index {++}, Px.n)

SPIPlay: execute next command after SPI data playback ends.

SPIPlayS: execute next command after SPI data playback starts.

Ch: specify the channel (**NX1 Only**).

Index: index of data on SPI Flash.

- Use the immediate value index.
- Use Ri / Xi index, supports 1~4 Ri or 1 ~ 2 Xi (NY6 / NY7) or 1 variable (NX1).

SPIGroup: SPI channel.

- NY7 supports SPI1 or SPI2, default is SPI1.
- NX1 supports SPI0 or SPI1, default is SPI0..

Px.n: action output pin if SPI sentence is action.

Voice format and sampling rates supported by NY7 SPIPlay are as following table:

Sample Rate	PCM8	PCM12	ADPCM6
7.8K	✓	✓	✓
15.6K	✓	✓	✓

Note:

1. *Voice files must be encoded to .bin file by SPI_Encoder and write to SPI Flash.*
2. *The Header of encoded file contains sentence or voice, user only need to give index to play.*
3. *If index exceeds the maximum number of data in SPI Flash, this command will play data at Index 0.*
4. *SPIPlay cannot be used with Voice commands (PlayV / PlayVS) / MIDI commands (PlayM / PlayMS) at the same time.*
5. *When SPIPlay is used for playing action sentence in NY6 / NY7, action channel is fixed at Ch1.*
6. *If user specifies the Px.n parameter, it means SPIPlay is applied to action playback. Ch parameter showed action channel can be Ch1 ~ 32. The selected file must be action file otherwise the command will be no effect. If user doesn't specify the Px.n parameter, it means SPIPlay is applied to Voice / MIDI playback, Ch parameter showed audio channel can be Ch0 ~ Ch7 only. The selected file must be Voice / MIDI file otherwise the command will be no effect.*
7. *In NY6, when playing both the voice and action data from SPI flash at the same time, if the voice factor is 1~9 and the sampling rate exceeds 16kHz (or the factor is 10~PCM and sampling rate exceeds 10kHz), the action data will not be output.*

Ex. Specify data to be played in NY7.

SPIPlay(1, SPI1) / SPIPlayS(2, SPI1) ; Use immediate value.

SPIPlay(R0, SPI1) / SPIPlayS(R2:R1:R0, SPI1) ; Use the value of Ri.

SPIPlay(X0, SPI1) / SPIPlayS(X1:X0, SPI1) ; Use immediate value.

Ex. Specify SPI channel in NY7.

SPIPlay(1, SPI1) / SPIPlay(1, SPI2) ; Speicfy SPI channel.

SPIPlay(R1:R0) ; Not specify SPI channel, use SPI1 by default.

Ex. Play action on SPI Flash in NY6.

SPIPlay(0, PA.3)

; Index 5 playback, action output to PA.3.

Ex. Play action on SPI Flash in NX1.

SPIPlay(Ch0, 0, PA.3, SPI0)

; Index 5 playback, action play on Ch1 and output to PA.3.

Ex.

P1: SPIPlay(5), PB.3=1

; After the end of Index 5 playback, execute PB.3=1.

P2: SPIPlayS(5), PB.3=1

; Execute PB.3=1 immediately after Index 5 starts to play.

P3: SPIPlay(R1:R0, SPI1)

; Use the index value "R1:R0" to play, R1=High 4-bit, R0=Low4-bit.

P4: SPIPlay(X0++)

; Use the index value "X0" to play, The value of X0 is automatically increase 1.

5.8.2 SPIWaitN(Ch)

[NY6B / NY6C / NY7 / NX1]

Execute next command after SPIPlay playback ends. If no SPIPlay playback is in progress, executes next command immediately.

Ch: SPI playback channel (audio channel only) to wait, wait all playing in all channels if omitted.

- NY6 / NY7 do not support Ch parameter.
- NX1 supports the audio channel Ch0 ~ Ch7.

5.8.3 SPIStop(Ch)

[NY6B / NY6C / NY7 / NX1]

Stop playback of SPI data.

Ch: SPI playback channel (audio channel only) to stop, stop all playing in all channels if omitted.

- NY6 / NY7 do not support Ch parameter.
- NX1 supports the audio channel Ch0 ~ Ch7.

5.8.4 SPIPause(Ch)

[NY6B / NY6C / NY7 / NX1]

Stop the playback of SPI data.

Ch: SPI playback channel (audio channel only) to pause, wait all playing in all channels if omitted.

- NY6 / NY7 do not support Ch parameter.
- NX1 supports the audio channel Ch0 ~ Ch7.

Note: If user doesn't set SPIPlay, this command does not take effect.

5.8.5 SPIResume(Ch)

[NY6B / NY6C / NY7 / NX1]

This command can resume the playback of SPI data.

Ch: SPI playback channel (audio channel only) to pause, wait all playing in all channels if omitted.

- NY6 / NY7 do not support Ch parameter.
- NX1 supports the audio channel Ch0 ~ Ch7.

5.8.6 SPIVol = n /SPIVol = Ri

[NY6B / NY6C / NY7]

Set the volume of SPIPlay.

n: volume value, can be immediate value or Ri.

- NY6 supports 0 ~ 15; n=0 means mute, n=15 means the loudest volume.
- NY7 supports 0 ~ 4; n=0 means mute, n=4 means the loudest volume.

Ex. SPIVol = 3

; Set the SPI volume as 3rd level.

Ex. SPIVol = R0

; Access the parameter of R0 as SPI volume level.

5.8.7 Ri = SPIVol

[NY6B / NY6C / NY7]

Get the volume of SPIPlay.

Ex. R0 = SPIVol

; Get the SPI playback volume and save to R0.

5.8.8 SPIGetIndex

[NY6B / NY6C / NY7 / NX1]

Read the number of sentences in SPI Flash, which is equivalent to Sentence No in SPI_Encoder.

SPIGetIndex(Result[, SPIGroup])

Result: number of file index in SPI Flash.

- NY6 / NY7 support 4 4-bit registers, Ri:Rk:Rj:Ri or 2 8-bit registers, Xj:Xi. Ri=bit0~3, Rj=bit4~7, Rk=bit8~11, Ri=bit12~15 / Xi=bit0~7, Xj=bit8~15 (NY6 / NY7 Only).
- 16-bit variable (NX1 Only).

SPIGroup: specify the accessed interface of SPI.

- NY6 / NX1 do not support the specified SPIGroup.
- NY7 can be either SPI1 or SPI2; the default is SPI1 if omitted.
- NX1 can be either SPI0 or SPI1, the default is SPI0 if omitted.

Ex. SPIGetIndex(R1:R0) ; Read the sentence no in first SPI.
Ex. SPIGetIndex(R2:R1:R0, SPI2) ; Read the sentence no in second SPI.
Ex SPIGetIndex(X0) ; Read the sentence no in first SPI.
Ex. SPIGetIndex(X1:X0, SPI2) ; Read the sentence no in second SPI.

5.9 SPI Flash Command

SPI Flash Command				
SPI_WREN	SPI_WRDIS	SPI_RDID	SPI_CE	SPI_SE
SPI_BE	SPI_DP	SPI_RDP	SPI_WRSR	SPI_RDSR
SPI_WRD	SPI_RDD	SPI_GetAddr	-	-

Note:

1. Please make sure SPI Flash is in readable / writable state before executing any commands on SPI Flash. According to different SPI Flash, the stable time after power-up is also different, ranging from about 200us to 15ms. The capacitance on the PCB also affects the time that SPI Flash reaches the operable voltage after power-up, as well as the discharge time after power-off. Therefore, if the SPI Flash is powered on after power-off, the status of SPI Flash should be checked, or there should be an adequate delay.
2. Different SPI Flash have different settings for Quad Mode. Please be aware if the QE bit has been properly set.

5.9.1 SPI_WREN

[NY6B / NY6C / NY7 / NX1]

Transmit the Write Enable command to the SPI Flash, and set the status register WEL bit (bit1) as 1 to make the SPI Flash stay in a writable state.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note:

1. This command is applicable to the SPI Flash of N25Q series; the instruction is 0x06.
2. After this command is executed, WEL bit (bit1) of status register will be set as 1, and writing data to SPI Flash is valid at this time. User can confirm the bit status through SPI_RDSR command.

Ex.

SPI_WREN ; Transmit Write Enable command

Ex. NY7 / NX1

SPI_WREN(SPI1) ; Transmit Write Enable command through SPI1.

Ex. NY7

SPI_WREN(SPI2) ; Transmit Write Enable command through SPI2.

5.9.2 SPI_WRDIS

[NY6B / NY6C / NY7 / NX1]

Transmit the Write Disable command to the SPI Flash, and clear the status register WEL bit (bit1) to 0 to make the SPI Flash stay in an unwritable state.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note:

1. *This command is applicable to the SPI Flash of N25Q series; the instruction is 0x04.*
2. *After this command is executed, WEL bit (bit1) of status register will be cleared to 0, and writing data to SPI Flash is invalid at this time. User can confirm the bit status through SPI_RDSR command.*

Ex.

SPI_WRDIS ; Transmit Write Disable command.

Ex. NY7 / NX1

SPI_WRDIS(SPI1) ; Transmit Write Disable command through SPI1.

Ex. NY7

SPI_WRDIS(SPI2) ; Transmit Write Disable command through SPI2.

5.9.3 SPI_RDID(Result, SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Transmit Read Identification to SPI Flash and read the information about Manufacturer ID, Memory Type and Capacity of SPI Flash.

Result: Save the variable of the read-back information, which can be a combination of Ri or Xi and requires 24-bit; the high 8-bit is Manufacturer ID, the middle 8-bit is Memory Type and the low 8-bit is Capacity.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.

- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note: This command is applicable to the SPI Flash of N25Q series; the instruction is 0x9F.

Ex. NY6B / NY6C / NY7

[Path]

TR1: SPI_RDID(R5:R4:R3:R2:R1:R0)

; Read SPI Flash Information via SPI1, and save Manufacturer ID in [R5: R4], Memory Type in [R3: R2], Capacity in [R1: R0].

Ex. NY7

[Path]

TR1: SPI_RDID(X2:X1:X0, SPI2)

; Read SPI Flash Information via SPI2, and save Manufacturer ID in X2, Memory Type in X1, Capacity in X0.

Ex. NX1

[Variable]

Var32: ID

[Path]

SPI_RDID(ID, SPI1)

; Read SPI Flash Information via SPI1, and save Manufacturer ID in ID[8:11], Memory Type in ID[4:7], Capacity in ID[0:3].

5.9.4 SPI_CE

[NY6B / NY6C / NY7 / NX1]

Transmit Chip Erase command to SPI Flash and excute erasing all memory.

The system will not enter sleep mode when the erase action is executing. After the erasing is completed, the "SPI_EraseEnd" path will be automatically executed. User can read the WIP bit (bit0) status of the status register by using the SPI_RDSR command. When the erase is in progress, the WIP bit is 1; when the erase is completed, the WIP bit is 0.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note:

1. This command is applicable to the SPI Flash of N25Q series; the instruction is 0xC7.
2. This command will automatically set WEL bit to 1, users do not need to execute SPI_WREN command additionally.
3. Depending on the different size of the SPI Flash, erasing may take seconds to tens of seconds. All commands are invalid during erasing process except SPI_RDSR command.

4. After the erasing is completed, the “SPI_EraseEnd” path is automatically executed. User will know the SPI_EraseEnd completing time through this path to perform subsequent actions.

Ex. NY6B / NY6C / NY7 / NX1

[Path]

TR1: SPI_CE

; Transmit Chip Erase command to SPI Flash to erase all memory.

Ex. NY7

[Path]

TR1: SPI_CE(SPI2)

; Transmit Chip Erase command to SPI Flash through SPI2 to erase all memory.

Ex. NY6B / NY6C / NY7 / NX1

[Path]

Erase: SPI_CE, R0=1

SPI_EraseEnd: R0=0

; Transmit Chip Erase command to SPI Flash. When erase is in progress, set R0 as 1. When the erase is completed, set R0 as 0.

5.9.5 SPI_SE(Addr, Count, SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Transmit Sector Erase command to SPI Flash and erase sector(s). The size of a sector is 4KB, users can specify the amount of sector to be erased once.

The system will not enter sleep mode when the erase action is executing. After the erasing is completed, the “SPI_EraseEnd” path will be automatically executed. User can read the WIP bit (bit0) status of the status register by using the SPI_RDSR command. When the erase is in progress, the WIP bit is 1; when the erase is completed, the WIP bit is 0.

Addr: Specify the address to be erased which can be an immediate value or specified by variable. The size of a sector is 4KB. When using immediate value addressing, user needs to fill in the 24-bit address, but Bit[11: 0] is invalid. When using variable addressing, user only needs to fill in the valid bit Bit[23:12].

Count: The amount of sector to be erased, the range is from 1 to 16; default is 1 if omitted.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note:

1. This command is applicable to the SPI Flash of N25Q series; the instruction is 0x20.
2. This command will automatically set WEL bit to 1, users do not need to execute SPI_WREN

command additionally.

3. *Depending on the number of the erased sector, erasing may take tens of milliseconds to seconds. All commands are invalid during erasing process except SPI_RDSR command.*
4. *After the erasing is completed, the "SPI_EraseEnd" path is automatically executed. User will know the SPI_EraseEnd completing time through this path to perform subsequent actions.*

Ex. NY6B / NY6C / NX1

[Path]

TR1: SPI_SE(0x4321, 3)

; Erase the data in address 0x4000~0x6FFF of SPI Flash.

TR2: R0=0x2, R1=0x1, SPI_SE(R1:R0, 2)

; Erase the data in address 0x12000~0x13FFF of SPI Flash.

Ex. NY7

[Path]

Erase: X0=0x23, R2=0x1, SPI_SE(R2:X0, SPI1), R3=1

SPI_EraseEnd: R3=0

; Erase the data in address 0x123000~0x123FFF of SPI Flash through SPI1. When the erase is in progress, set R3 as 1. When the erase is completed, set R3 as 0.

Ex. NX1

[Variable]

Var32: Addr

Var8: R3

[Path]

TR1: SPI_SE(0x4321, 3)

; Erase the data in address 0x4000 ~ 0x6FFF of SPI Flash through SPI0.

TR2: Addr = 0x12, SPI_SE(Addr, 2, SPI1)

; Erase the data in address 0x12000 ~ 0x13FFF of SPI Flash through SPI1.

Erase: Addr = 0x123, SPI_SE(Addr, SPI0), R3=1

SPI_EraseEnd: R3=0

; Erase the data in address 0x123000 ~ 0x123FFF of SPI Flash through SPI0. When erasing is in progress, Set R3 as 1, set R3 as 0 when erasing is completed.

5.9.6 SPI_BE(Addr, Count, SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Transmit Block Erase command to SPI Flash and erase block(s). The size of a block is 64KB. Users can specify the amount of block to be erased once.

The system will not enter sleep mode when the erase action is executing. After the erasing is completed, the "SPI_EraseEnd" path will be automatically executed. User can read the WIP bit (bit0) status of the status register by using the SPI_RDSR command. When the erase is in progress, the WIP bit is 1; when the erase is completed, the WIP bit is 0.

Addr: Specify the address to be erased which can be an immediate value or specified by variable. The size of a block is 64KB. When using immediate value addressing, user needs to fill in the 24-bit address, but Bit[15: 0] is invalid. When using variable addressing, user only needs to fill in the valid bit Bit[23:16].

Count: The amount of block to be erased, the range is from 1 to 16; default is 1 if omitted.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note:

1. *This command is applicable to the SPI Flash of N25Q series, the instruction is 0xD8.*
2. *This command will automatically set WEL bit to 1, users do not need to execute SPI_WREN command additionally.*
3. *Depending on the number of the erased block, erasing may take seconds. All commands are invalid during erasing process except SPI_RDSR command.*
4. *After the erasing is completed, the "SPI_EraseEnd" path is automatically executed. User will know the SPI_EraseEnd completing time through this path to perform subsequent actions.*

Ex. NY6B / NY6C / NY7 / NX1

[Path]

TR1: SPI_BE(0x14321, 2)

; Erase the data in address 0x10000~0x2FFFF of SPI Flash.

TR2: R0=0x3, SPI_BE(R0, 3)

; Erase the data in address 0x30000~0x5FFFF of SPI Flash.

Ex. NY7

[Path]

Erase: X0=0x17, SPI_BE(X0, SPI1), R3=1

SPI_EraseEnd: R3=0

; Erase the data in address 0x170000~0x17FFFF of SPI Flash through SPI1. When the erase is in progress, set R3 as 1. When the erase is completed, set R3 as 0.

Ex. NX1

[Variable]

Var32: Addr

[Path]

TR1: SPI_BE(0x14321, 2)

; Erase the data in address 0x10000~0x2FFFF of SPI Flash through SPI0.

TR2: Addr=0x3, SPI_BE(Addr, 3, SPI1)

; Erase the data in address 0x30000~0x5FFFF of SPI Flash through SPI1.

Erase: Addr=0x17, SPI_BE(Addr, SPI1), R3=1

SPI_EraseEnd: R3=0

; Erase the data in address 0x170000~0x17FFFF of SPI Flash through SPI1. When the erase is in progress, set R3 as 1. When the erase is completed, set R3 as 0.

5.9.7 SPI_DP(SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Transmit Deep Power-down command to SPI Flash to enter deep power-down mode. **It is recommended to execute this command before entering the sleep mode to save SPI Flash power consumption.**

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note:

1. **This command is applicable to the SPI Flash of N25Q series; the instruction is 0xB9.**
2. **After SPI Flash entering deep power-down, all commands are invalid except SPI_RDP command to wake up.**

Ex. NY6B / NY6C / NY7 / NX1

SPI_DP ; Transmit Deep Power-down command.

Ex. NY7 / NX1

SPI_DP(SPI1) ; Transmit Deep Power-down command through SPI1.

Ex. NY7

SPI_DP(SPI2) ; Transmit Deep Power-down command through SPI2.

5.9.8 SPI_RDP(Result, SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Transmit Release Deep Power-down command to SPI Flash. Wake up SPI Flash from deep power-down mode and read back 8-bit Device ID.

Result: Save the read-back Device ID which can be a combination of Ri or Xi, not reading the ID if omitted.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.

- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note: This command is applicable to the SPI Flash of N25Q series; the instruction is 0xAB.

Ex. NY7

[Path]

TR1: SPI_RDP(R1:R0)

; Wake up SPI Flash from deep power-down through SPI1 and save Device ID into [R1:R0].

TR2: SPI_RDP(SPI1)

; Wake up SPI Flash from deep power-down through SPI1 and do not read Device ID.

TR3: SPI_RDP(X0, SPI2)

; Wake up SPI Flash from deep power-down through SPI2 and save Device ID into X0.

Ex. NX1

[Variable]

Var8: DevID

[Path]

TR1: SPI_RDP(DevID)

; Wake up SPI Flash from deep power-down through SPI1 and save Device ID into DevID.

TR2: SPI_RDP(SPI1)

; Wake up SPI Flash from deep power-down through SPI1 and do not read Device ID.

TR3: SPI_RDP(X0, SPI2)

; Wake up SPI Flash from deep power-down through SPI2 and save Device ID into DevID.

5.9.9 SPI_WRSR(Data, SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Transmit Write Status Register command to SPI Flash to modify the data of Status Register.

Data: The Status Register data can be immediate value or a combination of Ri or Xi.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note:

1. This command is applicable to the SPI Flash of N25Q series; the instruction is 0x01.
2. This command will automatically set WEL bit to 1, users do not need to execute SPI_WREN command additionally.

Ex. NY7

[Path]

TR1: SPI_WRSR(0x18) ; Write 0x18 into Status Register through SPI1.
 TR2: R0=0x8, R1=0x1, SPI_WRSR(R1:R0, SPI1) ; Write 0x18 into Status Register through SPI1.
 TR3: X0=0x18, SPI_WRSR(X0, SPI2) ; Write 0x18 into Status Register through SPI2.

Ex. NX1

[Variable]

Var8: Data

[Path]

TR1: SPI_WRSR(0x18) ; Write 0x18 into Status Register through SPI0.
 TR2: Data = 0x18, SPI_WRSR(Data, SPI0) ; Write 0x18 into Status Register through SPI0.
 TR3: Data=0x18, SPI_WRSR(Data, SPI1) ; Write 0x18 into Status Register through SPI1.

5.9.10 SPI_RDSR(Result, SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Transmit Read Status Register command to the SPI Flash to read Status Register data.

Result: Save the read-back data of Status Register which can be a combination of Ri or Xi.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note: This command is applicable to the SPI Flash of N25Q series; the instruction is 0x05.

Ex. NY7

[Path]

TR1: SPI_RDSR(R1:R0) ; Read Status Register data through SPI1 and save into [R1:R0].
 TR2: SPI_RDSR(X0, SPI1) ; Read Status Register data through SPI1 and save into X0.
 TR3: SPI_RDSR(X0, SPI2) ; Read Status Register data through SPI2 and save into X0.

Ex. NX1

[Variable]

Var8: SR

[Path]

TR1: SPI_RDSR(SR) ; Read Status Register data through SPI0 and save into SR.
 TR2: SPI_RDSR(SR, SPI0) ; Read Status Register data through SPI0 and save into SR.
 TR3: SPI_RDSR(SR, SPI1) ; Read Status Register data through SPI1 and save into SR.

5.9.11 SPI_WRD(Addr, Data, SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Transmit Page Program command to the SPI Flash to write data into specified address. User can write many bytes of data at one time with SPI_TX command. After the data sent, it needs to issue SPI_CS_Off command to write in. User needs to use SPI_RDSR command to read WIP bit (bit0) register status. When writing, this bit is 1. After writing finished, this bit is 0.

Addr: Specify 24-bit address of reading data which can be immediate value or variable specified.

When specified by a variable, high bit variable can be omitted, the omitted bit is defaulted as 0. °

Data: The writing data can be immediate value or variable specified.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note:

1. *This command is applicable to the SPI Flash of N25Q series; the instruction is 0x02.*
2. *This command will automatically set WEL bit to 1, users do not need to execute SPI_WREN command additionally.*
3. *While the command executes, it is automatically set CS pin as low level. After the writing data are sent, issue SPI_CS_Off command to write in.*
4. *When writing to SPI Flash, WIP bit (bit0) of status register is set as 1, users can read bit status through SPI_RDSR command. When it is 0, it means finished writing.*
5. *Page Program command can write 256 bytes at most. The address is added by 1 automatically every time a byte is sent, but only for low 8 bit. If current writing address is 0x1FF, it will be 0x100 after adding 1.*
6. *According to amount of writing data, it may take hundred ms. All commands are invalid in this period except SPI_RDSR command.*

Ex. NY7

[Path]

TR1: SPI_WRD(0x4321, 0x32), SPI_CS_Off

; Write 0x32 into address 0x4321 through SPI1.

TR2: X0=0xFF, R2=0x2, R3=0x1, X2=0x34, SPI_WRD(R3:R2:X0, X2, SPI2),

SPI_TX(0x56, SPI2), SPI_TX(0x78, SPI2), SPI_CS_Off(SPI2)

; Through SPI2, write 0x34 into address 0x12FF, 0x56 into address 0x1200 and 0x78 into address 0x1201.

TR3: SPI_WRD(0x4321, 0xAB, SPI1), R0=0xD, R1=0xC, SPI_TX(R1:R0, SPI1),

X0=0xEF, SPI_TX(X0, SPI1), SPI_CS_Off(SPI1)

; Through SPI1, write 0xAB into address 0x4321, 0xCD into address 0x4322 and 0xEF into address 0x4323.

Ex. NX1

[Variable]

Var32: Addr

Var8: Data

[Path]

TR1: SPI_WRD(0x4321, 0x32), SPI_CS_Off

; Write 0x32 into address 0x4321 through SPI1.

TR2: Addr=0x12FF, Data=0x34, SPI_WRD(Addr, Data, SPI1),

SPI_TX(0x56, SPI1), SPI_TX(0x78, SPI1), SPI_CS_Off(SPI1)

; Through SPI1, write 0x34 into address 0x12FF, 0x56 into address 0x1200 and 0x78 into address 0x1201.

TR3: SPI_WRD(0x4321, 0xAB), Data=0xCD, SPI_TX(Data),

Data=0xEF, SPI_TX(Data), SPI_CS_Off

; Through SPI0, write 0xAB into address 0x4321, 0xCD into address 0x4322 and 0xEF into address 0x4323.

5.9.12 SPI_RDD(Addr, Result, SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Send Read Data command to SPI Flash. Read data from specified address, user can read many bytes data at one time with SPI_RX commad. *After finished reading, it needs to issue SPI_CS_Off command to end the reading action.*

Addr: Specify 24-bit address of reading data which can be immediate value or variable specified.

When specified by a variable, high bit variable can be omitted, the omitted bit is defaulted as 0.

Result: Save the variable of reading data.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note:

1. *This command is applicable to the SPI Flash of N25Q series; the instruction is 0x03.*
2. *While the command executes, it is automatically set CS pin as low level. After the data finished reading, it needs to issue SPI_CS_Off command to end the read action.*
3. *The address is added by 1 automatically every time a byte is read.*

Ex. NY7

[Path]

TR1: SPI_RDD(0x4321, R1:R0), SPI_CS_Off

; Save the data in address 0x4321 into [R1:R0] through SPI1.

TR2: R0=0x3, X1=0x12, SPI_RDD(X1:R0, X2, SPI2), SPI_CS_Off(SPI2)

; Save the data in address 0x123 into X2 through SPI2.

TR3: SPI_RDD(0x1234, X0, SPI1), SPI_RX(R3:R2, SPI1), SPI_RX(X2, SPI1), SPI_CS_Off(SPI1)

; Save the data in address 0x1234 into X0 through SPI1, the data in address 0x1235 into [R3:R2],

the data in address 0x1236 into X2.

Ex. NX1

[Variable]

Var32: Addr

Var8: Data, Data0, Data1, Data2

[Path]

TR1: SPI_RDD(0x4321, Data), SPI_CS_Off

; Save the data in address 0x4321 into Data through SPI1.

TR2: Addr=0x123, SPI_RDD(Addr, Data, SPI1), SPI_CS_Off(SPI1)

; Save the data in address 0x123 into X2 through SPI2.

TR3: SPI_RDD(0x1234, Data0, SPI1), SPI_RX(Data1, SPI1), SPI_RX(Data2, SPI1), SPI_CS_Off(SPI1)

; Save the data in address 0x1234 into Data0 through SPI1, the data in address 0x1235 into Data1, the data in address 0x1236 into Data2.

5.9.13 SPI_GetAddr(Index, Result, SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Users can add user defined file or reserved blocks through User Defined Sections page in *SPI_Encoder*. Then use this command to get the address of each block to excute read, write or erase operations.

When users add a file or defines the reserved blocks, the adding order (the Sec field) is the index value. Specify the index value of the block in the parameter of command to obtain the corresponding address. Please refer to *SPI_Encoder* User Manual for detailed operation.

Index: Specify the index value of the block address which can be an immediate value or specified by variable, total is 16-bit and the range is 0~65535. When specified by a variable, high bit variables can be omitted; the omitted bit is defaulted as 0.

Result: The variable stores the start address of block. Total is 24-bit, and high bit variables can be omitted and not to be stored.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note: This command is applicable to the SPI Flash of N25Q series; the instruction is 0x03.

Ex. NY7

[Path]

TR1: SPI_GetAddr(0x8, X2:X1:R1:R0)

; Read the block address of index value 8 through SPI1, and save Bit[23:16] into X2, Bit[15: 8]

into X1, Bit[7: 4] into R1, Bit[3: 0] into R0.

TR2: R0=0x3, R1=0x2, SPI_GetAddr(R1:R0, X2:X1, SPI2)

; Read the block address of index value 0x23 through SPI2, and save Bit[15:8] into X2, Bit[7:0] into X1.

TR3: X0=0x12, SPI_GetAddr(X0, R7:R6:R5:R4:R3:R2, SPI1)

; Read the block address of index value 0x12 through SPI1, and save the address into [R7:R2].

Ex. NX1

[Variable]

Var32: Addr

Var16: Index

[Path]

TR1: SPI_GetAddr(0x8, Addr)

; Read the block address of index value 0x8 through SPI0, and save the address into Addr.

TR2: Index=0x23, SPI_GetAddr(Index, Addr, SPI1)

; Read the block address of index value 0x23 through SPI1, and save the address into Addr.

5.10 SPI Command

SPI Flash Command				
SPI_CS_On	SPI_CS_Off	SPI_TX	SPI_RX	SPI_CLKDIV
SPI_CPOL	SPI_CPHA	-	-	-

5.10.1 SPI_CS_On(SpiGroup)

[NY6B / NY6C / NY7 / NX1]

Set the CS pin as low level.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note: When any one of the CS pins is set to low level, the system will not be able to enter sleep mode.

Ex.

SPI_CS_On ; Set the CS pin of SPI1 as low level.

SPI_CS_On(SPI1) ; Set the CS pin of SPI1 as low level.

SPI_CS_On(SPI2) ; Set the CS pin of SPI2 as low level.

5.10.2 SPI_CS_Off(SpiGroup)

[NY6B / NY6C / NY7 / NX1]

Set the CS pin as high level.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Ex.

```
SPI_CS_Off ; Set the CS pin of SPI1 as high level.
SPI_CS_Off(SPI1) ; Set the CS pin of SPI1 as high level.
SPI_CS_Off(SPI2) ; Set the CS pin of SPI2 as high level.
```

5.10.3 SPI_TX(Data, SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Send 8-bit data to SPI Flash via SCK and MOSI pins. This command can be used to send commands or write data with SPI_WRD command. It can also be used to combine the commands that Q-Code does not support.

Data: Transmit 8-bit data which can be immediate value or specified by variable.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note:

1. This command transmits 8-bit data only through the SCK and MISO pins, and does not affect the status of CS pin. Therefore, users need to control the CS pin level independently.
2. This command needs to be used with SPI_WREN and SPI_WRD commands to write the data into SPI Flash. Please refer to the description of SPI_WRD command for writing method.

Ex.

```
SPI_CS_On, SPI_TX(0xB9), SPI_CS_Off
; Transmit 0xB9 command to SPI Flash via SPI1.
R0=0xB, R1=0xA, SPI_CS_On(SPI2), SPI_TX(R1:R0, SPI2), SPI_CS_Off(SPI2)
; Transmit 0xAB command to SPI Flash via SPI2.
X0=0x06, SPI_CS_On(SPI1), SPI_TX(X0, SPI1), SPI_CS_Off(SPI1)
; Transmit 0x06 command to SPI Flash via SPI1.
```

5.10.4 SPI_RX(Result, SPIGroup)

[NY6B / NY6C / NY7 / NX1]

Receive 8-bit data from SPI Flash through SCK and MISO pins. This command can be used to read ID or receive data from SPI Flash with SPI_RDD command. It can also be used to combine the commands that Q-Code does not support.

Result: Read 8-bit data which can be a combination of Ri / Xi.

SPIGroup: Select the corresponding SPI channel.

- NY6 does not support SPIGroup parameter.
- NY7 supports SPI1 or SPI2; default is SPI1 if omitted.
- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Note:

1. *This command only receives 8-bit data through SCK and MISO pins, and does not affect the status of CS pin. Therefore, users need to control the CS pin level independently.*
2. *This command needs to be used with SPI_RDD command to read the data in SPI Flash. Please refer to the description of SPI_RDD command for reading method.*

Ex.

SPI_RDD(0x0, X0), SPI_RX(R5:R4), SPI_CS_Off

; Read the data in address 0x0~0x1 of SPI Flash through SPI1. Then save the data of 0x0 in address X0 and data of 0x1 to [R5: R4].

SPI_RDD(0x105, X0, SPI2), SPI_RX(X1, SPI2), SPI_CS_Off(SPI2)

; Read the data in address 0x105~0x106 of SPI Flash through SPI2. Then save the data of address 0x105 and 0x106 to X0 and X1.

5.10.5 SPI_CLKDIV(Divisor, SPIGroup)

[NX1]

Change IHRC divisor of SCK.

Divide: IHRC divisor, accepts 1 / 2 / 4 / 8 / 16 / 32 / 64 / 128. Default is 2 when IHRC>32MHz, default is 1 otherwise.

SPIGroup: Select the corresponding SPI channel.

- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Ex.

SPI_CLKDIV(16, SPI1)

; Change SCK frequency of SPI1 to 1/16 of IHRC.

5.10.6 SPI_CPOL(Polarity, SPIGroup)

[NX1]

Change the polarity of the clock signal during the idle state.

Polarity: 0 (Low)、1 (High). Default is 0.

SPIGroup: Select the corresponding SPI channel.

- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Ex.

SPI_CPOL(0, SPI1) ; Set SPI1 polarity to Low.

5.10.7 SPI_CPHA(Phase, SPIGroup)

[NX1]

Change SPI clock phase.

Phase: 0 (1st Edge), 1 (2nd Edge). Default is 0.

SPIGroup: Select the corresponding SPI channel.

- NX1 supports SPI0 or SPI1; default is SPI0 if omitted.

Ex.

SPI_CPHA(0, SPI1) ; Set SPI1 clock phase to First Edge.

5.11 Embedded Flash Command

Embedded Flash Command				
EF_SE	EF_WRD	EF_RDD	EF_GetAddr	-

5.11.1 EF_SE

[NX1 EF]

Transmit Sector Erase command to Embedded Flash and erase sector(s). The size of a sector is 512B, users can specify the amount of sector to be erased once. The system will not enter sleep mode when the erase action is executing.

EF_SE(Addr)

EF_SE(Addr, Count)

Addr: Specify the address to be erased which can be an immediate value or specified by variable. The size of a sector is 512B. When using immediate value addressing, user needs to fill in the 20-bit address, but Bit[8:0] is invalid. When using variable addressing, user only needs to fill in the valid bit Bit[20:9].

Count: The amount of sector to be erased, the range is from 1 to 16; default is 1 if omitted.

Note: Depending on the number of the erased sector, erasing may take tens of milliseconds to seconds. All commands are invalid during erasing process.

Ex.

[Variable]

Var32: Addr

[Path]

TR1: EF_SE(0x4321, 3)

; Erase the data in address 0x4200~0x47FF of Embedded Flash.

TR2: Addr=0x12, EF_SE(Addr, 2)

; Erase the data in address 0x2400~0x27FF of Embedded Flash.

5.11.2 EF_WRD

[NX1 EF]

Transmit Program command to the Embedded Flash to write data into specified address.

EF_WRD(Addr, Data)

Addr: Specify 20-bit address of reading data which can be immediate value or variable specified.

When specified by a variable, high bit variable can be omitted, the omitted bit is defaulted as 0. °

Data: The writing data can be immediate value or variable.

Note: According to amount of writing data, it may take hundred ms. All commands are invalid in this period.

Ex.

[Variable]

Var32: Addr

Var8: Data

[Path]

TR1: EF_WRD(0x4321, 0x32)

; Write 0x32 into address 0x4321.

TR2: Addr=0x12FF, Data=0x34, EF_WRD(Addr, Data)

; Write 0x34 into address 0x12FF.

5.11.3 EF_RDD

[NX1 EF]

Send Read Data command to Embedded Flash. Read data from specified address.

EF_RDD(Addr, Result)

Addr: Specify 20-bit address of reading data which can be immediate value or variable specified.

When specified by a variable, high bit variable can be omitted, the omitted bit is defaulted as 0.

Result: Save the variable of reading data.

Ex.

[Variable]

Var8: Data

Var32: Addr

[Path]

TR1: EF_RDD(0x4321, Data)

; Save the data in address 0x4321 into Data.

TR2: Addr=0x123, EF_RDD(Addr, Data)

; Save the data in address 0x123 into Data.

5.11.4 EF_GetAddr

[NX1 EF]

Users can add user defined file or reserved blocks through User Defined Sections page in *Q-Code*. Then use this command to get the address of each block to excute read, write or erase operations.

When users add a file or defines the reserved blocks, the adding order (the Sec field) is the index value. Specify the index value of the block in the parameter of command to obtain the corresponding address.

EF_GetAddr(Index, Result)

Index: Specify the index value of the block address which can be an immediate value or specified by variable, total is 16-bit and the range is 0~65535. When specified by a variable, high bit variables can be omitted; the omitted bit is defaulted as 0.

Result: The variable stores the start address of block. Total is 20-bit, and high bit variables can be omitted and not to be stored.

Ex.

[Variable]

Var32: Addr

Var8: Index

[Path]

TR1: EF_GetAddr(0x8, Addr)

; Read the block address of index value 8, and save to Addr.

TR2: Index=0x23, EF_GetAddr(Index, Addr)

; Read the block address of index value 0x23, and save into Addr.

5.12 Storage Command

Storage Command				
Storage_Save	-	-	-	-

5.12.1 Storage_Save

[NX1]

Store variables defined in [Storage Variable] to SPI Flash / Embedded Flash.

5.13 Comparator Command

Comparator Command				
CMP_ON	CMP_ON(Source)	CMP_OFF	CMP_Read(Rj:Ri)	CMP_Read(Xi)
CMP_CNT_ON(Count)	CMP_CNT_OFF	-	-	-

5.13.1 CMP_ON / CMP_ON(Source)

[NY6B / NY6C]

Activate Comparator function. If specify timing source frequency, the time of two comparisons will be captured. Timing source frequency are 62.5K, 250K, 1M and 4M.

Open Comparator or Capture function. If comparator triggeration is interrupted, the corresponding system path will be executed immediately.

Note:

1. When users activate Capture in the first time, the captured time of comparator will be wrong.
2. If Capture occupies Timer resource, the Timer function can not be used anymore.

Ex. Open Comparator function. If comparator is triggered, the system will execute INT_CMP.

TR1: CMP_ON ; Open Comparator function.
INT_CMP: !PB ; If comparator is triggered, the PB output is inverted immediately.

Ex. Open Capture function. When comparator is triggered, the system will execute INT_CMP. If comparator is untriggered, the system will execute INT_TMR.

TR1: CMP_ON(1M) ; Open Capture function.
INT_CMP: CMP_Read(X0) ; If comparator is triggered, the captured time will be read immediately.
INT_TMR: !PB ; If comparator is untriggered, the PB output is inverted immediately.

5.13.2 CMP_OFF

[NY6B / NY6C]

Close Comparator function.

Ex. TR2: CMP_OFF ; Close Comparator function.

5.13.3 CMP_Read(Rj:Ri) / CMP_Read(Xi)

[NY6B / NY6C]

Read the count of Capture.

Ex. Save the count of Capture to X0.

INT_CMP: CMP_Read(X0) ; Save the received count of Capture to Register X0.

Ex. Save the count of Capture to R0, R1.

INT_CMP: CMP_Read(R1:R0) ; Save the received count of Capture to Register R0, R1.

5.13.4 CMP_CNT_ON (Count)

[NY6B / NY6C]

Activate the external input count function, it needs to specify count times. If the count times achieve, system will execute INT_TMR.

Note: If Counter occupy Timer resource, the Timer function can not be used anymore.

Ex. Open Counter function. If Counter is set to 10 times, the PB output will be inverted immediately.

TR1: CMP_CNT_ON(10) ; Open Counter function and set to count 10 times.
INT_TMR:!PB ; If count up to 10 times, the PB output will be inverted immediately.

5.13.5 CMP_CNT_OFF

[NY6B / NY6C]

Close the Counter function.

Ex. TR2: CMP_CNT_OFF ; Close the Counter function.

5.14 Counter Command

Counter Command				
TMR_ON	TMR_OFF	TMR_Read(Rj:Ri)	TMR_Read(Xi)	-

5.14.1 TMR_ON

[NY5+ / NY6 / NX1]

Activate Timer function. It needs to specify count times.

TMR_On(Time)

TMR_On(Timer, Time)

Timer: timer.

- NY5 supports TMR0 / TMR1 / TMR2 / TMR3.
- NX1 supports TMR0 / TMR1 / TMR2 / TMR3 / PWMA / PWMB.

Time: time to count.

- NY5 supports 64 ~ 256us.
- NY6 supports 1.00ms ~ 4.00ms.
- In NX1 OTP, time interval depends on High Clock Frequency.
 - 64 ~ 5461us when High Clock Frequency is 12MHz.
 - 64 ~ 4096us when High Clock Frequency is 16MHz.
 - 64 ~ 2730us when High Clock Frequency is 24MHz.
 - 64 ~ 2048us when High Clock Frequency is 32MHz.
- NX1 EF supports 64 ~ 1365us.

Note:

1. **NY6 does not support Timer parameter.**
2. **In NY6, the minimum time unit is 64us when Time is between 1.00ms ~ 2.00ms, 256us when Time is between 2.01ms ~ 4.00ms. If Time is not divisible by minimum time unit, every interrupt time will have errors.**
3. **In NY5+, only 1 Timer can be activated at the same time. Activating one Timer will deactivate another timer.**

Ex. Open Timer function. It's interrupted once per 1.5ms. If the interrupt occurs, system will execute INT_TMR.

TR1: TMR_ON(1.5ms)

; Open Timer function and interrupt once per 1.5ms.

INT_TMR:!PA

; If Timer is interrupted, the PA output is inverted immediately.

5.14.2 TMR_OFF

[NY5+ / NY6 / NX1]

Close the Timer function.

TMR_Off

TMR_Off(Timer)

Timer: timer.

- NX1 supports TMR0 / TMR1 / TMR2 / TMR3 / PWMA / PWMB.

Note: NY5+ / NY6 does not support Timer parameter.

Ex. TR2:TMR_OFF ; Close the Timer function.

5.14.3 TMR_Read(Rj:Ri) / TMR_Read(Xi)

[NY6]

Read count of Timer.

Ex. Save the count of Timer X0.

TR1: TMR_Read(X0) ; Save the received count of Timer to Register X0.

Ex. Save the count of Timer to R0, R1.

TR1: TMR_Read(R1:R0) ; Save the received count of Timer to Register R0, R1.

5.15 MIDI Command

MIDI Command				
PlayM	PlayMS	WaitMN	PauseM	ResumeM
StopM	Instrument(Ch, i)	M_Chx_Vol = n	M_Chx_Vol = Ri	Ri = M_Chx_Vol
Tempo + n	Tempo - n	Tempo++	Tempo--	TempoRst
Tempo(Rj:Ri)	Tempo(Xi)	ReadTempo(Rj:Ri)	ReadTempo(Xi)	Mute_On(Ch)
Mute_Off(Ch)	OKON_On	OKON_Off	OKON_Play	OKON_SustainOn
OKON_SustainOff	OKON_SustainEnd	DynamicOn	DynamicOff	StopMNote
MIDI_Pitch	Mask_On(Ch)	Mask_Off(Ch)	ReadFileCountM	MIDI_Loop_On
MIDI_Loop_Off	-	-	-	-

5.15.1 PlayM / PlayMS

[NY5 / NY5+ / NY6 / NY7 / NX1]

This command provides user with simple commands to play music files.

The format of command is shown as following:

PlayM ({Ch,} Parameter {++} {, Storage}) {*n} {& [BG1{,BG2 {,BG3}}] }

PlayMS ({Ch,} Parameter {++} {, Storage}) {& [BG1{,BG2 {,BG3}}] }

PlayM: execute next command after finishing playing Melody file.

PlayMS: execute next command after starting playing Melody file.

Ch: specify audition channel.

- NY5 / NY5+ / NY6 / NY7 do not support Ch parameter.
- NX1 supports 0 ~ 3 or Ch0 ~ Ch3.

Parameter: specify music file.

- Melody file label.
- Ri (Supports 1 ~ 4 Ri) or Xi (Supports 1 ~ 2 Xi). The value of Ri / Xi indicates the nth melody file. Ri / Xi is automatically increased by 1 if “++” is specified.

Storage: Location of voice file to be played. Internal Storage if omitted.

- NY5 / NY5+ / NY6 / NY7 do not support Storage parameter.
- NX1 supports SPI0 / SPI1.

n: numbers of repeated playback.

- NY5 / NY5+ / NY6 / NY7 support constant 1 ~ 15.
- NX1 supports 1 ~ 255.

BG1: background1 action:

- BG1: can call path defined in **[Background1]**.
- X: does not change the current status of Background1 path.
- OFF: stop the execution of current Background1 path.

BG2: background2 action:

- BG2: can call path defined in **[Background2]**.
- X: does not change the current status of Background2 path.
- OFF: stop the execution of current Background2 path.

BG3: background3 action (**NX1 Only**):

- BG3: can call path defined in **[Background3]**.
- X: does not change the current status of Background3 path.
- OFF: stop the execution of current Background3 path.

Ex.

P1: PlayM(\$M0),PB.3=1

; Execute PB.3=1 after playing Melody file.

P2: PlayM(R3:R2:R1:R0),PB.3=1

; Execute PB.3=1 after playing the music file specified by register value (If R3=0x0, R2=0x3, R1=0xF, R0=0x1, plays no.M1009 music file).

P3: PlayM(X1:X0),PB.3=1

; Execute PB.3=1 after playing the music file specified by register value (If X1=0x1, X2=0xF2,

P4: PlayM(X1:X0++),PB.3=1

plays no.M498 music file).

; Execute PB.3=1 after playing the music file specified by register value (If X1=0x01, X0=0xF2, plays no.M498 music file, X0 automatically increase by 1, X0=0xF3).

P5: PlayMS(\$M0),PB.3=1

; Play music file and simultaneously execute PB.3=1.

P6: PlayMS(R3:R2:R1:R0),PB.3=1

; Play the music file by register value and simultaneously execute PB.3=1 (If R3=0x0, R2=0x3, R1=0xF, R0=0x1, plays no.M1009 music file).

P7: PlayMS(X1:X0),PB.3=1

; Play music file specified by register value and simultaneously execute PB.3=1 (If X1=0x01, X0=0xF2, plays no.M498 music file).

P8: PlayMS(X1:X0++),PB.3=1

; Play music file specified by register value and simultaneously execute PB.3=1 (If X1=0x01, X0=0xF2, plays no.M498 music file, X0 automatically increases by 1, X0=0xF3).

Ex. Play MIDI on SPI Flash.

[Variable]

Var8: R0

[Memory]

SPI0_File = Spi.spirj

[Path]

PowerOn: R0 = 0

Path1: PlayM(Ch0, R0++, SPI0)

; Play nth MIDI on SPI Flash, which n is specified by R0.

5.15.1.1 PlayM / PlayMS with Table

Ex.

[Table]

Tab:

```
{
[0x001, 0x003, 0x005, 0x007, 0x009],
[0x004, 0x005, 0x3F6, 0x008, 0x010],
[0x000, 0x001, 0x002, 0x003, 0x0F4] }
```

[Path]

TR1: TableL(Tab,1,1,R0), PlayM(R0), PB.3=1

; Store the specified address value by Table to R0, execute PB.3=1 after playing the music file specified by R0 (R0=0x5, plays

TR2: TableL(Tab,1,1,X0), PlayM(X0), PB.3=1

no.M5 music file).

; Store the specified address value by Table to X0, execute PB.3=1 after playing the music file specified by X0 (X0=0x05, plays no.M5 music file).

TR4: TableL(Tab,1,1,R0), PlayMS(R0), PB.3=1

; Store the specified address value by Table to R0, play the music file specified by R0 and execute PB.3=1 simultaneously (R0=0x5, plays no.M5 music file).

TR5: TableL(Tab,1,1,X0), PlayMS(X0), PB.3=1

; Store the specified address value by Table to X0, play music file specified by X0 and simultaneously execute PB.3=1 (X0=0x05, plays no.M5 music file).

Action	Command	Description
Labeled Music file	PlayM(\$M0)	Play Label "M0".
Play by register value	PlayM(X0)	Play X0 value to music file.
Numbers of repeated playback	PlayM(\$M0)*n	Label "M0" will be played n times.
Call path defined in Background while playing Voice file.	PlayM(\$M0)&[BG1,BG2]	Call path defined in Background1 and Background2 while playing Label "M0".
	PlayM(\$M0)&[X,BG2]	Does not change the current status of BG1.
	PlayM(\$M0)&[BG1,X]	Does not change the current status of BG2.
	PlayM(\$M0)&[OFF,BG2]	Stop the execution of current BG1.
	PlayM(\$M0)&[BG1,OFF]	Stop the execution of current BG2.

Ex.

[Melody Database]

D:\.qcODE\TEST\ADSR.qmd

; M0 = Music.mid, 4 channel

; M1 = Start.mid, 4 channel

[Path]

TR1: PlayM(\$M0)

; Play number M0 of melody file.

TR2: PlayM(\$M0)*3

; Play number M0 of the Melody file 3 times.

TR3: PlayM(\$M0)&[BG1, BG2]

; Play number M0 while calling Background1 and Background2.

[Background1]

BG1:

[Background2]

BG2:

5.15.2 WaitMN

[NY5 / NY5+ / NY6 / NY7 / NX1]

If Melody is being played, it will execute next command after finishing playing Melody. Otherwise, it will immediately execute the next command.

Note: $PlayM = PlayMS + WaitMN$.

Ex.

[Path]

TR1: PlayM(\$M0)

TR2: PlayMS(\$M0), WaitMN ; The result is the same as TR1.

5.15.3 PauseM

[NY5 / NY5+ / NY6 / NY7 / NX1]

PauseM is to pause the current playback of melody.

Ex.

PowerOn: PlayM(\$M0)

TR1: PauseM ; Pause the playback of melody.

5.15.4 ResumeM

[NY5 / NY5+ / NY6 / NY7 / NX1]

This command can be used to resume the paused playback of melody data. This command has no effect for the following conditions:

1. The playback of melody data is finished.
2. StopM or Stop commands have been executed.

Ex.

PowerOn: PlayM(\$M0)

TR1: PauseM ; Pause the playback of melody.

TR2: ResumeM ; Resume the playback of melody.

5.15.5 StopM

[NY5 / NY5+ / NY6 / NY7 / NX1]

StopM command is to immediately stop the current playback of melody data.

Ex.

PowerOn: PlayM(\$M0)

TR1: StopM ; Stop the playback of melody.

5.15.6 Instrument(Ch, i)

[NY5+ / NY6 / NY7 / NX1]

Instrument command is to change the timbre of the playing Melody.

Ch: MIDI channel to change instrument.

- NY5+ supports 1 ~ 4, 10.
- NY6 supports 1 ~ 6, 10.
- NY7 supports 1 ~ 16.
- NX1 supports 1 ~ 16

i: GM timbre, can be immediate value or variable. Range 0 ~ 127.

Note:

1. *This command only takes effect when Melody is playing.*
2. *If the GM timbre is specified by an variable and timbre does not exist, it may result in switching error or abnormal playback.*

Ex. Instrument(Ch2, 1) ; Change the timbre of MIDI channel 2 as timbre number 1.

Ex. Instrument(Ch1, R1:R0) ; Replace the timbre of MIDI channel 1 to R1:R0 of GM timbre forcibly; R1 is high 3 bit and R0 low 4 bit.

Ex. Instrument(Ch3, X0) ; Replace the timbre of MIDI channel 3 to the GM timbre number of X0 forcibly.

5.15.7 M_Chx_Vol = n / M_Chx_Vol = Ri

[NY5+ / NY6 / NY7 / NX1]

Modify the channel volume of the playing Melody respectively. The level of channel can be immediate value or Ri.

Chx: specify MIDI channel.

- NY5+ supports 1 ~ 4, 10.
- NY6 supports 1 ~ 6, 10.
- NY7 supports 1 ~ 16.
- NX1 supports 1 ~ 16.

n: set channel volume

- NY5+ supports 0 ~ 7 °
- NY6 supports 0 ~ 7 °
- NY7 supports 0 ~ 15 °
- NX1 supports 0 ~ 15 °

Note: *When Melody starts playing, the level of each MIDI channel volume is default as 15.*

Ex. M_Ch1_Vol = 10 ; Set the volume of Ch1 as 10, 67% to the original.

Ex. M_Ch1_Vol = R0 ; Set the volume of Ch1 by the value of R0.

5.15.8 Ri = M_Chx_Vol

[NY5+ / NY6 / NY7 / NX1]

Access the individual channel volume of the Melody being played.

Chx: specify MIDI channel.

- NY5+ supports 1 ~ 4, 10.
- NY6 supports 1 ~ 6, 10.
- NY7 supports 1 ~ 16.
- NX1 supports 1 ~ 16.

Note: When the melody is playing, the default volume of each channel is set to 15.

Ex. R0 = M_Ch1_Vol ; Access the volume of Ch1 and save to R0.

5.15.9 Tempo + n

[NY5+ / NY6 / NY7 / NX1]

Tempo + n command can accelerate the playing speed of Melody. “n” is the percentage of acceleration, and its unit is 5%. When the ratio is less than the multiple of 5, Q-Code will round the number to the nearest.

Note: This command only takes effect when Melody is playing.

Ex. Tempo=100

[Path]

PowerOn: PlayMS(\$M0), Tempo + 10 ; Accelerate 10% of Tempo speed as 110.

5.15.10 Tempo - n

[NY5+ / NY6 / NY7 / NX1]

Tempo – n command can slow down the playing speed of Melody. “n” is the percentage of slowdown, and its unit is 5%. When the ratio is less than the multiple of 5, Q-Code will round the number to the nearest.

Note: This command only takes effect when Melody is playing.

Ex. Tempo=100

[Path]

PowerOn: PlayMS(\$M0), Tempo - 5 ; Slow down 5% of Temp speed as 95%.

5.15.11 Tempo++

[NY5 / NY5+ / NY6 / NY7 / NX1]

Tempo++ command can control the playback speed of melody file to be faster by shifting up one level.

Note:

1. NY5 tempo level refer to Tempo = n.
2. For NY5+ / NY6 / NY7 / NX1, this command control the playback speed to be faster by shifting 5% up of each level. And this command only takes effect when Melody is playing.

Ex.**PowerOn:** PlayMS(\$M0)**TR1:** Tempo++ ; Set Tempo to 105%.**5.15.12 Tempo--****[NY5 / NY5+ / NY6 / NY7 / NX1]**

Tempo-- command can control the playback speed of melody file to be slower by shifting down one level.

Note:

1. NY5 tempo level refer to Tempo = n.
2. For NY5+ / NY6 / NY7 / NX1, this command control the playback speed to be slower by shifting 5% down of each level. And this command only takes effect when Melody is playing.

Ex.**PowerOn:** PlayMS(\$M0)**TR1:** Tempo-- ; Set Tempo to 95%.**5.15.13 TempoRst****[NY5+ / NY6 / NY7 / NX1]**

TempoRst command can reset the play speed of Melody.

Note: This command only takes effect when Melody is playing.

Ex. Tempo=100**PowerOn:** PlayMS(\$M0), Tempo - 5**TR1:** TempoRst ; Set Tempo to the original.**5.15.14 Tempo = n****[NY5]**

Tempo could control the playback speed of melody.

n: Tempo value, 34~500.

Tempo List			
34	51	76	150
36	53	86	170
38	55	93	200
41	58	100	235
45	64	110	300
47	68	120	500
49	72	135	

Ex. Tempo=100

[Path]

PowerOn: Tempo=100 ; Set Tempo as 100.

5.15.15 Tempo(Rj:Ri) / Tempo(Xi)

[NY5]

Set Tempo value according to register value.

The formula of Tempo value is as follows:

Time base = 2.048ms

Value = 60000(ms) / Tempo / 24 / Timebase

Ex. Tempo=100

PowerOn: X0=12, Tempo(X0) ; Set Tempo as 100.

5.15.16 ReadTempo(Rj:Ri) / ReadTempo(Xi)

[NY5 / NY5+ / NY6 / NY7 / NX1]

Save the current Tempo value to RAM.

Ex.

[Path]

TR1: PlayM(\$M0) ; Play melody.

TR2: ReadTempo(X0), [PD,PC]=X0 ; Output the tempo value to PD and PC.

5.15.17 Mute_On(Ch)

[NY5 / NY5+ / NY6 / NY7 / NX1]

This command activates the mute function. Mute_On can be used to mute specific channel during the playback of melody.

Ch: MIDI channel to mute. If Ch is not designated, it is regarded as all MIDI channels.

- NY5: 0 ~ 3.
- NY5+: 1 ~ 4, 10.
- NY6: 1 ~ 6, 10.

- NY7: 1 ~ 16.
- NX1: 1 ~ 16.

Note:

1. For NY5+ / NY6 / NY7, this command only takes effect when Melody is playing.
2. For NX1, the Mute_Off command isn't restricted by melody playback in Q-Code 6.50 or above version

Ex.

[Path]

PowerOn : PlayM(\$M0)

TR1: Mute_on(1) ; Activate mute function on MIDI Ch1 during melody playback.

TR2: Mute_on(3) ; Activate mute function on MIDI Ch3 during melody playback.

5.15.18 Mute_Off(Ch)

[NY5 / NY5+ / NY6 / NY7 / NX1]

This command stops the mute function. If a channel has been muted, Mute_off command can be used to stop muting the channel.

Ch: channel to stop muting. If Ch is not designated, it is regarded as all MIDI channels.

- NY5: 0 ~ 3.
- NY5+ :1 ~ 4, 10.
- NY6: 1 ~ 6, 10.
- NY7: 1 ~ 16.
- NX1: 1 ~ 16.

Note: For NY5+ / NY6 / NY7 / NX1, this command only takes effect when Melody is playing. It doesn't work when executed before playing Melody.

Ex.

[Path]

PowerOn : PlayM(\$M0)

TR1: Mute_on(1) ; Activate mute function on MIDI Ch1 during melody playback.

TR2: Mute_on(3) ; Activate mute function on MIDI Ch3 during melody playback.

TR3: Mute_off(1) ; Stop the mute function on MIDI Ch1.

TR4: Mute_off(3) ; Stop the mute function on MIDI Ch3.

5.15.19 OKON_On / SingleNote_On

[NY5 / NY5+ / NY6 / NY7 / NX1]

One-Key-One-Note is activated, and user can use this function with the command OKON_on.

Note:

1. NY5 / NY5+ OKON channel is fixed on MIDI channel 0.

2. *NY6 / NY7 OKON channel is fixed on MIDI channel 1. This command only takes effect when Melody is playing.*
3. *NX1 OKON channel is fixed on MIDI channel 1.*

Ex.

[Path]

PowerOn:

TR1: PlayM(\$M0)	; Play melody.
TR2: OKON_On	; Activate One-Key-One-Note function and play
	; Note on channel 1.

5.15.20 OKON_Off / SingleNote_Off

[NY5 / NY5+ / NY6 / NY7 / NX1]

User can use the command OKON_Off to stop the function of One-Key-One-Note.

Note: For NY5+ / NY6 / NY7, this command only takes effect when Melody is playing.

Ex.

[Path]

PowerOn:

TR1: PlayM(\$M0)	; Play melody.
TR2: OKON_On	; Activate One-Key-One-Note function and play;
	Note on MIDI channel 1.
TR3: OKON_Off	; Deactivate One-Key-One-Note function.

5.15.21 OKON_Play / SinglePlay

[NY5 / NY5+ / NY6 / NY7 / NX1]

User can use OKON_Play command to start the playback of One-Key-One-Note.

Note:

1. *NY5 / NY5+ OKON channel is fixed on MIDI channel 0.*
2. *NY6 / NY7 OKON channel is fixed on MIDI channel 1. This command only takes effect when Melody is playing.*
3. *NX1 OKON channel is fixed on MIDI channel 1.*

Ex.

[Path]

PowerOn:

TR1: PlayM(\$M0)	; Play melody.
TR2: OKON_On	; Activate One-Key-One-Note function and play
	; Note on channel 1.
TR3: OKON_Play	; Execute SinglePlay.

5.15.22 OKON_SustainOn

[NY5+]

Enable OKON sustain function. When OKON sustain is enabled and OKON_Play is executed, midi note is played continuously until OKON_SustainEnd is executed if note contains sustain stage.

Ex.

[Input State]

Input_0: TR1 TR2 TR3 TR4/TR4_R

[Path]

PowerOn: Input_0

TR1: PlayMS(\$M0), OKON_On	; Play mdi, and enable OKON.
TR2: OKON_SustainOn	; Enable OKON sustain function.
TR3: OKON_SustainOff	; Disable OKON sustain function.
TR4: OKON_Play	; Press key and start playing note. If OKON sustain is enabled, sustain stage of midi note is played continuously.
TR4_R: OKON_SustainEnd	; Release key, and stop playing of midi note.

5.15.23 OKON_SustainOff

[NY5+]

Disable OKON sustain function. When OKON sustain function is disabled, midi note will be played according to the duration set in Q-MIDI when OKON_Play is executed.

5.15.24 OKON_SustainEnd

[NY5+]

Stop sustain effect of current midi note of OKON playing. If the duration between OKON_Play and OKON_SustainEnd is less than sustain duration set in Q-MIDI, note is still played according to the duration set in Q-MIDI, otherwise, midi note playing will enter release stage immediately.

5.15.25 DynamicOn

[NY5+ / NY6 / NY7]

Open the channel dynamic allocation function. After achieving this function, Q-Code will allocate a suitable channel to play the notes of melody file which is based on the current status of synthesizer channels. When users compose, the notes can be playing on channel 1 ~ 4, 10 channel (NY5+) / 1 ~ 6, 10 channel (NY6) / 1 ~ 16 channel (NY7) normally. Users must notice that the number of notes cannot exceed the number of synthesizer channels. The default is DynamicOn.

Ex.

PowerOn: PlayMS(\$M0)

TR1: DynamicOn ; Open the dynamic allocation function.

5.15.26 DynamicOff

[NY5+ / NY6 / NY7]

Close the channel dynamic allocation function. After closing this function, Q-Code will play the notes of melody file by using the same number of synthesizer channels. When users compose, notes can be on channel 1 ~ 4, 10 (NY5+) / 1 ~ 6 (NY6) / 1 ~ 8 (NY7) only, and one note at once per channel; otherwise, the fore note will be replaced by the later note as notes overlap. When the MIDI channel is playing voice or single note, all notes on the channel will be ignored until the playback is finished.

Note: When the dynamic allocation function is disabled, the note of Percussion will be designated to play on the hardware channel 3 (NY5+) / channel 5 (NY6) / channel 7 (NY7).

Ex.

PowerOn: PlayMS(\$M0)

TR1: DynamicOff ; Close the dynamic allocation function.

5.15.27 StopMNote

[NY7]

Stop the playing note, but it will not stop the MIDI playing. That is, the other notes will still play normally after stopping the current note. In addition, StopMNote command does not affect the playback of single note which played using the InstNoteOn and DrumNoteOn commands.

StopMNote command can be used with the PauseM command to stop the note immediately when pausing MIDI. That will prevent unexpected delay.

Ex.

PauseM, StopMNote ; Pause MIDI playing and stop the playing note.

5.15.28 MIDI_Pitch(Semitone)

[NX1]

This command is for up and down the tone of MIDI playback.

Semitone: The setting range is limited by Q-MIDI, the maximum can be ± 12 semitones. The default is 0.

Ex.

MIDI_Pitch(-3), PlayMS(Ch0, \$M0) ; Lower the pitch by 3 semitones and play MIDI.

5.15.29 Mask_On(Ch)

[NX1]

Mask on the MIDI channel. When the melody file plays, the Mask_on command can mask the specified MIDI channel to reduce the loading of CPU.

Ch: The specified MIDI channel, it will mask all MIDI channels when not specified.

- NX1 supports 1 ~ 16.

Ex.

[Path]

PowerOn : PlayM(Ch0, \$M0)

TR1: Mask_on(1) ; Turn on the Mask function of MIDI channel 1.

TR2: Mask_on(3) ; Turn on the Mask function of MIDI channel 3.

5.15.30 Mask_Off(Ch)

[NX1]

Turn off the Mask function on the specified MIDI channels. When the Mask_On command is executed, user can revert the MIDI channels by using the Mask_Off command.

Ch: The specified MIDI channel, it will turn off the mask function on all MIDI channels when not specified.

- NX1 supports 1 ~ 16.

Ex.

[Path]

PowerOn : PlayM(Ch0, \$M0)

TR1: Mask_on(1) ; Turn on the Mask function of MIDI channel 1.

TR2: Mask_on(3) ; Turn on the Mask function of MIDI channel 3.

TR3: Mask_off(1) ; Turn off the Mask function of MIDI channel 1.

TR4: Mask_off(3) ; Turn off the Mask function of MIDI channel 3.

5.15.31 ReadFileCountM

[NX1]

Read MIDI file count.

ReadFileCountM(Ri)

ReadFileCountM(Ri, Storage)

Ri: Result MIDI file count. Range 0~65535.

Storage: Specify storage of melody database. Accepts SPI0 / SPI1. MIDI count in **[Melody Database]** if omitted.

Ex.

[Variable]

Var16: midi_max=0, midi_idx=0

[Path]

PowerOn: ReadFileCountM(midi_max) ; Read midi file count.

TR1: midi_idx++, ; midi_idx+1.
midi_idx>=midi_max?{midi_idx=0}, ; index exceed max value, reset to 0.
PlayM(ch0, midi_idx) ; Play MIDI which index is midi_idx.

5.15.32 MIDI_Loop_On

[NX1]

Enable MIDI loop play.

Ex.

PowerOn: InputState

TR1: PlayM(ch0, 0, SPI0) ; Play first MIDI stored on SPI0.

TR2: MIDI_Loop_On ; Enable MIDI loop play.

TR3: MIDI_Loop_Off ; Disable MIDI loop play.

Note: This command is deprecated since Q-Code 8.20, its recommended to replace with [Audio Loop On](#).

5.15.33 MIDI_Loop_Off

[NX1]

Disable MIDI loop play. Current playing MIDI will not be stopped immediately until the end of entire song.

Note: This command is deprecated since Q-Code 8.20, its recommended to replace with [Audio Loop Off](#).

5.16 Keyboard Command

Instrument Command				
InstNoteOn	InstNoteOff	InstNoteAllOff	DrumNoteOn	DrumNoteOff
NoteVibrato	Gliss	MaxSingleNote	LongInst_HoldTime	ShortInst_HoldTime
KRecord	KRecordS	WaitKRN	StopKR	PlayK
PlayKS	WaitKN	StopK	-	-

5.16.1 InstNoteOn(Index, Note, Vol) & InstNoteOff(Note)

[NY5+ / NY6 / NY7 / NX1]

The InstNoteOn and InstNoteOff commands can be used as single note of instruments' key, but these commands must be paired. When user presses one key, the InstNoteOn command will play one note and occupy one channel until user releases the key or the end of play. When the key is released, the InstNoteOff command will stop play and release the occupied channel.

Index: Specify the index of GM Pitched Instrument. User can designate the value via an immediate value or variables.

Note: {"C1", "C#1", "D1", "D#1", "E1", "F1", "F#1", "G1", "G#1", "A2", "A#2", "B2"... "B8"}

Vol: The velocity of single tone, its range is 0 ~ 127. User can designate the value via an immediate value or variables.

Ex.

Path1: InstNoteOn (0, C1, 127) ; **Falling Edge. Play timbre number 0, pitch C1, intensity 127.**

Path2: InstNoteOff (C1) ; **Rising Edge. Release pitch C1.**

Ex.

Path1: InstNoteOn (X0, C3, 127) ; **Falling Edge. Use the X0 designated timbre, pitch C3, intensity 127**

Path2: InstNoteOff (C3) ; **Rising Edge. Release pitch C3.**

Ex.

Path1: InstNoteOn (0, C4, X1) ; **Falling Edge. Play timbre number 0, pitch C4, using the velocity specified by X1.**

Path2: InstNoteOff (C4) ; **Rising Edge. Release pitch C4.**

Ex.

Path1: InstNoteOn (X0, C4, X1) ; **Falling Edge. Use the X0 designated timbre, pitchC4 and the intensity specified by X1.**

Path2: InstNoteOff (C4) ; **Rising Edge. Release pitch C4.**

5.16.2 InstNoteAllOff

[NX1]

The InstNoteAllOff command can be applied to keyboard functions of instrument and percussion. It will release all the notes of current playback instruments and percussions.

Ex.

Path1: InstNoteAllOff ; **Release all the current playing notes.**

5.16.3 DrumNoteOn(Index, Vol) & DrumNoteOff(Index)

[NY5+ / NY6 / NY7]

The DrumNoteOn and DrumNoteOff commands can be used as single note of percussions' key, but these commands must be paired. When user presses one key, the DrumNoteOn command will play one note and occupy one channel until user releases the key or the end of play. When the key is released, the DrumNoteOff command will stop play and release the occupied channel.

Index: Specify the index of GM Non-Pitched Instrument. User can designate the value via an immediate value or variables.

Vol: The velocity of single tone, its range is 0 ~ 127. User can designate the value via an immediate value or variables.

Ex.

Path1: DrumNoteOn (0, 127) ; Falling Edge. Play the percussion number 0, intensity 127.

Path2: DrumNoteOff (0) ; Rising Edge. Release the percussion number 0.

Ex.

Path1: DrumNoteOn (X0, 127) ; Falling Edge. Use the X0 designated percussion, intensity 127.

Path2: DrumNoteOff (X0) ; Rising Edge. Release the X0 designated percussion timbre.

Ex.

Path1: DrumNoteOn (35, X1) ; Falling Edge. Play the no.35 percussion timbre and use the X1 designated intensity.

Path2: DrumNoteOff (35) ; Rising Edge. Release the no.35 percussion timbre.

Ex.

Path1: DrumNoteOn (X0, X1) ; Falling Edge. Use the X0 designated percussion timbre and the X1 designated intensity

Path2: DrumNoteOff (X0) ; Rising Edge. Release the X0 designated percussion timbre.

5.16.4 NoteVibrato = n / NoteVibrato = Ri

[NY7 / NX1]

When setting the vibrato effect of timbre through Q-MIDI, users can use this command to achieve the vibrato effect of single note.

n / Ri: is the control parameter of vibrato, user can set the value by immediate value or Ri. The range is 0~8, 0 means close, whereas 1 ~ 8 can set the effect of depth—the greater the value, the wider the depth.

Note: The timbre of Percussion doesn't support the vibrato function.

Ex.

Path1: NoteVibrato = 8, InstNoteOn (0, C1, 127) ; Open the vibrato effect and play the No.0 GM, pitch C1, dynamics 127.

Path3: NoteVibrato = 0 ; Close the vibrato effect.

5.16.5 Gliss(Note, Semitone, Time)

[NY7 / NX1]

When a single note is playing, user can achieve the effect of glissando by using this command.

Note: The pitch of note which applied the effect.

Semitone: Designated the pitch number of semitone. The number of pitch range is +/-12 semitones.

Time: The duration.

- NY7 accepts 100ms, 200ms, 300ms, 400ms, 500ms, 600ms, 700ms, 800ms, 900ms, 1000ms, 1200ms, 1400ms, 1600ms, 1800ms and 2000ms.
- NX1 supports 100ms ~ 2000ms.

Note:

1. *Percussion timbres do not support the glissando effect.*
2. *The command is available only when the note is playing and corresponding to the designated pitch. According to different conditions, there will be about +/-10% deviation of time.*
3. *In NY7, user needs to add .mid files with control code "Wheel" to define the pitch changing range when making MIDI Database (.qmd) by Q-MIDI, or Gliss command will be ineffective.*
4. *In NY7, the largest pitch range is the range of the original sub-patch plus the range of pitch bend. For example, the sub-patch is C3~D4, and the range of pitch bend is +/-4 semitones, so the available pitch range is G#2~F#4.*

Ex.

Path1: InstNoteOn(0, F4, 127),Gliss(F4, -2, 300ms), Delay(300ms), Gliss(F4, -4, 100ms)

Path2: InstNoteOff(F4)

; After playing the F4 tone, gliding 2 semitones as D#4 for 300ms then lowering 4 semitones as B3 for 100ms.

5.16.6 MaxSingleNote = n / MaxSingleNote = Ri

[NY5+ / NY6 / NY7]

User can adjust the maximum number of notes based on requirements through this command which can be applied to electronic organ. The command can make sure that there are enough channels for playing MIDI.

Ex.

Path1: MaxSingleNote = 3 ; Set the max note number as 3 notes.

Path2: R0=5, MaxSingleNote = R0 ; Set the max note number as 5 notes via R0.

Note: The value of the command must be less than or equal to the value of Melody_MaxSingleNote option. Otherwise, it will be invalid.

5.16.7 LongInst_HoleTime

[NX1]

This command is for instruments which contains Sustain. When the instrument is NoteOn, it will hold

the longest playing note.

Time: Specify the longest playing time of the instrument, the supported time is 1ms ~ 131068ms. 0 means play instrument with unlimite time, and the default is 0.

Ex.

Path1: LongInst_HoldTime(16000) ; Set the instrument holds for 16 seconds.
Path2: LongInst_HoldTime(0) ; Set the instrument holds for unlimite time.

5.16.8 ShortInst_HoldTime

[NX1]

This commands is for instruments which doesn't contain Sustain. When the instrument id NoteOff, it will hold the time then connect with Envelope Release.

Time: Specify the delay time for connecting with Envelope Release, the support time is 0ms~1020ms, and the default is 0.

Ex.

Path1: ShortInst_HoldTime(300) ; Set the instrument delay 300ms then coonects with Envelope Release after NoteOff.

5.16.9 KRecord / KRecordS

[NX1]

Start the keyboard recording to records the Note and Vol of InstNoteOn / InstNoteOff. After recording, user can play the recording through PlayK.

KRecord(Label)

KRecordS(Label)

KRecord: After recording (includes Timeout), executes the next command.

KRecordS: Execute the next command immediately as starting recording.

Label: the recording section defined in [Record].

Note:

1. When KRecord / KRecordS is executed, the InstNoteOn or InstNoteOff must be triggered once to enter the keyboard recording mode completely to enable the Timeout function.
2. When KRecord / KRecordS is executed, any trigger to InstNoteOn / InstNoteOff will reset the timeout counter.
3. KRecord / KRecordS don't support Realtime Erasing, please work with EraseR / EraseRS to erase the recording memory.
4. For PlayKS command, when the recording memory is full, it causes Timeout, the KRecord

will end recording and execute the next command.

Ex.

[Path]

TR1: EraseR(\$Rec0), KRecord(\$Rec0)	; Erase Rec0 and than use KRecord to recording.
TR2: PlayK(\$Rec0)	; Use PlayK to playback the recording.
TR3: InstNoteOn (0, C4, 127)	; Falling Edge. Play No.0, pitch C4, dynamics 127.
TR4: InstNoteOff (C4)	; Rising Edge. Release pitch C4 single note.
TR5: InstNoteOn (0, D4, 127)	; Falling Edge. Play No.0, pitch D4, dynamics 127.
TR6: InstNoteOff (D4)	; Rising Edge. Release D4 single note.
TR7: InstNoteOn (0, E4, 127)	; Falling Edge. Release No.0, pitch E4, dynamics 127.
TR8: InstNoteOff (E4)	; Rising Edge. Release pitch E4 single note.

5.16.10 WaitKRN

[NX1]

If the keyboard recording is executing, executes the next command after recording. If there's no keyboard recording, executes the next command immediately.

Note: KRecord=KRecordS+WaitKRN.

Ex.

[Path]

TR1: KRecord(\$Rec0)	
TR2: KRecordS(\$Rec0), WaitKRN	; Same executing result with TR1.

5.16.11 StopKR

[NX1]

The StopKR command is for stop the current keyboard recording immediately.

Ex.

TR1: KRecordS(\$Rec0)	
TR2: StopKR	; Stop keyboard recording.

5.16.12 PlayK / PlayKS

[NX1]

Execute the playback of keyboard recording. After recording, user can play the recording through PlayK.

PlayK(Label, Instrument)

PlayKS(Label, Instrument)

PlayK: After the playback of keyboard ends, executing the next command.

PlayKS: After the playback of keyboard starts, executing the next command immediately.

Label: the recording section defined in **[Record]**

Instrument: Use specified instrument to play. Accepts 0 ~ 127.

Ex.

[Path]

TR1: EraseR(\$Rec0),KRecord(\$Rec0) ; After erasing Rec0, using KRecord to start keyboard recording.

TR2: PlayK(\$Rec0, 0) ; Use PlayK to play the keyboard recording, with instrument 0.

5.16.13 WaitKN

[NX1]

If the playback of keyboard is playing, executing the next command when the playback ends. If there's no keyboard playback, executes the next command immediately.

Note: PlayK=PlayKS+WaitKN.

Ex.

[Path]

TR1: PlayK(\$Rec0)

TR2: PlayKS(\$Rec0), WaitKN ; Same executing result with TR1.

5.16.14 StopK

[NX1]

The StopK command is for stopping the current playback of keyboard immediately.

Ex.

TR1: PlayKS(\$Rec0)

TR2: StopK ; Stop the playback of keyboard.

5.17 Volume Command

Volume Command				
Vol_Max	Vol_Min	Vol = n	Vol = Ri	Vol++
Vol--	Ri = Vol	Px = Vol	Vol += n	Vol -= n
VolX1	VolX2	CHx Vol = n	PP_Gain = n	PP_Gain = Ri
PP_Gain++	PP_Gain--	Ri = PP_Gain	PGA_Gain = n	PGA_Gain = Ri
Ri = PGA_Gain	Ri = MixCtrl	Px = MixCtrl	MixCtrl	-

5.17.1 Vol_Max

[NY5 / NY5+ / NY6 / NY7 / NX1]

Vol_Max can tune the volume to the maximum value (Vol = 15).

Ex. PowerOn: Vol_Max ; Set the volume to be the maximum.

5.17.2 Vol_Min

[NY5 / NY5+ / NY6 / NY7 / NX1]

Vol_Min can tune the volume to the minimum value (Vol=0).

Note: In NY5 / NX1, volume = 0 does not mean mute.

Ex. PowerOn: Vol_Min ; Set the volume to be the minimum.

5.17.3 Vol = n

[NY5 / NY5+ / NY6 / NY7 / NX1]

Directly set volume level.

n: Volume.

- In NY5 / NY5+, NY6, NY7, NX1, n = 0 ~ 15.

Note: In NY5 / NX1, volume = 0 does not mean mute.

Ex. Vol=7 ; Set volume to 8th level.

5.17.4 Vol = Ri

[NY5 / NY5+ / NY6 / NY7 / NX1]

Set volume to the value of Ri.

Note: In NY5 / NX1, volume = 0 does not mean mute.

Ex. R1=3, Vol=R1 ; Volume will be 3.

5.17.5 Vol++

[NY5 / NY5+ / NY6 / NY7 / NX1]

Vol++ command can shift up one level of current volume setting. The maximum volume in is 15.

Note: In NY5 / NX1, volume = 0 does not mean mute.

Ex. Vol=8, Vol++ ; Volume will be 9.

5.17.6 Vol--

[NY5 / NY5+ / NY6 / NY7 / NX1]

Vol-- command can shift down one level of current volume setting with the minimum volume as 0.

Note: In NY5 / NX1, volume = 0 does not mean mute.

Ex. Vol=8, Vol-- ; Volume will be 7.

5.17.7 Ri = Vol

[NY5 / NY5+ / NY6 / NY7 / NX1]

Keeps the current volume setting in Ri.

Ex. R1=Vol ; Obtain current volume level and put it at R1.

5.17.8 Px = Vol

[NY5 / NY5+ / NY6 / NY7 / NX1]

Output the current volume to Port.

Ex. PB=Vol ; Output the current level of Volume to PB.

5.17.9 Vol += n

[NX1]

Vol += n command can shift up n level of current volume setting. If the current volume increased by more than 15, it's set to 15.

Ex.

Vol += 2

5.17.10 Vol -= n

[NX1]

Vol -= n command can shift down n level of current volume setting. If the current volume is less than 0, it's set to 0.

Note: In NX1, volume = 0 does not mean mute.

Ex.

Vol -= 2

5.17.11 VolX1

[NY6]

The volume level is the original Vol level..

Ex. Vol=8, VolX1 ; Volume=1.

5.17.12 VolX2

[NY6]

The volume level is the original VOL level multiplied by 2.

Note: This command can increase volume, but it might cause burst sound, please pay attention to use.

Ex. Vol=0x8, VolX2 ; Volume=16.

5.17.13 CHx_VOL = n

[NX1]

CHx_VOL = n command controls the volume of the specified audio channel.

n: volume, from 0~15.

CHx: corresponds to the channel used by [PlayV / PlayVS](#), [SPIPlay / SPIPlayS](#) and [PlayM / PlayMS](#).

Note: Volume = 0 means mute.

Ex.

SPIPlay(Ch0,0),Ch0_vol=5 ; Set volume to 5 in channel 0.

5.17.14 PP_Gain = n

[NX1 OTP]

User can adjust Push-Pull gain by PP_Gain command, n=0~16. PP_Gain=0 means mute, PP_Gain=16 means the largest volume. The corresponding setting value to Push-Pull volume is shown below.

NX1 PP_Gain	Push-Pull Volume
16	130%
15	122%
14	114%
13	107%
12	100%
11	94%
10	88%
9	83%
8	77%
7	71%
6	66%
5	60%
4	55%

NX1 PP_Gain	Push-Pull Volume
3	50%
2	45%
1	40%
0	Mute

Ex. PP_Gain=9 ; Set PP_Gain as the 9th level.

5.17.15 PP_Gain = Ri

[NX1 OTP]

User can adjust Push-Pull gain by PP_Gain command, Ri=0 ~ 16.

Ex. R1=3, PP_Gain=R1 ; Control PP_Gain by R1.

5.17.16 PP_Gain++

[NX1 OTP]

Use PP_Gain++ to add one level on the current Push-Pull gain setting, the max value of PP_Gain is 16.

Ex. PP_Gain=8, PP_Gain++ ; PP_Gain = 9.

5.17.17 PP_Gain--

[NX1 OTP]

Use PP_Gain-- to minus one level on the current Push-Pull gain, the minimum value of PP_Gain is 0.

Ex. PP_Gain=8, PP_Gain-- ; PP_Gain = 7.

5.17.18 Ri = PP_Gain

[NX1 OTP]

Use Ri=PP_Gain to access the current Push-Pull gain setting and store at Ri.

Ex. R1=PP_Gain ; Access the current PP_Gain and store at Ri.

5.17.19 PGA_Gain = n

[NX1]

User can use PGA_Gain command to adjust the appropriate PGA_Gain, and can amplify the signals recorded by the microphone. n=0~31, 0 is the minimum (still have signal), 31 is the maximum. The default is 12.

Please refer to the table below for PGA_Gain's setting value and the amplified gain to input signal. The user can adjust the value of PGA_Gain according to the application. For voice recognition, the gain is

recommended to be 60~70. If the gain is too large, the waveform will be saturated and distorted which is bad for the voice identification.

Please note that Ropi (R10) on NX1_FDB Ver.A 2016/11/21 is 5.1K Ω , and Ropi (R5) on NX1_FDB Ver.B 2017/10/5 (informal blue board) is 2K Ω . NX1_FDB Ver.B 2018/8/22 (formal blue board) is fixed to 0 Ω .

PGA_Gain	gain@Ropi = 5.1K Ω	gain@Ropi = 2K Ω	gain@Ropi = 0 Ω
31	74.5	149.9	387.1
30	73.5	144.6	356.0
29	72.4	139.3	326.9
28	71.2	134.0	300.6
27	69.9	128.7	276.2
26	68.5	123.1	253.2
25	67.0	117.7	232.5
24	65.4	112.4	213.6
23	63.7	107.0	195.9
22	61.8	101.6	179.3
21	59.8	96.5	164.8
20	57.8	90.7	149.4
19	55.6	85.0	135.3
18	53.3	79.6	122.6
17	50.9	74.2	110.9
16	48.5	68.1	98.4
15	46.0	63.2	88.9
14	43.3	58.6	80.5
13	40.8	53.8	72.1
12	38.2	49.0	64.2
11	35.6	44.5	57.1
10	33.0	40.5	50.9
9	30.4	36.7	45.3
8	27.9	33.1	40.3
7	25.5	29.7	35.5
6	23.1	26.5	31.2
5	20.8	23.6	27.4
4	18.7	20.9	24.0
3	16.6	18.3	20.8
2	14.6	15.9	17.9
1	12.8	13.8	15.3
0	11.0	11.8	13.0

Ex. PGA_Gain=10

; Set PGA_Gain as the 10th level.

5.17.20 PGA_Gain = Ri

[NX1]

Set PGA_Gain by register.

Ex. R1=10, PGA_Gain=R1 ; Set PGA_Gain as the 10th level.

5.17.21 Ri = PGA_Gain

[NX1]

Read the current PGA_Gain to the register.

Ex. R1 = PGA_Gain

5.17.22 Ri = MixCtrl

[NY5]

Keep current setting value of MixCtrl to RAM.

Ex. R0=MixCtrl ; Keep current setting of MixCtrl in R0.

5.17.23 Px = MixCtrl

[NY5]

Assign current setting value of MixCtrl to Port.

Ex. PB=MixCtrl ; Output current setting value of MixCtrl to PB.

5.17.24 MixCtrl

[NY5 / NX1]

This command is to control the ratio of volume output for channels. For NY5 series, CH0, CH1, CH2 and CH3 are available. The available settings for all series are listed below.

MixCtrl Command	Description	Setting Value
MixCtrl(2, 0, 2, 0)	Half volume for CH0, and half volume for CH2.	0x3
MixCtrl(2, 0, 1, 1)	Half volume for CH 0, and one fourth volume for CH2 and CH3 respectively.	0x1
MixCtrl(1, 1, 2, 0)	One fourth volume for CH0 and CH1 repectively, and half volume for CH2.	0x2
MixCtrl(1, 1, 1, 1)	One fourth volume for each channel.	0x0
MixCtrl(4, 0, 0, 0)	Full volume for CH0, and mute the other channels.	0x9

MixCtrl Command	Description	Setting Value
MixCtrl(2, 2, 0, 0)	Half volume for CH0 and CH1 respectively.	0x8
MixCtrl(0, 0, 4, 0)	Full volume for CH2, and mute the other channels.	0xE
MixCtrl(0, 0, 2, 2)	Half volume for CH2 and CH3 respectively.	0xC

NX1 controls the mixing output of each channel, please set the options by selecting 0%, 25%, 50%, 75% and 100%.

Ex. MixCtrl(50%,25%,75%,100%)

; Output the current MixCtrl setting to PB.

Note:

1. For NY5 series, if MixCtrl is not defined, default is MixCtrl(1, 1, 1, 1).
2. For NX1 series, if MixCtrl is not defined, default is MixCtrl(100%, 100%, 100%, 100%).

5.18 TouchKey Command

TouchKey Command			
TouchKey_ON	TouchKey_OFF	TouchKey_CLR	TouchKey_Scan_Slow
TouchKey_Scan_Normal	TouchKey_Sensitivity	Calibrate_ON	Calibrate_OFF
AutoJudge_Calibrate	Enforce_Calibrate_Normal	Enforce_Calibrate_Sleep	Ri = TouchKey(Px)
Touchkey_Count	Touchkey_BGCount	-	-

5.18.1 TouchKey_ON

[NY9T]

Turn on the Touch-Key scanning function. When user enables the function, the Touch-Keys will keep scanning.

Note:

1. The system default is TouchKey_ON, user doesn't need to issue TouchKey_On command after the PowerOn command.
2. After issuing PowerOn command, the default scanning speed is TouchKey_Scan_Normal.
3. After issuing TouchKey_ON, the scanning status will keep the last setting.

5.18.2 TouchKey_OFF

[NY9T]

Turn off the Touch-Key scanning function. If user turns off the function, the Touch-Keys won't scan.

Ex.

[Input State]

KEY1:

[Path]

PowerOn: TouchKey_OFF, Delay(0.3), KEY1, TouchKey_ON

; When PowerOn is issued, closing the Touch-Key scanning function; then rescan Touch-Keys after delaying 0.3 second.

5.18.3 TouchKey_CLR

[NY9T]

The command is used to clear the status of Touch-Keys. When the status is clean, Touch-Keys will rescan.

Ex.

[Input State]

KEY1: TR1

[Path]

PowerOn: KEY1 ; Set the status of Touch-Key State as KEY1.

TR1: PlayA(Ch1,\$VIO0), TouchKey_CLR

; After pressing TR1, executing PlayA(Ch1,\$VIO0). When PlayA(Ch1,\$VIO0) finished playing, executing the TouchKey_CLR command, clearing the current status of Touch-Keys, then, rescanning Touch-Keys. If TR1 is pressed when VIO0 finished playing, Q-Code will keep executing PlayA(Ch1,\$VIO0). If TR1 isn't pressed, Q-Code will enter Sleep mode.

5.18.4 TouchKey_Scan_Slow

[NY9T]

When user turns on the TouchKey_Scan_Slow function, the Touch-Key will scan keys less frequently which can achieve energy saving.

Note:

1. Before entering sleep mode, please switch to the TouchKey_Scan_Slow function for saving energy.
2. When Touch-Key is detected, Q-Code will switch to the TouchKey_Scan_Normal function.
3. If the key is pressed, Q-Code will not enter TouchKey_Scan_Slow.

Ex.

[Path]

PowerOn: TouchKey_ON

Sleep: TouchKey_Scan_Slow ; Switch to slow-scan mode.

5.18.5 TouchKey_Scan_Normal

[NY9T]

When user turns on the TouchKey_Scan_Normal function, the Touch-Key will resume to normal scanning. (The default is TouchKey_Scan_Normal)

Note: When executing the TouchKey_Scan_Normal command, the reaction time of Touch-Key is faster than TouchKey_Scan_Slow.

Ex.

[Path]

PowerOn: TouchKey_ON

Sleep: TouchKey_Scan_Slow ; Switch to TouchKey_Scan_Slow.

TR1: TouchKey_Scan_Normal ; Switch to TouchKey_Scan_Normal.

5.18.6 TouchKey_Sensitivity (Level)

[NY9T / NX1]

User can switch the NY9T sensitivity of Touch-Keys by 8 levels. (The default is 4, the lower value, the better sensitivity) NX1 sensitivity of Touch-Keys has 4 levels.

Level: Set the sensitivity of Touch-Keys. User can assign the level or adjust sensitivity by register or external resistor for NY9T series. Please refer to [ReadRadj\(Ri\)](#) for description of Radj value table. NX1 can assign level, 0 ~ 3.

- Set the level of sensitivity (0~7) **Ex. TouchKey_Sensitivity(4)**
- Ri (supports Ri) **Ex. TouchKey_Sensitivity(R0)**
- Radj (supports external resistor for adjusting sensitivity) **Ex.TouchKey_Sensitivity(Radj)**

Ex. Set sensitivity of different Touch-Keys by bonding option.

[Path]

PowerOn: Switch(PE[xdd])=[Sens0, Sens1, Sens2, Sens3] ; PE[1:0] as bonding option.

Sens0: TouchKey_Sensitivity(0) ; Set PE[1:0] as 00, the sensitivity is the highest.

Sens1: TouchKey_Sensitivity(2) ; Set PE[1:0] as 01, the sensitivity is between the 2nd highest and intermediate.

Sens2: TouchKey_Sensitivity(4) ; Set PE[1:0] as 10, the sensitivity is intermediate.

Sens3: TouchKey_Sensitivity(7) ; Set PE[1:0] as 11, the sensitivity is the lowest.

Ex. Using register to dynamically adjust sensitivity for different modes.

[Path]

PowerOn:

Main: R0=0, R_Mode_Count>=8?SetSens, R0=4, SetSens

SetSens: TouchKey_Sensitivity(R0) ; Sensitivity is set by R0.

Ex. Set sensitivity of Touch-Keys directly by the level of the external resistor when power up.

[Path]

PowerOn: TouchKey_Sensitivity(Radj) ; Adjust sensitivity by the level of the external resistor.

Note: When the external resistor is not mounted or the resistor value exceeds the table range, the sensitivity will remain as the original setting.

5.18.7 Calibrate_ON

[NY9T]

Turn on the Touch-Keys calibration function. (The default is enabled.)

5.18.8 Calibrate_OFF

[NY9T]

Turn off the Touch-Keys calibration function.

5.18.9 AutoJudge_Calibrate

[NY9T]

If AutoJudge_Calibrate function is enabled, IC will calibrate under the environment conditions automatically. If any Touch-Key is pressed during calibration, the function will be stopped.

Ex. Calibrate the Touch-Keys every 4 seconds.

[Path]

4Sec: AutoJudge_Calibrate

; Calibrate the Touch-Keys every 4 seconds. If the function is turned off, IC won't do AutoJudge calibration.

TR1R: Calibrate_ON

; Turn on the calibration of Touch-Keys.

TR2R: Calibrate_OFF

; Turn off the calibration of Touch-Keys.

5.18.10 Enforce_Calibrate_Normal

[NY9T / NX1]

When Enforce_Calibrate_Normal function is enabled, IC will calibrate under the environment conditions automatically. If any Touch-Key is pressed, IC will still enforce the calibration.

Note:

- 1. If user sets Calibrate_OFF, the Touch-Keys won't be calibrated while executing the Enforce_Calibrate_Normal command, and it will continue to execute the next command.**
- 2. It's strongly suggested that calibrate Touch-Keys once in a while to avoid changed environment conditions causing poor reactions.**
- 3. It's suggested to enforce calibration on Touch-Keys when key is held over 40 seconds.**
- 4. If there is an Enforce_Calibrate path, it will be executed immediately after Enforce_Calibrate_Normal is done.**
- 5. Touch-Key release path is executed only if user chooses not to clear Touch-Key status after Enforce_Calibrate_Normal is done.**

Ex. Long- press key for 40 seconds to enforce calibration on Touch-Key once.

[Symbol]

KeyPress = R0

Count = R1

[Path]

4Sec: KeyPress=1?Check40sec, AutoJudge_Calibrate

Check40sec: Count++, Count=10?Timeout_40sec

Timeout_40sec: Count=0, Enforce_Calibrate_Normal

; Calibrate the Touch-Keys every 4 seconds. When keys are held over 40 seconds, enforce calibration on Touch-Keys again.

TR1R: Calibrate_ON ; Turn on the calibration of Touch-Keys.

TR2R: Calibrate_OFF ; Turn off the calibration of Touch-Keys.

TR3R: KeyPress=1, Count=0, Play..... ; When KEY3 is pressed, KeyPress=1 (key is pressed).

TR3F: KeyPress=0, Count=0 ; When KEY3 is released, KeyPress=0 (key is released).

Enforce_Calibrate: SW_Reset

; If there is no need to keep previous status after Enforce_Calibrate_Normal is done, it can do a SW_Reset to restore to power-up initial status. If SW_Reset is not issued, it will go on to execute Touch-Key release path.

5.18.11 Enforce_Calibrate_Sleep

[NY9T]

When Enforce_Calibrate_Sleep function is enabled, IC will calibrate under the environment conditions automatically. If any Touch-Key is pressed, IC will still enforce the calibration.

Note:

1. If user sets Calibrate_OFF, the Touch-Keys won't be calibrated while executing the Enforce_Calibrate_Sleep command, and it will continue to execute the next command.
2. It's strongly suggested that calibrate Touch-Keys once in a while to avoid changed environment conditions causing poor reactions.
3. If there is an Enforce_Calibrate path, it will not be executed after Enforce_Calibrate_Sleep is done.
4. After the Enforce_Calibrate_Sleep command is executed, IC will enter Sleep without waiting for calibration to complete.

Ex. Every 0.5 seconds to enforce calibration on Touch-Key once.

[Path]

500ms: Enforce_Calibrate_Sleep ; Every 500ms to enforce calibration on Touch-Key once.

5.18.12 Ri = TouchKey(Px)

[NY9T]

Save the status of TouchKey to Ri in the unit of Port and access the designated 4 pins of Port everytime.
 NY9T001A/004A:x = PA, NY9T008A:x = PA/PB, NY9T016A:x = PA~PD.

Ex. R0=TouchKey(PA) ;Save the status of PA.0~PA.3 to the current R0.

5.18.13 Var = Touchkey_Count(TouchKey)

[NX1]

This command will return the specified Count value that scanned byTouchKey to variable Var, it's applied for adjusting TouchKey setting.

TouchKey: Key in the used Count value from 0 to (n-1) for returning the Count value to TouchKey.

Ex. Var32_Count=TouchKey_Count(0) ; Sent the Count value of TouchKey0 to variable Var32_Count.

5.18.14 Var = Touchkey_BGCount(TouchKey)

[NX1]

This command will return the specified Count value of TouchKey background to variable Var, it's applied for adjusting TouchKey setting.

TouchKey: Key in the used Count value from 0 to (n-1) for returning the Count value to TouchKey.

Ex. Var32_BGCount=TouchKey_BGCount(0) ; Sent the Count value of TouchKey0 to variable Var32_BGCount.

5.19 Table Command

Table Command	
TableL(TableName, Rx/Xx, Ry/Xy, Ri)	TableM(TableName, Rx/Xx, Ry/Xy, Ri)
TableH(TableName, Rx/Xx, Ry/Xy, Ri)	Table(TableName, Rx/Xx, Ry/Xy, Rh, Rm, RI)
TableL(TableName, X, Y, Ri)	TableM(TableName, X, Y, Ri)
TableH(TableName, X, Y, Ri)	Table(TableName, X, Y, Rh, Rm, RI)
TableL(TableName, Rx/Xx, Ry/Xy, Xi)	TableH(TableName, Rx/Xx, Ry/Xy, Xi)
Table(TableName, Rx/Xx, Ry/Xy, Xh, XI)	TableL(TableName, X, Y, Xi)
TableH(TableName, X, Y, Xi)	Table(TableName, X, Y, Xh, XI)
Table(TableName, VarX, VarY, VarR)	

5.19.1 TableL(TableName, Rx/Xx, Ry/Xy, Ri)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

TableL(TableName, Rx/Xx, Ry/Xy, Ri) command is for indirect data addressing and it can be used to read the Low-Nibble of value defined in **[Table]** section.

TableName: name of the table defined in **[Table]**.

Rx/Xx: the content of Rx represents the address of X axis. User can use 4-bit or 8-bit variable addressing.

Ry/Xy: the content of Ry represents the address of Y axis. User can use 4-bit or 8-bit variable addressing.

Ri: save the Low-Nibble of obtained data to Ri.

5.19.2 TableM(TableName, Rx/Xx, Ry/Xy, Ri)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

TableM(TableName, Rx/Xx, Ry/Xy, Ri) command is for indirect data addressing and it can be used to read the Middle-Nibble of value defined in **[Table]** section.

TableName: name of the table defined in **[Table]**.

Rx/Xx: the content of Rx represents the address of X axis. User can use 4-bit or 8-bit variable addressing.

Ry/Xy: the content of Ry represents the address of Y axis. User can use 4-bit or 8-bit variable addressing.

Ri: save the Middle-Nibble of obtained data to Ri.

5.19.3 TableH(TableName, Rx/Xx, Ry/Xy, Ri)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

TableH(TableName, Rx/Xx, Ry/Xy, Ri) command is for indirect data addressing and it can be used to read the High-Nibble of value defined in **[Table]** section.

TableName: name of the table defined in **[Table]** .

Rx/Xx: the content of Rx represents the address of X axis. User can use 4-bit or 8-bit variable addressing.

Ry/Xy: the content of Ry represents the address of Y axis. User can use 4-bit or 8-bit variable addressing.

Ri: save the High-Nibble of obtained data to Ri.

5.19.4 Table(TableName, Rx/Xx, Ry/Xy, Rh, Rm, RI)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Table (TableName, Rx/Xx, Ry/Xy, Rh, Rm, RI) command is for indirect data addressing and it can be used to read the entry of table defined in **[Table]** section.

TableName: name of table defined in **[Table]** .

Rx/Xx: the content of Rx represents the address of X axis. User can use 4-bit or 8-bit variable addressing.

Ry/Yy: the content of Ry represents the address of Y axis. User can use 4-bit or 8-bit variable addressing.

Rh: save the High-Nibble of obtained data to Rh.

Rm: save the Middle-Nibble of obtained data to Rm.

RI: save the Low-Nibble of obtained data to RI.

Ex.

[Table]

Trans:

```
{
[0x001, 0x003, 0x005, 0x007, 0x009],
[0x004, 0x005, 0x3F6, 0x008, 0x010],
[0x000, 0x001, 0x002, 0x003, 0x0F4]
}
```

[Path]

P1: R0=2, R1=1, TableL(trans,R0,R1,R2) ; **R2=0x6.**

P2: R0=2, R1=1, TableM(trans,R0,R1,R2) ; **R2=0xF.**

P3: R0=2, R1=1, TableH(trans,R0,R1,R2) ; **R2=0x3.**

P4: R0=2, R1=1, Table(trans,R0,R1,R4,R3,R2) ; **R2=0x6 , R3=0xF , R4=0x3.**

5.19.5 TableL(TableName, X, Y, Ri)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

TableL(TableName, X, Y, Ri) command is for direct data addressing and it can be used to read the Low-Nibble of entry of table defined in **[Table]** section.

TableName: name of table defined in **[Table]**.

X: represents the address of X axis.

Y: represents the address of Y axis.

Ri: save the Low-Nibble of obtained data to Ri.

5.19.6 TableM(TableName, X, Y, Ri)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

TableM(TableName, X, Y, Ri) command is for direct data addressing and it can be used to read the Middle-Nibble of entry of table defined in **[Table]**section.

TableName: name of table defined in [Table].

X: represents the address of X axis.

Y: represents the address of Y axis.

Ri: save the Middle-Nibble of obtained data to Ri.

5.19.7 TableH(TableName, X, Y, Ri)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

TableH(TableName, X, Y, Ri) command is for direct data addressing and it can be used to read the High-Nibble of entry of table defined in [Table] section.

TableName: name of table defined [Table].

X: represents the address of X axis.

Y: represents the address of Y axis.

Ri: save the High-Nibble of obtained data to Ri.

5.19.8 Table(TableName, X, Y, Rh, Rm, RI)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Table(TableName, X, Y, Rh, Rm, RI) command is for direct data addressing and it can be used to read the entry of table defined in [Table] section.

TableName: name of table defined in [Table].

X: represents the address of X axis.

Y: represents the address of Y axis.

Rh: save the High-Nibble of obtained data to Rh.

Rm: save the Middle-Nibble of obtained data to Rm.

RI: save the Low-Nibble of obtained data to RI.

Ex.

[Table]

Trans:

```
{
[0x001, 0x003, 0x005, 0x007, 0x009],
[0x004, 0x005, 0x3F6, 0x008, 0x010],
[0x000, 0x001, 0x002, 0x003, 0x0F4 ]
}
```

[Path]

```
P1: TableL(trans, 2, 1,R2)           ; R2=0x6
P2: TableM(trans, 2, 1,R2)           ; R2=0xF.
P3: TableH(trans, 2, 1,R2)           ; R2=0x3.
P4: Table(trans, 2, 1,R4,R3,R2)       ; R2=0x6 , R3=0xF , R4=0x3.
```

5.19.9 TableL(TableName, Rx/Xx, Ry/Xy, Xi)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

TableL(TableName, Rx/Xx, Ry/Xy, Xi) is the in-direct data addressing command. It could read the Low-Byte of entry of table defined in the [Table] section.

TableName: name of table defined in [Table].

Rx/Xx: the content of Xx represents the address of X axis. User can use 4-bit or 8-bit variable addressing.

Ry/Xy: the content of Xy represents the address of Y axis. User can use 4-bit or 8-bit variable addressing.

Xi: save the Low-Byte of obtained data to Xi.

5.19.10 TableH(TableName, Rx/Xx, Ry/Xy, Xi)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

TableH(TableName, Rx/Xx, Ry/Xy, Xi) is the in-direct data addressing command. It could read the High-Byte of entry of table defined in the [Table] section.

TableName: name of table defined in [Table].

Rx/Xx: the content of Xx represents the address of X axis. User can use 4-bit or 8-bit variable addressing.

Ry/Xy: the content of Xy represents the address of Y axis. User can use 4-bit or 8-bit variable addressing.

Xi: save the High-Byte of obtained data to Xi.

5.19.11 Table(TableName, Rx/Xx, Ry/Xy, Xh, Xi)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Table(TableName, Rx/Xx, Ry/Xy, Xh, Xi) command is for direct data addressing and it can be used to read the entry of table defined in [Table] section.

TableName: name of table defined in [Table].

Rx/Xx: The content of Xx represents the address of X axis. User can use 4-bit or 8-bit variable addressing.

Ry/Xy: The content of Xy represents the address of Y axis. User can use 4-bit or 8-bit variable addressing.

Xh: save the High-Byte of obtained data to Xh.

Xi: save the Low-Byte of obtained data to Xi.

Ex.

[Table]

Trans:

```
{
[0x001, 0x003, 0x005, 0x007, 0x009],
[0x004, 0x005, 0x3F6, 0x008, 0x010],
[0x000, 0x001, 0x002, 0x003, 0x0F4]
}

[Path]
P1 : X0=2, X1=1, TableL(trans,X0,X1,X2) ; X2=0xF6.
P2 : X0=2, X1=1, TableH(trans,X0,X1,X2) ; X2=0x03.
P3 : X0=2, X1=1, Table(trans,X0,X1,X2,X3) ; X2=0x03 , X3=0xF6.
```

5.19.12 TableL(TableName, X, Y, Xi)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

TableL(TableName, X, Y, Xi) is the in-direct data addressing command. It could read the Low-Byte of entry of table defined in the [Table] section.

TableName: name of table defined in [Table].

X: represents the address of X axis.

Y: represents the address of Y axis.

Xi: save the Low-Byte of obtained data to Xi.

5.19.13 TableH(TableName, X, Y, Xi)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

TableH(TableName, X, Y, Xi) is the in-direct data addressing command. It could read the High-Byte of entry of table defined in the [Table] section.

TableName: name of table defined in [Table].

X: represents the address of X axis;

Y: represents the address of Y axis.

Xi: save the High-Byte of obtained data to Xi.

5.19.14 Table(TableName, X, Y, Xh, Xi)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

Table(TableName, X, Y, Xh, Xi) command is for direct data addressing and it can be used to read the entry of table defined in [Table] section.

TableName: name of table defined in [Table].

X: represents the address of X axis.

Y: represents the address of Y axis;

Xh: save the High-Byte of obtained data to Xh.

XI: save the Low-Byte of obtained data to XI.

Ex.

[Table]

Trans:

```
{
[0x001, 0x003, 0x005, 0x007, 0x009],
[0x004, 0x005, 0x3F6, 0x008, 0x010],
[0x000, 0x001, 0x002, 0x003, 0x0F4]
}
```

[Path]

```
P1 : TableL(trans,2,1,X2)           ; X2=0xF6.
P2 : TableH(trans,2,1,X2)           ; X2=0x03.
P3 : Table(trans,2,1,X2,X3)         ; X2=0x03 , X3=0xF6.
```

5.19.15 Table(TableName, VarX, VarY, VarR)

[NX1]

Indirect addressing command, can be used to read data defined in **[Table]**.

Table(TableName, VarX, VarY, VarR)

VarR = TableName[VarX][VarY]

TableName: Table name defined in **[Table]**.

VarX: Address of X axis.

- Immediate value.
- Variable defined in **[Variable]**.

VarY: Address of Y axis.

- Immediate value.
- Variable defined in **[Variable]**.

VarR: Save the obtained data to VarR.

Ex.

[Table]

Trans:

```
{
[0x001, 0x003, 0x005, 0x007, 0x009],
[0x004, 0x005, 0x3F6, 0x008, 0x010],
[0x000, 0x001, 0x002, 0x003, 0x0F4]
}
```

[Variable]

Var8: R0, R1, R2

[Path]

P1: R0=2, R1=1, Table(trans,R0,R1,R2) ; R2=0x3F6.

P2: R0=2, R1=1, R2=trans[R0][R1] ; R2=0x3F6.

5.20 IR Command

IR Command				
IR_TX=data	IR_TX(Rl:Rk:Rj:Ri)	IR_TX(Xj:Xi)	IR_TX_WaitN	IR_RX_ON
IR_RX_OFF	[Rl,Rk,Rj,Ri] = IR_RX	[Xj,Xi] = IR_RX	IR_RX = data?Path	
IR_RX != data?Path		IR_TX_Busy?Path		IR_Carrier_On
IR_Carrier_Off	-	-	-	

5.20.1 IR_TX=data

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Transmit data as IR Code directly.

Ex. IR_TX=0xF or IR_TX=15 ; IR Code = 0x0F.

Ex. IR_TX=0xFFFF or IR_TX=4095 ; IR Code = 0xFFFF.

Ex. IR_TX=0xFFFFF or IR_TX=65535 ; IR Code = 0xFFFFF.

5.20.2 IR_TX(Rl:Rk:Rj:Ri)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Transmit IR code using Ri, Rj, Rk and Rl. Ri=bit0 ~ 3, Rj=bit4 ~ 7, Rk=bit8 ~ 11, Rl=bit12 ~ 15.

Ex. R0=0xF for 4-bit mode.

TR1: IR_TX(R0) ; IR Code = 0xF.

Ex. R0=0xF, R1=0x3 for 8-bit mode.

TR1: IR_TX(R1:R0) ; IR Code = 0x3F.

Ex. R0=0xF, R1=0xF, R2=0x1 for 12-bit mode.

TR1: IR_TX(R2:R1:R0) ; IR Code = 0x13F.

Ex. R0=0xF, R1=0x3, R2=0x01, R3=0x0 for 16-bit mode.

TR1: IR_TX(R3:R2:R1:R0) ; IR Code = 0x13F.

5.20.3 IR_TX(Xj:Xi)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Transmit IR code using Xi as high-byte and Xj as low-byte.

Ex. X0=0x3F for 4-bit mode.

TR1: IR_TX(X0) ; IR Code = 0x0F.

Ex. X0=0x3F for 8-bit mode.

TR1: IR_TX(X0) ; IR Code = 0x3F.

Ex. X0=0x3F, X1=0x01 for 12-bit mode.

TR1: IR_TX(X1, X0) ; IR Code = 0x13F.

Ex. X0=0x3F, X1=0x01 for 16-bit mode.

TR1: IR_TX(X1:X0) ; IR Code = 0x13F.

5.20.4 IR_TX_WaitN

[NY5+ / NX1]

If IR data is being transmitting, it will execute next command after transmitting is completed. Otherwise, it will immediately execute the next command.

5.20.5 IR_RX_ON

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Activate IR receiving function. IR code cannot be received if this function is not activated.

Ex. IR_RX_ON ; Activate the IR reception function.

5.20.6 IR_RX_OFF

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Deactivate IR receiving function.

Ex. IR_RX_OFF ; Deactivate IR reception function.

5.20.7 [RI, Rk, Rj, Ri] = IR_RX

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Write the received IR code to RAM. Ri=bit0 ~ 3, Rj=bit4 ~ 7, Rk=bit8 ~ 11, RI=bit12 ~ 15.

Note: Using this command before receiving IR data will get unknown data.

Ex. IR_Bit = 12

```
[R0] = IR_RX ; Write Bit[3:0] of IR code to R0.
[R1, R0] = IR_RX ; Write Bit[3:0] of IR code to R0, Bit[7:4] to R1.
[R2, R1, R0] = IR_RX ; Write Bit[3:0] of IR code to R0, Bit[7:4] to R1,
; Bit[11:8] to R2.
```

5.20.8 [Xj, Xi] = IR_RX

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Write the received IR code to RAM. Xi=bit0~7, Xj=bit8~15.

Note: Using this command before receiving IR data will get unknown data.

Ex. IR_Bit = 12

```
[X0] = IR_RX ; Write Bit[7:0] of IR code to X0.
[X1, X0] = IR_RX ; Bit[7:0] of IR code will be written to X0.
; Bit[11:8] of IR code will be written to X1 and
; X1[7:0] will be 0.
```

5.20.9 IR_RX = data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path when received IR code equals to data.

Note: NY9T does not support this command.

Ex. IR_RX = 0? Path ; Jump to Path if received code is 0.

5.20.10 IR_RX != data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Jump to Path if received IR code not equal to data.

Ex. IR_RX != 0? Path ; Jump to Path if received code is not 0.

5.20.11 IR_TX_Busy?Path

[NX1]

Jump to Path if IR is transmitting data.

5.20.12 IR_Carrier_On

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Activate IR Carrier output. This command is used for manual control of IR Carrier.

5.20.13 IR_Carrier_Off

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

Deactivate IR Carrier output. This command is used for manual control or IR Carrier.

5.21 Serial Control TX Command

Serial Control Command		
SC_TX(Mode, data)	SC_TX(Mode, Ri:Rj:Rl:Rk)	SC_TX(Mode, Xi:Xj)

5.21.1 SC_TX(Mode, Parameter)

[NY9T008A / NY9T016A]

User can use the NY9T as the keys of MCUs. Q-Code provides 3 kinds of transfer protocols: SPI_Like, NY3 Serial_Trigger and IR_Trigger. Only supports 16-bit mode.

SPI_Like: Use 2 pins to achieve a SPI-like communication protocol.

NY3 Serial_Trigger: User can connect NY3 via the serial signal, and NY9T acted as NY3 trigger buttons. Please refer to NY3C/3D series specification for the detailed connection usage.

IR_Trigger: Use a pin to simulate IR transmission signal, and can be directly used to connect current NY4/5/7 series with IR communication.

Mode: Select communication protocol of Serial Control signal or use pins to auto detect the communication protocol.

- Auto Detect **Ex.** SC_TX(Auto, 0xFFFF)
- SPI_Like **Ex.** SC_TX(SPI_Like, 0xFFFF)
- NY3 **Ex.** SC_TX(NY3, 0xFFFF)
- IR_Trigger **Ex.** SC_TX(IR_Trigger, 0xFFFF)

Parameter: Data transmission.

- Data (supports 16-bit) **Ex.** SC_TX(Auto, 0xFFFF)
- Ri (supports to 1 ~ 4 Ri) **Ex.** SC_TX(Auto, R3:R2:R1:R0)
- Xi (supports to 1 ~ 2 Ri) **Ex.** SC_TX(Auto, X1:X0)

Note:

1. Set the communication protocol on Option before using this function.
2. The signal is transmitted by the left register which is the highest bit, the data transmission is fixed at 16-bit.

5.21.2 SC_TX(Mode, Ri:Rk:Rj:Ri)

[NY9T008A / NY9T016A]

Use RAM to transfer serial signal. Ri=bit0~3, Rj=bit4~7, Ri=bit8~11, Rk=bit12~15.

Ex. Auto_Detect Mode

R3=0x0, R2=0, R1=0, R0=0, SC_TX(Auto, R3:R2:R1:R0)

; Auto detect communication protocol, and output signal of [R3:R2:R1:R0] register value.

Ex. SPI_Like Mode

R3=0x0, R2=0, R1=0, R0=0, SC_TX(SPI_Like, R3:R2:R1:R0)

; Use SPI_Like communication protocol, and output signal of [R3:R2:R1:R0] register value.

Ex. NY3 Serial_Trigger Mode

R3=0x0, R2=0, R1=0, R0=0, SC_TX(NY3, R3:R2:R1:R0)

; Use NY3 Serial_Trigger communication protocol, and output signal of [R3:R2:R1:R0] register value. If all the register values are 0, and only output signal Reset.

Ex. IR_Trigger Mode

R3=0x0, R2=0, R1=0, R0=0, SC_TX(IR_Trigger, R3:R2:R1:R0)

; Use IR_Trigger communication protocol, and output signal of [R3:R2:R1:R0] register value.

5.21.3 SC_TX(Mode, Xj:Xi)

[NY9T008A / NY9T016A]

Use RAM to transfer serial signal. Xi=bit0~7, Xj=bit8~15.

Ex. Auto_Detect Mode

X1=0x0, X0=0, SC_TX(Auto, X1:X0)

; Auto detect communication protocol, and output signal of [X1:X0] register value.

Ex. SPI_Like Mode

X1=0x0, X0=0, SC_TX(SPI_Like, X1:X0)

; Use SPI_Like communication protocol, and output signal of [X1:X0] register value.

Ex. NY3 Serial_Trigger Mode

R3=0x0, R2=0, R1=0, R0=0, SC_TX(NY3, X1:X0)

; Use NY3 Serial_Trigger communication protocol, and output signal of [X1:X0] register value. If all the register values are 0, and only output signal Reset.

Ex. IR_Trigger Mode

R3=0x0, R2=0, R1=0, R0=0, SC_TX(IR_Trigger, X1:X0)

; Use IR_Trigger communication protocol, and output signal of [X1:X0] register value.

5.22 Serial Control Command

Serial Control Command				
SC_RX_ON	SC_RX_OFF	[Rl,Rk,Rj,Ri] = SC_RX	[Xj,Xi] = SC_RX	SC_RX = data?Path
SC_RX != data?Path	-	-	-	-

5.22.1 SC_RX_ON

[NY4 / NY5 / NY5+ / NY6 / NY7]

Activate SC receiving function. SC code cannot be received if this function is not activated.

Ex. SC_RX_ON ; Activate the SC reception function.

5.22.2 SC_RX_OFF

[NY4 / NY5 / NY5+ / NY6 / NY7]

Deactivate SC receiving function.

Ex. SC_RX_OFF ; Deactivate SC reception function.

5.22.3 [Rl, Rk, Rj, Ri] = SC_RX

[NY4 / NY5 / NY5+ / NY6 / NY7]

Write the received SC code to RAM. Ri=bit0 ~ 3, Rj=bit4 ~ 7, Rk=bit8 ~ 11, Rl=bit12 ~ 15.

Note: Using this command before receiving SC data will get unknown data.

Ex.

[R0] = SC_RX ; Write Bit[3:0] of SC code to R0.
[R1, R0] = SC_RX ; Write Bit[3:0] of SC code to R0, Bit[7:4] to R1.
[R2, R1, R0] = SC_RX ; Write Bit[3:0] of SC code to R0, Bit[7:4] to R1, Bit[11:8] to R2.

5.22.4 [Xj, Xi] = SC_RX

[NY4 / NY5 / NY5+ / NY6 / NY7]

Write the received IR code to RAM. Xi=bit0~7, Xj=bit8~15.

Note: Using this command before receiving SC data will get unknown data.

Ex.

[X0] = SC_RX ; Write Bit[7:0] of SC code to X0.
[X1, X0] = SC_RX ; Bit[7:0] of SC code will be written to X0.
; Bit[15:8] of SC code will be written to X1.

5.22.5 SC_RX = data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7]

Jump to Path when received SC code equals to data.

Ex. SC_RX = 0? Path ; Jump to Path if received code is 0.

5.22.6 SC_RX != data?Path

[NY4 / NY5 / NY5+ / NY6 / NY7]

Jump to Path if received SC code not equal to data.

Ex. SC_RX != 0? Path ; Jump to Path if received code is not 0.

5.23 I2C Command

I2C Command				
I2C_TX	I2C_RX	I2C_RX=data?Path		
I2C_RX!=Data?Path		I2C_Ack?Path	I2C_Start	I2C_Stop
I2C_MReadAck	I2C_MReadNAck	I2C_Reset	-	-

5.23.1 I2C_TX

[NY5+ / NX1]

Transmit I2C data.

I2C_TX(Data)

I2C_TX(Rj:Ri)

I2C_TX(Xi)

Data: Immediate value to be transmitted.

Rj:Ri: Transmit data stored in RAM. Ri=bit0 ~ 3 , Rj=bit4 ~ 7.

Xi: Transmit data stored in RAM, Xi=bit0 ~ 7.

5.23.2 I2C_RX

[NY5+ / NX1]

Store received I2C data to specified RAM.

[Rj, Ri] = I2C_RX

[Xi] = I2C_RX

Var = I2C_RX

Rj, Ri: Write the received I2C code to RAM. Ri=bit0 ~ 3, Rj=bit4 ~ 7.

Xi: Write the received SC code to RAM. Xi=bit0 ~ 7.

Var: Write the received SC code to RAM. Var=bit0 ~ 7.

5.23.3 I2C_RX=data?Path

[NY5+ / NX1]

Jump to Path when received I2C data is equal to Data.

5.23.4 I2C_RX!=data?Path

[NY5+ / NX1]

Jump to Path when received I2C data is not equal to Data.

5.23.5 I2C_Ack?Path

[NY5+ / NX1]

When acting as Master, jump to Path if ACK is responded from receiver after data is transmitted.

5.23.6 I2C_Start

[NY5+ / NX1]

When acting as Master, transmit device address and inform to start reading data from or writing data to slave device.

I2C_Start

I2C_Start(Address, Mode)

I2C_Start(Address, Bits, Mode)

Address: device address.

- Data (supports 0 ~ 255)
- Ri (supports 1 ~ 2 Ri)
- Xi (support 1 Xi)

Bits: Accepts 7 / 10, default is 7.

Mode: transmit mode, accepts R / W.

Note: This command is only available when I2C is Master mode.

5.23.7 I2C_Stop

[NY5+ / NX1]

When acting as Master, transmit stop signal, and end current transmission.

Note: This command is only available when I2C is Master mode.

5.23.8 I2C_MReadAck

[NY5+ / NX1]

Master sends clock signal to receive data by other device. ACK is transmitted after receiving.

Note: This command is only available when I2C is Master mode.

5.23.9 I2C_MReadNAck

[NY5+ / NX1]

Master sends clock signal to receive data by other device. No ACK is transmitted after receiving.

Note: This command is only available when I2C is Master mode.

5.23.10 I2C_Reset

[NX1]

Reset I2C.

Note: This command is only available when I2C is Master mode.

5.24 UART Command

UART Command			
UART_TX	UART_TX WaitN	UART_RX	UART_RX=data?Path
UART_RX!=data?Path		UART_TX_Busy?Path	-

5.24.1 UART_TX

[NY5+ / NX1]

Transmit UART data.

UART_TX(Data)

UART_TX(Rj:Ri)

UART_TX(Xi)

Data: Immediate value to be transmitted.

Rj:Ri: Transmit data stored in RAM. Ri=bit0 ~ 3, Rj=bit4 ~ 7.

Xi: Transmit data stored in RAM, Xi=bit0 ~ 7.

5.24.2 UART_TX_WaitN

[NY5+ / NX1]

If UART data is being transmitting, it will execute next command after transmitting is completed. Otherwise, it will immediately execute the next command.

5.24.3 UART_RX

[NY5+ / NX1]

$[Rj, Ri] = \text{UART\_RX}$

$Xi = \text{UART\_RX}$

Store received UART data to specified RAM, Ri=bit0 ~ 3, Rj=bit4 ~ 7, Xi=bit0 ~ 7.

5.24.4 UART_RX=data?Path

[NY5+ / NX1]

Jump to Path when received UART data is equal to Data.

5.24.5 UART_RX!=data?Path

[NY5+ / NX1]

Jump to Path when received UART data is equal to Data.

5.24.6 UART_TX_Busy?Path

[NX1 EF]

Jump to Path if UART TX is in busy state.

5.25 PWMIO Command

PWMIO Command				
PWMOut	PWMOutS	PlayPWM	PlayPMWS	WaitPN
StopPWM	PausePWM	ResumePWM	HoldPWM	PWMCtrl

5.25.1 PWMOut / PWMOutS

5.25.1.1 PWMOut / PWMOutS(Px.n, Duty, Time, Type)

[NY4 / NY5 / NY5+ / NY6 / NY7]

This command is provided for user to output PWM-IO. The format of command is as following.

PWMOut: After the time of the PWM-IO output is finished, the next command will be executed.

PWMOutS: After PWM-IO output, the following command will be executed immediately.

Pin: Specify output pin.

- **Px.n:** IC pin.
- **EXPx_Py.n:** I/O Expander pin.

Duty: User can designate the output percentage and levels via an immediate value, or designate levels via variables.

- Output level via immediate value. **Ex. PWMOut(PC.0, 12, 1s)**
- Output percentage via immediate value **Ex. PWMOut(PC.0, 30%, 1s)**
- Output level via Ri (Fixed using 2 Ri) **Ex. PWMOut(PC.0, R1:R0, 1s)**
- Output level via Xi ((Fixed using 1 Xi) **Ex. PWMOut(PC.0, X0, 1s)**

Time: Set the PWM-IO output time, default 16ms if omitted. Loop is also accepted to output PWM (Time=16ms~30sec)

Type: use software PWM-IO (0) or hardware PWM-IO (1), default is software PWM-IO (0).

Note: NY4 / NY5 / NY6 / NY7 do not support Type parameter.

5.25.1.2 PWMOut / PWMOutS (Para8, Para7, Para6, Para5, Para4, Para3, Para2, Para1)

[NY9T]

The PWMOut command is for setting the Duty cycle of all PWM-IO pins, it can output directly without the PlayPWM command.

Para1~8: It presents the serial number of each pin, Para1=PE.0, Para2=PE.1, and so forth. (Only the pins of PWM-IO can set the serial number.)

- Set the percentage of output directly. (0%~100%) **Ex. PWMOut(30%, 50%, 70%, 5%)**
- Set the levels of output. (0~255) **Ex. PWMOut(30, 50, 70, 5)**

Time: User can set the output time of PWM-IO, the default is 16ms. (Time=16ms~30sec)

Note: In NY9T, when user set two PWM-IO output commands, the commands have to space out 16ms above to avoid working error.

5.25.1.3 PWMOut / PWMOutS (@PWME n, Percentage, Time)

[NY9T]

PWME n: The corresponding usage of pins and files is according to the sequence of PWM-IO output

pin and the Label "A1~A8" of Action files in the **[I/O Pin]** section. There are 8 pins corresponding to PE.0~PF.3 at most. PWME_n=1 or 0, Activate PWM-IO output function when the designated pin plays Action file; 0=Not turned. (The unfilled part will be filled in 0 automatically, the MSB is on the leftmost)

- Set the designated output pin **Ex. PWMOut(@1111, 30%)**
- Ri (supports 2 Ri) **Ex. PWMOut(R1:R0, 30%)**
- Xi (supports 1 Xi) **Ex. PWMOut(X0, 30%)**
- Keep the last setting when filling in X **Ex. PWMOut(X, 30%)**

Percentage: Set the output parameter by percentage or level.

- Set the output percentage ratio (0%~100%) **Ex. PWMOut(@1111, 30%)**
- Set the output level parameter (0~255) **Ex. PWMOut(@1111, 127)**
- Ri (supports 2 Ri) **Ex. PWMOut(@1111, Rj:Ri)**
- Xi (supports 1 Xi) **Ex. PWMOut(@1111, Xi)**

Time: User can set the PWM-IO output time, the default time is 16ms. (Time=16ms~30sec)

Ex.

[Path]

TR1: PWMOut(@1111, 30%)	; PE0~PE3 output 30% luminance.
TR2: PWMOut(@1111, 30)	; PE0~PE3 output 30/256 luminance.
TR3: R1=0x1, R0=0xE, PWMOut(@1111, R1:R0)	; PE0~PE3 output 30/256 luminance.
TR4: X0=0x1E, PWMOut(@1111,X0)	; PE0~PE3 output 30/256 luminance.
TR5: PWMOut(@1111, 30%, 16ms)	; PE0~PE3 output 30% luminance and play for 16 ms.
TR6: PWMOut(@1111, 30, 2s)	; PE0~PE3 output 30/256 luminance and play for 2 seconds.

Note:

1. When user set two PWM-IO output commands, the commands have to space out 16ms above to avoid working error.
2. If the parameter of percentage is decided by register, it will occupy the space that DATA ROM stores PWM-IO signals. (DATA Size calculation = (pin number+1)*256).

5.25.2 PlayPWM / PlayPWMS

5.25.2.1 PlayPWM / PlayPWMS

[NY5+]

User can draw the output Action signals by Q-Visio, and play signals by using PWM-IO command in Q-Code.

PlayPWM({Px.n, }Ch, \$PIO) { & [BG1 {,BG2}] }

PlayPWM({Px.n, }Ch, \$PIO, Loop) { & [BG1 {,BG2}] }

PlayPWMS({Px.n, }Ch, \$PIO)

PlayPWMS({Px.n, }Ch, \$PIO, Loop)

PlayPWM: After the end of playing Action file, executing the next command.

PlayPWMS: When Action file starts playing, executing the next command immediately.

Px.n: Pin to output PWM-IO.

- NY5+ supports PB.0 ~ PB.3 and PD.0 ~ PD.3 ◦

Ch: PWM-IO channel.

- NY5+ supports Ch1 ~ Ch4.

PIO: Output config defined in **[PWM-IO]**.

- Label of output config (SPIOx and MPIOx).
- Ri (Supports 1 ~ 3 Ri) or Xi (Supports 1 ~ 2 Xi). The value of Ri / Xi indicates the nth output config defined in **[PWM-IO]**. Ri / Xi is automatically increased by 1 if “++” is specified. If 2 Xi is applied, Most 4 significant bits are omitted.

Loop: Output in a loop continuously, until stopped by commands like `StopPWM`. This parameter is optional.

Note:

1. In NY9T, when user set two PWM-IO output commands, the commands have to space out 16ms above to avoid working error.
2. In NY5+, when Px.n is specified, only Multi Signal in **[PWM-IO]** can be used. In a contrary, when Px.n is not specified, only Single Signal in **[PWM-IO]** can be used.

Ex. NY5+

[Action File]

VIO0 = Action1.vio

; Assign the Action file that contains the signal of ActionLabel "A1 ~ A6".

[PWM-IO]

Multi_Pins = [PB.0, PB.1, PB.2, PB.3, PD.0, PD.1, PD.2, PD.3]

MPIO1 = [VIO0.A1, VIO0.A2, VIO0.A3, VIO0.A4]

MPIO2 = [X, X, X, X, VIO0.A4, VIO0.A3, VIO0.A2, VIO0.A1]

SPIO1 = [VIO0.A6]

[Path]

Path1: PB=[P P P P], PlayPWM(Ch1, \$MPIO1)

; Output VIO0.A1 to PB.0, Output VIO0.A2 to PB.1, and vice versa.

Path2: PD=[P P P P], PlayPWM(Ch2, \$MPIO2)

; Output VIO0.A4 to PD.0, Output VIO0.A3 to PD.1, and vice versa.

Path3: PlayPWM(PB.0, Ch1, \$SPIO1)

; Output VIO0.A6 to PB.0.

5.25.2.2 PlayPWM / PlayPWMS(PWME_n, \$VIOLabel, Extension)

[NY9T]

User can draw the output Action signals by Q-Visio, and play signals by using PWM-IO command in Q-Code.

PlayPWM(@PWME_n, \$VIOLabel) { & [BG1 {,BG2}] }

PlayPWM(@PWME_n, \$VIOLabel, Extension) { & [BG1 {,BG2}] }

PlayPWMS(@PWME_n, \$VIOLabel)

PlayPWMS(@PWME_n, \$VIOLabel, Extension)

PlayPWM: After the end of playing Action file, executing the next command.

PlayPWMS: When Action file starts playing, executing the next command immediately.

PWME_n: The corresponding usage of pins and files is according to the sequence of PWM-IO output pin and the Label "A1~A8" of Action files in [I/O]. There are 8 pins corresponding to PE.0~PF.3 at most. PWME_n=1 or 0, 1=Activate PWM-IO output function when the designated pin plays Action file; 0=Not turned. (The unfilled part will be filled in 0 automatically, the MSB is on the leftmost)

- Set the designated output pin **Ex.** PlayPWM(@1111, \$VIO0, 4)
- Ri (supports 1 ~ 2 Ri) **Ex.** PlayPWM(R1:R0, \$VIO0, 4)
- Xi (supports 1 Xi) **Ex.** PlayPWM(X0, \$VIO0, 4)
- Keep the last setting when filling in X **Ex.** PlayPWM(X, \$VIO0, 4)

VIOLabel: When user adds multiple of VIO files, user has to assign the cited location of Action File.

- Set the desired output file **Ex.** PlayPWM(@1111, \$VIO0)
- Ri (supports 1 ~ 4 Ri) **Ex.** PlayPWM(@1111, R3:R2:R1:R0)
- Xi (supports 1 ~ 2 Xi) **Ex.** PlayPWM(@1111, X1:X0)

Extension: The extension function can extend length of signals to save ROM size. Extension=1/2/4/8.

- Assign value. **Ex.** PlayPWM(@1111, \$VIO0, 4)
- Ri (supports 1 Ri) **Ex.** PlayPWM(@1111, \$VIO0, R0)
- Keep the last setting when filling in X. **Ex.** PlayPWM(@1111, \$VIO0, X)

Note:

1. When 2 PWM-IO outputs command, the interval must be more than 16 ms for avoiding incorrect actions.
2. The leftmost parameter of the PWME_n is set as the MSB of PWM-IO, and Action file corresponds to the LSB of A1.
3. Action signals correspond to PWM-IO pin in order. A1 is the first priority for PE.0, if PE.0 is not PWM-IO, A1 will correspond to PE.1, and so forth.
4. When user plays VIO file and outputs multiple of signals simultaneously, the time lengths

of signals have to be the same. If they are different, Q-Code will base on the longest output signal and force the other signals with insufficient time to output 0%.

5. *When specifying extension by Ri, the value of Ri must be 1 / 2 / 4 / 8. If the parameter is other than the above, it will be 1 eventually.*

Ex.

[Action File]

VIO0 = Action1.vio ; Assign the Action file that contains the signal of ActionLabel "A1~A6".

VIO1 = Action2.vio ; Assign the Action file that contains the signal of ActionLabel "A1~A6".

[Path]

Path1: PlayPWM(@1100, \$VIO0) ; Play "VIO0" and output to PE2, PE3.

Path2: PlayPWM(@0011, \$VIO0) ; Play "VIO0" and output to PE0, PE1.

Path3: PlayPWM(@1111, \$VIO1) ; Play "VIO1", and output to PE0, PE1, PE2, PE3.

Path4: PlayPWM(@1111, \$VIO1, 4) ; Extend VIO1 signal length in multiple of 4 and output to PE0, PE1, PE2, PE3.

Path5: PlayPWM(@1111, \$VIO1), PlayA... ; After end of playing "VIO1", executing PlayA command immediately.

Path6: PlayPWMS(@1111, \$VIO1), PlayA... ; After playing "VIO1", executing PlayA command immediately.

5.25.3 WaitPN

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Execute next command after PWM-IO playback ends. If no PWM-IO playback is in progress, executes next command immediately.

WaitPN

WaitPN(Pin)

WaitPN(Ch)

WaitPN(PIO)

Pin: Wait for PWM-IO by PWMOut / PWMOutS on specified pin.

- **Px.n:** IC pin.
- **EXPx_Py.n:** I/O Expander pin.

Ch: PWM-IO channel by PlayPWM / PlayPWMS.

PIO: All PlayPWM / PlayPWMS channels.

Note:

1. **NY4 / NY5 / NY6 / NY7 / NY9T do not support Pin / Ch / PIO parameter.**
2. **NX1 does not support Ch / PIO parameter.**

5.25.4 StopPWM

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Stop output of PWM-IO.

StopPWM

StopPWM(Pin)

StopPWM(Ch)

StopPWM(PIO)

Pin: Stop PWM-IO by PWMOut / PWMOutS on specified pin.

- **Px.n:** IC pin.
- **EXPx_Py.n:** I/O Expander pin.

Ch: PWM-IO channel by PlayPWM / PlayPWMS.

PIO: All PlayPWM / PlayPWMS channels.

Note:

1. **NY4 / NY5 / NY6 / NY7 / NY9T do not support Px.n / Ch / PIO parameter.**
2. **NX1 does not support Ch / PIO parameter.**

5.25.5 PausePWM

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Pause the PWM-IO output.

- In NY9T , PausePWM apply to all PWM-IO output. IC is capable of entering sleep mode.
- In NY4 / NY5 / NY6 / NY7, PlayPWM is not supported. PausePWM pauses PWM timer, and keep current output duty. IC is not capable of entering sleep mode.
- In NY5+
 - ◆ For PWM-IO output invoked by PWMOut, PausePWM pauses PWM timer, and keep current output duty. IC is not capable of entering sleep mode.
 - ◆ For PWM-IO output invoked by PlayPWM, PausePWM pauses output. IC is capable of entering sleep mode.

PausePWM

PausePWM(Pin)

PausePWM(Ch)

PausePWM(PIO)

Pin: pause PWM-IO by PWMOut / PWMOutS on specified pin.

- **Px.n:** IC pin.
- **EXPx_Py.n:** I/O Expander pin.

Ch: PWM-IO channel by PlayPWM / PlayPWMS.

PIO: All PlayPWM / PlayPWMS channels.

Note:

1. If user doesn't set PWM-IO to output, this command does not take effect.
2. NY4 / NY5 / NY6 / NY7 / NY9T do not support Pin / Ch / PIO parameter.
3. NX1 does not support Ch / PIO parameter.

5.25.6 ResumePWM

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command can resume the playback of PWM-IO.

ResumePWM

ResumePWM(Pin)

ResumePWM(Ch)

ResumePWM(PIO)

Pin: Resume PWM-IO by PWMOut / PWMOutS on specified pin.

- **Px.n:** IC pin.
- **EXPx_Py.n:** I/O Expander pin.

Ch: PWM-IO channel by PlayPWM / PlayPWMS.

PIO: All PlayPWM / PlayPWMS channels.

Note:

1. NY4 / NY5 / NY6 / NY7 / NY9T do not support Pin / Ch / PIO parameter.
2. NX1 does not support Ch / PIO parameter.

5.25.7 HoldPWM

[NY5+ / NY9T]

Pause the PWM-IO output, the output will keep on current duty, IC is not capable of entering sleep mode.

- In NY9T, HoldPWM apply to PWM-IO invoked by PWMOut / PlayPWM.
- In NY5+, HoldPWM only apply to PWM-IO invoked by PlayPWM, but not PWM-IO invoked by PWMOut.

HoldPWM

HoldPWM(Ch)

HoldPWM(PIO)

Ch: PWM-IO channel by PlayPWM / PlayPWMS.

PIO: All PlayPWM / PlayPWMS channels.

Note:

1. NY9T does not support Px.n / Ch / PIO parameter.
2. For the HoldPWM of NY5+, the PWMOut doesn't work.

5.25.8 PWMCtrl(@PWME_n, Extension)

[NY9T]

User can control the set PWM-IO pins in the [I/O Pin] section. The usage of PWME_n is same to the PlayPWM command, please refer to 4.20.2

Ex. User can turn on or off the output of PWM-IO during playing PWM-IO (PE.0~PE.3).

[Path]

PowerOn: PWMCtrl(@1111,1)	; Default PWMIO as1111, Extension=1.
TR1: PlayPWM(X, \$VIO0)	; Play “VIO0” and output to PE.
TR2: PWMCtrl(@0011,X)	; Close the output of PE.2 and PE.3.
TR3: PWMCtrl(X, 4)	; Set the Extension in the multiple of 4.

5.25.9 PWMDuty

[NY5+]

Change the PWM-IO on specified pin.

PWMDuty(Pin, Duty)

Pin: specify PWM-IO pin to change.

- **Px.n:** IC pin.
- **EXPx_Py.n:** I/O Expander pin.

Duty: User can designate the output percentage and levels via an immediate value, or designate levels via variables.

- Output level via immediate value.
- Output percentage via immediate value
- Output level via Ri (Fixed using 2 Ri)
- Output level via Xi ((Fixed using 1 Xi)

Note: Changing of PWM duty only takes effect when PWM-IO is executing.

Ex.

PWMDuty(PB.0, 50%)

PWMDuty(PB.0, 8)

PWMDuty(PB.0, R1:R0)

PWMDuty(PB.0, X0)

PWMDuty(EXP0_P0.3, X0)

5.26 Interrupt Command

Interrupt Command				
INT_ON	INT_OFF	INT_RET	INT = n	-

5.26.1 INT_ON

[NY5 / NY5+ / NX1]

Activate interrupt.

Note:

1. Even though interrupt is activated, system will enter sleep mode if no play or delay commands are executing.
2. In NX1, its INT_ON by default after PowerOn.
3. In NX1, INT_ON must have corresponding INT_OFF, and vice versa.

Ex. NY5 / NY5+

[Path]

PowerOn: KEY1, INT_ON ; Turn on interrupt.

Ex. NX1

[Variable]

Var8: count500ms = 0

[Path]

PowerOn: KEY1, INT_ON

500ms: INT_OFF, count500ms++, INT_ON ; Disable interrupt before modifying count500ms variable, in case of unexpected result.

[Interrupt]

RTC_2Hz: count500ms++

5.26.2 INT_OFF

[NY5 / NY5+ / NX1]

Deactivate interrupt.

Note:

1. In NX1, INT_ON must have corresponding INT_OFF, and vice versa.
2. Don't turn off the NX1 Interrupt program for a long time. It might affect the system's operation.

Ex.

[Path]

PowerOn: KEY1, INT_OFF ; Disable interrupt.

5.26.3 INT_RET

[NY5 / NY5+]

Return to interrupt service vector.

Note: If jump is used inside interrupt routine, "INT_RET" is needed to jump back to interrupt back, or system may not work correctly.

Ex.

[Path]

PowerOn: KEY1, INT_ON, INT=1.024 ; Turn on interrupt and set interrupt time = 1ms.
Interrupt: R0=0xF?Change, PA=0x0, ; If R0 = 0xF, jump to "change", else execute
INT_RET ; next command.
Change: PA=0xF, INT_RET ; After executing PA=0xF, return back to interrupt
; vector.

5.26.4 INT = n

[NY5 / NY5+]

Set the interval between interrupts. n = 0.256 ms/ 0.512 ms/ 1.024 ms.

Ex.

[Path]

PowerOn: KEY1, INT_ON, INT=1.024
Interrupt: !PA, INT_RET ; Invert the output value of PA every 1ms
; (each time interrupt occurred).

5.27 Delay Command

Delay Command				
Delay(Time)	Delay(Ri:Rj:Rk)	WaitDN(n)	StopD(n)	PauseD(n)
ResumeD(n)	SDelay(time)	-	-	-

5.27.1 Delay(Time)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command is for providing delay in Q-Code for foreground and backgrounds, respectively.

Time: unit: sec or ms (default is sec), with minimum value of 4ms. Less than 4ms, will be rounded up.

- NY4 / NY5 / NY6 / NY7 / NY9T accepts max 15s.
- NY5+ / NX1 accepts max 0x7FFFFFFF milliseconds.

Note:

1. Due to deviation between RTC_1024Hz and 1ms (0.9765625ms - 1ms) / 1ms $\div$ -2.34%, there is -2.34% delay time deviation in Q-Code 6.51 or older versions. This deviation will be compensated in Q-Code 6.60 or newer version.

2. ***NX1 counter is using RTC, which contains $\pm 1.5\%$ deviation.***
3. ***NX1 is using Time-Sharing Mutli-Process Architecture. Deviations described in Note 1, Note 2 do not include time of context switching.***

Ex. Delay 0.1sec.

[Path]

PowerOn: Delay(0.1) ; Delay 0.1 second.

or

PowerOn: Delay(100ms) ; Delay 100ms.

5.27.2 Delay(Rk:Rj:Ri)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

According to the value of RAM to set delay duration. Ri=Low Nibble, Rj=Middle Nibble, Rk=High Nibble.

- In, NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T, the time unit is 4.096ms.
- In NX1, the time unit is 1ms ◦

Ex. In NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T, calculate Ri, Rj, Rk to delay 1 second.

1000 ms / 4.096 ms = 244.1 (244.1 is rounded up to 245), express 245 to hexadecimal 0xF5. (4.096ms is the minimum unit of counter)

TR1: R2=0x0, R1=0xF, R0=0x5, Delay(R2:R1:R0) ; Delay for 1 second.

Ex. In NX1, 1000 means 1sec.

[Variable]

Var16: Delay_Time

[Path]

TR1: Delay_Time=1000, Delay(Delay_Time) ; Delay for 1 second.

5.27.3 WaitDN(n)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

If Delay command is being executed, the system will execute next command after finishing. Otherwise, the system will immediately execute next command.

n: 0 = foreground, 1 = background1, 2 = background2 and 3=background3. If n is omitted, wait all Delay command .

Ex.

[Path]

TR1: Delay(5)

TR2: WaitDN(1), BG1 ; Wait till delay at background1 ends,
; execute Out command.

[Background1]

BG1: Delay(5) ; Delay 5 seconds at background1.

5.27.4 StopD(n)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Stop delay.

n: 0 = foreground, 1 = background1, 2 = background2 and 3=background3. If n is omitted, all Delay will be stopped..

Ex.

[Path]

TR1: BG1 ; Call BG1.

TR2: StopD(1) ; Stop delay at background1.

[Background1]

BG1: Delay(5) ; Delay 5 seconds at background1.

Ex.

[Path]

PowerOn: BG1, Delay(3) ; Call BG1 and delay 3 seconds at foreground.

TR1: StopD ; Stop delay at foreground and background1.

[Background1]

BG1: Delay(5) ; Delay 5 seconds at background1.

5.27.5 PauseD(n)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command can be used to pause the designated Delay.

n: 0 = foreground, 1 = background1, 2 = background2 and 3=background3. If n is omitted, all Delay will be paused.

[Path]

TR1: BG1 ; Call BG1.

TR2: PauseD(1) ; Pause delay at background1.

TR3: ResumeD(1) ; Resume delay at background1.

[Background1]

BG1: Delay(5) ; Delay 5 seconds at background1.

5.27.6 ResumeD(n)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command can be used to resume Delay.

n: 0 = foreground, 1 = background1, 2 = background2 and 3=background3. If n is omitted, all Delay will be resumed.

[Path]

TR1: BG1 ; Call BG1.
 TR2: PauseD(1) ; Pause delay at background1.
 TR3: ResumeD(1) ; Resume delay at background1.
 [Background1]
 BG1: Delay(5) ; Delay 5 seconds at background1.

5.27.7 SDelay(time)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

SDelay command is for setting short delay. Time unit: ms or us (default is ms).

Note:

1. No other step can be conducted during the delay. The next step can only be executed after the delay is finished.
2. For NY4 / NY5 / NY5+ / NY9T, time range is 50us ~ 12500us, increased by 50us.
3. For NY6 / NY7, time range is 10us ~ 4000us, increased by 10us.
4. In NX1, time range is 1us ~ 4000us, increased by 1us.

Ex. Delay 250us.

DELAY_TEST: SDelay(250us) ; Delay 250us.

5.28 Action Command

Action Command				
PlayA	PlayAS	WaitAN(Ch)	PauseA(Ch)	ResumeA(Ch)
HoldA(Ch)	StopA(Ch)	-	-	-

5.28.1 PlayA / PlayAS

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

User can use this command in Q-Code to output signals of Action, which is drawn using Q-Visio.

PlayA (Px.n, Ch, {VIOLabel.}Label{, Extension}{, Loop}) { *n } { &[BG1 {BG2 {BG3}}] }

PlayAS (Px.n, Ch, {VIOLabel.}Label{, Extension}{, Loop}) { &[BG1 {BG2 {BG3}}] }

PlayA (Ch, VIOLabel{, Extension}{, Loop}) { *n } { &[BG1 {BG2 {BG3}}] }

PlayAS (Ch, VIOLabel{, Extension}{, Loop}) { &[BG1 {BG2 {BG3}}] }

PlayA: Next command will be executed after Action play ends.

PlayAS: Next command will be executed right after Action play starts.

Px.n: Output to assigned pin.

Ch: Action channels.

- NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T support Ch1 ~ Ch8.

- NX1 supports Ch1 ~ Ch32.

VIOLabel: Indicate which action file to play.

Label: Indicate which signal from action file to play.

Extension: The extension function can extend length of signals to save ROM size. Range 1 ~ 15.

- Assign value. Ex. PlayA(PE.0, 1, \$VIO0.A1, 4)
- Ri (supports 1 Ri) Ex. PlayA(PE.0, 1, \$VIO0.A1, R0)

Loop: Output action repeatedly until other command stops action output, ex, StopA. This parameter is optional.

Note:

1. *NX1 does not support play by whole action file, only supports single signal in action file.*
2. *NY4 / NY5 / NY5+ / NY6 / NY7 / NX1 do not support Extension parameter.*
3. *When user plays VIO files and outputs multiple signals simultaneously, the signals must have equal output time length. If the time lengths are unequal, the output time would base on the longest time length of the signal, and the deficiency time length of other signals would be output as 0% forcibly.*
4. *In NY9T, the resolution of Action counter is only 1ms, the variable range is narrow, and it is not recommended that set Ascend/Descend as output signals.*
5. *When user opens Loop function and outputs multiple signals, the signals must have equal output time length to avoid signal discontinuity.*

Ex.

Path1: PlayA(PB.0, 4, \$A1) ; Play signal "A1" at channel 4, output to PB.2.

Path2: PlayAS(PB.1, Ch3, \$EyeMotion) ; Play signal "EyeMotion" at channel 3, output to PB.1.

Path3: PlayAS(PB.2, Ch2, \$A3) ; Play signal "A3" at channel 2, output to PB.2.

Path4: PlayA(PB.2, Ch2, \$VIO1.A3) ; Play signal "A3" of VIO1 action file at channel 2,
; output to PB.2

Ex. Use [Action] to output multiple action signals at once.

[Action File]

VIO0 = Action1.vio ; Add action file.

VIO1 = Action2.vio

[Output State]

Output_0: A A A A A A A A ; Use PA and PB as action signal output.

[Action]

Compression = Enable

FrameRate = 80

Multi_Pins = [PA.0, PA.1, PA.2, PA.3, PB.0, PB.1, PB.2, PB.3] ; Use PA / PB as action output.

MVIO0 = [VIO0.A1, VIO0.A2, VIO0.A3, VIO0.A4, VIO0.A5, VIO0.A6, VIO0.A7, VIO0.A8]

; MVIO0 uses A1 ~ A8 from VIO0.

MVIO1 = [VIO1.A8, VIO1.A7, VIO1.A6, VIO1.A5, VIO1.A4, VIO1.A3, VIO1.A2, VIO1.A1]

; MVIO1 uses A8 ~ A1 from VIO1.

[Path]

TR1: Output_0, PlayA(Ch1, \$MVIO0) ; Use MVIO0 to output Action signal to PA and PB.

TR2: Output_0, PlayA(Ch1, \$MVIO1) ; Use MVIO1 to output Action signal to PA and PB.

5.28.2 WaitAN(Ch)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

If Action file on the designated channel is being executed, the system will execute next command after finishing. Otherwise, the system will immediately execute next command.

Ch: Action channel to wait, wait all action if omitted.

- NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T support Ch1 ~ Ch8.
- NX1 supports Ch1 ~ Ch32.

5.28.3 PauseA(Ch)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command can be used to pause Action file on the designated channel currently being played.

Ch: Action channel to pause, pause all action if omitted.

- NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T support Ch1 ~ Ch8.
- NX1 supports Ch1 ~ Ch32.

5.28.4 ResumeA(Ch)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command can be used to resume Action file on the designated channel.

Ch: Action channel to resume, resume all action if omitted.

- NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T support Ch1 ~ Ch8.
- NX1 supports Ch1 ~ Ch32.

5.28.5 HoldA(Ch)

[NY9T]

Pause the Action playing, and hold the current output status.

Ch: Action channel to hold. If not assigned, all action channels will be hold. Ch=1 ~ 8 or Ch1 ~ Ch8.

5.28.6 StopA(Ch)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Stop Action file on the designated channel.

Ch: Action channel to stop, stop all action if omitted.

- NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T support Ch1 ~ Ch8.
- NX1 supports Ch1 ~ Ch32.

Ex.

[Path]

```

TR1: PlayA(PB.0, 1, $A0)           ; Play $A0 on channel 1, and assign output value to PB0.
TR2: PauseA(1)                     ; Pause Action on channel 1.
TR3: Resume(1)                     ; Resume Action on channel 1.
TR4: StopA(1)                       ; Stop Action on channel 1.
TR5: WaitAN(1), Out                 ; If Action on channel 1 is being executing, the system
                                   ; wait till Action on channel 1 ends. Then execute Out.

```

5.29 LED Strip Command

LED Strip Command				
LEDStr_Play	LEDStr_PlayS	LEDStr_Stop	LEDStr_Clear	LEDStr_Brightness
LEDSync_Play	LEDSync_PlayS	LEDSync_Stop	LEDSync_Clear	LEDSync_Brightness
LEDText_Play	LEDText_PlayS	LEDText_Stop	LEDText_Clear	LEDText_Brightness

5.29.1 LEDStr_Play / LEDStr_PlayS

[NX1]

The LED strip commands can play the Single-String strip file.

LEDStr_Play(Px.n, LEDLabel)

LEDStr_Play(Px.n, index)

LEDStr_Play(Px.n, index, Storage)

LEDStr_PlayS(Px.n, LEDLabel)

LEDStr_PlayS(Px.n, index)

LEDStr_PlayS(Px.n, index, Storage)

LEDStr_Play: After the LED strip playback ends, executing the next command.

LEDStr_PlayS: When the LED strip playback starts, executing the next command immediately.

Px.n: Output pin.

LEDLabel: Specify the target LED file to play.

Index: Play the index value of file, it could be immediate value or variable.

Storage: Location of LED strip data. Accepts SPI0 / SPI1, Internal ROM if omitted.

Ex.

[LED Strip]

Timer = PWMA

Period = 50ms

Single_LED_Order = RGB

Single_Data0 = H300_L900

Single_Data1 = H900_L300

Single_LED_Count = [PA.2: 100]

LED0 = Strip.csv /Color_Order=RGB /Format=Single ; Add LED Strip files (.csv).

[Path]

TR1: LEDStr_PlayS(PA.2, \$LED0) ; Play LED0, outputto PA.2 °

TR2: SpiPlayS(ch,0), LEDStr_Play(PA.2, 0) ; Play 0th Speech and 0th Strip file.

5.29.2 LEDStr_Stop

[NX1]

This command is for stopping the playback of Single-String LED strip.

LEDStr_Stop

LEDStr_Stop(Px.n)

Px.n: Stop the Single-String of LED strip output on specified pin. Stop Single-String of LED strip output on all pin if omitted.

Ex.

TR1: LEDStr_Stop ; Stop the LED strip playback.

5.29.3 LEDStr_Clear

[NX1]

This command is for clearing the Single-String LED Strip and turning off all LEDs.

LEDStr_Clear

LEDStr_Clear(Px.n)

Px.n: Specify the pin to clear Single-String output of LED Strip, clear all Single-String output of LED Strip on all pins if omitted.

Ex.

TR1: LEDStr_Clear ; Turn off the LED strip.

5.29.4 LEDStr_Brightness

[NX1]

This command is for adjusting the brightness of Single-String LED strip, n=0 ~ 15.

Ex.

TR1: LEDStr_Brightness(15) ; Set the brightest luminance of LED strip.

5.29.5 LEDSync_Play / LEDSync_PlayS

[NX1]

The LED strip commands can play the Sync-Play strip file.

LEDSync_Play(Px.n, LEDLabel)

LEDSync_Play(Px.n, index)

LEDSync_Play(Px.n, index, Storage)

LEDSync_PlayS(Px.n, LEDLabel)

LEDSync_PlayS(Px.n, index)

LEDSync_PlayS(Px.n, index, Storage)

LEDSync_Play: After the LED strip playback ends, executing the next command.

LEDSync_PlayS: When the LED strip playback starts, executing the next command immediately.

LEDLabel: Specify the target Sync-Play LED file to play.

Index: Play the index value of file, it could be immediate value or variable.

Storage: Location of LED strip data. Accepts SPI0 / SPI1, Internal ROM if omitted.

Ex.

[LED Strip]

Timer = PWMB

Period = 50ms

Sync_LED_Order = RGB

Sync_Data0 = H300_L900

Sync_Data1 = H900_L300

Sync_LEDCount = [PA.0:7, PA.1:10]

LED0 = Sync.csv /Color_Order=RGB /Format=Sync ; Add LED Strip files (.csv).

[Path]

TR1: LEDSync_PlayS(\$LED0) ; Play LED0, outputto PA.2 °

TR2: SpiPlayS(ch,0), LEDSync_Play(0) ; Play 0th Speech and 0th Strip file.

5.29.6 LEDSync_Stop

[NX1]

This command is for stopping the playback of Sync-Play LED strip.

LEDSync_Stop

Ex.

TR1: LEDSync_Stop ; Stop the LED strip playback.

5.29.7 LEDSync_Clear

[NX1]

This command is for clearing the Sync-Play LED Strip and turning off all LEDs.

LEDSync_Clear

Ex.

TR1: LEDSync_Clear ; Turn off the LED strip.

5.29.8 LEDSync_Brightness

[NX1]

This command is for adjusting the brightness of Sync-Play LED strip, n=0 ~ 15.

Ex.

TR1: LEDSync_Brightness(15) ; Set the brightest luminance of LED strip.

5.29.9 LEDText_Play / LEDText_PlayS

[NX1]

The Scrolling-Text LED strip commands can play the Single-String strip file.

LEDText_Play(LEDLabel)

LEDText_Play(index)

LEDText_Play(index, Storage)

LEDText_PlayS(LEDLabel)

LEDText_PlayS(index)

LEDText_PlayS(index, Storage)

LEDText_Play: After the LED strip playback ends, executing the next command.

LEDText_PlayS: When the LED strip playback starts, executing the next command immediately.

LEDLabel: Specify the target Scrolling-Text LED file to play.

Index: Play the index value of file, it could be immediate value or variable.

Storage: Location of LED strip data. Accepts SPI0 / SPI1, Internal ROM if omitted.

Ex.

[LED Strip]

Timer = PWMB

Period = 50ms

Text_LED_Order = RGB

Text_Data0 = H300_L900

```

Text_Data1 = H900_L300
Text_MatrixUnit = 8x8
Text_MatrixType = Type1
Text_Pins_Matrix = 4x2
Text_Pins = [PA.0, PA.1, PA.2, PA.3, PA.4, PA.5, PA.6, PA.7]
LED0 = Text1.csv /Color_order=RGB /Format=Text ; Add LED Strip files (.csv).
[Path]
TR1: LEDText_PlayS($LED0) ; Play LED0, outputto PA.2 °
TR2: SpiPlayS(ch,0), LEDText_Play(0) ; Play 0th Speech and 0th Strip file.

```

5.29.10 LEDText_Stop

[NX1]

This command is for stopping the playback of Scrolling-Text LED strip.

LEDText_Stop

Ex.

```
TR1: LEDText_Stop ; Stop the LED strip playback.
```

5.29.11 LEDText_Clear

[NX1]

This command is for clearing the Scrolling-Text LED Strip and turning off all LEDs.

LEDText_Clear

Ex.

```
TR1: LEDText_Clear ; Turn off the LED strip.
```

5.29.12 LEDText_Brightness

[NX1]

This command is for adjusting the brightness of Scrolling-Text LED strip, n=0 ~ 15.

Ex.

```
TR1: LEDStr_Brightness(15) ; Set the brightest luminance of LED strip.
```

5.30 QFID Command

QFID Command				
QFID_GroupID	QFID_TagId	QFID_TagInput	QFID_On	QFID_Off
QFID_SlowOn	QFID_SlowOff	-	-	-

5.30.1 QFID_GroupID(Ri)

[NX1]

Scan all the QFID Group. Ri[7:0] is the state of Group. When the corresponding bit shows 1, it means the tags of corresponding Group is detected. 0 means un-detected.

5.30.2 QFID_TagId (GroupID, Ri:Rk:Rj:Ri)

[NY5+ / NY7 / NX1]

Scan all the Tag states.

GroupID: specify Group ID.

Ri is the state of Tag0 ~ Tag3. When the corresponding bit shows 1, it means the corresponding Tag is detected. 0 means un-detected.

Rj is the state of Tag4 ~ Tag7. When the corresponding bit shows 1, it means the corresponding Tag is detected. 0 means un-detected.

Rk is the state of Tag8 ~ Tag11. When the corresponding bit shows 1, it means the corresponding Tag is detected. 0 means un-detected.

Ri is the state of Tag12 ~ Tag15. When the corresponding bit shows 1, it means the corresponding Tag is detected. 0 means un-detected.

Note: NY5+ / NY7 does not support GroupID parameter.

5.30.3 QFID_TagInput (ID, Ri)

[NY5+ / NY7 / NX1]

Read the input state of 3 I/O pins on Tag.

GroupID: specify Group ID.

ID is the specified Tag ID, its range is 0~15 which can be immediate value or specified by variable.

Ri: result Ri[2:0] means the read I/O state, Ri[0] means state of I/O1, Ri[1] means I/O2 and Ri[2] means I/O3.

Note: NY5+ / NY7 does not specify the GroupID parameter.

5.30.4 QFID_On

[NY5+ / NY7 / NX1]

Turn on the QFID scanning function. This function is disable by default.

5.30.5 QFID_Off

[NY5+ / NY7 / NX1]

Turn off the QFID scanning function.

5.30.6 QFID_SlowOn

[NY5+ / NY7]

Turn on the QFID slow mode. This function will excute in scanning once and resting several times, and the rest times is determined by the value of option QFID Scan Interval. Please refer to QFID section for details. It can also reduce the current consumption, but the response speed will be reduced as well.

Note:

1. *This command and the Slow command of MISC command can not be executed at the same time. Otherwise, the length of rest time will be incorrect.*
2. *Before executing this command, users must select the slow pin in the QFID Option and connect a resistor and a capacitor in parallel with the pin. Otherwise, the operation will be abnormal. Please refer to the QFID Reader Application Circuit of QFID Option for Resistance and capacitance values.*

5.30.7 QFID_SlowOff

[NY5+ / NY7]

Turn off the QFID slow mode.

5.31 WaveID Command

WaveID Command				
WaveID_TX	WaveID_RX_ON	WaveID_RX_OFF	WaveID_RX	

5.31.1 WaveID_TX(Ri)

[NX1]

Send the WaveID code.

Ri: It's the specified code, it could be immediately value or variable.

Ex.

[Option]

WaveID = TX

; Enable WaveID TX.

WaveID_CommandCount = 8

[Path]

TR1: WaveID_TX(5)

; Send the code 5.

TR2: WaveID_TX(R0)

; Send the code R0.

5.31.2 WaveID_RX_ON

[NX1]

Turn on the WaveID receiving function.

Ex.

[Path]

TR1: WaveID_RX_ON

; Turn on the receiving function.

5.31.3 WaveID_RX_OFF

[NX1]

Turn off the WaveID receiving function.

Ex.

[Path]

TR1: WaveID_RX_OFF

; Turn off the receiving function.

5.31.4 WaveID_RX

[NX1]

User can store the received WaveID code in the specified variable through this command.

Ex.

[Option]

WaveID = RX

; Enable WaveID RX.

WaveID_CommandCount = 8

[Variable]

Var8: WaveId_Value

[Path]

TR1: WaveID_RX_On

WaveID_RX: WaveId_Value = WaveID_RX

; Store the received code in WaveId_Value.

5.32 Sound Localization Command

Sound Localization Command				
SL_On	SL_Off	SL_RX	-	-

5.32.1 SL_On

[NX1]

Enable scan of Sound Localization. Default is off after poweron.

Ex.

[Option]

SoundLoc = Enable ; Enable Sound Localization.

SoundLoc_Amp_Pin = PA.15

SoundLoc_Amp_Shutdown_Control = Active_Low

[Variable]

Var8: R_Angle

[Path]

TR1: SL_On ; Activate Sound Localization.

SL_RX: R_Angle=SL_RX, R_Angle=30?SL_30, R_Angle=60?SL_60

SL_30: PlayV(Ch0, \$V0) ; Play voice \$V0 fi R_Angle is 30.

SL_60: PlayV(Ch0, \$V1) ; Play voice \$V1 if R_Angle is 60.

5.32.2 SL_Off

[NX1]

Disable scan of Sound Localization.

5.32.3 Var=SL_RX

[NX1]

Read scanned angle of Sound Localization and restore to Var. Angle can be 0 / 30 / 60 / 90 / 120 / 150 / 180 / 210 / 240 / 270 / 300 / 330. Its recommended to use this command inside SL_RX path.

5.33 Sound Detect Command

Sound Detect Command				
SoundDetect_On	SoundDetect_Off	-	-	-

5.33.1 SoundDetect_On

[NX1]

Activate sound detection.

Ex.

[Option]

SoundDetect = Enable

SoundDetect_High_Threshold = 300

SoundDetect_Low_Threshold = 300

SoundDetect_Active_Time = 80ms

SoundDetect_Mute_Time = 650ms

[Path]

TR1: SoundDetect_On

; Activate sound detection.

Sound_Detect: PA.0 = 1

; Set PA.0 to 1 when sound is detected.

Sound_Mute: PA.0 = 0

; Set PA.0 to 0 when sound is not detected for more
; than 150ms.

5.33.2 SoundDetect_Off

[NX1]

Deactivate sound detection.

5.34 Pitch Detect Command

Pitch Detect Command				
PitchDetect_On	PitchDetect_Off	ReadPitch	-	-

5.34.1 PitchDetect_On

[NX1]

Activate pitch detection.

Ex.

[Option]

PitchDetect = Tone

; Pure-tone detection.

PitchDetect_Range = Middle

; Detection range: 1000 ~ 3000Hz.

[Variable]

Var16: Pitch_Value

[Path]

TR1: PitchDetect_On	; Activate pitch detection.
TR2: PitchDetect_Off	; Deactivate pitch detection.
Pitch_Detected: ReadPitch(Pitch_Value)	; Pure tone in 1000 ~ 3000Hz is detected, ; save detected pitch to variable Pitch_Value.

5.34.2 PitchDetect_Off

[NX1]

Deactivate pitch detection.

5.34.3 ReadPitch

[NX1]

Read detected pitch.

ReadPitch(Var)

Var: Save detected pitch to Var, range 31 ~ 4000.

5.35 RFC Command

RFC Command				
RFC_On	RFC_Off	RFC_Level(Ri)	-	-

5.35.1 RFC_On

[NY5+ / NY6 / NY7 / NX1]

Turn on the RFC scanning function. This function is disabled by default.

5.35.2 RFC_Off

[NY5+ / NY6 / NY7 / NX1]

Turn off the RFC scanning function.

5.35.3 RFC_Level(Ri)

[NY5+ / NY6 / NY7 / NX1]

Read the detected RFC value and store the value in Ri.

Note:

1. Use the RFC_On command to enable the RFC function and wait for about 30ms before

getting the correct RFC value. Otherwise, user may get an unknown value.

2. *Before executing this command, users must connect a resistor and a capacitor to the RFC pin. Please refer to the RFC Application Circuit of RFC Option.*

5.36 ADC Input Command

ADC Input Command				
ADC_Input	=	=	-	-

5.36.1 ADC_Input

[NX1]

Read from A/D Converter.

ADC_Input(Px.n, Var)

Px.n: Read A/D Converter result from specified pin.

Var: Store result to Var, range is 0 ~ 0xFFFF0.

Ex.

[Option]

ADC_Input_Pins = PA.1, PA.2

[Variable]

Var16: Result

[Path]

TR: ADC_Input(PA.2, Result) ; Read from A/D converter on PA.2 and
; save to Result °

5.37 VR Command

VR Command				
VR State	VR_ON	VR_OFF	VR_VAD=n	VR_VAD_On
VR_VAD_Off	VRGC_Timeout_CLR		Ri = VR_HitScore	Ri = VR_HitID
Ri=VR_Loading	VT_Training	VT_Delete	VT_DeleteAll	VT_TrainingNum

5.37.1 VR State

[NX1]

VR state is for defining the corresponding state of voice command. The VR state means the corresponding reactions while the voice commands are recognized. User can set different VR States to react differently when voice commands are identified in different scenarios. The voice command can

be divided into groups, but only one group of commands can be selected as the current voice recognition commands.

Ex.

[VR]

VRG1_0: VR1 VR2 VR3 ; Set the voice recognition group 1 state 0

VRG2_0: VR4 VR5 ; Set the voice recognition group 2 state 0

[Path]

PowerOn: VRG1_0 ; Activate VRG1_0, and only commands of group 1 can be recognized

5.37.2 VR_ON

[NX1]

Enable the voice recognition function. When the VR function is enabled, *Q-Code_NX1* will continue detecting if there is a voice command. Please note that when the speech or music is played, the voice recognition function will pause automatically, and won't automatically resume until the end of the voice playback. Remember to set the VR State before enabling the voice recognition function. User can switch VR State at any time when the function is enabled.

Ex.

[VR]

VRG1_0: VR1 VR2 VR3

[Path]

PowerOn: VRG1_0, VR_ON ; Enable the voice recognition function.

5.37.3 VR_OFF

[NX1]

Disable the voice recognition function.

Ex.

[Input State]

KEY1: TR1 TR2 X/TR4

[VR]

VRG1_0: VR1 VR2 VR3

[Path]

PowerOn: KEY1,VRG1_0,VR_ON

TR1: VR_OFF ; Disable the voice recognition function.

5.37.4 VR_VAD=n

[NX1]

When using VR, using this command to set the VAD (Voice Activity Detection) threshold of input sound.

VAD is used to detect the existence of voice signals, set the threshold means that the the recognition must in above a certain volume. Larger thresholds can filter out larger background noise, but it may also filter the smaller volume commands. If the threshold is too small may lead that the background noise can't be filtered to prevent maloperation. Users need to set the threshold according to the environment. The default threshold is 4, and VR_VAD_Off when starting up.

This VAD threshold is equivalent to the energy threshold in the CSMT (Cyberon CSpotter Modeling Tool), but their definitions are not the same. User can choose one of them to use. If energy threshold > 0, user can turn off the VAD command. If user turns on the VAD command, the energy threshold can be set as 0. After setting the threshold, VAD needs to be executed by VR_VAD_On command. The default is VR_VAD_Off.

n: The threshold which can be immediate value or variable. n = 0 ~ 16.

Threshold	Average Volume (%)	Average Volume (Sample)	Decription
0	0.00%	0	Background noise is not filtered, equivalent to VAD is disabled.
1	0.46%	150	Suitable for very quiet enclosed room.
2	0.61%	200	Suitable for closed room with a little background noise but no talk.
3	0.76%	250	Suitable for room with background noise or music but not no talk.
4	0.92%	300	Suitable for room with people talk 3 meters away
5	1.07%	350	Suitable for room with people talk 1 meter away
6	1.22%	400	Suitable for room with slight activity within 1 meter
7	1.53%	500	Suitable for some activities within 1 meter
8	1.83%	600	Suitable for rooms with people talk within 1 meter
9	2.14%	700	Suitable for more quiet open space
10	2.44%	800	Suitable for open space with slight background noise
11	3.66%	1000	Suitable for open space with relatively large background noise
12	3.66%	1200	Suitable for open space with larger background noise or people talk
13	4.58%	1500	Suitable for open space with activities or people talk within 1 meter
14	6.10%	2000	Suitable for open space with activities or have a larger voice within 1 meter
15	9.16%	3000	For more noisy environment, requires a large volume to identify
16	13.73%	4500	For a very noisy environment such as an exhibition, need larger volume to identify

Note: If energy threshold > 0, RAM consumes about 300 bytes, while VAD only consumes 12

bytes.

Ex.

VR_VAD=1 ; Set the VAD threshold as 1.

5.37.5 VR_VAD_On

[NX1]

Turn on the VAD function of voice recognition. When VAD is enabled, the volume of voice command must be larger than threshold to be interpreted and recognized.

Ex.

VR_VAD_On ; Turn on VAD.

5.37.6 VR_VAD_Off

[NX1]

Turn off the VAD function of voice recognition.

Ex.

VR_VAD_Off ; Trun off VAD.

5.37.7 VRGC_Timeout_CLR

[NX1]

Clear the Timeout timer of VRGC state. If a command in any VR groups is recognized successfully, the timeout timer will be activated. When all the voice commands in VR groups have been recognized, user can use this command to clear the Timeout timer.

Ex.

VRGC_Timeout_CLR ; Clear Timeout timer.

5.37.8 Ri = VR_HitScore

[NX1]

Get the score of the successfully identified voice command. The higher score means hitting the voice commands better. This command only can be used after the voice command has been successfully recognized.

Ex.

R0 = VR_HitScore ; Store the recognized score of voice command to variable R0.

5.37.9 Ri = VR_HitID

[NX1]

Get the index value of the successfully identified voice command. This command only can be used after the voice command has been successfully recognized.

Ex.

R0 = VR_HitID ; Store the index value of voice command to variable R0.

5.37.10 VR_Loading

[NX1]

Access the real-time CPU loading while executing VR. The unit is percentage.

Ex.

R0 = VR_Loading ; Store the CPU loading to variable R0.

5.37.11 VT_Training

[NX1]

This command is for training voice tag. If the training is successive, it can be applied to the usage of voice recognition.

Ex.

VT_Training(1) ; Train the first group of voice command.

5.37.12 VT_Delete

[NX1]

Delete the successfully trained voice command.

Ex.

VT_Delete(1) ; Delete the first group of voice command.

5.37.13 VT_DeleteAll

[NX1]

Delete all the successfully trained voice commands.

Ex.

VT_DeleteAll ; Delete all voice commands.

5.37.14 VT_TrainingNum

[NX1]

Set the training number of recording, the number is up to 3, and 1 at least. The more the number, the easier it is to recognize voice commands. The default of Voice Tag is 3, and 1 for the Voice Password.

Ex.

VT_TrainingNum=3 ; Set the training number as 3.

5.38 Sound Effect Command

Sound Effect Command				
PitchChange	PitchChange_Off	SpeedChange	SpeedChange_Off	Robot1
Robot1_Off	Robot2	Robot2_Off	Robot3	Robot3_Off
Robot4	Robot4_Off	Echo	Echo_Off	Reverb
Reverb_off	Darth	Darth_Off	AnimalRoar	AnimalRoar_Off

5.38.1 PitchChange

[NX1]

This command is used to change the pitch of the voice file. The maximum can be adjusted to two times of the normal pitch, the minimum can be adjusted to half of the normal pitch.

PitchChange(Channel, Pitch)

Channel: channel to change pitch.

Pitch: Constant or variable, the range must be between -12 ~ +12.

Value	Description
-12	0.5X
-11	0.53X
-10	0.56X
-9	0.59X
-8	0.63X
-7	0.67X
-6	0.72X
-5	0.75X
-4	0.79X
-3	0.84X
-2	0.89X
-1	0.94X
0	Close PitchChange

Value	Description
1	1.06X
2	1.12X
3	1.19X
4	1.26X
5	1.33X
6	1.41X
7	1.50X
8	1.59X
9	1.68X
10	1.78X
11	1.89X
12	2X

Note:

1. *Pitch = 0 means PitchChange_Off.*
2. *If the program uses 2 channels, the PitchChange only can use 1 channel. 2 channels for the command at the same time is not allowed.*
3. *The channel specified by PitchChange and the channel specified by SpeedChange must operate on the same channel at the same time*
4. *This command is only available for Voice files, no effect on MIDI files.*
5. *This command only applies to the voice with sampling rate less then or equal to 16KHz.*
6. *This command is only available for ADPCM / PCM / SBC-1 / CELP.*

Ex. PitchChange(ch1, 1)

5.38.2 PitchChange_Off

[NX1]

Close PitchChange effect.

Note: *The PitchChange_Off command is available only when PitchChange command is used.*

5.38.3 SpeedChange

[NX1]

This command is used to change the voice speed of playing file. The maximum can be adjusted to two times of the normal speed, the minimum can be adjusted to half of the normal speed.

SpeedChange(channel, number)

Channel: channel to change speed.

Number: Constant or variable, the range must be between -12 ~ +12.

Value	Description
-12	0.5X
-11	0.54X
-10	0.58X
-9	0.62X
-8	0.67X
-7	0.71X
-6	0.75X
-5	0.79X
-4	0.83X
-3	0.87X
-2	0.92X
-1	0.96X
0	Close SpeedChange
1	1.08X
2	1.17X
3	1.25X
4	1.33X
5	1.42X
6	1.45X
7	1.58X
8	1.67X
9	1.75X
10	1.83X
11	1.92X
12	2X

Note:

1. **Number = 0 means SpeedChange_Off.**
2. **If the program uses 2 channels, the SpeedChange only can use 1 channel. 2 channels for the command at the same time is not allowed.**
3. **The channel specified by PitchChange and the channel specified by SpeedChange must operate on the same channel at the same time.**
4. **This command is only available for Voice files, no effect on MIDI files.**
5. **This command is only useful to the Voice file that is less than 16K of the sampling rate.**
6. **This command is only available for ADPCM / PCM / SBC-1 / CELP.**

Ex. SpeedChange(ch1, 1)

5.38.4 SpeedChange_Off

[NX1]

Close SpeedChange effect.

Note: The SpeedChange_Off command is available only when SpeedChange command is used.

5.38.5 Robot1

[NX1]

The effect of robot speech.

Robot1(Channel, number)

Channel: channel to apply robot speech effect.

Number: the range is between 0 ~ 15.

Note:

1. This command is only available for Voice files, no effect on MIDI files.
2. This command is only available for ADPCM / PCM / SBC-1 / CELP.

5.38.6 Robot1_Off

[NX1]

Close Robot1 effect.

Note: The Robot1_Off command is available only when Robot1 command is used.

5.38.7 Robot2

[NX1]

To generate the effect of robot speech.

Robot2(Channel, Type,Thres)

Channel: voice channels.

Type: The range is 0 ~ 2, means different scales.

Thres: The noise threshold range is 0~7. The normal voice playback can be set 0 because there is no external noise.

Note:

1. If the program uses 2 channels, the Robot2 only can use 1 channel. 2 channels for the command at the same time is not allowed.
2. This command is only available for Voice files, no effect on MIDI files.
3. This command is only available for ADPCM / PCM / SBC-1 / CELP.

Ex. Robot2(ch0, 2,0)

5.38.8 Robot2_Off

[NX1]

Turn off the effect of Robot2.

Note: The Robot2_Off command is available only when Robot2 command is used.

5.38.9 Robot3

[NX1]

To generate the effect of robot speech.

Robot3(Channel, Type, Thres)

Channel: voice channels.

Type: The range is 0 ~ 2, means different scales.

Thres: The noise threshold range is 0~7. The normal voice playback can be set 0 because there is no external noise.

Note:

- 1. If the program uses 2 channels, the Robot3 only can use 1 channel. 2 channels for the command at the same time is not allowed.*
- 2. This command is only available for Voice files, no effect on MIDI files.*
- 3. This command is only available for ADPCM / PCM / SBC-1 / CELP.*

Ex. Robot3(ch0, 2,0)

5.38.10 Robot3_Off

[NX1]

Turn of the Robot3 effect.

Note: The Robot3_Off command is available only when Robot3 command is used.

5.38.11 Robot4

[NX1]

To generate the effect of robot speech.

Robot4(Channel)

Channel : voice channel, only supports Ch0 ◦

Note:

1. *Robot4 effect only apply to ADPCM.*
2. *This command is only available for Voice files, no effect on MIDI files.*

Ex. Robot4(Ch0)

5.38.12 Robot4_Off

[NX1]

Turn off Robot4 effect.

Note: The Robot4_Off command is available only whe Robot4 command is used.

5.38.13 Echo

[NX1]

To generate the echo effect.

Echo(Channel, Level)

Channel: voice channels

Level: The echo range is 0 ~ 7, means different level, and 7 is the longest delay. The parameteris only availabe for SBC / ADPCM Channel 1 / CELP / PCM.

Option ADPCM_ECHO_DELAY: short means short delay, and long means long delay effect. This option is only available for Channel0 and the voice file uses ADPCM algorithm.

Note:

1. *If the program uses 2 channels, the Echo only can use 1 channel. 2 channels for the command at the same time is not allowed.*
2. *For a better effect, it's recommended to apply the Echo command by selecting ADPCM with Channel0.*
3. *This command is only available for Voice files, no effect on MIDI files.*
4. *This command is only available for ADPCM / PCM / SBC-1 / CELP.*

Ex.

[Option]

ADPCM_ECHO_DELAY = Short ; Selects short delay for echo

[Path]

PowerOn: PlayV(ch0,\$V0), Echo(ch0,0) ;Play the Echo effect

5.38.14 Echo_Off

[NX1]

Turn off the echo effect.

Note: *The Echo_Off command is available only when Echo command is used.*

5.38.15 Reverb

[NX1]

To generate the reverb effect.

Reverb(Channel, Level)

Channel: The voice channel.

Level: The range is 0 ~ 7 which means different level of reverb, 7 is the highest level.

Note:

1. *If the program uses 2 channels, the Reverb only can use 1 channel. 2 channels for the command at the same time is not allowed.*
2. *This command is only available for Voice files, no effect on MIDI files.*
3. *This command is only available for ADPCM / PCM / SBC-1 / CELP.*

5.38.16 Reverb_Off

[NX1]

Turn off the Reverb effect.

Note: *The Reverb_Off command is available only when Reverb command is used.*

5.38.17 DARTH

[NX1]

To generate the DARTH effect.

DARTH(Channel, Pitch, Type)

Channel: The voice channel, only Ch0 is available.

Pitch: Accepts -12 ~ 12, refer to Number parameter of PitchChange.

Type: Accepts 0 ~ 3, type of DARTH effect: 0 is for metallic sound, 1 / 2 is for metallic + husky sound, 3 is for husky sound.

Note:

1. *This command only takes effect on ADPCM algorithm.*
2. *This command only applies to voice, does not apply to MIDI playback.*

Ex. DARTH(Ch0,-4,2)

5.38.18 Darth_Off

[NX1]

Turn off the Darth effect.

Note: The Darth_Off command is available only when Darth command is used.

5.38.19 AnimalRoar

[NX1]

Apply AnimalRoar sound effect.

AnimalRoar(Channel, Speed, Pitch, Reverb)

Channel: Voice channel, only accepts Ch0.

Speed: Speed, accepts -12 ~ 12. Same as Number parameter of **SpeedChange**.

Pitch: Pitch, accepts -12 ~ 12. Same as **PitchChange**.

Reverb: Accepts 0 ~ 3.

Note:

1. This command only takes effect on ADPCM algorithm.
2. This command only applies to voice, does not apply to MIDI playback.
3. Ch0 is occupied by AnimalRoar sound effect, cannot be shared with other sound effect.

Ex. AnimalRoar(Ch0, -4, -4, 2)

5.38.20 AnimalRoar_Off

[NX1]

Turn off the AnimalRoar effect.

Note: The AnimalRoar_Off command is available only when AnimalRoar command is used.

5.39 Real Time Play Command

Real Time Play Command				
RT_Play	RT_Play_Off	RT_PitchChange	RT_PitchChange_Off	RT_Robot1
RT_Robot1_Off	RT_Robot2	RT_Robot2_Off	RT_Robot3	RT_Robot3_Off
RT_Robot4	RT_Robot4_Off	RT_Echo	RT_Echo_Off	RT_Reverb
RT_Reverb_Off	RT_Ghost	RT_Ghost_Off	RT_Darth	RT_Darth_Off
RT_Chorus	RT_Chorus_Off	RT_Vol = n	RT_Vol = Var	Var = RT_Vol

5.39.1 RT_Play

[NX1]

The real-time play command.

Note: When the command is using, the voice channel supports up to 2 channels.

Ex.

[Option]

RT_Sampling_Rate = 8000

;Set the sample rate of the real-time playback as 8k, it also can be 12k or 16k.

[Path]

PowerOn: RT_Play

;Real-time playback

5.39.2 RT_Play_Off

[NX1]

Turn off the real-time playback.

Note: The RT_Play_Off command is available only when RT_Play command is used.

5.39.3 RT_PitchChange

[NX1]

This command is used to set the pitch of the real-time playback. The maximum can be adjusted to two times of the normal pitch, the minimum can be adjusted to half of the normal pitch.

RT_PitchChange(Pitch)

Pitch: Constant or variable, the range must be between -12 ~ +12.

Value	Description
-12	0.5X
-11	0.53X
-10	0.56X
-9	0.59X
-8	0.63X
-7	0.67X
-6	0.72X
-5	0.75X
-4	0.79X
-3	0.84X
-2	0.89X
-1	0.94X
0	Close PitchChange

Value	Description
1	1.06X
2	1.12X
3	1.19X
4	1.26X
5	1.33X
6	1.41X
7	1.50X
8	1.59X
9	1.68X
10	1.78X
11	1.89X
12	2X

Note: Pitch = 0 means RT_PitchChange_Off.

Ex. RT_PitchChange(1)

5.39.4 RT_PitchChange_Off

[NX1]

Turn off the real-time playback of pitch changes.

Note: The RT_PitchChange_Off command is available only when RT_PitchChange command is used.

5.39.5 RT_Robot1

[NX1]

To generate the real-time robot sound.

RT_Robot1(Number)

Number: The range is 0 ~ 15.

Ex. RT_Robot1(1)

5.39.6 RT_Robot1_Off

[NX1]

Turn off real-time playback of robot sound.

Note: The RT_Robot1_Off command is available only when RT_Robot1 command is used.

5.39.7 RT_Robot2

[NX1]

To generate the real-time robot sound.

RT_Robot2(Type,Thres)

Type: The range is 0 ~ 2, means different scale.

Thres: The noise threshold range is 0~7, the higher parameter, the higher threshold value and is suitable for noisy environments.

Ex. *RT_Robot2(1,3)*

5.39.8 RT_Robot2_Off

[NX1]

Turn off real-time playback of robot sound.

Note: *The RT_Robot2_Off command is available only when RT_Robot2 command is used.*

5.39.9 RT_Robot3

[NX1]

To generate the real-time robot sound.

RT_Robot3(Type,Thres)

Type: The range is 0 ~ 2, means different scale.

Thres: The noise threshold range is 0~7, the higher parameter, the higher threshold value and is suitable for noisy environments.

Ex. *RT_Robot3(1,3)*

5.39.10 RT_Robot3_Off

[NX1]

Turn off the real-time playback of robot sound.

Note: *The RT_Robot3_Off command is available only when RT_Robot3 command is used.*

5.39.11 RT_Robot4

[NX1]

To generate the real-time robot sound.

Ex. *RT_Robot4*

5.39.12 RT_Robot4_Off

[NX1]

Turn off the real-time playback of robot sound.

Note: *The RT_Robot4_Off command is available only when RT_Robot4 command is used.*

5.39.13 RT_Echo

[NX1]

To generate the real-time effect of echo.

Option RT_ECHO_DELAY: short means short delay effect, long means long delay echo.

Ex.

[Option]

RT_ECHO_DELAY = Short

;Selects short delay for echo

[Path]

PowerOn: RT_Play, RT_Echo

;Use the Echo effect for the playback

5.39.14 RT_Echo_Off

[NX1]

Turn off the real-time playback of Echo.

Note: The RT_Echo_Off command is available only when RT_Echo command is used.

5.39.15 RT_Reverb

[NX1]

To generate the real-time effect of reverb.

RT_Reverb(Level)

Level: The range is 0 ~ 7 which means different level of reverb, 7 is the highest level.

Ex. RT_Reverb(7)

5.39.16 RT_Reverb_Off

[NX1]

Turn off the real-time playback of Reverb.

Note: The RT_Reverb_Off command is available only when RT_Reverb command is used.

5.39.17 RT_Ghost

[NX1]

To generate the real-time effect of ghost sound.

RT_Ghost(Pitch,Type)

Pitch: The range is -12 ~ 12 by pitch. 1~12 is recommend for female, -1~-12 for male.

Type: The range is 0 ~ 2. It's recommended to set 0~1 to simulate ghost speech, 2 for ghost scream.

Ex. RT_Ghost(12,2)

5.39.18 RT_Ghost_Off

[NX1]

Turn off the real-time playback of ghost sound.

Note: The RT_Ghost_Off command is available only when RT_Ghost command is used.

5.39.19 RT_Darth

[NX1]

To generate the effect of Darth speech.

RT_Darth(Pitch, Type)

Pitch: Range must be between -12 ~ +12. Same as Number parameter of RT_PitchChange.

Type: Range 0 ~ 3. Type of Darth speech, 0 for artificial sound, 1 / 2 for artifical + husky sound, 3 for husky sound.

Ex. RT_Darth(-4, 2)

5.39.20 RT_Darth_Off

[NX1]

Turn off the real-time playback of Darth speech.

Note: The RT_Darth_Off command is available only when RT_Darth command is used.

5.39.21 RT_Chorus

[NX1]

To generate the effect of Chorus sound.

RT_Chorus(Pitch, Vol_1, Vol_2, Vol_3, Gain)

Pitch: Accepts -12 ~ 12, refer to Number parameter of RT_PitchChange. Chorus track 1 and 3 will change pitch as specified value. Chorus track 2 will not change.

Vol_1: Accepts 0 ~ 100, volume of chorus track 1.

Vol_2: Accepts 0 ~ 100, volume of chorus track 2.

Vol_3: Accepts 0 ~ 100, volume of chorus track 3.

Gain: Accepts 0 ~ 10, overall gain.

Value	Gain
0	1.0
1	1.1
2	1.2

Value	Gain
3	1.3
4	1.4
5	1.5
6	1.6
7	1.7
8	1.8
9	1.9
10	2.0

Ex. RT_Chorus(-8, 80, 80, 80, 6)

5.39.22 RT_Chorus_Off

[NX1]

Turn off the real-time playback of Chorus speech.

Note: The RT_Chorus_Off command is available only when RT_Chorus command is used.

5.39.23 RT_Vol = n / RT_Vol = Var

[NX1]

Change the volume of real time play. Volume can be number or variable.

n: volume, accepts 0 ~ 15.

Note: RT_Vol command is available only when RT command is used.

Ex. RT_Vol = 10 ; Set RT_Vol to 10, about 67% of original volume.

Ex. RT_Vol = R0 ; Set RT_Vol as R0.

5.39.24 Var = RT_Vol

[NX1]

Get RT_Vol and save to Var.

Note: RT_Vol command is available only when RT command is used.

Ex. R0 = RT_Vol ; Get RT_Vol and save to R0.

5.40 Animaltalks Command

Animaltalks Command			
Animaltalks_On	Animaltalks_Off	Animaltalks_Record	Animaltalks_RecordS

Animaltalks_StopR	Animaltalks_Play	Animaltalks_PlayS	Animaltalks_Stop
Animaltalks_SetVoice		Animaltalks_LongSound_On	-
Animaltalks_LongSound_Off		-	-

5.40.1 Animaltalks_On

[NX1]

Activate animaltalks.

Note: *Animaltalks_On must be executed before other animaltalks commands to allocate resource.*

Ex.

[Path]

TR1: AnimalTalks_On ; Activate animaltalks.

5.40.2 Animaltalks_Off

[NX1]

Deactivate animaltalks.

Note: *Animaltalks_Off must be executed after animaltalks commands are finished to release used resources.*

Ex.

[Path]

TR1: AnimalTalks_Off ; Deactivate animaltalks.

5.40.3 Animaltalks_Record / Animaltalks_RecordS

[NX1]

Execute animalTalks recording. After recording is finished, use AnimalTalks_Play / Animaltalks_PlayS to play.

Animaltalks_Record(Label {,Time})

Animaltalks_RecordS (Label {, Time})

Animaltalks_Record: execute next command after recording is finished.

Animaltalks_RecordS: execute next command after recording is started.

Label: Record block defined in [Record].

Time: Recording time, record block length is used if omitted.

Ex.

[Path]

TR1: **Animaltalks_Record(\$Rec0)**

; Start animaltalks recording, use Rec0 block.

TR2: **Animaltalks_Play**

; Play recoded result by using Animaltalks_Play.

5.40.4 Animaltalks_StopR

[NX1]

Stops animaltalks recording.

Ex.

[Path]

TR1: **Animaltalks_Record(\$Rec0)**

TR2: **Animaltalks_StopR**

; Stop recording.

5.40.5 Animaltalks_Play / Animaltalks_PlayS

[NX1]

Execute animalTalks playback.

Animaltalks_Play: execute next command after playback is finished.

Animaltalks_PlayS: execute next command after playback is started.

Note: Animaltalks playback uses audio channel with sound effect, any playback on chosen channel will be stopped directly.

Ex.

[Path]

TR1: **Animaltalks_Play**

; Play recoded result.

5.40.6 Animaltalks_Stop

[NX1]

Stops animaltalks playback.

Ex.

[Path]

TR1: **Animaltalks_Play**

TR2: **Animaltalks_Stop**

; Stop animaltalks playback.

5.40.7 Animaltalks_SetVoice

[NX1]

Set animal voice to be used in AnimalTalks_Play / Animaltalks_PlayS.

Animaltalks_SetVoice(Index {,Storage})

Index: Index of animal voice.

Storage: Location of animal voice file to be played. Internal Storage if omitted, accepts udr, spi0, spi1.

Ex.

[Path]

TR1: Animaltalks_SetVoice(2, spi0) ; Set animal voice, 2nd voice on SPI0.

TR2: Animaltalks_Play ; Play recorded result.

5.40.8 Animaltalks_LongSound_On

[NX1]

Enable long sound mode when using Animaltalks_Play / Animaltalks_PlayS, mute section in recorded voice will be skipped.

Animaltalks_LongSound_On(Factor)

Factor: Duration factor when play, supports range: 1% ~ 65535%.

Ex.

[Path]

TR1: Animaltalks_SetVoice(2, spi0) ; Set animal voice, 2nd voice on SPI0.

TR2: Animaltalks_LongSound_On(300%) ; Enable long sound mode, duration factor to 300%.

TR3: Animaltalks_Play ; Play recorded result.

5.40.9 Animaltalks_LongSound_Off

[NX1]

Disable long sound mode when using Animaltalks_Play A / Animaltalks_PlayS.

Ex.

[Path]

TR1: Animaltalks_SetVoice(2, spi0) ; Set animal voice, 2nd voice on SPI0.

TR2: Animaltalks_LongSound_Off ; Disable long sound mode.

TR3: Animaltalks_Play ; Play recorded result.

5.41 Animalsings Command

Animalsings Command			
Animalsings_On	Animalsings_Off	Animalsings_Record	Animalsings_RecordS
Animalsings_StopR	Animalsings_Play	Animalsings_PlayS	Animalsings_Stop
Animalsings_SetVoice		Animalsings_LongSound_On	
Animalsings_LongSound_Off		Animalsings_NC_On	Animalsings_NC_Off
Animalsings_NC_Auto		-	-

5.41.1 Animalsings_On

[NX1]

Activate animalsings.

Note: *Animalsings_On must be executed before other animalsings commands to allocate resource.*

Ex.

[Path]

TR1: Animalsings_On ; Activate animalsings.

5.41.2 Animalsings_Off

[NX1]

Deactivate animalsings.

Note: *Animalsings_Off must be executed after animalsings commands are finished to release used resources.*

Ex.

[Path]

TR1: Animalsings_Off ; Deactivate animalsings.

5.41.3 Animalsings_Record / Animalsings_RecordS

[NX1]

Execute animalsings recording. After recording is finished, use Animalsings_Play / Animalsings_PlayS to play.

Animalsings_Record(Label)

Animalsings_Record(Label, Time)

Animalsings_RecordS(Label)

Animalsings_RecordS (Label, Time)

Animalsings_Record: execute next command after recording is finished.

Animalsings_RecordS: execute next command after recording is started.

Label: Record block defined in [**Record**].

Time: Recording time, record block length is used if omitted.

Ex.

[**Path**]

TR1: Animalsings_Record(\$Rec0) ; Start animalsings recording, use Rec0 block.

TR2: Animalsings_Play ; Play recoded result by Animalsings_Play.

5.41.4 Animalsings_StopR

[**NX1**]

Stops animalsings recording.

Ex.

[**Path**]

TR1: Animalsings_Record(\$Rec0)

TR2: Animalsings_StopR ; Stop recording.

5.41.5 Animalsings_Play / Animalsings_PlayS

[**NX1**]

Execute animalsings playback.

Animalsings_Play: execute next command after playback is finished.

Animalsings_PlayS: execute next command after playback is started.

Note: *Animalsings playback uses audio channel with sound effect, any playback on chosen channel will be stopped directly.*

Ex.

[**Path**]

TR1: Animalsings_Play ; Play recoded result.

5.41.6 Animalsings_Stop

[**NX1**]

Stops animalsings playback.

Ex.

[Path]

TR1: Animalsings_Play

TR2: Animalsings_Stop ; Stop animalsings playback.

5.41.7 Animalsings_SetVoice

[NX1]

Set animal voice to be used in Animalsings_Play / Animalsings_PlayS.

Animalsings_SetVoice(Index)

Animalsings_SetVoice(Index, Storage)

Index: Index of animal voice.

Storage: Location of animal voice file to be played. Internal Storage if omitted, accepts udr, spi0, spi1.

Ex.

[Path]

TR1: Animalsings_SetVoice(2, spi0) ; Set animal voice, 2nd voice on SPI0.

TR2: Animalsings_Play ; Play recorded result.

5.41.8 Animalsings_LongSound_On

[NX1]

Enable long sound mode when using Animalsings_Play / Animalsings_PlayS, mute section in recorded voice will be skipped.

Animalsings_LongSound_On(Factor)

Factor: Duration factor when play, supports range: 1% ~ 65535%.

Ex.

[Path]

TR1: Animalsings_SetVoice(2, spi0) ; Set animal voice, 2nd voice on SPI0.

TR2: Animalsings_LongSound_On(300%) ; Enable long sound mode, duration factor to 300%.

TR3: Animalsings_Play ; Play recorded result.

5.41.9 Animalsings_LongSound_Off

[NX1]

Disable long sound mode when using Animalsings_Play A / Animalsings_PlayS.

Ex.

[Path]

TR1: Animalsings_SetVoice(2, spi0) ; Set animal voice, 2nd voice on SPI0.

TR2: Animalsings_LongSound_Off ; Disable long sound mode.

TR3: Animalsings_Play ; Play recorded result.

5.41.10 Animalsings_NC_On

[NX1]

Activate noise cancellation of animalsings.

Ex.

[Path]

TR1: Animalsings_NC_On ; Activate noise cancellation.

5.41.11 Animalsings_NC_Off

[NX1]

Deactivate noise cancellation of animalsings.

Ex.

[Path]

TR1: Animalsings_NC_Off ; Deactivate noise cancellation.

5.41.12 Animalsings_NC_Auto

[NX1]

Enable automatic noise cancellatio of animalsings.

Ex.

[Path]

TR1: Animalsings_NC_Auto ; Enable automatic noise cancellation.

5.42 I/O Expander Command

I/O Expander Command				
IoExp Input State	IoExp Output State	IoExp_Key_CLR	IoExp_Key_ON	IoExp_Key_OFF
IoExp Sleep	IoExp_ErrId	IoExp_ReadInfo	IoExp_ReadSN	-
IoExp_SetInputPullHigh		IoExp_SetInputPullLow		
IoExp_SetInputFloating		IoExp_SetOutputHigh		
IoExp_SetOutputLow		-		

5.42.1 IoExp Input State

[NY5+ / NX1]

This command is for changing current key corresponding pattern on each I/O Expander. User can set different input states for each I/O Expander in [Input State Exp0] / [Input State Exp1] / [Input State Exp2] / [Input State Exp3] to control requirements of different key triggered.

Ex.

[Input State Exp0]

IoExp0_KEY1: TR1 TR2 X/TR4

[Path]

PowerOn: IoExp0_KEY1 ; Set input state on I/O Expander 0 to IoExp0_KEY1.

5.42.2 IoExp Output State

[NY5+ / NX1]

This command is for changing output setting of corresponding pin on I/O Expander. User can set different output states for each I/O Expander in [Output State Exp0] / [Output State Exp1] / [Output State Exp2] / [Output State Exp3] to control output of pin.

Ex.

[Output State Exp0]

IoExp0_Output_0: 0 1 0 1

[Path]

PowerOn: IoExp0_Output_0 ; Set output state on I/O Expander 0 to IoExp0_Output_0.

5.42.3 IoExp_Key_CLR

[NY5+ / NX1]

This command is for clearing current key state of specified I/O Expander, all I/O Expander if ExpID is omitted. Once it is cleared, the key will be rescanned.

IoExp_Key_CLR

IoExp_Key_CLR(ExpId)

ExpId: I/O Expander ID.

5.42.4 IoExp_Key_ON

[NY5+ / NX1]

This command is for activating scan key function on specified I/O Expander, all I/O Expander if ExpID is omitted. When user activates Key_ON command, the system will continually scan key.

IoExp_Key_ON

IoExp_Key_ON(ExpId)

ExpId: I/O Expander ID.

*Note: The system default value is **IoExp_Key_ON**. After PowerOn, no need to issue another **IoExp_Key_ON** instruction*

5.42.5 IoExp_Key_OFF

[NY5+ / NX1]

This command is for turning off scan key function on specified I/O Expander, all I/O Expander if ExpID is omitted. When user turns off scan key function, the key will not be scanned.

IoExp_Key_OFF

IoExp_Key_Off(ExpId)

ExpId: I/O Expander ID.

5.42.6 IoExp_Sleep

[NY5+ / NX1]

Make all I/O Expander enter sleep.

Note: It requires at least 1ms to operate on I/O Expander after IC is woken up.

5.42.7 IoExp_ErrId

[NY5+ / NX1]

When communication with I/O Expander failed, use this command to get failed I/O Expander ID.

Var = IoExp_ErrId

Var: I/O Expander ID which communication failed.

5.42.8 IoExp_ReadInfo

[NY5+ / NX1]

Read information of I/O Expander.

IoExp_ReadInfo(ExpId, Body, Func, FwVer)

ExpId: I/O Expander ID.

Body: Body code of I/O Expander.

Func: Function code of I/O Expander.

FwVer: Firmware version of I/O Expander.

5.42.9 IoExp_ReadSN

[NY5+ / NX1]

SN is increased while I/O Expander received command. Use this command to get SN.

IoExp_ReadSN(ExpId, SN)

ExpID: I/O Expander ID.

SN: SN of I/O Expander.

5.42.10 IoExp_SetInputPullHigh

[NY5+ / NX1]

Change I/O Expander pin to Pull-High input mode.

IoExp_SetInputPullHigh(Port, Value)

Port: Port on I/O Expander, ex. Exp0_P0 is P0 port on I/O Expander 0.

Value: Accepts number or variable. To set pin as Pull-High input mode, set corresponding bit to 1.

5.42.11 IoExp_SetInputPullLow

[NY5+ / NX1]

Change I/O Expander pin to Pull-Low input mode.

IoExp_SetInputPullLow(Port, Value)

Port: Port on I/O Expander, ex. Exp0_P0 is P0 port on I/O Expander 0.

Value: Accepts number or variable. To set pin as Pull-Low input mode, set corresponding bit to 1.

5.42.12 IoExp_SetInputFloating

[NY5+ / NX1]

Change I/O Expander pin to Floating input mode.

IoExp_SetInputFloating(Port, Value)

Port: Port on I/O Expander, ex. Exp0_P0 is P0 port on I/O Expander 0.

Value: Accepts number or variable. To set pin as Floating input mode, set corresponding bit to 1.

5.42.13 IoExp_SetOutputHigh

[NY5+ / NX1]

Change I/O Expander pin to Output High mode.

IoExp_SetOutputHigh(Port, Value)

Port: Port on I/O Expander, ex. Exp0_P0 is P0 port on I/O Expander 0.

Value: Accepts number or variable. To set pin as Output High mode, set corresponding bit to 1.

5.42.14 IoExp_SetOutputLow

[NY5+ / NX1]

Change I/O Expander pin to Output High mode.

IoExp_SetOutputLow(Port, Value)

Port: Port on I/O Expander, ex. Exp0_P0 is P0 port on I/O Expander 0.

Value: Accepts number or variable. To set pin as Output High mode, set corresponding bit to 1.

5.43 MISC Command

MISC Command				
Input State	Output State	Action Mark State	Wave Mark State	Melody Mark State
NoteOn State	PWM-IO Mark State		QFID State	Key_CLR
Key_ON	Key_OFF	Stop	Pause(n)	Resume(n)
ReadChannel	PauseDown	ResumeUp	Audio_Loop_On	Audio_Loop_Off
Audio_ON	Audio_OFF	AudioMode	NoiseFilter_ON	NoiseFilter_OFF
EQ_Filter	EQ_Filter_Off	AGC_On	AGC_Off	RampUp
RampDown	AutoSleep_On	AutoSleep_Off	Sleep	WDT_CLR
Repeat	Background	Slow	SlowOff	END
ChMode(n)	ReadRollingCode		ReadRadj(Ri)	SW_Reset
Debounce	Direct_Debounce(Time)		Matrix_Debounce(Time)	
Ri = LVD	Enforce_Wakeup	GetMicVol	Millis	Srand

5.43.1 Input State

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command is for changing current key corresponding pattern. User can set different input states in **[Input State]** to control requirements of different key triggered.

Ex.

[Input State]

KEY1: TR1 TR2 X/TR4

[Path]

PowerOn: KEY1

; Set Input State as KEY1.

5.43.2 Output State

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command is for changing output setting of corresponding pin. User can set different output states in [Output State] to control output pin.

Ex.

[Output State]

Output_0: 0 1 0 1

[Path]

PowerOn: Output_0

; Set Output State as Output_0.

5.43.3 Action Mark State

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

User can insert marks in action files while editing file through Q-Visio. When Q-Code accesses the marks, it will execute the corresponding path of current Action Mark state.

There can be as many as 255 such marks from A1 to A255. After Action Marks are defined, users just have to insert corresponding Action Mark path and .vio file into Q-Code to activate Action Mark feature. User can define the sets of corresponding states up to maximum 255.

Ex.

[Action Mark]

	A1	A2	A3	A4
ActionMark1:	AM1	AM2	AM3	AM4

[Path]

PowerOn: ActionMark1, PlayA(PF.3, CH1, \$VIO1.A1)

[Background1]

AM1: PB=0x1	; When system reads ActionMark A1, AM1 is executed.
AM2: PB=0x2	; When system reads ActionMark A2, AM2 is executed.
AM3: PB=0x3	; When system reads ActionMark A3, AM3 is executed.
AM4: PB=0x4	; When system reads ActionMark A4, AM4 is executed.

5.43.4 Wave Mark State

[NY4 / NY5 / NY5+ / NX1]

This feature is for user to add action according to the Wave Mark in *Quick-IO* file while playing Voice file. User can insert Mark anywhere in Voice file to manage the background actions while using *Quick-IO* for editing files. There can be as many as 15 such numbers from [M1] to [M15]. User can switch Wave Mark State to change actions to be taken while playing voice. There can be as many as 255 Wave Mark States.

[Wave Mark]

;	M1	M2	M3	M4
WaveMark1:	WM1	WM2	WM3	WM4

[Path]

PowerOn: WaveMark1, PlayV(\$V0)

[Background1]

WM1: PB=0x1	; When Wave Mark M1 is reached, WM1 is executed.
WM2: PB=0x2	; When Wave Mark M2 is reached, WM2 is executed.
WM3: PB=0x3	; When Wave Mark M3 is reached, WM3 is executed.
WM4: PB=0x0	; When Wave Mark M4 is reached, WM4 is executed.

5.43.5 Melody Mark State

[NY5 / NY5+ / NY6 / NY7 / NX1]

This feature is for user to add the background path action according to the Melody Mark in MIDI file while playing Melody file. User can insert Mark anywhere in the Voice file to manage the background actions while using *Q-MIDI* for editing files. There can be as many as 255 such numbers from [M1] to [M255] with 255 background actions attached to each number. There can be as many as 255 Melody Mark State.

[Melody Mark]

;	M1	M2	M3	M4
MelodyMark1:	MM1	MM2	MM3	MM4

[Path]

PowerOn: MelodyMark1, PlayM(\$M0)

[Background1]

MM1: PB=0x1	; When system reads Melody Mark M1, MM1 is executed.
MM2: PB=0x2	; When system reads Melody Mark M2, MM2 is executed.
MM3: PB=0x3	; When system reads Melody Mark M3, MM3 is executed.
MM4: PB=0x0	; When system reads Melody Mark M4, MM4 is executed.

5.43.6 NoteOn State

[NY5 / NY5+ / NY6 / NY7 / NX1]

If Q-MIDI / Cakewalk is not used for inserting marks for MIDI playing, [Note On] section can be used to directly execute steps for each “channel note on” and jump to the corresponding path. There can be as many as 255 NoteOn State.

- NY5 supports Ch0 ~ Ch3.
- NY5+ supports Ch1 ~ Ch4, Ch10.
- NY6 supports MIDI channel Ch1 ~ Ch6, Ch10.
- NY7 supports MIDI channel Ch1 ~ Ch16.
- NX1 supports MIDI channel Ch1 ~ Ch16.

Ex.

[Note On]

	CH1	CH2	CH3	CH4
CH1: CH1_ON	X		X	X
CH2: X		CH2_ON	X	X

[Path]

TR1: PlayM(\$M0)	; Play Melody.
TR2: SWITCH(R0)=[NO1, NO2], R0=0, TR2	; Choose Note On State.
NO1: CH1, R0=1	; Set Note On State=CH1.
NO2: CH2, R0=0	; Set Note On State=CH2.
CH1_ON: BG1	; Call BG1 when every Ch1 note is on.
CH2_ON: BG2	; Call BG2 when every Ch2 note is on.

[Background1]

BG1:PB=[X X X 1], DELAY(0.08), PB=[X X X 0]

[Background2]

BG2:PB=[X X 1 X], DELAY(0.08), PB=[X X 0 X]

5.43.7 PWM-IO Mark State

[NY5+]

User can insert marks anywhere in the action file to manage the background actions while using Q-Visio for editing files. When Q-Code uses PlayPWM and reaches the marks, it will execute the corresponding path of PWM-IO Mark state automatically. User can define the PWM-IO Mark states in [PWM-IO Mark] section.

There can be as many as 255 such marks from A1 to A255. User can define the setting of corresponding states up to maximum 255.

Ex. Use Q-Visio to insert A1 ~ A4 marks.

[PWM-IO]

SPIO0 = [VIO0.A1]

[PWM-IO Mark]

;	A1	A2	A3	A4
PWMIOMark_0:	PM1	PM2	PM3	PM4

[Path]

PowerOn: PWMIOMark_0, PlayPWM(PA.0, Ch1, \$SPIO0)

[Background1]

PM1: PB=0x1	; System will execute PM1 when PWM-IO mark 1 is read.
PM2: PB=0x2	; System will execute PM2 when PWM-IO mark 2 is read.
PM3: PB=0x3	; System will execute PM3 when PWM-IO mark 3 is read.
PM4: PB=0x0	; System will execute PM4 when PWM-IO mark 4 is read.

5.43.8 QFID State

[NY5+ / NY7 / NX1]

This command is for changing the reaction status of the current QFID. Users can set different QFID State in [QFID] to control the demand of different QFID Tag detection.

Ex.

[QFID]

QFIDGroup0_0: TR1 TR2 X/TR4

[Path]

PowerOn: QFIDGroup0_0 ; Set QFID State to QFIDGroup0_0.

5.43.9 Key_CLR

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command is for clearing current key state. Once it is cleared, the key will be rescanned.

Ex.

[Input_State]

KEY1: TR1 TR2

[Path]

PowerOn: KEY1 ; Set input state KEY1.

TR1: PlayV(Ch0, \$V0), KEY_CLR

; Execute PLAYV(ch0,\$V0) after clicking TR1, then execute KEY_CLR command to clear current key state and rescan key state. After the playback of V0 is finished, the program will continue

the execution of PLAYV(ch0, \$V0) if TR1 is being held. If TR1 is not held, the program will enter sleep mode.

5.43.10 Key_ON

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command is for activating scan key function. When user activates Key_ON command, the system will continually scan key.

Note: The system default is Key_ON. It is not necessary that user activate additional Key_ON command after PowerOn.

5.43.11 Key_OFF

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command is for turning off scan key function. When user turns off scan key function, the key will not be scanned.

Ex.

[Input_State]

KEY1: TR1

[Path]

PowerOn: KEY_OFF, DELAY(0.3), KEY1, KEY_ON

; Scan key will be turned off as PowerOn and activated until 0.3 seconds after PowerOn.

5.43.12 Stop

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command is to stop all currently executing steps in foreground and backgrounds including PlayV, PlayM, Delay, PlayA, PlayPWM, PWMOOut and all background actions.

Note: For NY5, executing Stop command also stops interrupt.

Ex.

[Path]

PowerOn: PlayV(ch0,\$V0) & [BG1, BG2]

TR1: Stop ; Stop all playbacks and delays.

[Background1]

BG1:

[Background2]

BG2:

5.43.13 Pause(n)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command is to pause the designated steps in foreground, background1 and background2 including PlayV, PlayA, PlayM, Delay, PlayPWM / PlayPWMS and PWMOOut / PWMOOutS.

n: 0 = foreground, 1 = background1, 2 = background2 and 3=background3. If n is not designated, the system will pause all execution.

Ex.

[Path]

TR1: Pause(1) ; Pause all steps in background1.

[Background1]

BG1: PlayV(Ch0, \$V0), Delay(5) ; BG1 PlayV(Ch0, \$V0) then delay for 5s.

5.43.14 Resume(n)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

This command is to resume the designated steps in foreground, background1 and background2 including PlayV, PlayA, PlayM, Delay, PlayPWM / PlayPWMS and PWMOOut / PWMOOutS.

n: 0 = foreground, 1 = background1, 2 = background2 and 3=background3. If n is not designated, the system will resume all execution.

Ex.

[Path]

TR1: Pause(1) ; Pause all steps in background1.

TR2: Resume(1) ; Resume all steps in background1.

[Background1]

BG1: PlayV(ch0, \$V0), Delay(5) ; BG1 PlayV(Ch0, \$V0) then delay for 5s.

5.43.15 ReadChannel(Rj:Ri) / ReadChannel(Xi)

[NY5 / NY5+ / NY6 / NY7]

Keep current status of channel in RAM.

Ex.

[Path]

TR1: PlayV(CH2, \$V0) ; Play Voice.

TR2: ReadChannel(X0), [PC,PB]=X0 ; Assign current status of channel to PC and PB.

5.43.16 PauseDown

[NY5 / NY5+]

After PauseDown command is finished then system will immediately Ramp Down. User only can use ResumeUp command to resume playing after PauseDown execution (It is only valid on Channel2).

Note:

1. *PauseDown command is only valid on Channel2, and system can't pause playbacks while data merely is played on Channel0/Channel1. If PauseDown command is executed, data that is playing on Channel0/Channel1 will be paused too.*
2. *PauseDown command will be executed for entering sleep mode.*

Ex.

[Path]

TR1: PlayV(ch2, \$V0)

; Play Voice.

TR2: PauseDown

; Pause the playbacks of data, and IC will enter sleep mode.

TR3: ResumeUp

; Resume the playbacks of data.

5.43.17 ResumeUp

[NY5 / NY5+]

This command can only be used to resume the playbacks by PauseDown command.

Ex.

[Path]

TR1: PlayV(ch2, \$V0)

; Play Voice.

TR2: PauseDown

; Pause the playbacks of data, and IC will enter sleep mode.

TR3: ResumeUp

; Resume the playbacks of data.

5.43.18 Audio_Loop_On

[NX1]

Activate looping of audio playback.

Audio_Loop_On(Ch)

Ch: Specify which audio channel to activate looping of playback.

- NX1 supports Ch0 ~ Ch7.

5.43.19 Audio_Loop_Off

[NX1]

Deactivate looping of audio playback.

Audio_Loop_Off(Ch)

Ch: Specify which audio channel to deactivate looping of playback.

- NX1 supports Ch0 ~ Ch7.

5.43.20 Audio_ON

[NY5 / NY5+ / NY6 / NY7]

This command is for activating Audio output function.

Note:

1. *The default output setting is Audio_ON that is activated as PowerOn.*
2. *If user uses DAC / Push-Pull output then turns on Audio output, it might give out “pop” sound.*

5.43.21 Audio_OFF

[NY5 / NY5+ / NY6 / NY7]

This command is for closing Audio output function.

Note:

1. *For voice output again, please use Audio_ON.*
2. *If user uses DAC / Push-Pull output then turns off Audio output, it might give out “pop” sound.*
3. *Using PWM / DAC output, calling Audio_OFF will makes PWM / DAC output pin to be in floating status.*

Ex.

[Path]

TR1: PlayV(Ch0,\$V0)	; Play Voice.
TR2: Audio_ON	; Activate audio output.
TR3: Audio_OFF	; Deactivate audio output.

5.43.22 AudioMode=n

[NY5+ / NY6 / NY7]

Select the audio output mode.

n: audio output mode. DAC / PWM / PP.

- NY5+ supports DAC / PWM.

- NY6A only supports PWM.
- NY6B / NY6C provide DAC / PWM.
- NY7A series provide DAC / PWM.
- NY7B / NY7C series provide DAC / Push-Pull.

Ex. NY7A

[Path]

TR1: AudioMode=DAC ; Set as DAC output.
 TR2: AudioMode=PWM ; Set as PWM output.
 TR3: PlayV(Ch0, \$V0) ; Play voice files.

Ex. NY7B/NY7C

[Path]

TR1: AudioMode=DAC ; Set as DAC output.
 TR2: AudioMode=PP ; Set as Push-Pull output.
 TR3: PlayV(Ch0, \$V0) ; Play voice files.

5.43.23 NoiseFilter_ON(Ch)

[NY6 / NY7 / NX1]

Enable the noise filter function.

NoiseFilter_On

NoiseFilter_On(Ch)

Ch: The audio channel. If user doesn't specified the channel, Q-Code will enable the noise filter function on all audio channels.

- NY6 supports 0 ~ 5 or Ch0 ~ Ch5.
- NY7 / NX1 do not support the specified channel.

Note:

1. If opens the noise filter function, it stays open until closedown.
2. The NoiseFilter_OFF function can't work when user plays any voice.
3. The default is set as On.

5.43.24 NoiseFilter_OFF

[NY6 / NY7 / NX1]

Disable the noise filter function.

NoiseFilter_Off

NoiseFilter_Off(Ch)

Ch: The audio channel. If user doesn't specified the channel, Q-Code will close the noise filter function on all audio channels.

- NY6 supports 0 ~ 5 or Ch0 ~ Ch5.
- NY7 / NX1 do not support the specified channel.

Note:

1. *If opens the noise filter function, it stays open until closedown.*
2. *The NoiseFilter_OFF function can't work when use plays any voice.*

Ex. When the sounds issue noises, user can open the Noise Filter function to reduce noises.

[Path]

```
TR1: PlayV(Ch0,$V0)           ; Play Voice.
TR2: NoiseFilter_ON           ; Open Noise Filter.
TR3: NoiseFilter_OFF          ; Close Noise Filter.
```

5.43.25 EQ_Filter

[NX1]

Enable audio filter and specify filter parameters.

EQ_Filter(Ch, Index)

Ch: audio channel.

- NX1 accepts Ch0 ~ Ch7 °

Index: Filter index defined in Audio Filter file.

- 0 ~ Max value of Filter index.

Note:

1. *Filter parameters are set by Audio Filter option.*
2. *EQ Filter is only available for SBC-1 / ADPCM.*

Ex. Use EQ_Filter to apply audio filter for voice playing.

[Option]

```
Custom_Filter = Filter.fnx      ; Add Audio Filter.
```

[Path]

```
TR1: PlayV(Ch0,$V0)           ; Play Voice.
TR2: EQ_Filter(Ch0, 1)         ; Apply filter parameter of index=1.
```

5.43.26 EQ_Filter_Off

[NX1]

Disable audio filter.

EQ_Filter_Off(Ch)

Ch: audio channel

- NX1 accepts Ch0 ~ Ch7。

Note: *The EQ_Filter_Off command is available only when EQ_Filter command is used.*

5.43.27 AGC_On

[NX1]

Enable the AGC function. When using VR or the Record command in Q-Code, AGC (Auto Gain Control) can avoid excessive volume and signal saturation. On the other hand, it also can improve the recognition rate of VR and reduce the Record noises.

Note: *If the project contains VR, and the IC body is not NX12P44, the default is Enable.*

5.43.28 AGC_Off

[NX1]

Disable the AGC function. When using VR or the Record command in Q-Code, AGC (Auto Gain Control) can avoid excessive volume and signal saturation. On the other hand, it also can improve the recognition rate of VR and reduce the Record noises.

Note: *If the project contains VR, and the IC body is not NX12P44, the default is Enable.*

Ex. If the recording sounds noisy at high volume, user can reduce the noise by turning on the AGC.

[Path]

TR1: Record(\$Rec0,5s)	; Start to record
TR2: PlayV(CH0,\$Rec0)	; Play the recording
TR3: AGC_On	; Turn on AGC
TR4: AGC_Off	; Turn off AGC

5.43.29 RampUp

[NY5 / NY5+ / NY6 / NY7 / NX1]

The system will conduct RampUp for user automatically when Q-Code is about to play voice file or melody file. If user has special requirements of conducting RampUp command manually, Q-Code language provides RampUp command.

Note:

1. *If RampUp command is executed, system will not execute RampUp procedure before IC enter sleep mode or execute RampDn command.*
2. *In NY5 / NY5+ / NY6 / NY7, no other step can be conducted during the executing. The next step can only be executed after the execution is finished.*
3. *In NX1, next command will be executed right after RampUp is executed. Execution time of RampUp is determined by Ramp Up Time option.*

Ex. RampUp, Delay(1), PlayV.....

; After RampUp command is executed, PlayV/PlayM command will not execute RampUp procedure.

Ex. When OP Amp with control pin is applied.

[Path]

TR1: RampUp, PB.3=0, PlayV.....

; After RampUp command is executed, send control signal to enable OP Amp circuit. It can prevent “pop” sound noise while playing.

5.43.30 RampDown

[NY5 / NY5+ / NY6 / NY7 / NX1]

The Q-Code system will execute RampDown command for user after the playback of voice or melody files is finished. Users still can execute this command to achieve function if needs.

Note:

- 1. If there is certain delay before entering sleep mode, there could be issues of electricity consumption or noise. Q-Code language provides RampDown command for user to stop Audio Output to solve the issues of electricity consumption and “pop” sound noise.*
- 2. In NY5 / NY5+ / NY6 / NY7, no other step can be conducted during the executing. The next step can only be executed after the executing is finished.*
- 3. In NX1, next command will be executed right after RampDown is executed. Execution time of RampDown is determined by Ramp Down Time option.*

Ex. PlayV..... , RampDown, Delay(15)

; If playing is finished and it doesn't go to sleep immediately, RampDn command must be executed first to turn off audio output to solve issues of electricity consumption and noise.

5.43.31 AutoSleep_On

[NX1]

Q-Code automatically enters sleep mode while idling.

Note: The default setting is AutoSleep_ON.

5.43.32 AutoSleep_Off

[NX1]

Q-Code do not enter sleep mode while idling.

Note: When AutoSleep is off, IC wil not enter sleep until AutoSleep_On is executed.

5.43.33 Sleep

[NX1]

Change IC sleep mode while system has nothing to do.

Sleep(Mode)

Mode: IC sleep mode.

- NX1 accepts Halt / Standby ◦

5.43.34 WDT_CLR

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

WDT_CLR (Watch Dog Clear) command is for clearing Watch Dog Counter. If there are too many commands executed in Q-Code without Play or Delay commands in-between, the WDT counter might overflow and cause IC being reset. User can manually use WDT_CLR command to forcefully reset WDT counter.

Ex. WDT_CLR ; Watch Dog Clear.

5.43.35 Repeat

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The repeat function is for user to achieve the purpose of code size saving.

- In NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T, there are 3 repeat function provided, which can be used independently in foreground and background1 / background2.
- In NX1, there are 4 repeat function provided, which can be used independently in foreground and background1 / background2 / background3.

{ ... } * n

n: Repeat time.

- NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T support 1 ~ 15.
- NX1 supports 1 ~ 255.

Ex. Loop counter is used to achieve repeat function during the repeat of single section.

{ PlayV(\$V0), PlayV(\$V1), PlayV(\$v2) } * 3 ; Porgram inside the "{....}" will be executed for 3 times.

5.43.36 Background

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

Execute background paths while executing command.

... & [BG1]

... & [BG1, BG2]

... & [BG1, BG2, BG3]

... & [BG1, BG2, BG3, BG4]

... & [BG1, BG2, BG3, BG4, BG5]

BG1: Background1 path.

- Background path defined in [Background1].
- X does not change the current status of background1.
- OFF: stop background1 path.

BG2: Background2 path.

- Background path defined in [Background2].
- X does not change the current status of background2.
- OFF: stop background2 path.

BG3: Background3 path.

- Background path defined in [Background3].
- X does not change current status of background3.
- OFF: stop background3 path.

BG4: Background4 path.

- Background path defined in [Background4].
- X does not change current status of background4.
- OFF: stop background4 path.

BG5: Background5 path.

- Background path defined in [Background5].
- X does not change current status of background5.
- OFF: stop background5 path.

Ex.

PlayV(\$V0) & [BG1, BG2] ; Start up BG1 / BG2 while V0 starts playing.

5.43.37 Slow

[NY5+ / NY6 / NY7]

Switch the Sleep mode to Slow mode. The Slow mode can be applied to count for a long time or check signals in the fixed period of time. After executing this command, the Slow mode will replace the original Sleep mode. The system will wakeup automatically in the fixed time; after wakeup, Q-Code will execute the WakeUp path once, and execute Sleep path once before entering Slow mode. User can close the Slow mode by using the SlowOff command for resuming the Sleep mode. The wakeup time of Slow mode is 16 times of interruption time which means when user sets interruption time as 256us, the system will wakeup at 4ms; set as 1ms and the system will wakeup at 16ms.

Ex. Switch to Slow.

[Path]

TR1: Slow ; Switch to Slow.

5.43.38 SlowOff

[NY5+ / NY6 / NY7]

This command is for closing Slow mode and resume the Sleep mode.

Ex. Close the Slow mode.

[Path]

TR1: SlowOff ; Turn off the Slow mode.

5.43.39 END

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T / NX1]

The END command here is not for forcefully stopping all the steps currently being executed. The system will enter sleep mode only after all executions of paths are completed. If user needs to stop running steps and enter sleep mode, Please use Stop command.

Note: The “END” command will be appended to each path automatically when compiling. Therefore, adding “END” command to the end of path is not necessary.

Ex. PlayV..... , RampDown, Delay(15), END

5.43.40 ReadRollingCode

[NY4 / NY5 / NY5+ / NY6 / NY7 / NX1]

ReadRollingCode(Rm:Rl:Rk:Rj:Ri)

Rolling Code is a 20 bit serial number that is written to OTP while programming. The number is automatically incremented after each write, thus each OTP has a unique number. The command allows OTP serial number to be read and saved to registers. The saved content is: Ri=bit0~3, Rj=bit4~7, Rk=bit8~11, Rl=bit12~15, Rm=bit16~19.

Ex. **ReadRollingCode(R4:R3:R2:R1:R0)** ; Save the read rolling code in R0 ~ R4.

5.43.41 ChMode(n)

[NY6 / NY7]

Set the number of channels.

n: number of channels

- NY6 supports 2 / 4 / 6 (default is 6).
- NY7 supports 2 / 4 / 6 / 8 (default is 8).

Note:

1. When the voice or melody is playing, switch the number of channels will cause the incorrect sample rate which results in abnormal playback. Please avoid switching channels while playing voice or melody.
2. In NY6, when ChMode is set to 6, only support voice sample rate up to 41.6KHz. Other ChMode can use up to 44.1KHz.

Ex.

ChMode(4), PlayV(ch0, \$V1) ; Set 4 channels to play V1.

Quality	Available Channel Serial Number
High Quality	0 ~ 1
Good	0 ~ 3
Medium	0 ~ 5
Normal	0 ~ 7

5.43.42 ReadRadj(Ri)

[NY9T]

When using ReadRadj to get resistor value, it will return a value of 0x0~0x7. If no resistor is connected, it will return 0xF. This command has no effect to sensitivity of Touch-Keys. Users may define their own level of sensitivity based on the return value.

Recommended Resistor (+/- 1%)	Level	Ri
750K	0	0x0
360K	1	0x1
180K	2	0x2
120K	3	0x3
82K	4	0x4
51K	5	0x5
33K	6	0x6
16K	7	0x7
Other	-	0xF

Note: If not using external resistor to adjust sensitivity, user needs to connect Radj pad to VDD.

Ex.

[Path]

PowerOn: ReadRadj(R0), Switch (R0)=[Sens0, Sens1, Sens2, Sens3]

	; Read the value of external resistor.
Sens0: TouchKey_Sensitivity(0)	; Set the highest sensitivity.
Sens1: TouchKey_Sensitivity(2)	;Set sensitivity between the 2 nd highest and intermediate.
Sens2: TouchKey_Sensitivity(4)	; Set the intermediate sensitivity.
Sens3: TouchKey_Sensitivity(7)	; Set the lowest sensitivity.

5.43.43 SW_Reset

[NY9T]

SW_Reset (Software Reset) is able to restore the system to power-up initial status. It is different to hardware reset that IC does not reload the hardware initial option, and only reset RAM and IO. After SW_Reset is executed, the program will return to the starting point of entry, the instructions followed by SW_Reset will not be executed.

Note: If user wants to reset the entire system after the execution of Enforce_Calibrate, may write a SW_Reset command or enable SW_Reset Option.

Ex. Enforce_Calibrate: SW_Reset ; Software Reset.

5.43.44 Debounce(Time)

[NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T]

The Debounce(Time) command provides a method to change the scanning Debounce time of the Direct and the Matrix buttons at the same time. The time unit is millisecond or second (default is second), minimum=0ms, maximum=1s.

Note:

- 1. The Debounce(Time) command will affect both the Direct and the Matrix buttons.*
- 2. For NY9T, Debounce Option or of Debounce(Time) are exclusive, and time must be multiples of 4ms.*

Ex.

[Path]

PowerOn: KEY1, KEY_ON

TR1: Debounce(16ms)	; Set the Debounce times of Direct and Matrix buttons as 16ms.
TR2: Debounce(0.06)	; Set the Debounce times of Direct and Matrix buttons as 60ms.

5.43.45 Direct_Debounce(Time)

[NY5+ / NY6 / NY7]

The Direct_Debounce(Time) command provides a method to change the scanning Debounce time of the Direct button. The time unit is millisecond or second (default is second), minimum=0ms, maximum=1s.

Note: The Direct_Debounce(Time) command is valid only for the Direct button, without affecting the Matrix button.

Ex.

[Path]

PowerOn: KEY1, KEY_ON

TR1: Direct_Debounce(16ms) ; Set the Debounce time of Direct button as 16ms.

TR2: Direct_Debounce(0.06) ; Set the Debounce time of Direct button as 60ms.

5.43.46 Matrix_Debounce(Time)

[NY5+ / NY6 / NY7]

The Matrix_Debounce command provides a method to change the scanning Debounce time of the Matrix button. The time unit is millisecond or second (default is second), minimum=0ms, maximum=1s.

Note: The Matrix_Debounce(Time) command is valid only for the Matrix button, without affecting the Direct button.

Ex.

[Path]

PowerOn: KEY1, KEY_ON

TR1: Matrix_Debounce(16ms) ; Set the Debounce time of Matrix button as 16ms.

TR2: Matrix_Debounce(0.06) ; Set the Debounce time of Matrix button as 60ms.

5.43.47 Ri = LVD

[NY5+ / NY6B / NY6C / NX1]

This command reads the current level of LVD. LVD level is save to the specified RAM Ri.

The mapping of LVD level is as following:

Level	NY4PxxxC / NY5+	NY6P025A LVD1	NY6P025A LVD2	NY6B / NY6C	NX1 OTP	NX1 EF
1	<2.0V	<2.0V	<2.4V	<2.4V	<2.2V	<2.0V
2	2.0V ~ 2.2V	2.0V ~ 2.2V	2.4V ~ 2.7V	2.4V ~ 2.7V	2.2V ~ 2.4V	2.0V ~ 2.2V
3	2.2V ~ 2.4V	2.2V ~ 3.0V	2.7V ~ 3.6V	2.7V ~ 3.6V	2.4V ~ 2.6V	2.2V ~ 2.4V
4	2.4V ~ 2.8V	3.0V ~ 3.2V	3.6V ~ 4.1V	3.6V ~ 4.1V	2.6V ~ 3.2V	2.4V ~ 2.8V
5	2.8V ~ 3.0V	>3.2V	>4.1V	>4.1V	3.2V ~ 3.4V	2.8V ~ 3.0V
6	3.0V ~ 3.3V				3.4V ~ 3.6V	3.0V ~ 3.2V

7	3.3V ~ 3.6V				> 3.6V	3.2V ~ 3.4V
8	>3.6V					3.4V ~ 3.6V
9						> 3.6V

Ex. Save the level parameter of LVD to R0.

TR1: R0 = LVD

; Save the LVD level to R0.

5.43.48 Enforce_Wakeup

[NX1]

Enforce_Wakeup allows user to wake system up from sleep in RTC_2Hz path.

Note: *Enforce_Wakeup is only available in RTC_2Hz path and under Standby Mode.*

Ex.

[Variable]

Var8: event_trig=0, count_500ms=0

[Path]

PowerOn:

main: event_trig=1?TR1

; play voice everytime system is waked up.

TR1: event_trig=0, PlayVS(CH0,\$V0)

[Interrupt]

RTC_2Hz: count_500ms++, count_500ms>=10?Per5Sec

Per5Sec: count_500ms=0, event_trig=1, Enforce_Wakeup ; wakeup every 5 seconds.

5.43.49 GetMicVol

[NX1]

GetMicVol command gets microphone volume by directly reading ADC value.

GetMicVol(Var)

Var: Result will be stored in Var, range 0~65535.

Note:

1. Reasonable volume value is fetched only when Mic is used. Var will be 0 when mic is not used.
2. There may be 5% background noise due to environment and microphone sensitivity. Result is estimated volume.

Ex.

[Variable]

Var16: var_mic=0

[Path]

```
PowerOn: RT_Play ; Use mic by RT command.
8ms: GetMicVol(var_mic), ; Get mic volume every 8ms.
var_mic>6000?{PA.1=0}:{PA.1=1} ; PA.1=0 when volume is greater than
6000, or PA.1=1.
```

5.43.50 Millis

[NX1]

Millis command reads duration in milliseconds after system initialized by reading NX1 system counter. Read counter is Var32.

Millis(Var)

Var: Counter value read will be stored at Var.

Note: Counting will be paused while sleeping.

Ex.

[Variable]

Var32: millis_var

[Path]

PowerOn: Millis(millis_var) ; Get the duration after system initialized.

5.43.51 Srand

[NX1]

Srand is used to set random number seed.

Srand(Seed)

Seed: Random number seed.

5.44 Debug Command

Debug Command				
Printf	-	-	-	-

5.44.1 Printf

[NX1]

Use UART_TX to send debug message, like printf in C language.

Printf(Format_str, Args)

Format_str: Message string.

Args: Variables, allows any number of variable, even without any.

Note:

1. *UART TX must be enabled to use this command.*
2. *Variable used in Format_str accepts: %o, %d, %u, %x and %X.*
3. *Q-Code will not check consistency between Format_str and Args, user must check by themselves.*

Ex.

[Variable]

Var32: cnt = 0

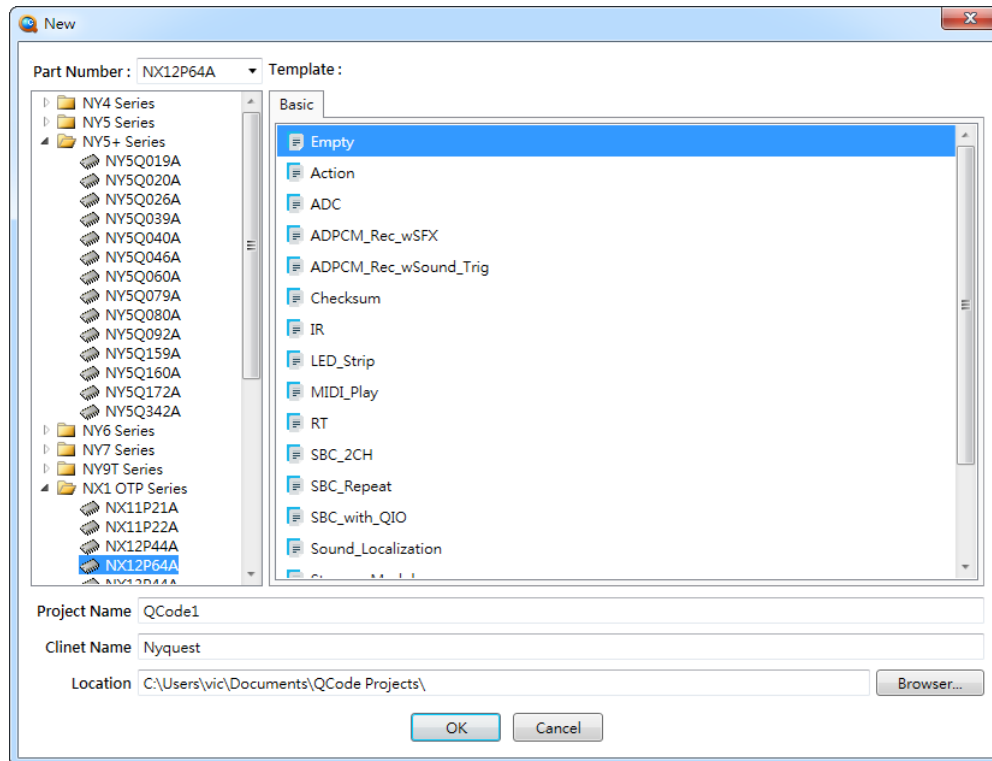
[Path]

PowerOn: Printf("Hello\r\n")	; Output Hello
8ms: cnt++	; Increase cnt every 8ms.
500ms: Printf("cnt%d\r\n", cnt)	; Output cnt value every 500ms.

6 Appendix

6.1 New File Window

[File] → [New]



Part Number : The IC body is selected here.

Template : Use the built-in file template to edit the content and structure of file. If you click on the series item, the series IC information will be displayed on the right as shown below.

Part Number : NY5Q046A	Template :										
		P/N	OTP	RAM	SPI Flash	I/O	SPI	16-bit Timer	PWM-IO	12-bit ADC	MIC
		NX11P21A	32Kb	4Kb	-	18	0 / -	2	-	-	-
		NX11P22A	32Kb	4Kb	-	24	0 / -	2	4	-	-
		NX12P44A	64Kb	8Kb	-	32	0 / 1	3	4	8-ch	v
		NX12P64A	64Kb	12Kb	-	32	0 / 1	3	8	8-ch	v
		NX13P44A	64Kb	8Kb	-	32	0 / 1	3	4	8-ch	v
		NX13P64A	64Kb	12Kb	-	32	0 / 1	3	8	8-ch	v
		NX11S21A	32Kb	4Kb	2Mb	12	0 / -	2	-	-	-
		NX11S22A	32Kb	4Kb	4Mb	12	0 / -	2	-	-	-
		NX11M22A	32Kb	4Kb	4Mb	12	0 / -	2	-	-	-
		NX11M23A	32Kb	4Kb	8Mb	12	0 / -	2	-	-	-
		NX11M24A	32Kb	4Kb	16Mb	12	0 / -	2	-	-	-
		NX11M25A	32Kb	4Kb	32Mb	12	0 / -	2	-	-	-
		NX12M52A	96Kb	10Kb	4Mb	16	0 / -	3	4	4-ch	v
		NX12M53A	96Kb	10Kb	8Mb	16	0 / -	3	4	4-ch	v
		NX12M54A	96Kb	10Kb	16Mb	16	0 / -	3	4	4-ch	v
		NX12M55A	96Kb	10Kb	32Mb	16	0 / -	3	4	4-ch	v
		NX13M52A	96Kb	10Kb	4Mb	16	0 / -	3	4	4-ch	v
		NX13M53A	96Kb	10Kb	8Mb	16	0 / -	3	4	4-ch	v

Project Name : Key in project name.

Client Name : For protect the interests of clients, please key in client name.

Location : Key in the location of file.

6.2 Tools

6.2.1 Related Software Tools

Using the Q-Code programming, there are some peripheral tools to complete the functions. For example, NYASM is used to compile program and generate the .bin file and .htm file. (This is an important file to make the code).

The following tools may be used along with Q-Code:



Q-MIDI: This software is used to edit MIDI and timbre. Generated file's extension is .qmd.



Quick-IO: This software is for drawing the signals of output pin. It must be used together with WAV file and the extension name of the generated file is .nyq.



Q-Visio: This software is for drawing the signals of output pin and the extension name of the generated file is .vio.



Q-Touch: This software is for Touch-Key calibration, and the extension of generated file is .t9x.



Q-Writer: This software is for programming the BIN file into Flash Demo Board, Romter or OTP for verification.

Note: When installing the Q-Code, it is recommended to install related tools simultaneously. About the usage of these tools, please refer to the user manual.

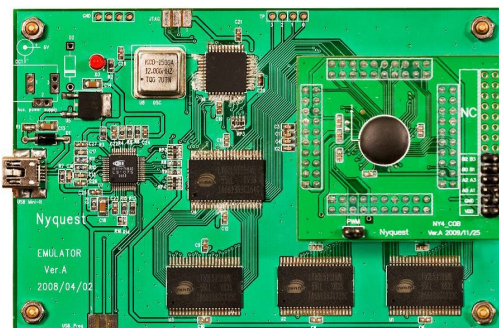
6.2.2 Related Hardware Tools

After finish programming and compiler, user can download the program to ICE to demonstrate. It can also be demonstrated through FDB_Writer or Q-FDB_Writer by writing onto the flash demo board.

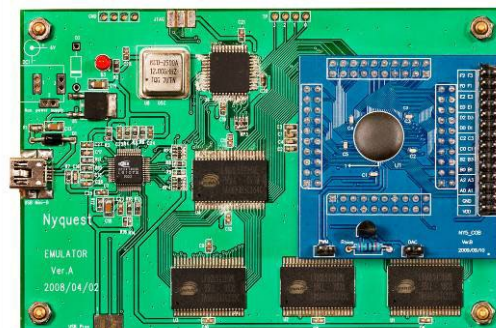
What is ICE? ICE is the abbreviation of In Circuit Emulator. Just connect ICE to their personal computers through USB, then download the finished program to ICE for simulating. (Detail installation of ICE, please refer to NYIDE user manual). NY4_COB / NY5_COB / NY7_COB can be used with ICE.

Note:

1. ICE has V1 / V2 (In following pictures, NY4 / NY5 uses ICE V1, NY7 uses ICE V2).
2. ICE V1 only supports NY4 / NY5.
3. ICE V2 supports NY4 / NY5 / NY5+ / NY6 / NY7 / NY9T.



NY4_COB on ICE V1



NY5_COB on ICE V1



NY7_COB on ICE V2

What is Q-Writer? It is a recording system of graphical interfaces. User can use it to write produced .bin file rapidly on Flash Demo Board or Romter. Also user can burn directly to OTP for verification. (More details about the installation and usage of Q-Writer's software and hardware, please check the Q-Writer user manual.)

FDB_Writer: As the following picture.



Q-FDB_Writer: As the following picture.



Romter: As the following picture.



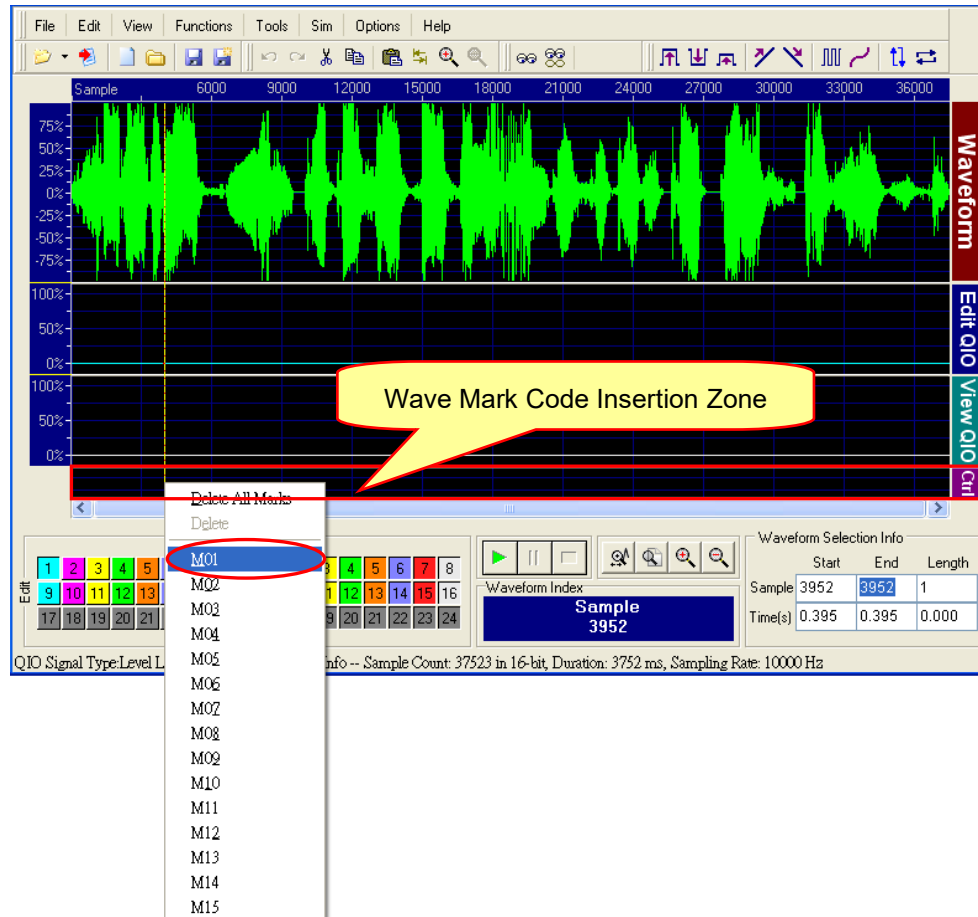
NX_Programmer: As the following picture.



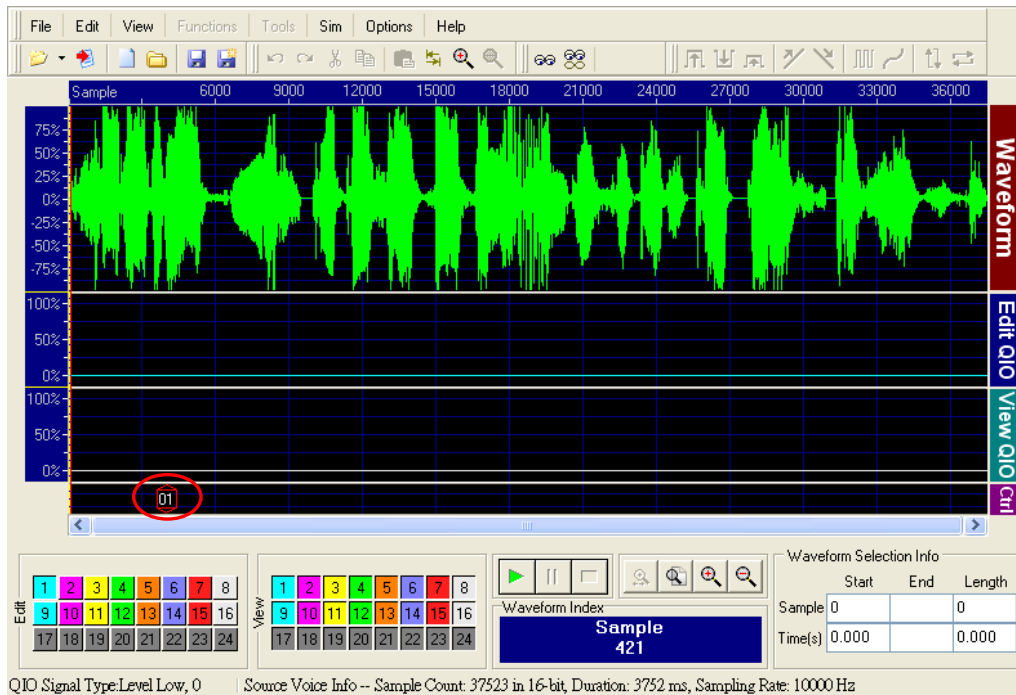
6.3 Insert Control Code to WAV File by Using Quick-IO

Quick-IO provides functions for inserting control code. User can insert control code at any position for Q-Code program to read. When Q-Code program read Wave Mark, the corresponding path in **[Wave Mark]** section will be executed.

After a WAV file is opened, move the mouse cursor to Ctrl rail and click the right button, then Mark Number will appear.



After selecting Mark Number to be inserted, the inserted Mark will appear in control rail, and the code insertion is completed as shown in the following figure:



6.4 Using Q-Sound Functions in Q-Code

To use Q-Sound functions in Q-Code, please operate by following steps:

[Step 1: How to Open Q-Sound Window.](#)

[Step 2: How to Open Split Window.](#)

[Step 3: Set Noise Gate.](#)

[Step 4: Set Minimum Mute Length.](#)

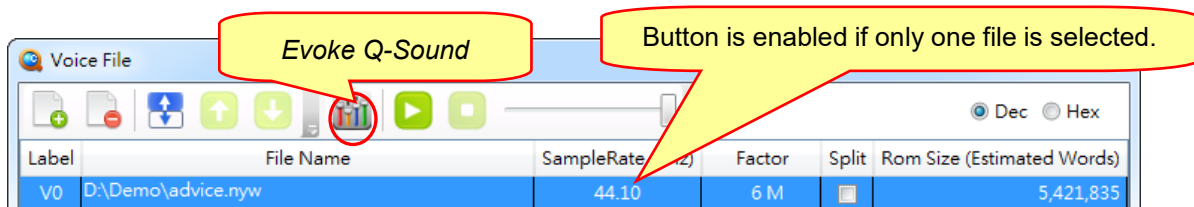
[Step 5: Use Auto Mark.](#)

[Step 6: Adjust Mark.](#)

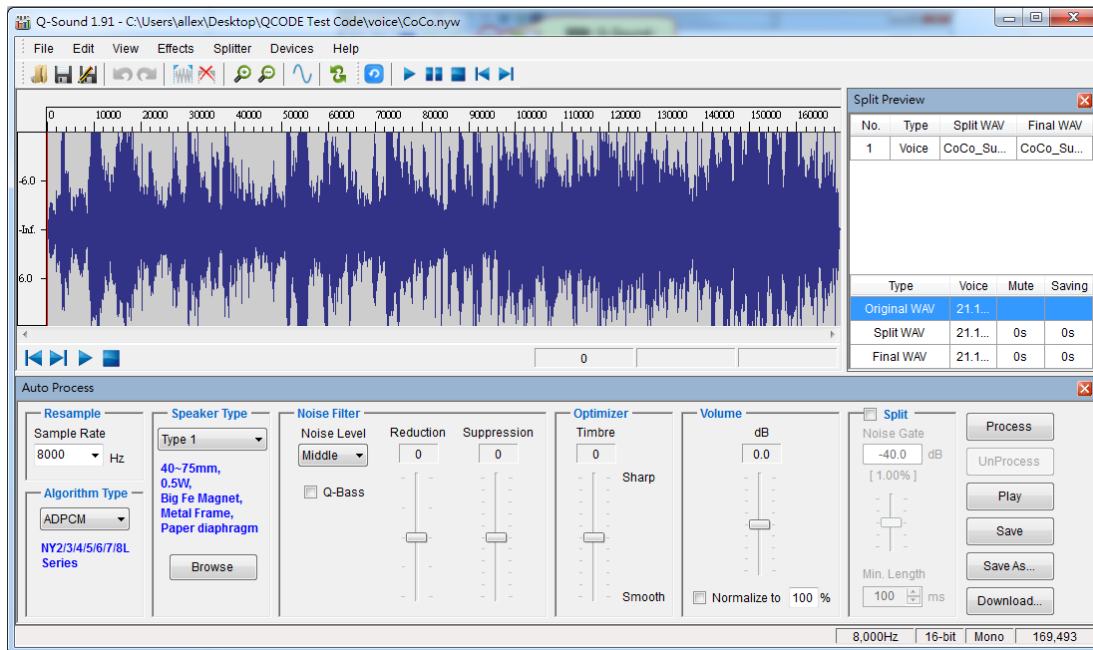
[Step 7: Using .nyw Files in Q-Code.](#)

Step 1: How to Open Q-Sound Window.

Click on Add File icon in the Voice File section to open Voice File dialog.



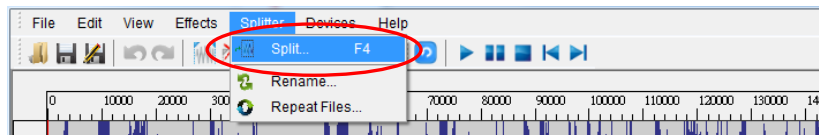
Click on  button to evoke Q-Sound.



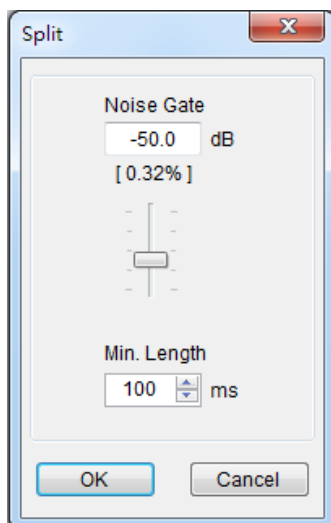
Note: Q-Sound supports .wav, .nyw formats only.

Step 2: How to Open Split Window.

Select the [Splitter] in menu.

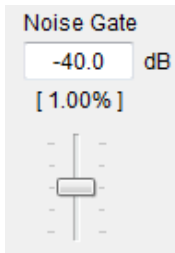


Click Split for opening Split window.



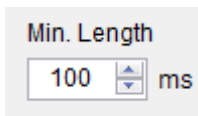
Step 3: Set Noise Gate.

Adjusting level of Noise Gate via threshold can set the range of mute section.



Step 4: Set Minimum Mute Length

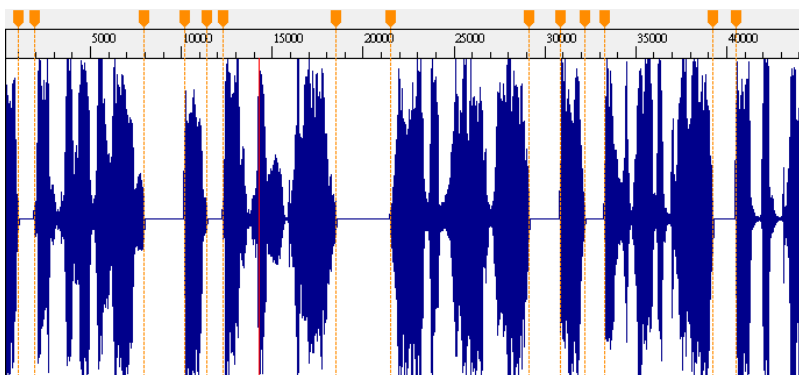
For adjusting minimum mute length, user can press Up / Down button or directly keyed in number. Mute section length shorter than minimum length will be ignored by Q-Sound.



Step 5: Process Auto Mark Function

After function settings, Q-Sound will automatically insert marks according to the setting of Noise Gate and Minimum Mute Length.

After inserting the marks, the below dialog box will display the numbers of mute sections, and preview panel will display splitting process.



Split Preview			
No.	Type	Split WAV	Final WAV
1	Voice	Vocal_bin_dec_Sub1(134ms)	Vocal_bin_dec_Sub1(134ms)
2	Mute	112ms	112ms
3	Voice	Vocal_bin_dec_Sub2(753ms)	Vocal_bin_dec_Sub2(753ms)
4	Mute	282ms	282ms
5	Voice	Vocal_bin_dec_Sub3(147ms)	Vocal_bin_dec_Sub3(147ms)
6	Mute	110ms	110ms
7	Voice	Vocal_bin_dec_Sub4(780ms)	Vocal_bin_dec_Sub4(780ms)

If the splitting result is not as expected, go back to [Step 2](#) for operation.

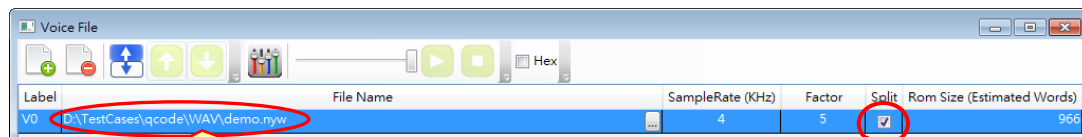
Step 6: Adjust Mark

If user want to manually adjust the mark, please move the cursor to mark,  will appear then press and hold the left button to adjust.

After completion, if the splitting result is not as expected, please repeat [Step 2](#) and [Step 3](#). After confirmation, press Save and close Q-Sound.

Step 7: Using .nyw Files in Q-Code

After closing Q-Sound, back to Voice File Dialog. (If Split option for selected file is not checked, Q-Sound works including splitting and mark will be ignored)



.wav is converted to .nyw format automatically

Check to enable splitting process

Click OK button to close Voice File dialog. And generated code for voice file process will be added / replaced in text editor.

[Voice File]

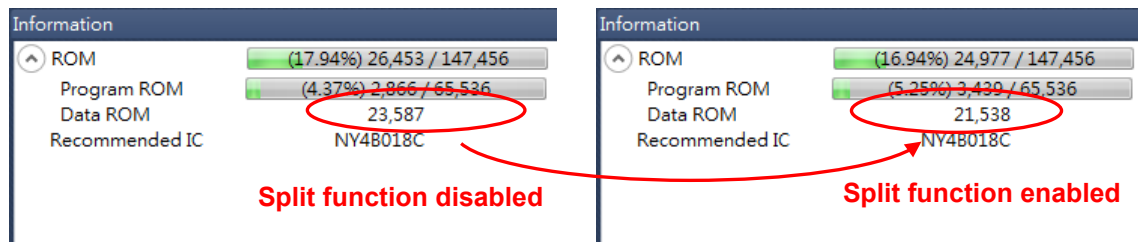
```
V0 = D:\Demo\advice.nyw /5 /fs /s
```

The command to play splitted voice remains the same.

[Path]

```
PowerOn:playV($v0), input_0
p1: playV($v0)
p2: playV($v0)
```

The ROM consumption of a Q-Code program with enabled / disabled Split function is illustrated as below. User can check the ROM Size difference from status bar.



Note: For more detail Q-Sound functions, see Q-Sound User Manual.

6.5 Command Table of Q-Code

6.5.1 Command Table of NY4

6.5.1.1 Arithmetic Command

Arithmetic Logic Command			
Var1 = Var2	Var1 = Var2 + Var3	Var1 = Var2 - Var3	Var1 = Var2 * Var3
Var1 = Var2 / Var3	Var1 = Var2 % Var3	Var++	Var--
Var1 = Var2 & Var3	Var1 = Var2 Var3	Var1 = Var2 ^ Var3	Var1 = Var2 << Var3
Var1 = Var2 >> Var3	!Var	Var=RandomL	Var=RandomH
Var=Random	-	-	-

6.5.1.2 Flow Control Command

Flow Control Command				
Var = Var ? Path	Var != Var ? Path	Var >= Var ? Path	Var <= Var ? Path	Var > Var ? Path
Var < Var ? Path	Px = data?Path	Px != data?Path	Px.n = 0?Path	Px.n != 0?Path
Px.n = 1?Path	Px.n != 1?Path	-	-	-
RandomL=data?Path	RandomH=data?Path	RandomL!=data?Path	RandomH!=data?Path	
Random=data?Path	Random!=data?Path	Px[1 X 0 X]?Path	Voice?Path	
PaueV?Path	Delay(n)?Path	Action(Ch)?Path	PWMIO(n)?Path	PausePWM?Path
CheckSum?Path		!Px[1 X 0 X]?Path	!Voice?Path	!PauseV?Path
!Delay?Path	!Action?Path	!PWMIO?Path	!PausePWM?Path	
!CheckSum?Path		If-Else	-	-
Switch(Ri) = [Path0, Path1, Path2...Path15]		Switch(Px) = [Path0, Path1, Path2...Path15]		
Switch(RandomL) = [Path0, Path1,..Path15]		Switch(RandomH) = [Path0, Path1,..Path15]		
Switch(Px[d x d x]) = [Path0, Path1, Path2....Path15]				
Switch(Xi)=[Path0, Path1, Path2,..Path255]		-		
Switch(Random)=[Path0, Path1,Path2...Path255]				
While	Do-While	For	-	:

6.5.1.3 I/O Command

4-bit I/O Command				
Ri = Px	Ri = PxM	Ri.n = Px.n	PxM = data	PxM = Ri
PxM.n = 0	PxM.n = 1	Px = data	Px = Ri	Px = Py

4-bit I/O Command				
!Px	!Px.n	Px.n = 0	Px.n = 1	Px.n = Ri.n
Px.n = Py.n	Px = Py + Ri	Px = Py - Ri	Px = Py Ri	Px = Py ^ Ri
Px = Py & Ri	Ri = Px + Ri	Ri = Px - Ri	Ri = Px Ri	Ri = Px ^ Ri
Ri = Px & Ri	Px = Py + data	Px = Py - data	Px = Py data	Px = Py ^ data
Px = Py & data	Ri = Px + data	Ri = Px - data	Ri = Px data	Ri = Px ^ data
Ri = Px & data	Px = [1 x 0 FD An]	Px.n = 1KHz(time)	-	-
8-bit I/O Command				
XiL = Px	XiH = Px	XiL.n = Px.n	XiH.n = Px.n	Xi.n = Px.n
Xi = [Px, Py]	Px = XiL	Px = XiH	Px.n = XiL.n	Px.n = XiH.n
Px.n = Xi.n	[Px, Py] = Xi	-	-	-

6.5.1.4 Path Command

Path Command				
ASM	BG	BreakFG	StopFG	StopBG
StopBG1	StopBG2	Subroutine	Label(Pathname)	Macro

6.5.1.5 Voice Command

Voice Command				
PlayV	PlayVS	Voicename	WaitVN(Ch)	PauseV(Ch)
ResumeV(Ch)	StopV(Ch)	FreqCH=nK	-	-

6.5.1.6 Sentence Command

Sentence Command				
PlayS	-	-	-	-

6.5.1.7 Table Command

Table Command	
TableL(TableName, Rx, Ry, Ri)	TableM(TableName, Rx, Ry, Ri)
TableH(TableName, Rx, Ry, Ri)	Table(TableName, Rx, Ry, Rh, Rm, Ri)
TableL(TableName, X, Y, Ri)	TableM(TableName, X, Y, Ri)
TableH(TableName, X, Y, Ri)	Table(TableName, X, Y, Rh, Rm, Ri)
TableL(TableName, Xx, Xy, Xi)	TableH(TableName, Xx, Xy, Xi)

Table Command	
Table(TableName, Xx, Xy, Xh, Xi)	TableL(TableName, X, Y, Xi)
TableH(TableName, X, Y, Xi)	Table(TableName, X, Y, Xh, Xi)

6.5.1.8 IR Command

IR Command				
IR_TX=data	IR_TX(Rl:Rk:Rj:Ri)	IR_TX(Xj:Xi)	IR_RX_ON	IR_RX_OFF
[Rl,Rk,Rj,Ri] = IR_RX	[Xj,Xi] = IR_RX	IR_RX = data?Path	IR_RX != data?Path	
IR_Carrier_On	IR_Carrier_Off	-	-	

6.5.1.9 Serial Control Command

Serial Control Command			
SC_RX_ON	SC_RX_OFF	[Rl,Rk,Rj,Ri] = SC_RX	[Xj,Xi] = SC_RX
SC_RX = data?Path		SC_RX != data?Path	

6.5.1.10 PWMIO Command

PWMIO Command				
PWMOut	PWMOutS	WaitPN	StopPWM	PausePWM
ResumePWM	-	-	-	-

6.5.1.11 Delay Command

Delay Command				
Delay(time)	Delay(Ri:Rj:Rk)	WaitDN(n)	StopD(n)	PausedD(n)
ResumeD(n)	SDelay(time)	-	-	-

6.5.1.12 Action Command

Action Command				
PlayA	PlayAS	WaitAN(Ch)	PauseA(Ch)	ResumeA(Ch)
StopA(Ch)	-	-	-	-

6.5.1.13 MISC Command

MISC Command				
Input State	Output State	Action Mark State	Wave Mark State	Key_CLR

MISC Command				
Key_ON	Key_OFF	Stop	Pause(n)	Resume(n)
WDT_CLR	Repeat	Background	END	Debounce(Time)
ReadRollingCode	Ri = LVD	-	:	-

6.5.2 Command Table of NY5

6.5.2.1 Arithmetic Command

Arithmetic Logic Command			
Var1 = Var2	Var1 = Var2 + Var3	Var1 = Var2 - Var3	Var1 = Var2 * Var3
Var1 = Var2 / Var3	Var1 = Var2 % Var3	Var++	Var--
Var1 = Var2 & Var3	Var1 = Var2 Var3	Var1 = Var2 ^ Var3	Var1 = Var2 << Var3
Var1 = Var2 >> Var3	!Var	Var=RandomL	Var=RandomH
Var=Random	-	-	-

6.5.2.2 Flow Control Command

Flow Control Command				
Var = Var ? Path	Var != Var ? Path	Var >= Var ? Path	Var <= Var ? Path	Var > Var ? Path
Var < Var ? Path	Px = data?Path	Px != data?Path	Px.n = 0?Path	Px.n != 0?Path
Px.n = 1?Path	Px.n != 1?Path	Vol = n?Path	Vol !=n ?Path	-
MixCtrl = data?Path	MixCtrl != data ?Path	RandomL=data?Path	RandomH=data?Path	
RandomL!=data?Path	RandomH!=data?Path	Random=data?Path	Random!=data?Path	
Px[1 X 0 X]?Path		Voice(Ch)?Path	PaueV(Ch)?Path	Melody?Path
PauseM?Path	Delay(n)?Path	Action(Ch)?Path	PWMIO(n)?Path	PausePWM?Path
CheckSum?Path		!Px[1 X 0 X]?Path	!Voice?Path	!PauseV?Path
!Melody?Path	!PauseM?Path	!Delay?Path	!Action?Path	!PWMIO?Path
!PausePWM?Path		!CheckSum?Path	If-Else	-
Switch(Ri) = [Path0, Path1, Path2,..Path15]		Switch(Px) = [Path0, Path1, Path2,..Path15]		
Switch(RandomL) = [Path0, Path1,..Path15]		Switch(RandomH) = [Path0, Path1,..Path15]		
Switch(Px[d x d x]) = [Path0, Path1, Path2....Path15]				
Switch(Xi)= [Path0, Path1, Path2,..Path255]				
Switch(Random)= [Path0, Path1, Path2,..Path255]				
While	Do-While	For	-	:

6.5.2.3 I/O Command

4-bit I/O Command				
Ri = Px	Ri.n = Px.n	Px = data	Px = Ri	Px = Py
!Px	!Px.n	Px.n = 0	Px.n = 1	Px.n = Ri.n
Px.n = Py.n	Px = Py + Ri	Px = Py - Ri	Px = Py Ri	Px = Py ^ Ri
Px = Py & Ri	Ri = Px + Ri	Ri = Px - Ri	Ri = Px Ri	Ri = Px ^ Ri
Ri = Px & Ri	Px = Py + data	Px = Py - data	Px = Py data	Px = Py ^ data
Px = Py & data	Ri = Px + data	Ri = Px - data	Ri = Px data	Ri = Px ^ data
Ri = Px & data	Px = [1 x 0 FD An]	Px.n = 1KHz(time)	-	-
8-bit I/O Command				
XiL = Px	XiH = Px	XiL.n = Px.n	XiH.n = Px.n	Xi.n = Px.n
Xi = [Px, Py]	Px = XiL	Px = XiH	Px.n = XiL.n	Px.n = XiH.n
Px.n = Xi.n	[Px, Py] = Xi	-	-	-

6.5.2.4 Path Command

Path Command				
ASM	BG	BreakFG	StopFG	StopBG
StopBG1	StopBG2	Subroutine	Label(Pathname)	Macro

6.5.2.5 Voice Command

Voice Command				
PlayV	PlayVS	Voicename	WaitVN(Ch)	PauseV(Ch)
ResumeV(Ch)	StopV(Ch)	FreqCH=nK	-	-

6.5.2.6 Sentence Command

Sentence Command				
PlayS	-	-	-	-

6.5.2.7 MIDI Command

MIDI Command				
PlayM	PlayMS	WaitMN	PauseM	ResumeM

MIDI Command				
StopM	Tempo = n	Tempo++	Tempo--	Tempo(Rj:Ri)
Tempo(Xi)	ReadTempo	Mute On(Ch)	Mute Off(Ch)	OKON On
OKON Off	OKON Play	-	-	-

6.5.2.8 Volume Command

Volume Command				
Vol_Max	Vol_Min	Vol = n	Vol = Ri	Vol++
Vol--	Ri = Vol	Px = Vol	Ri = MixCtrl	Px = MixCtrl
MixCtrl	-	-	-	-

6.5.2.9 Table Command

Table Command	
TableL(TableName, Rx, Ry, Ri)	TableM(TableName, Rx, Ry, Ri)
TableH(TableName, Rx, Ry, Ri)	Table(TableName, Rx, Ry, Rh, Rm, Ri)
TableL(TableName, X, Y, Ri)	TableM(TableName, X, Y, Ri)
TableH(TableName, X, Y, Ri)	Table(TableName, X, Y, Rh, Rm, Ri)
TableL(TableName, Xx, Xy, Xi)	TableH(TableName, Xx, Xy, Xi)
Table(TableName, Xx, Xy, Xh, Xi)	TableL(TableName, X, Y, Xi)
TableH(TableName, X, Y, Xi)	Table(TableName, X, Y, Xh, Xi)

6.5.2.10 IR Command

IR Command				
IR_TX=data	IR_TX(Ri:Rk:Rj:Ri)	IR_TX(Xj:Xi)	IR_RX_ON	IR_RX_OFF
[Ri,Rk,Rj,Ri] = IR_RX	[Xj,Xi] = IR_RX	IR_RX = data?Path	IR_RX != data?Path	
IR_Carrier_On	IR_Carrier_Off	-	-	

6.5.2.11 Serial Control Command

Serial Control Command			
SC_RX_ON	SC_RX_OFF	[Ri,Rk,Rj,Ri] = SC_RX	[Xj,Xi] = SC_RX
SC_RX = data?Path		SC_RX != data?Path	

6.5.2.12 PWMIO Command

PWMIO Command

PWMIO Command				
PWMOut	PWMOutS	WaitPN	StopPWM	PausePWM
ResumePWM	-	-	-	-

6.5.2.13 Interrupt Command

Interrupt Command				
INT_ON	INT_OFF	INT_RET	INT = n	-

6.5.2.14 Delay Command

Delay Command				
Delay(time)	Delay(Ri:Rj:Rk)	WaitDN(n)	StopD(n)	PauseD(n)
ResumeD(n)	SDelay(time)	-	-	-

6.5.2.15 Action Command

Action Command				
PlayA	PlayAS	WaitAN(Ch)	PauseA(Ch)	ResumeA(Ch)
StopA(Ch)	-	-	-	-

6.5.2.16 MISC Command

MISC Command				
Input State	Output State	Action Mark State	Wave Mark State	Melody Mark State
NoteOn State	Key_CLR	Key_ON	Key_OFF	Stop
Pause(n)	Resume(n)	ReadChannel	PauseDown	ResumeUp
Audio_ON	Audio_OFF	RampUp	RampDown	WDT_CLR
Repeat	Background	END	ReadRollingCode	Debounce(Time)

6.5.3 Command Table of NY5+

6.5.3.1 Arithmetic Command

Arithmetic Logic Command			
Var1 = Var2	Var1 = Var2 + Var3	Var1 = Var2 - Var3	Var1 = Var2 * Var3
Var1 = Var2 / Var3	Var1 = Var2 % Var3	Var++	Var--
Var1 = Var2 & Var3	Var1 = Var2 Var3	Var1 = Var2 ^ Var3	Var1 = Var2 << Var3
Var1 = Var2 >> Var3	!Var	Var=RandomL	Var=RandomH

Arithmetic Logic Command			
Var=Random	-	-	-

6.5.3.2 Flow Control Jump

Flow Control Command				
Var = Var ? Path	Var != Var ? Path	Var >= Var ? Path	Var <= Var ? Path	Var > Var ? Path
Var < Var ? Path	Px = data?Path	Px != data?Path	Px.n = 0?Path	Px.n != 0?Path
Px.n = 1?Path	Px.n != 1?Path	Vol = n?Path	Vol !=n ?Path	-
RandomL=data?Path	RandomH=data?Path	RandomL!=data?Path	RandomH!=data?Path	
Random=data?Path	Random!=data?Path	Px[1 X 0 X]?Path	ChUsed(Ch)?Path	
Voice(Ch)?Path	PaueV(Ch)?Path	Melody?Path	PauseM?Path	Delay(n)?Path
Action(Ch)?Path	PauseA(Ch)?Path	PWMIO(n)?Path	PausePWM?Path	
IoExp Exists?Path		CheckSum?Path	!Px[1 X 0 X]?Path	
!ChUsed?Path	!Voice?Path	!PauseV?Path	!Melody?Path	!PauseM?Path
!Delay?Path	!Action?Path	!PauseA?Path	!PWMIO?Path	-
!PausePWM?Path		!IoExp Exists?Path		-
!CheckSum?Path		If-Else	-	-
Switch(Ri) = [Path0, Path1, Path2,..Path15]		Switch(Px) = [Path0, Path1, Path2,..Path15]		
Switch(RandomL) = [Path0, Path1,..Path15]		Switch(RandomH) = [Path0, Path1,..Path15]		
Switch(Px[d x d x]) = [Path0, Path1, Path2....Path15]				
Switch(Xi)=[Path0, Path1, Path2,..Path255]		-		
Switch(Random)=[Path0, Path1,Path2,..Path255]				
While	Do-While	For	-	-

6.5.3.3 I/O Command

4-bit I/O Command				
Ri = Px	Ri=PXM	Ri.n = Px.n	PxM = data	PxM = Ri
PxM.n = 0	PxM.n = 1	Px = data	Px = Ri	Px = Py
!Px	!Px.n	Px.n = 0	Px.n = 1	Px.n = Ri.n
Px.n = Py.n	Px = Py + Ri	Px = Py - Ri	Px = Py Ri	Px = Py ^ Ri
Px = Py & Ri	Ri = Px + Ri	Ri = Px - Ri	Ri = Px Ri	Ri = Px ^ Ri
Ri = Px & Ri	Px = Py + data	Px = Py - data	Px = Py data	Px = Py ^ data

4-bit I/O Command				
Px = Py & data	Ri = Px + data	Ri = Px - data	Ri = Px data	Ri = Px ^ data
Ri = Px & data	Px = [1 x 0 FD An]	Px.n= 1KHz(time)	-	-
8-bit I/O Command				
XiL = Px	XiH = Px	XiL.n = Px.n	XiH.n = Px.n	Xi.n = Px.n
Xi = [Px, Py]	Px = XiL	Px = XiH	Px.n = XiL.n	Px.n = XiH.n
Px.n = Xi.n	[Px, Py] = Xi	-	-	-

6.5.3.4 Path Command

Path Command				
ASM	BG	BreakFG	StopFG	StopBG
StopBG1	StopBG2	Subroutine	Label(Pathname)	Macro

6.5.3.5 Voice Command

Voice Command				
PlayV	PlayVS	WaitVN(Ch)	PauseV(Ch)	ResumeV(Ch)
StopV(Ch)	V Chx Freq = nK	V Chx Vol = n	V Chx Vol = Xi	Xi = V Chx Vol

6.5.3.6 Sentence Command

Sentence Command				
PlayS	-	-	-	-

6.5.3.7 Timer Command

Timer Command				
TMR_ON	TMR_OFF	-	-	-

6.5.3.8 MIDI Command

MIDI Command				
PlayM	PlayMS	WaitMN	PauseM	ResumeM
StopM	Instrument	M Chx Vol = n	M Chx Vol = Ri	Ri = M Chx Vol
Tempo + n	Tempo - n	Tempo++	Tempo--	TempoRst
ReadTempo(Ri:Rj)	ReadTempo(Xi)	Mute_On(Ch)	Mute_Off(Ch)	OKON_On
OKON_Off	OKON_Play	OKON_SustainOn	OKON_SustainOff	OKON_SustainEnd
DynamicOn	DynamicOff			

6.5.3.9 Keyboard Command

Instrument Command				
InstNoteOn	InstNoteOff	DrumNoteOn	DrumNoteOff	MaxSingleNote

6.5.3.10 Volume Command

Volume Command				
Vol Max	Vol Min	Vol = n	Vol = Ri	Vol++
Vol--	Ri = Vol	Px = Vol	-	-

6.5.3.11 Table Command

Table Command	
TableL(TableName, Rx, Ry, Ri)	TableM(TableName, Rx, Ry, Ri)
TableH(TableName, Rx, Ry, Ri)	Table(TableName, Rx, Ry, Rh, Rm, Ri)
TableL(TableName, X, Y, Ri)	TableM(TableName, X, Y, Ri)
TableH(TableName, X, Y, Ri)	Table(TableName, X, Y, Rh, Rm, Ri)
TableL(TableName, Xx, Xy, Xi)	TableH(TableName, Xx, Xy, Xi)
Table(TableName, Xx, Xy, Xh, Xi)	TableL(TableName, X, Y, Xi)
TableH(TableName, X, Y, Xi)	Table(TableName, X, Y, Xh, Xi)

6.5.3.12 IR Command

IR Command				
IR_TX=data	IR_TX(Ri:Rk:Rj:Ri)	IR_TX(Xj:Xi)	IR_TX WaitN	IR_RX_ON
IR_RX_OFF	[Ri,Rk,Rj,Ri] = IR_RX		[Xj,Xi] = IR_RX	-
IR_RX = data?Path		IR_RX != data?Path		-
IR_Carrier On	IR_Carrier Off	-	-	-

6.5.3.13 Serial Control Command

Serial Control Command			
SC_RX_ON	SC_RX_OFF	[Ri,Rk,Rj,Ri] = SC_RX	[Xj,Xi] = SC_RX
SC_RX = data?Path		SC_RX != data?Path	

6.5.3.14 I2C Command

I2C Command				
I2C_TX	I2C_RX	I2C_RX=data?Path		
I2C_RX!=Data?Path		I2C_Ack?Path	I2C_Start	I2C_Stop
I2C_MReadAck	I2C_MReadNAck	-	-	-

6.5.3.15 UART Command

UART Command				
UART_TX	UART_TX WaitN	UART_RX	UART_RX=data?Path	
UART_RX!=data?Path	-	-	-	

6.5.3.16 PWMIO Command

PWMIO Command				
PWMOOut	PWMOOutS	PlayPWM	PlayPWMS	WaitPN
StopPWM	PausePWM	ResumePWM	PWMDuty	-

6.5.3.17 Interrupt Command

Interrupt Command				
INT_ON	INT_OFF	INT_RET	INT = n	-

6.5.3.18 Delay Command

Delay Command				
Delay(time)	Delay(Ri:Rj:Rk)	WaitDN(n)	StopD(n)	PauseD(n)
ResumeD(n)	SDelay(time)	-	-	-

6.5.3.19 Action Command

Action Command				
PlayA	PlayAS	WaitAN(Ch)	PauseA(Ch)	ResumeA(Ch)
StopA(Ch)	-	-	-	-

6.5.3.20 QFID Command

QFID Command				
QFID_TagId	QFID_TagInput	QFID_On	QFID_Off	QFID_SlowOn
QFID_SlowOff	-	-	-	-

6.5.3.21 I/O Expander Command

I/O Expander Command				
IoExp Input State	IoExp Output State	IoExp_Key_CLR	IoExp_Key_ON	IoExp_Key_OFF
IoExp_Sleep	IoExp_ErrId	IoExp_ReadInfo	IoExp_ReadSN	-

IoExp_SetInputPullHigh	IoExp_SetInputPullLow
IoExp_SetInputFloating	IoExp_SetOutputHigh
IoExp_SetOutputLow	-

6.5.3.22 MISC Command

MISC Command				
Input State	Output State	Action Mark State	Wave Mark State	Melody Mark State
NoteOn State	PWM-IO Mark State		QFID State	Key_CLR
Key_ON	Key_OFF	Stop	Pause(n)	Resume(n)
ReadChannel	PauseDown	ResumeUp	Audio_ON	Audio_OFF
AudioMode	RampUp	RampDown	WDT_CLR	Repeat
Background	Slow	SlowOff	END	Debounce
Direct_Debounce	Matrix_Debounce		Ri = LVD	ReadRollingCode

6.5.4 Command Table of NY6

6.5.4.1 Arithmetic Command

Arithmetic Logic Command			
Var1 = Var2	Var1 = Var2 + Var3	Var1 = Var2 - Var3	Var1 = Var2 * Var3
Var1 = Var2 / Var3	Var1 = Var2 % Var3	Var++	Var--
Var1 = Var2 & Var3	Var1 = Var2 Var3	Var1 = Var2 ^ Var3	Var1 = Var2 << Var3
Var1 = Var2 >> Var3	!Var	Var=RandomL	Var=RandomH
Var=Random	-	-	-

6.5.4.2 Flow Control Command

Flow Control Command				
Var = Var ? Path	Var != Var ? Path	Var >= Var ? Path	Var <= Var ? Path	Var > Var ? Path
Var < Var ? Path	Px = data?Path	Px != data?Path	Px.n = 0?Path	Px.n != 0?Path
Px.n = 1?Path	Px.n != 1?Path	Vol = n?Path	Vol !=n ?Path	-
RandomL=data?Path	RandomH=data?Path	RandomL!=data?Path	RandomH!=data?Path	
Random=data?Path	Random!=data?Path	Px[1 X 0 X]?Path	ChUsed(Ch)?Path	
Voice(Ch)?Path	PaueV(Ch)?Path	Melody?Path	PauseM?Path	Delay(n)?Path
Action(Ch)?Path	PauseA(Ch)?Path	PWMIO(n)?Path	PausePWM?Path	
CheckSum?Path		!Px[1 X 0 X]?Path	!ChUsed?Path	!Voice?Path

Flow Control Command				
!PauseV?Path	!Melody?Path	!PauseM?Path	!Delay?Path	!Action?Path
!PauseA?Path	!PWMIO?Path	!PausePWM?Path		-
!CheckSum?Path		If-Else	-	-
Switch(Ri) = [Path0, Path1, Path2...Path15]		Switch(Px) = [Path0, Path1, Path2...Path15]		
Switch(RandomL) = [Path0, Path1,..Path15]		Switch(RandomH) = [Path0, Path1,..Path15]		
Switch(Px[d x d x]) = [Path0, Path1, Path2....Path15]				
Switch(Xi)=[Path0, Path1, Path2,..Path255]		-		
Switch(Random)=[Path0, Path1,Path2...Path255]				
While	Do-While	For	-	-

6.5.4.3 I/O Command

4-bit I/O Command				
Ri = Px	Ri = PxM	Ri.n = Px.n	PxM = data	PxM = Ri
PxM.n = 0	PxM.n = 1	Px = data	Px = Ri	Px = Py
!Px	!Px.n	Px.n = 0	Px.n = 1	Px.n = Ri.n
Px.n = Py.n	Px = Py + Ri	Px = Py - Ri	Px = Py Ri	Px = Py ^ Ri
Px = Py & Ri	Ri = Px + Ri	Ri = Px - Ri	Ri = Px Ri	Ri = Px ^ Ri
Ri = Px & Ri	Px = Py + data	Px = Py - data	Px = Py data	Px = Py ^ data
Px = Py & data	Ri = Px + data	Ri = Px - data	Ri = Px data	Ri = Px ^ data
Ri = Px & data	Px = [1 x 0 FD An]	Px.n= 1KHz(time)	-	-
8-bit I/O Command				
XiL = Px	XiH = Px	XiL.n = Px.n	XiH.n = Px.n	Xi.n = Px.n
Xi = [Px, Py]	Px = XiL	Px = XiH	Px.n = XiL.n	Px.n = XiH.n
Px.n = Xi.n	[Px, Py] = Xi	-	-	-

6.5.4.4 Path Command

Path Command				
ASM	BG	BreakFG	StopFG	StopBG
StopBG1	StopBG2	Subroutine	Label(Pathname)	Macro

6.5.4.5 Voice Command

Voice Command				
PlayV	PlayVS	WaitVN(Ch)	PauseV(Ch)	ResumeV(Ch)
StopV(Ch)	V_Chx_Vol = n	V_Chx_Vol = Xi	Xi = V_Chx_Vol	-

6.5.4.6 Sentence Command

Sentence Command				
PlayS	WaitSN(n)	PauseS(n)	ResumeS(n)	StopS(n)

6.5.4.7 SPIPlay Command

SPIPlay Command				
SPIPlay	SPIPlayS	SPIWaitN	SPIStop	SPIPause
SPIResume	SPIVol = n	SPIVol = Ri	Ri = SPIVol	-
SPIGetIndex(Result, SPIGroup)	-	-	-	-

6.5.4.8 SPI Flash Command

SPI Flash Command				
SPI_WREN	SPI_WRDIS	SPI_RDID	SPI_CE	SPI_SE
SPI_BE	SPI_DP	SPI_RDP	SPI_WRSR	SPI_RDSR
SPI_WRD	SPI_RDD	SPI_GetAddr	-	-

6.5.4.9 SPI Command

SPI Command				
SPI_CS_On	SPI_CS_Off	SPI_TX	SPI_RX	-

6.5.4.10 Comparator Command

Comparator Command				
CMP_ON	CMP_ON(Source)	CMP_OFF	CMP_Read(Rj:Ri)	CMP_Read(Xi)
CMP_CNT_ON(Count)	CMP_CNT_OFF	-	-	-

6.5.4.11 Timer Command

Timer Command				
TMR_ON	TMR_OFF	TMR_Read(Rj:Ri)	TMR_Read(Xi)	-

6.5.4.12 MIDI Command

MIDI Command				
PlayM	PlayMS	WaitMN	PauseM	ResumeM
StopM	Instrument(Ch, i)	M_Chx_Vol = n	M_Chx_Vol = Ri	Ri = M_Chx_Vol
Tempo + n	Tempo - n	Tempo++	Tempo--	TempoRst
ReadTempo(Ri:Ri)	ReadTempo(Xi)	Mute_On(Ch)	Mute_Off(Ch)	OKON_On
OKON_Off	OKON_Play	DynamicOn	DynamicOff	-

6.5.4.13 Keyboard Command

Instrument Command				
InstNoteOn	InstNoteOff	DrumNoteOn	DrumNoteOff	MaxSingleNote

6.5.4.14 Volume Command

Volume Command				
Vol_Max	Vol_Min	Vol = n	Vol = Ri	Vol++
Vol--	Ri = Vol	Px = Vol	VolX1	VolX2

6.5.4.15 Table Command

Table Command	
TableL(TableName, Rx/Xx, Ry/Xy, Ri)	TableM(TableName, Rx/Xx, Ry/Xy, Ri)
TableH(TableName, Rx/Xx, Ry/Xy, Ri)	Table(TableName, Rx/Xx, Ry/Xy, Rh, Rm, Ri)
TableL(TableName, X, Y, Ri)	TableM(TableName, X, Y, Ri)
TableH(TableName, X, Y, Ri)	Table(TableName, X, Y, Rh, Rm, Ri)
TableL(TableName, Rx/Xx, Ry/Xy, Xi)	TableH(TableName, Rx/Xx, Ry/Xy, Xi)
Table(TableName, Rx/Xx, Ry/Xy, Xh, Xi)	TableL(TableName, X, Y, Xi)
TableH(TableName, X, Y, Xi)	Table(TableName, X, Y, Xh, Xi)

6.5.4.16 IR Command

IR Command				
IR_TX=data	IR_TX(Ri:Rk:Rj:Ri)	IR_TX(Xj:Xi)	IR_RX_ON	IR_RX_OFF
[Ri,Rk,Rj,Ri] = IR_RX	[Xj,Xi] = IR_RX	IR_RX = data?Path	IR_RX != data?Path	
IR_Carrier_On	IR_Carrier_Off	-	-	-

6.5.4.17 Serial Control Command

Serial Control Command			
SC_RX_ON	SC_RX_OFF	[Ri,Rk,Rj,Ri] = SC_RX	[Xj,Xi] = SC_RX
SC_RX = data?Path		SC_RX != data?Path	

6.5.4.18 PWMIO Command

PWMIO Command				
PWMOut	PWMOutS	WaitPN	StopPWM	PausePWM
ResumePWM	-	-	-	-

6.5.4.19 Delay Command

Delay Command				
Delay(time)	Delay(Ri:Rj:Rk)	WaitDN(n)	StopD(n)	PauseD(n)
ResumeD(n)	SDelay(time)	-	-	-

6.5.4.20 Action Command

Action Command				
PlayA	PlayAS	WaitAN(Ch)	PauseA(Ch)	ResumeA(Ch)
StopA(Ch)	-	-	-	-

6.5.4.21 RFC Command

RFC Command				
RFC_On	RFC_Off	RFC_Level(Ri)	-	-

6.5.4.22 MISC Command

MISC Command				

MISC Command				
Input State	Output State	Action Mark State	Melody Mark State	NoteOn State
Key_CLR	Key_ON	Key_OFF	Stop	Pause(n)
Resume(n)	ReadChannel	Audio_ON	Audio_OFF	AudioMode
NoiseFilter_ON	NoiseFilter_OFF	RampUp	RampDown	WDT_CLR
Repeat	Background	Slow	SlowOff	END
ChMode(n)	ReadRollingCode		Ri = LVD	Debounce
Direct_Debounce(Time)		Matrix_Debounce(Time)		-

6.5.5 Command Table of NY7

6.5.5.1 Arithmetic Command

Arithmetic Logic Command			
Var1 = Var2	Var1 = Var2 + Var3	Var1 = Var2 - Var3	Var1 = Var2 * Var3
Var1 = Var2 / Var3	Var1 = Var2 % Var3	Var++	Var--
Var1 = Var2 & Var3	Var1 = Var2 Var3	Var1 = Var2 ^ Var3	Var1 = Var2 << Var3
Var1 = Var2 >> Var3	!Var	Var=RandomL	Var=RandomH
Var=Random	-	-	-

6.5.5.2 Flow Control Command

Flow Control Command				
Var = Var ? Path	Var != Var ? Path	Var >= Var ? Path	Var <= Var ? Path	Var > Var ? Path
Var < Var ? Path	Px = data?Path	Px != data?Path	Px.n = 0?Path	Px.n != 0?Path
Px.n = 1?Path	Px.n != 1?Path	Vol = n?Path	Vol != n?Path	-
RandomL=data?Path	RandomH=data?Path	RandomL!=data?Path	RandomH!=data?Path	
Random=data?Path	Random!=data?Path	Px[1 X 0 X]?Path		ChUsed(Ch)?Path
Voice(Ch)?Path	PaueV(Ch)?Path	Melody?Path	PauseM?Path	Delay(n)?Path
Action(Ch)?Path	PauseA(Ch)?Path	PWMIO(n)?Path	PausePWM?Path	-
CheckSum?Path		!Px[1 X 0 X]?Path		!ChUsed?Path
!Voice?Path	!PauseV?Path	!Melody?Path	!PauseM?Path	!Delay?Path
!Action?Path	!PauseA?Path	!PWMIO?Path	!PausePWM?Path	
!CheckSum?Path		If-Else	-	-

Flow Control Command				
Switch(Ri) = [Path0, Path1, Path2,..Path15]		Switch(Px) = [Path0, Path1, Path2,..Path15]		
Switch(RandomL) = [Path0, Path1,..Path15]		Switch(RandomH) = [Path0, Path1,..Path15]		
Switch(Px[d x d x]) = [Path0, Path1, Path2....Path15]				
Switch(Xi)=[Path0, Path1, Path2...Path255]		-		
Switch(Random)=[Path0, Path1,Path2,..Path255]				
While	Do-While	For	-	=

6.5.5.3 I/O Command

4-bit I/O Command				
Ri = Px	Ri = PxM	Ri.n = Px.n	PxM = data	PxM = Ri
PxM.n = 0	PxM.n = 1	Px = data	Px = Ri	Px = Py
IPx	IPx.n	Px.n = 0	Px.n = 1	Px.n = Ri.n
Px.n = Py.n	Px = Py + Ri	Px = Py - Ri	Px = Py Ri	Px = Py ^ Ri
Px = Py & Ri	Ri = Px + Ri	Ri = Px - Ri	Ri = Px Ri	Ri = Px ^ Ri
Ri = Px & Ri	Px = Py + data	Px = Py - data	Px = Py data	Px = Py ^ data
Px = Py & data	Ri = Px + data	Ri = Px - data	Ri = Px data	Ri = Px ^ data
Ri = Px & data	Px = [1 x 0 FD An]	Px.n= 1KHz(time)	-	-
8-bit I/O Command				
XiL = Px	XiH = Px	XiL.n = Px.n	XiH.n = Px.n	Xi.n = Px.n
Xi = [Px, Py]	Px = XiL	Px = XiH	Px.n = XiL.n	Px.n = XiH.n
Px.n = Xi.n	[Px, Py] = Xi	-	-	-

6.5.5.4 Path Command

Path Command				
ASM	BG	BreakFG	StopFG	StopBG
StopBG1	StopBG2	Subroutine	Label(Pathname)	Macro

6.5.5.5 Voice Command

Voice Command				
PlayV	PlayVS	WaitVN(Ch)	PauseV(Ch)	ResumeV(Ch)
StopV(Ch)	V Chx Vol = n	V Chx Vol = Xi	Xi = V Chx Vol	-

6.5.5.6 Sentence Command

Sentence Command				
PlayS	WaitSN(n)	PauseS(n)	ResumeS(n)	StopS(n)

6.5.5.7 SPIPlay Command

SPIPlay Command				
SPIPlay	SPIPlayS	SPIWaitN	SPIStop	SPIPause
SPIResume	SPIVol = n	SPIVol = Ri	Ri = SPIVol	-
SPIGetIndex(Result, SPIGroup)	-			-

6.5.5.8 SPI Flash Command

SPI Flash Command				
SPI_WREN	SPI_WRDIS	SPI_RDID	SPI_CE	SPI_SE
SPI_BE	SPI_DP	SPI_RDP	SPI_WRSR	SPI_RDSR
SPI_WRD	SPI_RDD	SPI_GetAddr	-	-

6.5.5.9 SPI Command

SPI Command				
SPI_CS_On	SPI_CS_Off	SPI_TX	SPI_RX	-

6.5.5.10 MIDI Command

MIDI Command				
PlayM	PlayMS	WaitMN	PauseM	ResumeM
StopM	Instrument(Ch, i)	M_Chx_Vol = n	M_Chx_Vol = Ri	Ri = M_Chx_Vol
Tempo + n	Tempo - n	Tempo++	Tempo--	TempoRst
ReadTempo(Ri:Rj)	ReadTempo(Xi)	Mute_On(Ch)	Mute_Off(Ch)	OKON_On
OKON_Off	OKON_Play	DynamicOn	DynamicOff	StopMNote

6.5.5.11 Keyboard Command

Instrument Command				
InstNoteOn	InstNoteOff	DrumNoteOn	DrumNoteOff	NoteVibrato
Gliss	MaxSingleNote	-	-	-

6.5.5.12 Volume Command

Volume Command				
Vol_Max	Vol_Min	Vol = n	Vol = Ri	Vol++
Vol--	Ri = Vol	Px = Vol	-	-

6.5.5.13 Table Command

Table Command	
TableL(Table Name, Rx/Xx, Ry/Yy, Ri)	TableM(Table Name, Rx/Xx, Ry/Yy, Ri)
TableH(Table Name, Rx/Xx, Ry/Yy, Ri)	Table(Table Name, Rx/Xx, Ry/Yy, Rh, Rm, Ri)
TableL(Table Name, X, Y, Ri)	TableM(Table Name, X, Y, Ri)
TableH(Table Name, X, Y, Ri)	Table(Table Name, X, Y, Rh, Rm, Ri)
TableL(Table Name, Rx/Xx, Ry/Yy, Xi)	TableH(Table Name, Rx/Xx, Ry/Yy, Xi)
Table(Table Name, Rx/Xx, Ry/Yy, Xh, Xi)	TableL(Table Name, X, Y, Xi)
TableH(Table Name, X, Y, Xi)	Table(Table Name, X, Y, Xh, Xi)

6.5.5.14 IR Command

IR Command				
IR_TX=data	IR_TX(Ri:Rk:Rj:Ri)	IR_TX(Xj:Xi)	IR_RX_ON	IR_RX_OFF
[Ri,Rk,Rj,Ri] = IR_RX	[Xj,Xi] = IR_RX	IR_RX = data?Path	IR_RX != data?Path	
IR_Carrier_On	IR_Carrier_Off	-	-	-

6.5.5.15 Serial Control Command

Serial Control Command			
SC_RX_ON	SC_RX_OFF	[Ri,Rk,Rj,Ri] = SC_RX	[Xj,Xi] = SC_RX
SC_RX = data?Path		SC_RX != data?Path	

6.5.5.16 PWMIO Command

PWMIO Command				
PWMOut	PWMOutS	WaitPN	StopPWM	PausePWM
ResumePWM	-	-	-	-

6.5.5.17 Delay Command

Delay Command				
Delay(time)	Delay(Ri:Rj:Rk)	WaitDN(n)	StopD(n)	PauseD(n)
ResumeD(n)	SDelay(time)	-	-	-

6.5.5.18 Action Command

Action Command				
PlayA	PlayAS	WaitAN(Ch)	PauseA(Ch)	ResumeA(Ch)
StopA(Ch)	-	-	-	-

6.5.5.19 QFID Command

QFID Command				
QFID_TagId	QFID_TagInput	QFID_On	QFID_Off	QFID_SlowOn
QFID_SlowOff	-	-	-	-

6.5.5.20 RFC Command

RFC Command				
RFC_On	RFC_Off	RFC_Level(Ri)	-	-

6.5.5.21 MISC Command

MISC Command				
Input State	Output State	Action Mark State	Melody Mark State	NoteOn State
QFID State	Key_CLR	Key_ON	Key_OFF	Stop
Pause(n)	Resume(n)	ReadChannel	Audio_ON	Audio_OFF
AudioMode	NoiseFilter_ON	NoiseFilter_OFF	RampUp	RampDown
WDT_CLR	Repeat	Background	Slow	SlowOff
END	ChMode(n)	ReadRollingCode	Debounce(Time)	-
Direct_Debounce(Time)		Matrix_Debounce(Time)		-

6.5.6 Command Table of NY9T

6.5.6.1 Arithmetic Command

Arithmetic Logic Command			
Var1 = Var2	Var1 = Var2 + Var3	Var1 = Var2 - Var3	Var1 = Var2 * Var3
Var1 = Var2 / Var3	Var1 = Var2 % Var3	Var++	Var--
Var1 = Var2 & Var3	Var1 = Var2 Var3	Var1 = Var2 ^ Var3	Var1 = Var2 << Var3
Var1 = Var2 >> Var3	!Var	Var=RandomL	Var=RandomH
Var=Random	-	-	-

6.5.6.2 Flow Control Command

Flow Control Command				
Var = Var ? Path	Var != Var ? Path	Var >= Var ? Path	Var <= Var ? Path	Var > Var ? Path
Var < Var ? Path	Px = data?Path	Px != data?Path	Px.n = 0?Path	Px.n != 0?Path
Px.n = 1?Path	Px.n != 1?Path	-	-	-
RandomL=data?Path	RandomH=data?Path	RandomL!=data?Path	RandomH!=data?Path	
Random=data?Path	Random!=data?Path	Px[1 X 0 X]?Path	Delay(n)?Path	
PauseD(n)?Path	Action(Ch)?Path	PauseA(Ch)?Path	PWMIO(n)?Path	
PausePWM?Path		HoldPWM?Path	Pause(n)?Path	Calibrate?Path
CheckSum?Path	!Px[1 X 0 X]?Path		!Delay?Path	!Paused?Path
!Action?Path	!PauseA?Path	!PWMIO?Path	!PausePWM?Path	
!HoldPWM?Path	!Pause?Path	!Calibrate?Path	!CheckSum?Path	If-Else
Switch(Ri) = [Path0, Path1, Path2...Path15]		Switch(Px) = [Path0, Path1, Path2...Path15]		
Switch(RandomL) = [Path0, Path1,...Path15]		Switch(RandomH) = [Path0, Path1,...Path15]		
Switch(Px[d x d x]) = [Path0, Path1, Path2....Path15]				
Switch(Xi)=[Path0, Path1, Path2...Path255]		-		
Switch(Random)=[Path0, Path1,Path2...Path255]				
While	Do-While	For	-	-

6.5.6.3 I/O Command

4-bit I/O Command				
Ri = Px	Ri = PxM	Ri.n = Px.n	PxM = data	PxM = Ri
PxM.n = 0	PxM.n = 1	Px = data	Px = Ri	Px = Py

4-bit I/O Command				
!Px	!Px.n	Px.n = 0	Px.n = 1	Px.n = Ri.n
Px.n = Py.n	Px = Py + Ri	Px = Py - Ri	Px = Py Ri	Px = Py ^ Ri
Px = Py & Ri	Ri = Px + Ri	Ri = Px - Ri	Ri = Px Ri	Ri = Px ^ Ri
Ri = Px & Ri	Px = Py + data	Px = Py - data	Px = Py data	Px = Py ^ data
Px = Py & data	Ri = Px + data	Ri = Px - data	Ri = Px data	Ri = Px ^ data
Ri = Px & data	Px = [1 x 0 FD An]	Px.n= 1KHz(time)	-	-
8-bit I/O Command				
XiL = Px	XiH = Px	XiL.n = Px.n	XiH.n = Px.n	Xi.n = Px.n
Xi = [Px, Py]	Px = XiL	Px = XiH	Px.n = XiL.n	Px.n = XiH.n
Px.n = Xi.n	[Px, Py] = Xi	-	-	-

6.5.6.4 Path Command

Path Command				
ASM	BG	BreakFG	StopFG	StopBG
StopBG1	StopBG2	Subroutine	Label(Pathname)	Macro
1ms_RET	4ms_RET	-	-	-

6.5.6.5 Sentence Command

Sentence Command				
PlayS	-	-	-	-

6.5.6.6 Touch-Key Command

Touch-Key Command			
TouchKey_ON	TouchKey_OFF	TouchKey_CLR	TouchKey_Scan_Slow
TouchKey_Scan_Normal	TouchKey_Sensitivity	Calibrate_ON	Calibrate_OFF
AutoJudge_Calibrate	Enforce_Calibrate_Normal	Enforce_Calibrate_Sleep	
Ri = TouchKey(Px)	-	-	

6.5.6.7 Table Command

Table Command	
TableL(Table Name, Rx/Xx, Ry/Yy, Ri)	TableM(Table Name, Rx/Xx, Ry/Yy, Ri)
TableH(Table Name, Rx/Xx, Ry/Yy, Ri)	Table(Table Name, Rx/Xx, Ry/Yy, Rh, Rm, Ri)

Table Command	
TableL(TableName, X, Y, Ri)	TableM(TableName, X, Y, Ri)
TableH(TableName, X, Y, Ri)	Table(TableName, X, Y, Rh, Rm, Ri)
TableL(TableName, Rx/Xx, Ry/Xy, Xi)	TableH(TableName, Rx/Xx, Ry/Xy, Xi)
Table(TableName, Rx/Xx, Ry/Xy, Xh, Xi)	TableL(TableName, X, Y, Xi)
TableH(TableName, X, Y, Xi)	Table(TableName, X, Y, Xh, Xi)

6.5.6.8 Serial Control TX Command

Serial Control TX Command		
SC_TX(Mode, data)	SC_TX(Mode, Ri:Rj:Rl:Rk)	SC_TX(Mode, Xi:Xj)

6.5.6.9 PWMIO Command

PWMIO Command				
PWMOut	PWMOutS	PlayPWM	PlayPMWS	WaitPN
StopPWM	PausePWM	ResumePWM	HoldPWM	PWMCtrl

6.5.6.10 Delay Command

Delay Command				
Delay(time)	Delay(Ri:Rj:Rk)	WaitDN(n)	StopD(n)	PauseD(n)
ResumeD(n)	SDelay(time)	-	-	-

6.5.6.11 Action Command

Action Command				
PlayA	PlayAS	WaitAN(Ch)	PauseA(Ch)	ResumeA(Ch)
HoldA(Ch)	StopA(Ch)	-	-	-

6.5.6.12 MISC Command

MISC Command				
Input State	Output State	Action Mark State	Key_CLR	Key_ON
Key_OFF	Stop	Pause(n)	Resume(n)	WDT_CLR
Repeat	Background	END	ReadRadj(Ri)	SW_Reset
Debounce(Time)	-	-	-	-

6.5.7 Command Table of NX1 OTP

6.5.7.1 Arithmetic Command

Arithmetic Logic Command			
Var1 = Var2	Var1 = Var2 + Var3	Var1 = Var2 – Var3	Var1 = Var2 * Var3
Var1 = Var2 / Var3	Var1 = Var2 % Var3	Var++	Var--
Var1 = Var2 & Var3	Var1 = Var2 Var3	Var1 = Var2 ^ Var3	Var1 = Var2 << Var3
Var1 = Var2 >> Var3	!Var	Var=RandomL	Var=RandomH
Var=Random	-	-	-

6.5.7.2 Flow Control Command

Flow Control Command				
Var = Var ? Path	Var != Var ? Path	Var >= Var ? Path	Var <= Var ? Path	Var > Var ? Path
Var < Var ? Path	Px = data?Path	Px != data?Path	Px.n = 0?Path	Px.n != 0?Path
Px.n = 1?Path	Px.n != 1?Path	Vol = n?Path	Vol !=n ?Path	-
RandomL=data?Path	RandomH=data?Path	RandomL!=data?Path	RandomH!=data?Path	
Random=data?Path	Random!=data?Path	Px[1 X 0 X]?Path	ChUsed(Ch)?Path	
Voice(Ch)?Path	PaueV(Ch)?Path	Melody?Path	PauseM?Path	Delay(n)?Path
Action(Ch)?Path	PauseA(Ch)?Path	PWMIO(n)?Path	PausePWM?Path	
SPIPlay?Path	SPIPause?Path	Record?Path	KRecord?Path	Recorded?Path
PlayK?Path	EraseR?Path	VR_VAD?Path	LEDStr?Path	LEDSync?Path
LEDText?Path	-	-	-	-
Animaltalks_Record?Path	Animaltalks_Play?Path			-
Animalsings_Record?Path	Animalsings_Play?Path			-
IoExp_Exists?Path	CheckSum?Path	SPI_CheckSum?Path		
!Px[1 X 0 X]?Path	!ChUsed?Path	!Voice?Path	!PauseV?Path	
!Melody?Path	!PauseM?Path	!Delay?Path	!Action?Path	!PauseA?Path
!PWMIO?Path	!PausePWM?Path	!SPIPlay?Path		-
!SPIPause?Path	!Record?Path	!KRecord?Path	!Recorded?Path	!PlayK?Path
!EraseR?Path	!VR_VAD?Path	!LEDStr?Path	!LEDSync?Path	!LEDText?Path
!Animaltalks_Record?Path	!Animaltalks_Play?Path			-
!Animalsings_Record?Path	!Animalsings_Play?Path			-
!IoExp_Exists?Path	!CheckSum?Path	!SPI_CheckSum?Path		

Flow Control Command				
<u>If-Else</u>	-	-	-	=
<u>Switch(Ri) = [Path0, Path1, Path2...Path15]</u>		<u>Switch(Px) = [Path0, Path1, Path2...Path15]</u>		
<u>Switch(RandomL) = [Path0, Path1,..Path15]</u>		<u>Switch(RandomH) = [Path0, Path1,..Path15]</u>		
<u>Switch(Px[d x d x]) = [Path0, Path1, Path2....Path15]</u>				
<u>Switch(Xi)=[Path0, Path1, Path2,..Path255]</u>		-		
<u>Switch(Random)=[Path0, Path1,Path2,..Path255]</u>				
<u>While</u>	<u>Do-While</u>	<u>For</u>	-	=

6.5.7.3 I/O Command

4-bit I/O Command				
Ri = Px	Ri = PxM	Ri.n = Px.n	PxM = data	PxM = Ri
PxM.n = 0	PxM.n = 1	Px = data	Px = Ri	Px = Py
!Px	!Px.n	Px.n = 0	Px.n = 1	Px.n = Ri.n
Px.n = Py.n	Px = Py + Ri	Px = Py - Ri	Px = Py Ri	Px = Py ^ Ri
Px = Py & Ri	Ri = Px + Ri	Ri = Px - Ri	Ri = Px Ri	Ri = Px ^ Ri
Ri = Px & Ri	Px = Py + data	Px = Py - data	Px = Py data	Px = Py ^ data
Px = Py & data	Ri = Px + data	Ri = Px - data	Ri = Px data	Ri = Px ^ data
Ri = Px & data	Px = [1 x 0 FD An]	-	-	-

6.5.7.4 Path Command

Path Command				
C-Code	BG	StopFG	StopBG	StopBG1
StopBG2	StopBG3	StopBG4	StopBG5	Subroutine
Label(Pathname)	Macro	-	-	-

6.5.7.5 Voice Command

Voice Command				
PlayV	PlayVS	WaitVN(Ch)	PauseV(Ch)	ResumeV(Ch)
StopV(Ch)	SBC Loop On	SBC Loop Off	ADPCM Loop On	ADPCM Loop Off
ADPCM UpSampling		ADM Loop On	ADM Loop Off	ADM UpSampling
PCM Loop On	PCM Loop Off	ReadFileCountV	-	-

6.5.7.6 Record Command

Record Command				
Record	RecordS	WaitRN	StopR	EraseR
EraseRS	WaitEN	-	-	-

6.5.7.7 Sentence Command

Sentence Command				
PlayS	WaitSN	PauseS(n)	ResumeS(n)	StopS(n)

6.5.7.8 SPIPlay Command

SPIPlay Command				
SPIPlay	SPIPlayS	SPIWaitN	SPIStop	SPIPause
SPIResume	SPIGetIndex(Result, SPIGroup)	-	-	-

6.5.7.9 SPI Flash Command

SPI Flash Command				
SPI_WREN	SPI_WRDIS	SPI_RDID	SPI_CE	SPI_SE
SPI_BE	SPI_DP	SPI_RDP	SPI_WRSR	SPI_RDSR
SPI_WRD	SPI_RDD	SPI_GetAddr	-	-

6.5.7.10 SPI Command

SPI Command				
SPI_CS_On	SPI_CS_Off	SPI_TX	SPI_RX	SPI_CLKDIV
SPI_CPOL	SPI_CPHA	-	-	-

6.5.7.11 Storage Command

Storage Command				
Storage_Save	-	-	-	-

6.5.7.12 Timer Command

Timer Command				
TMR_ON	TMR_OFF	-	-	-

6.5.7.13 MIDI Command

MIDI Command				
PlayM	PlayMS	WaitMN	PauseM	ResumeM
StopM	Instrument	M_Chx_Vol = n	M_Chx_Vol = Ri	Ri = M_Chx_Vol
Tempo + n	Tempo - n	Tempo++	Tempo--	TempoRst
ReadTempo(Ri)	Mute_On(Ch)	Mute_Off(Ch)	OKON_On	OKON_Off
OKON_Play	MIDI_Pitch	Mask_On	Mask_Off	ReadFileCountM
MIDI_Loop_On	MIDI_Loop_Off	-	-	-

6.5.7.14 Keyboard Command

Instrument Command				
InstNoteOn	InstNoteOff	InstNoteAllOff	DrumNoteOn	DrumNoteOff
NoteVibrato	Gliss	LongInst_HoldTime		-
ShortInst_HoldTime		KRecord	KRecordS	WaitKRN
StopKR	PlayK	PlayKS	WaitKN	StopK

6.5.7.15 Volume Command

Volume Command				
Vol_Max	Vol_Min	Vol = n	Vol = Ri	Vol++
Vol--	Ri = Vol	Px = Vol	Vol += n	Vol -= n
CHx_Vol = n	PP_Gain = n	PP_Gain = Ri	PP_Gain++	PP_Gain--
Ri=PP_Gain	PGA_Gain = n	PGA_Gain = Ri	Ri = PGA_Gain	MixCtrl

6.5.7.16 TouchKey Command

TouchKey Command			
TouchKey_Sensitivity	Enforce_Calibrate_Normal		Touchkey_Count
Touchkey_BGCount	-	-	-

6.5.7.17 Table Command

Table Command	
Table(TableName, VarX, VarY, VarR)	-

6.5.7.18 IR Command

IR Command				
IR_TX=data	IR_TX(Ri)	IR_TX WaitN	IR_RX_ON	IR_RX_OFF
Ri = IR_RX	IR_RX = data?Path		IR_RX != data?Path	
IR_TX_Busy?Path		IR_Carrier_On	IR_Carrier_Off	-

6.5.7.19 I2C Command

I2C Command				
I2C_TX	I2C_RX	I2C_RX=data?Path		
I2C_RX!=Data?Path		I2C_Ack?Path	I2C_Start	I2C_Stop
I2C_MReadAck	I2C_MReadNAck	I2C_Reset	-	-

6.5.7.20 UART Command

UART Command				
UART_TX	UART_RX	UART_RX=data?Path	-	
UART_RX!=data?Path		-	-	-

6.5.7.21 Interrupt Command

Interrupt Command				
INT_ON	INT_OFF	-	-	-

6.5.7.22 Delay Command

Delay Command				
Delay(time)	Delay(Ri:Rj:Rk)	WaitDN(n)	StopD(n)	PauseD(n)
ResumeD(n)	SDelay(time)	-	-	-

6.5.7.23 Action Command

Action Command				
PlayA	PlayAS	WaitAN(Ch)	PauseA(Ch)	ResumeA(Ch)
StopA(Ch)	-	-	-	-

6.5.7.24 LED Strip Command

LED Strip Command				

LEDStr_Play	LEDStr_PlayS	LEDStr_Stop	LEDStr_Clear	LEDStr_Brightness
LEDSync_Play	LEDSync_PlayS	LEDSync_Stop	LEDSync_Clear	LEDSync_Brightness
LEDText_Play	LEDText_PlayS	LEDText_Stop	LEDText_Clear	LEDText_Brightness

6.5.7.25 QFID Command

QFID Command				
QFID_GroupID	QFID_TagId	QFID_TagInput	QFID_On	QFID_Off

6.5.7.26 WaveID Command

WaveID Command				
WaveID_TX	WaveID_RX_ON	WaveID_RX_OFF	WaveID_RX	-

6.5.7.27 Sound Localization Command

Sound Localization Command				
SL_On	SL_Off	SL_RX	-	-

6.5.7.28 Sound Detect Command

Sound Detect Command				
SoundDetect_On	SoundDetect_Off	-	-	-

6.5.7.29 Pitch Detect Command

Pitch Detect Command				
PitchDetect_On	PitchDetect_Off	ReadPitch	-	-

6.5.7.30 RFC Command

RFC Command				
RFC_On	RFC_Off	RFC_Level(Ri)	-	-

6.5.7.31 ADC Input Command

ADC Input Command				
ADC_Input	=	=	-	-

6.5.7.32 VR Command

VR Command				
VR State	VR_ON	VR_OFF	VR_VAD=n	VR_VAD_On
VR_VAD_Off	VRGC_Timeout_CLR		Ri = VR_HitScore	Ri = VR_HitID
VR>Loading	VT_Training	VT_Delete	VT_DeleteAll	VT_TrainingNum

6.5.7.33 Sound Effect Command

Sound Effect Command				
PitchChange	PitchChange_Off	SpeedChange	SpeedChange_Off	Robot1
Robot1_Off	Robot2	Robot2_Off	Robot3	Robot3_Off
Robot4	Robot4_Off	Echo	Ecbo_Off	Reverb
Reverb_off	Darth	Darth_Off	AnimalRoar	AnimalRoar_Off

6.5.7.34 Real Time Play Command

Real Time Play Command				
RT_Play	RT_Play_Off	RT_PitchChange	RT_PitchChange_Off	
RT_Robot1	RT_Robot1_Off	RT_Robot2	RT_Robot2_Off	RT_Robot3
RT_Robot3_Off	RT_Robot4	RT_Robot4_Off	RT_Echo	RT_Echo_Off
RT_Reverb	RT_Reverb_Off	RT_Ghost	RT_Ghost_Off	RT_Darth
RT_Darth_Off	RT_Chorus	RT_Chorus_Off	RT_Vol = n	RT_Vol = Var
Var = RT_Vol	-	-	-	-

6.5.7.35 Animaltalks Command

Animaltalks Command			
Animaltalks_On	Animaltalks_Off	Animaltalks_Record	Animaltalks_RecordS
Animaltalks_StopR	Animaltalks_Play	Animaltalks_PlayS	Animaltalks_Stop
Animaltalks_SetVoice		Animaltalks_LongSound_On	-
Animaltalks_LongSound_Off		-	-

6.5.7.36 Animalsings Command

Animalsings Command			
Animalsings_On	Animalsings_Off	Animalsings_Record	Animalsings_RecordS
Animalsings_StopR	Animalsings_Play	Animalsings_PlayS	Animalsings_Stop

Animalsings_SetVoice	Animalsings_LongSound_On	
Animalsings_LongSound_Off	Animalsings_NC_On	Animalsings_NC_Off
Animalsings_NC_Auto	-	-

6.5.7.37 I/O Expander Command

I/O Expander Command				
IoExp_Input_State	IoExp_Output_State	IoExp_Key_Clr	IoExp_Key_On	IoExp_Key_Off
IoExp_Sleep	IoExp_ErrId	IoExp_ReadInfo	IoExp_ReadSN	-
IoExp_SetInputPullHigh		IoExp_SetInputPullLow		
IoExp_SetInputFloating		IoExp_SetOutputHigh		
IoExp_SetOutputLow		-		

6.5.7.38 MISC Command

MISC Command				
Input_State	Output_State	Action_Mark_State		Wave_Mark_State
Melody_Mark_State		NoteOn_State	QFID_State	Key_CLR
Key_ON	Key_OFF	Stop	Pause(n)	Resume(n)
Audio_Loop_On	Audio_Loop_Off		NoiseFilter_ON	NoiseFilter_OFF
EQ_Filter	EQ_Filter_Off	AGC_On	AGC_Off	AutoSleep_On
AutoSleep_Off	Sleep	WDT_CLR	Repeat	Background
END	ReadRollingCode		Ri = LVD	Enforce_Wakeup
GetMicVol	Millis	Srand	-	:

6.5.7.39 Debug Command

Debug Command				
Printf	-	-	-	-

6.5.8 Command Table of NX1 EF

6.4.8.1 Arithmetic Command

Arithmetic Logic Command			
Var1 = Var2	Var1 = Var2 + Var3	Var1 = Var2 - Var3	Var1 = Var2 * Var3
Var1 = Var2 / Var3	Var1 = Var2 % Var3	Var++	Var--
Var1 = Var2 & Var3	Var1 = Var2 Var3	Var1 = Var2 ^ Var3	Var1 = Var2 << Var3

Var1 = Var2 >> Var3	!Var	Var=RandomL	Var=RandomH
Var=Random	-	-	-

6.4.8.2 Flow Control Command

Flow Control Command				
Var = Var ? Path	Var != Var ? Path	Var >= Var ? Path	Var <= Var ? Path	Var > Var ? Path
Var < Var ? Path	Px = data?Path	Px != data?Path	Px.n = 0?Path	Px.n != 0?Path
Px.n = 1?Path	Px.n != 1?Path	Vol = n?Path	Vol !=n ?Path	-
RandomL=data?Path	RandomH=data?Path	RandomL!=data?Path	RandomH!=data?Path	
Random=data?Path	Random!=data?Path	Px[1 X 0 X]?Path	ChUsed(Ch)?Path	
Voice(Ch)?Path	PaueV(Ch)?Path	Melody?Path	PauseM?Path	Delay(n)?Path
Action(Ch)?Path	PauseA(Ch)?Path	PWMIO(n)?Path	PausePWM?Path	
SPIPlay?Path	SPIPause?Path	Record?Path	KRecord?Path	Recorded?Path
PlayK?Path	EraseR?Path	VR_VAD?Path	LEDStr?Path	LEDSync?Path
LEDText?Path	-	-	-	-
Animaltalks_Record?Path		Animaltalks_Play?Path		-
Animalsing_Record?Path		Animalsings_Play?Path		-
IoExp_Exists?Path		CheckSum?Path	SPI_CheckSum?Path	
!Px[1 X 0 X]?Path		!ChUsed?Path	!Voice?Path	!PauseV?Path
!Melody?Path	!PauseM?Path	!Delay?Path	!Action?Path	!PauseA?Path
!PWMIO?Path	!PausePWM?Path		!SPIPlay?Path	-
!SPIPause?Path	!Record?Path	!KRecord?Path	!Recorded?Path	!PlayK?Path
!EraseR?Path	!VR_VAD?Path	!LEDStr?Path	!LEDSync?Path	!LEDText?Path
!Animaltalks_Record?Path		!Animaltalks_Play?Path		-
!Animalsings_Record?Path		!Animalsings_Play?Path		-
!IoExp_Exists?Path		!CheckSum?Path	!SPI_CheckSum?Path	
If-Else	-	-	-	-
Switch(Ri) = [Path0, Path1, Path2,...Path15]		Switch(Px) = [Path0, Path1, Path2,...Path15]		
Switch(RandomL) = [Path0, Path1,...Path15]		Switch(RandomH) = [Path0, Path1,...Path15]		
Switch(Px[d x d x]) = [Path0, Path1, Path2....Path15]				
Switch(Xi)=[Path0, Path1, Path2,...Path255]		-		
Switch(Random)=[Path0, Path1,Path2,...Path255]				
While	Do-While	For	-	-

6.4.8.3 I/O Command

4-bit I/O Command				
Ri = Px	Ri = PxM	Ri.n = Px.n	PxM = data	PxM = Ri
PxM.n = 0	PxM.n = 1	Px = data	Px = Ri	Px = Py
!Px	!Px.n	Px.n = 0	Px.n = 1	Px.n = Ri.n
Px.n = Py.n	Px = Py + Ri	Px = Py - Ri	Px = Py Ri	Px = Py ^ Ri
Px = Py & Ri	Ri = Px + Ri	Ri = Px - Ri	Ri = Px Ri	Ri = Px ^ Ri
Ri = Px & Ri	Px = Py + data	Px = Py - data	Px = Py data	Px = Py ^ data
Px = Py & data	Ri = Px + data	Ri = Px - data	Ri = Px data	Ri = Px ^ data
Ri = Px & data	Px = [1 x 0 FD An]	-	-	-

6.4.8.4 Path Command

Path Command				
C-Code	BG	StopFG	StopBG	StopBG1
StopBG2	StopBG3	StopBG4	StopBG5	Subroutine
Label(Pathname)	Macro	-	-	-

6.4.8.5 Voice Command

Voice Command				
PlayV	PlayVS	WaitVN(Ch)	PauseV(Ch)	ResumeV(Ch)
StopV(Ch)	SBC_Loop_On	SBC_Loop_Off	ADPCM_Loop_On	ADPCM_Loop_Off
ADPCM_UpSampling	ADM_Loop_On	ADM_Loop_Off	ADM_UpSampling	
PCM_Loop_On	PCM_Loop_Off	ReadFileCountV	-	

6.4.8.6 Record Command

Record Command				
Record	RecordS	WaitRN	StopR	EraseR
EraseRS	WaitEN	-	-	-

6.4.8.7 Sentence Command

Sentence Command				
PlayS	WaitSN	PauseS(n)	ResumeS(n)	StopS(n)

6.4.8.8 SPIPlay Command

SPIPlay Command				
SPIPlay	SPIPlayS	SPIWaitN	SPIStop	SPIPause
SPIResume	SPIGetIndex(Result, SPIGroup)	-	-	-

6.4.8.9 SPI Flash Command

SPI Flash Command				
SPI_WREN	SPI_WRDIS	SPI_RDID	SPI_CE	SPI_SE
SPI_BE	SPI_DP	SPI_RDP	SPI_WRSR	SPI_RDSR
SPI_WRD	SPI_RDD	SPI_GetAddr	-	-

6.4.8.10 SPI Command

SPI Command				
SPI_CS On	SPI_CS Off	SPI_TX	SPI_RX	SPI_CLKDIV
SPI_CPOL	SPI_CPHA	-	-	-

6.4.8.11 Embedded Flash Command

Embedded Flash Command				
EF_SE	EF_WRD	EF_RDD	EF_GetAddr	-

6.4.8.12 Storage Command

Storage Command				
Storage_Save	-	-	-	-

6.4.8.13 Timer Command

Timer Command				
TMR_ON	TMR_OFF	-	-	-

6.4.8.14 MIDI Command

MIDI Command				
PlayM	PlayMS	WaitMN	PauseM	ResumeM

MIDI Command				
StopM	Instrument	M Chx Vol = n	M Chx Vol = Ri	Ri = M Chx Vol
Tempo + n	Tempo - n	Tempo++	Tempo--	TempoRst
ReadTempo(Ri)	Mute On(Ch)	Mute Off(Ch)	OKON On	OKON Off
OKON Play	MIDI Pitch	Mask On	Mask Off	ReadFileCountM
MIDI Loop On	MIDI Loop Off	-	-	-

6.4.8.15 Keyboard Command

Instrument Command				
InstNoteOn	InstNoteOff	InstNoteAllOff	DrumNoteOn	DrumNoteOff
NoteVibrato	Gliss	LongInst HoldTime		-
ShortInst HoldTime		KRecord	KRecordS	WaitKRN
StopKR	PlayK	PlayKS	WaitKN	StopK

6.4.8.16 Volume Command

Volume Command				
Vol Max	Vol Min	Vol = n	Vol = Ri	Vol++
Vol--	Ri = Vol	Px = Vol	Vol += n	Vol -= n
CHx Vol = n	PP Gain = n	PP Gain = Ri	PP Gain++	PP Gain--
Ri=PP Gain	PGA Gain = n	PGA Gain = Ri	Ri = PGA Gain	MixCtrl

6.4.8.17 TouchKey Command

TouchKey Command			
TouchKey Sensitivity	Enforce Calibrate Normal		Touchkey Count
Touchkey BGCount	-	-	-

6.4.8.18 Table Command

Table Command	
Table(TableName, VarX, VarY, VarR)	-

6.4.8.19 IR Command

IR Command

IR Command				
IR_TX=data	IR_TX(Ri)	IR_TX WaitN	IR_RX_ON	IR_RX_OFF
Ri = IR_RX	IR_RX = data?Path		IR_RX != data?Path	
IR_TX_Busy?Path		IR_Carrier_On	IR_Carrier_Off	-

6.4.8.20 I2C Command

I2C Command				
I2C_TX	I2C_RX	I2C_RX=data?Path		
I2C_RX!=Data?Path		I2C_Ack?Path	I2C_Start	I2C_Stop
I2C_MReadAck	I2C_MReadNack	I2C_Reset	-	-

6.4.8.21 UART Command

UART Command				
UART_TX	UART_TX WaitN	UART_RX	UART_RX=data?Path	
UART_RX!=data?Path		UART_TX_Busy?Path		-

6.4.8.22 Interrupt Command

Interrupt Command				
INT_ON	INT_OFF	-	-	-

6.4.8.23 Delay Command

Delay Command				
Delay(time)	Delay(Ri:Rj:Rk)	WaitDN(n)	StopD(n)	PauseD(n)
ResumeD(n)	SDelay(time)	-	-	-

6.4.8.24 Action Command

Action Command				
PlayA	PlayAS	WaitAN(Ch)	PauseA(Ch)	ResumeA(Ch)
StopA(Ch)	-	-	-	-

6.4.8.25 LED Strip Command

LED Strip Command

LED Strip Command				
LEDStr_Play	LEDStr_PlayS	LEDStr_Stop	LEDStr_Clear	LEDStr_Brightness
LEDSync_Play	LEDSync_PlayS	LEDSync_Stop	LEDSync_Clear	LEDSync_Brightness
LEDText_Play	LEDText_PlayS	LEDText_Stop	LEDText_Clear	LEDText_Brightness

6.4.8.26 QFID Command

QFID Command				
QFID_GroupID	QFID_TagId	QFID_TagInput	QFID_On	QFID_Off

6.4.8.27 WaveID Command

WaveID Command				
WaveID_TX	WaveID_RX_ON	WaveID_RX_OFF	WaveID_RX	-

6.4.8.28 Sound Localization Command

Sound Localization Command				
SL_On	SL_Off	SL_RX	-	-

6.4.8.29 Sound Detect Command

Sound Detect Command				
SoundDetect_On	SoundDetect_Off	-	-	-

6.4.8.30 Pitch Detect Command

Pitch Detect Command				
PitchDetect_On	PitchDetect_Off	ReadPitch	-	-

6.4.8.31 RFC Command

RFC Command				
RFC_On	RFC_Off	RFC_Level(Ri)	-	-

6.4.8.32 ADC Input Command

ADC Input Command				
ADC_Input	=	=	-	-

6.4.8.33 VR Command

VR Command				
VR State	VR_ON	VR_OFF	VR_VAD=n	VR_VAD_On
VR_VAD_Off	VRGC_Timeout_CLR		Ri = VR_HitScore	Ri = VR_HitID
VR>Loading	VT_Training	VT_Delete	VT_DeleteAll	VT_TrainingNum

6.4.8.34 Sound Effect Command

Sound Effect Command				
PitchChange	PitchChange_Off	SpeedChange	SpeedChange_Off	Robot1
Robot1_Off	Robot2	Robot2_Off	Robot3	Robot3_Off
Robot4	Robot4_Off	Echo	Ecbo_Off	Reverb
Reverb_off	Darth	Darth_Off	AnimalRoar	AnimalRoar_Off

6.4.8.35 Real Time Play Command

Real Time Play Command				
RT_Play	RT_Play_Off	RT_PitchChange	RT_PitchChange_Off	
RT_Robot1	RT_Robot1_Off	RT_Robot2	RT_Robot2_Off	RT_Robot3
RT_Robot3_Off	RT_Robot4	RT_Robot4_Off	RT_Echo	RT_Echo_Off
RT_Reverb	RT_Reverb_Off	RT_Ghost	RT_Ghost_Off	RT_Darth
RT_Darth_Off	RT_Chorus	RT_Chorus_Off	RT_Vol = n	RT_Vol = Var
Var = RT_Vol	-	-	-	-

6.4.8.36 Animaltalks Command

Animaltalks Command			
Animaltalks_On	Animaltalks_Off	Animaltalks_Record	Animaltalks_RecordS
Animaltalks_StopR	Animaltalks_Play	Animaltalks_PlayS	Animaltalks_Stop
Animaltalks_SetVoice		Animaltalks_LongSound_On	-
Animaltalks_LongSound_Off		-	-

6.4.8.37 Animalsings Command

Animalsings Command			
Animalsings_On	Animalsings_Off	Animalsings_Record	Animalsings_RecordS
Animalsings_StopR	Animalsings_Play	Animalsings_PlayS	Animalsings_Stop

Animalsings_SetVoice	Animalsings_LongSound_On	
Animalsings_LongSound_Off	Animalsings_NC_On	Animalsings_NC_Off
Animalsings_NC_Auto	-	-

6.4.8.38 I/O Expander Command

I/O Expander Command				
IoExp_Input_State	IoExp_Output_State	IoExp_Key_Clr	IoExp_Key_On	IoExp_Key_Off
IoExp_Sleep	IoExp_ErrId	IoExp_ReadInfo	IoExp_ReadSN	-
IoExp_SetInputPullHigh		IoExp_SetInputPullLow		
IoExp_SetInputFloating		IoExp_SetOutputHigh		
IoExp_SetOutputLow		-		

6.4.8.39 MISC Command

MISC Command				
Input_State	Output_State	Action_Mark_State	Wave_Mark_State	-
Melody_Mark_State		NoteOn_State	QFID_State	Key_CLR
Key_ON	Key_OFF	Stop	Pause(n)	Resume(n)
Audio_Loop_On		Audio_Loop_Off	NoiseFilter_On	NoiseFilter_Off
EQ_Filter	EQ_Filter_Off	AGC_On	AGC_Off	AutoSleep_On
AutoSleep_Off	Sleep	WDT_CLR	Repeat	Background
END	ReadRollingCode		Ri = LVD	Enforce_Wakeup
GetMicVol	Millis	Srand	-	-

6.4.8.40 Debug Command

Debug Command				
Printf	-	-	-	-

6.6 NX1 Audio Channel

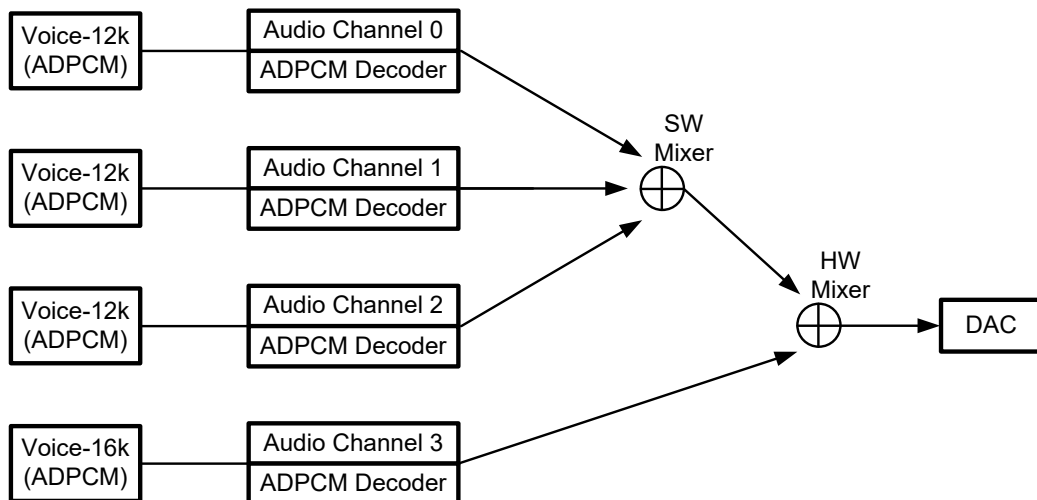
NX1 (uses C_Module as its kernel) supports eight Audio Channels: Audio Channel 0 ~ Audio Channel 7, the kernel program will correspond to the HW_CH0 and HW_CH1 of hardware channels automatically based on the sample rate of voice file. Q-Code can use two Audio Channels to play audio with different sampling rates at the same time, but it might cause playback error by playing 3 audio with different sample rates at the same time. If user uses voice record or sound effects (SpeedChange / PitchChange /

RobotX / Echo / Reverb / DARTH), the numbers of software supported channels will be reduced to 2.

The voice formats supported by Q-Code up to 2 sets of SBC-1, 2 sets of SBC-2, 8 sets of ADPCM, 8 sets of ADM, 3 sets of PCM, 1 set of CELP and MIDI decoding algorithm. Please note the number of decoding algorithm sets, if the played voice formats exceed the supported sets, the voice will not be played successfully. The application of multi-channel has to refer to the conditions of algorithm, please refer to [RAM Usage](#).

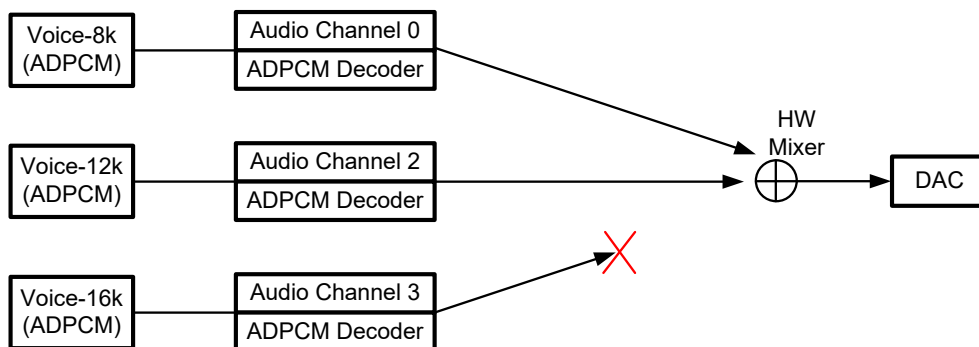
Ex. 4 channels ADPCM with 2 types of sample rate are simultaneous playback.

TR1: PlayVS(Ch0, \$v0), PlayVS(Ch1, \$v1), PlayVS(Ch2, \$v2), PlayV(Ch3, \$v3)



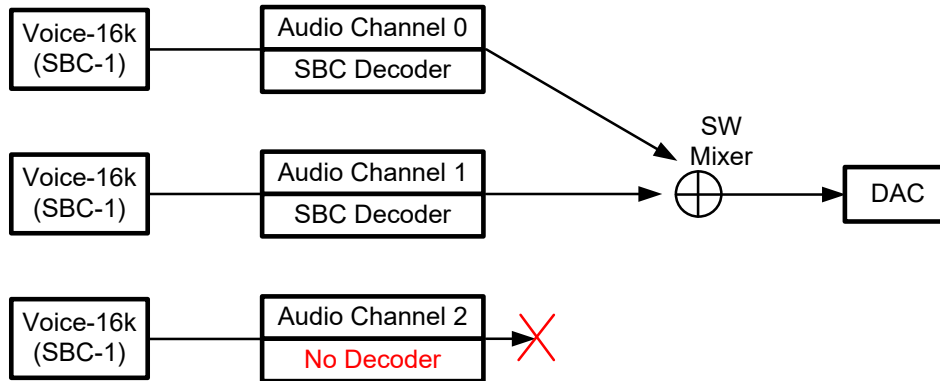
Ex.3 channels ADPCM with 3 types of sample rate cannot be played at the same time.

TR1: PlayVS(Ch0, \$v0), PlayVS(Ch1, \$v1), PlayV(Ch2, \$v2)



Ex. 3 channels SBC cannot be played at the same time.

TR1: PlayVS(Ch0, \$v0), PlayVS(Ch1, \$v1), PlayV(Ch2, \$v2)



6.7 RAM Resources

RAM will be reserved if specific function or command is used in *Q-Code*. The number of reserved RAMs will be different according to different functions or commands. The RAM will be occupied forward in sequence. The cross reference table of reserved RAMs and functions are shown in the following sections.

6.6.1 Command for NY4 RAM Resources

There are two pages of RAM, Page0 and Page1. There are 48 memory units for each page. Totally 86 registers are available for users. 76 memory units are available for users and the others are reserved by system.

Note: System will randomly configure in accordance with the function.

Q-Code Function	Description	Number of Occupied RAM
Reserved by system	Program flow control. The occupied numbers of NY4A are different from NY4B.	20 / 22
Action	When PlayA / PlayAS function is used, 9 RAMs will be used in each action channel. If user executes Action Compression function, each action channel will use one more RAM.	$(9+n) * ch$
Action Mark State	2 RAMs will be used when the setting is more than 15 states.	1~2
	Additional 1 RAM will be occupied if max path number is over 15.	1~2
BackGround	1 RAM will be occupied if Path number is more than 15 in background1.	1~2

Q-Code Function	Description	Number of Occupied RAM
	1 RAM will be occupied if Path number is more than 15 in background2.	1~2
Delay	When Delay function is used, 3 RAMs will be used in each layer and system will occupy 2 RAMs.	3*n+2
FD	The port using FD function will occupy 1 RAM.	1~2
Input State	2 RAMs will be used when the setting is more than 15 states.	1~2
IR	When IR Tx function is used, 1 RAM will be occupied if DATA Length increases by 4 bit.	1~4
	When IR Rx function is used, 1 RAM will be occupied if DATA Length increases by 4 bit.	1~4
Key Function	Each Matrix output pin up to a corresponding group of input ports will occupy 3 RAM. Each Direct port will occupy 3 RAM. When Debounce is more than 60ms, it will occupy 2 RAMs. System will occupy 2 RAMs.	(Key / 4)*3+3~4
Random	Additional 1 RAM will be occupied if max Random number is over 16.	3~4
Repeat	Repeat function is used in foreground.	1
	Repeat function is used in background1.	1
	Repeat function is used in background2.	1
Subroutine	4 RAMs will be occupied when first layer is called, and 4 more RAMs will be occupied if the second layer is called.	4 / 8
PausedD	When PausedD function is used, 3 RAMs will be used in each layer.	3*n
PlayVS	PlayVS function is used.	1
QIO	When QIO function is used, 1 RAM will be used and system will occupy 5 RAMs.	1*n+5
Wait	When WaitVN function is used, 1 RAM will be used and system will occupy 1 RAM.	1+1
	When WaitDN function is used, 1 RAM will be used and system will occupy 1 RAM.	1+1
	When WaitAN function is used, 2 RAMs will be used and system will occupy 1 RAM.	2+1
Wave Mark	1 RAM will be occupied when Wave Mark is used, and Additional 1 RAM will be occupied if Wave Mark State number is more then 15.	2~3

6.6.2 Command for NY5 RAM Resources

NY5: There are four pages of SRAM, Page0, Page1, Page2 and Page3 in NY5. There are 56 memory units for each page. Totally 224 registers are available, which 188 memory units are available for users and the others are reserved by system.

Note: System will randomly configure in accordance with the function.

Q-Code Function	Description	Number of occupied RAM
Reserved by system	Program flow control (NY5A / NY5B / NY5C*20 I/O; NY5C*24 I/O).	36 / 36 / 36 / 38
Action	When PlayA / PlayAS function is used, 10 RAMs will be used in each action channel. If user executes Action Compression function, each action channel will use one more RAM.	$(10+n)*ch$
	When PlayA / PlayAS function is used and InterruptService = Action, it will occupy additional 7 RAMs.	$(10+n)*ch+7$
Action Mark State	2 RAMs will be used when the setting is more than 15 states.	1~2
	Additional 1 RAM will be occupied if max path number is over 15.	1~2
BackGround	Additional 1 RAM will be occupied if Path number is more than 15 in background1.	1~2
	Additional 1 RAM will be occupied if Path number is more than 15 in background2.	1~2
Delay	When Delay function is used, 3 RAMs will be used in each layer and system will occupy 2 RAMs.	$3*n+2$
FD	The port using FD function will occupy 1 RAM.	1~6
Input State	2 RAMs will be used when the setting is more than 16 States.	1~2
IR	When IR Tx function is used, 1 RAM will be occupied if DATA Length increases by 4 bit.	1~4
	When IR Rx function is used, 1 RAM will be occupied if DATA Length increases by 4 bit.	1~4
Interrupt Path	Interrupt Path function is used.	8~12
Key Function	Each Matrix output pin up to a corresponding group of input ports will occupy 3 RAM. Each Direct port will occupy 3 RAM. When Debounce is more than 60ms, it will occupy 2 RAMs. System will occupy 2 RAMs.	$(Key / 4)*3+3\sim4$
Melody Mark	1 RAM will be occupied when Melody Mark is used, and	2~4

Q-Code Function	Description	Number of occupied RAM
	additional 1 RAM will be occupied if Wave Mark State number is more than 15.	
Note On	When Note On function is used, additional 1 RAM will be occupied if Note On State number is more than 15.	2~3
Paused	When Paused function is used, 3 RAMs will be used in each layer.	3*n
PauseDown / ResumeUp	When PauseDown/ResumeUp function is used, 4 RAMs will be used.	4
PauseV	PauseV function is used.	1
PlayM	Use PlayM + 4 channel PCT Mode.	99
	Use PlayM + 4 channel ADSR Mode (automatically open InterruptService = Melody)	117
	When using PlayM and InterruptService = Melody, it will occupy additional 7 RAMs.	7
PlayMS	When PlayMS function is used, in addition to the original PlayM, it needs to take additional 1 RAM.	1
PlayVS	PlayVS function is used in foreground.	1
	PlayVS function is used in background1.	1
	PlayVS function is used in background2.	1
QIO	When QIO function is used, 1 RAM will be occupied with each port using QIO function. System will occupy 6 RAMs.	1*n+6
	When QIO function is used and InterruptService = QIO, it will occupy additional 10 RAMs.	(1*n+6)+10
Random	1 RAM will be occupied when Random is used, and additional 1 RAM will be occupied if Random number is more than 16.	3~4
Repeat	Repeat function is used in foreground.	1
	Repeat function is used in background1.	1
	Repeat function is used in background2.	1
Single Play	Single Playfunction is used.	2
Subroutine	Provides nested stacking feature in Subroutine section. 4 RAMs will be occupied when first layer is called, and 4 more RAMs will be occupied if the second layer is called.	4 / 8
Wait	When WaitVN function is used, 2 RAMs will be used and system will occupy 1 RAM.	2+1
	When WaitDN function is used, 1 RAM will be used and system	1+1

Q-Code Function	Description	Number of occupied RAM
	will occupy 1 RAM.	
	When WaitAN function is used, 2 RAMs will be used and system will occupy 1 RAM.	2+1
	When WaitMN function is used, 1 RAM will be used and system will occupy 1 RAM.	1+1
Wave Mark	1 RAM will be occupied when Wave Mark is used, and additional 1 RAM will be occupied if Wave Mark State number is more than 15.	2~3

6.6.3 NY5+Memory Allocation

There are 215 registers are available. *Q-Code* will allocate the needed number of RAM by the used functions, user can close unnecessary functions to increase the available number of RAM. After executing Build, the used number of RAM will be listed in the window for Build Message.

6.6.4 NY6 Memory Allocation

There are 6 pages: Page0 ~ Page5. There are 56 memory units for each page. Totally 336 registers are available. *Q-Code* will allocate the needed number of RAM by the used functions, user can close unnecessary functions to increase the available number of RAM. After executing Build, the used number of RAM will be listed in the window for Build Message.

6.6.5 NY7 Memory Allocation

There are two pages of SRAM, Page0 and Page1. There are 224 memory units for each page. Totally 448 registers are available, which 395 memory units are available for users and the others are reserved by system. *Q-Code* will allocate the needed number of RAM by the used functions, user can close unnecessary functions to increase the available number of RAM. After executing Build, the used number of RAM will be listed in the window for Build Message.

6.6.6 NY9T Memory Allocation

For NY9T001A / NY9T004A, there is one page of Page0. There are 48 RAM, 41 RAM are available for user and the other are reserved by system.

Note: System will randomly configure in accordance with the function.

Q-Code function	Description	Number of occupied RAM
Reserved	Program flow control.	7
AutoJudge_Calibrate	When AutoJudge_Calibrate function is used, 1 RAM will be occupied.	1
Enforce_Calibrate	When Enforce_Calibrate function is used, 1 RAM will be occupied.	1
Process	When Process function is used in foreground, 4 RAMs will be occupied.	4
	When Process function is used in background1, 4 RAMs will be occupied.	4
	When Process function is used in background2, 4 RAMs will be occupied.	4
IO	The occupied RAM numbers are different that based on the different numbers of output port of Body.	1*n
Action	When PlayA / PlayAS function is used, 9 RAMs will be used in each channel.	9
	Each channel occupies 1 RAM when using the Period function.	1
	Each channel occupies 6 RAM when using the Extension function.	6
	When the Loop function is used, channel 1~4 and channel 5~8 will occupy 1 RAM.	1~2
	When VIO is playing, Q-Code will base on the numbers of output port to occupy RAM.	1*n
	When PlayA / PlayAS function is used and set Action = INT, the extra 6~7 RAM will be used.	8*ch+(6~7)
PWM-IO	When PlayPWM / PlayPWMS / PWMOOut is used, 1 RAM will be occupied.	1
BackGround	1 additional RAM will be occupied if Path number is more than 15 in backGround1.	1~2
	1 additional RAM will be occupied if Path number is more than 15 in backGround2.	1~2
Delay	When Delay function is used, 3 RAMs will be used in each layer and system will occupy 2 RAMs.	3*n+2
Input State	2 RAMs will be occupied when the setting is more than 15 States.	1~2
Key	Each Input Port up to 4 corresponding keys, in principle, each 4 keys will occupy 1 RAM. Direct and Matrix Keys are calculated separately.	(Key/4)+2
	When Debounce is more than 60ms, it will occupy 2 RAMs.	1~2

Q-Code function	Description	Number of occupied RAM
Touch-Key	Each Input Port corresponds to 4 keys as most. For every 4 keys, 1 RAM will be occupied.	(Key/4)+3
Random	1 additional RAM will be occupied if Path number is more than 16.	1~2
Repeat	When Repeat function is used in foreground.	1
	When Repeat function is used in background1.	1
	When Repeat function is used in background2.	1
Subroutine	When Subroutine function is used, 4 RAMs will be used.	4
Wait	When WaitDN function is used, 1 RAM will be used and system will occupy 1 RAM.	1+1
	When WaitAN function is used, 1 RAMs will be used and system will occupy 1 RAM.	1+1
	When WaitPN function is used, 1 RAM will be used and system will occupy 1 RAM.	1+1
Special Path	When 4ms path is used, system will occupy 1 RAM.	1
	When Enforce_Calibrate path is used, system will occupy 1 RAM.	
Radj	When the Radj related commands are used, system will occupy 1 RAM.	1

For NY9T008A / NY9T016A, there are two pages of Page0 and Page1. There are 96 RAM for each page, 89 RAM are available for users and the others are reserved by system.

Note: System will randomly configure in accordance with the function.

Q-Code function	Description	Number of occupied RAM
Reserved	Program flow control.	7
AutoJudge_Calibrate	When AutoJudge_Calibrate function is used, 1 RAM will be occupied.	1
Enforce_Calibrate	When Enforce_Calibrate function is used, 1 RAM will be occupied.	1
Process	When Process function is used in foreground, 4 RAMs will be occupied.	4
	When Process function is used in background1, 4 RAMs will be occupied.	4

Q-Code function	Description	Number of occupied RAM
	When Process function is used in background2, 4 RAMs will be occupied.	4
IO	The occupied RAM numbers are different that based on the different numbers of output port of Body.	1*n
Arithmetic Logic	When user uses multiplication and division, 2 RAM will be occupied.	2
Action	When PlayA / PlayAS function is used, 9 RAMs will be used in each channel.	9
	Each channel occupies 1 RAM when using the Period function.	1
	Each channel occupies 6 RAM when using the Extension function.	6
	When the Loop function is used, channel 1~4 and channel 5~8 will occupy 1 RAM.	1~2
	When VIO is playing, Q-Code will base on the numbers of output port to occupy RAM.	1*n
	When PlayA / PlayAS function is used and set Action = INT, the extra 7~8 RAM will be used.	9*ch+(7~8)
PWM-IO	When PlayPWM / PlayPWMS / PWMOut is used, 1 RAM will be occupied.	1
BackGround	1 additional RAM will be occupied if Path number is more than 15 in backGround1.	1~2
	1 additional RAM will be occupied if Path number is more than 15 in backGround2.	1~2
Delay	When Delay function is used, 3 RAMs will be used in each layer and system will occupy 2 RAMs.	3*n+2
Input State	2 RAMs will be occupied when the setting is more than 15 States.	1~2
Key	Each Input Port up to 4 corresponding keys, in principle, each 4 keys will occupy 1 RAM. Direct and Matrix Keys are calculated separately.	(Key/4)+2
	When Debounce is more than 60ms, it will occupy 2 RAMs.	1~2
Touch-Key	Each Input Port corresponds to 4 keys as most. For every 4keys, 1 RAM will be occupied.	(Key/4)+3
Random	1 additional RAM will be occupied if Path number is more than 16.	1~2
Repeat	When Repeat function is used in foreground.	1

Q-Code function	Description	Number of occupied RAM
	When Repeat function is used in background1.	1
	When Repeat function is used in background2.	1
Subroutine	When Subroutine function is used, 4 RAMs will be used.	4
Wait	When WaitDN function is used, 1 RAM will be used and system will occupy 1 RAM.	1+1
	When WaitAN function is used, 2 RAMs will be used and system will occupy 1 RAM.	2+1
	When WaitPN function is used, 1 RAM will be used and system will occupy 1 RAM.	1+1
Special Path	When 4ms path is used, system will occupy 1 RAM.	1
	When Enforce_Calibrate path is used, system will occupy 1 RAM.	
Serial Control	When Serial Control is used, system will occupy 4 RAM.	4
Radj	When the Radj related commands are used, system will occupy 1 RAM.	1

6.8 Corresponding Table of Ri and Xi

The corresponding table of Ri (4-bit RAM) and Xi (8-bit RAM) is shown below.

Ri	Xi	Ri	Xi	Ri	Xi	Ri	Xi	Ri	Xi	Ri	Xi	Ri	Xi	Ri	Xi
R0	X0L	R1	X0H	R2	X1L	R3	X1H	R4	X2L	R5	X2H	R6	X3L	R7	X3H
R8	X4L	R9	X4H	R10	X5L	R11	X5H	R12	X6L	R13	X6H	R14	X7L	R15	X7H
R16	X8L	R17	X8H	R18	X9L	R19	X9H	R20	X10L	R21	X10H	R22	X11L	R23	X11H
R24	X12L	R25	X12H	R26	X13L	R27	X13H	R28	X14L	R29	X14H	R30	X15L	R31	X15H
R32	X16L	R33	X16H	R34	X17L	R35	X17H	R36	X18L	R37	X18H	R38	X19L	R39	X19H
R40	X20L	R41	X20H	R42	X21L	R43	X21H	R44	X22L	R45	X22H	R46	X23L	R47	X23H
R48	X24L	R49	X24H	R50	X25L	R51	X25H	R52	X26L	R53	X26H	R54	X27L	R55	X27H
R56	X28L	R57	X28H	R58	X29L	R59	X29H	R60	X30L	R61	X30H	R62	X31L	R63	X31H
R64	X32L	R65	X32H	R66	X33L	R67	X33H	R68	X34L	R69	X34H	R70	X35L	R71	X35H
R72	X36L	R73	X36H	R74	X37L	R75	X37H	R76	X38L	R77	X38H	R78	X39L	R79	X39H
R80	X40L	R81	X40H	R82	X41L	R83	X41H	R84	X42L	R85	X42H	R86	X43L	R87	X43H
R88	X44L	R89	X44H	R90	X45L	R91	X45H	R92	X46L	R93	X46H	R94	X47L	R95	X47H
R96	X48L	R97	X48H	R98	X49L	R99	X49H	R100	X50L	R101	X50H	R102	X51L	R103	X51H
R104	X52L	R105	X52H	R106	X53L	R107	X53H	R108	X54L	R109	X54H	R110	X55L	R111	X55H
R112	X56L	R113	X56H	R114	X57L	R115	X57H	R116	X58L	R117	X58H	R118	X59L	R119	X59H
R120	X60L	R121	X60H	R122	X61L	R123	X61H	R124	X62L	R125	X62H	R126	X63L	R127	X63H
R128	X64L	R129	X64H	R130	X65L	R131	X65H	R132	X66L	R133	X66H	R134	X67L	R135	X67H

Ri	Xi	Ri	Xi	Ri	Xi	Ri	Xi	Ri	Xi	Ri	Xi	Ri	Xi	Ri	Xi
R136	X68L	R137	X68H	R138	X69L	R139	X69H	R140	X70L	R141	X70H	R142	X71L	R143	X71H
R144	X72L	R145	X72H	R146	X73L	R147	X73H	R148	X74L	R149	X74H	R150	X75L	R151	X75H
R152	X76L	R153	X76H	R154	X77L	R155	X77H	R156	X78L	R157	X78H	R158	X79L	R159	X79H
R160	X80L	R161	X80H	R162	X81L	R163	X81H	R164	X82L	R165	X82H	R166	X83L	R167	X83H
R168	X84L	R169	X84H	R170	X85L	R171	X85H	R172	X86L	R173	X86H	R174	X87L	R175	X87H
R176	X88L	R177	X88H	R178	X89L	R179	X89H	R180	X90L	R181	X90H	R182	X91L	R183	X91H
R184	X92L	R185	X92H	R186	X93L	R187	X93H	R188	X94L	R189	X94H	R190	X95L	R191	X95H
R192	X96L	R193	X96H	R194	X97L	R195	X97H	R196	X98L	R197	X98H	R198	X99L	R199	X99H
R200	X100L	R201	X100H	R202	X101L	R203	X101H	R204	X102L	R205	X102H	R206	X103L	R207	X103H
R208	X104L	R209	X104H	R210	X105L	R211	X105H	R212	X106L	R213	X106H	R214	X107L	R215	X107H
R216	X108L	R217	X108H	R218	X109L	R219	X109H	R220	X110L	R221	X110H	R222	X111L	R223	X111H
R224	X112L	R225	X112H	R226	X113L	R227	X113H	R228	X114L	R229	X114H	R230	X115L	R231	X115H
R232	X116L	R233	X116H	R234	X117L	R235	X117H	R236	X118L	R237	X118H	R238	X119L	R239	X119H
R240	X120L	R241	X120H	R242	X121L	R243	X121H	R244	X122L	R245	X122H	R246	X123L	R247	X123H
R248	X124L	R249	X124H	R250	X125L	R251	X125H	R252	X126L	R253	X126H	R254	X127L	R255	X127H
R256	X128L	R257	X128H	R258	X129L	R259	X129H	R260	X130L	R261	X130H	R262	X131L	R263	X131H
R264	X132L	R265	X132H	R266	X133L	R267	X133H	R268	X134L	R269	X134H	R270	X135L	R271	X135H
R272	X136L	R273	X136H	R274	X137L	R275	X137H	R276	X138L	R277	X138H	R278	X139L	R279	X139H
R280	X140L	R281	X140H	R282	X141L	R283	X141H	R284	X142L	R285	X142H	R286	X143L	R287	X143H
R288	X144L	R289	X144H	R290	X145L	R291	X145H	R292	X146L	R293	X146H	R294	X147L	R295	X147H
R296	X148L	R297	X148H	R298	X149L	R299	X149H	R300	X150L	R301	X150H	R302	X151L	R303	X151H
R304	X152L	R305	X152H	R306	X153L	R307	X153H	R308	X154L	R309	X154H	R310	X155L	R311	X155H
R312	X156L	R313	X156H	R314	X157L	R315	X157H	R316	X158L	R317	X158H	R318	X159L	R319	X159H
R320	X160L	R321	X160H	R322	X161L	R323	X161H	R324	X162L	R325	X162H	R326	X163L	R327	X163H
R328	X164L	R329	X164H	R330	X165L	R331	X165H	R332	X166L	R333	X166H	R334	X167L	R335	X167H
R336	X168L	R337	X168H	R338	X169L	R339	X169H	R340	X170L	R341	X170H	R342	X171L	R343	X171H
R344	X172L	R345	X172H	R346	X173L	R347	X173H	R348	X174L	R349	X174H	R350	X175L	R351	X175H
R352	X176L	R353	X176H	R354	X177L	R355	X177H	R356	X178L	R357	X178H	R358	X179L	R359	X179H
R360	X180L	R361	X180H	R362	X181L	R363	X181H	R364	X182L	R365	X182H	R366	X183L	R367	X183H
R368	X184L	R369	X184H	R370	X185L	R371	X185H	R372	X186L	R373	X186H	R374	X187L	R375	X187H
R376	X188L	R377	X188H	R378	X189L	R379	X189H	R380	X190L	R381	X190H	R382	X191L	R383	X191H
R384	X192L	R385	X192H	R386	X193L	R387	X193H	R388	X194L	R389	X194H	R390	X195L	R391	X195H
R392	X196L	R393	X196H	-	-	-	-	-	-	-	-	-	-	-	-

6.9 Sample Rate Table

6.8.1 NY4 / NY5 Sample Rate Table

When using PlayV to play a Wave, there are some parameters needs to determine:

$$TM = (F_{TCS} / F_{SR}) - 1$$

TM: Timer value.

F_{TCS} : System frequency (1MHz when playing voice).

F_{SR} : Playback speed.

Sample Rate Table:

TM(HEX)	SR(Hz)	TM(HEX)	SR(Hz)	TM(HEX)	SR(Hz)	TM(HEX)	SR(Hz)	TM(HEX)	SR(Hz)
F9	4000.0	CB	4901.9	9D	6329.1	6F	8928.5	41	15151.
F8	4016.0	CA	4926.1	9C	6369.4	6E	9009.0	40	15384.
F7	4032.2	C9	4950.5	9B	6410.2	6D	9090.9	3F	15625.
F6	4048.5	C8	4975.1	9A	6451.6	6C	9174.3	3E	15873.
F5	4065.0	C7	5000.0	99	6493.5	6B	9259.2	3D	16129.
F4	4081.6	C6	5025.1	98	6535.9	6A	9345.7	3C	16393.
F3	4098.3	C5	5050.5	97	6578.9	69	9433.9	3B	16666.
F2	4115.2	C4	5076.1	96	6622.5	68	9523.8	3A	16949.
F1	4132.2	C3	5102.0	95	6666.6	67	9615.3	39	17241.
F0	4149.3	C2	5128.2	94	6711.4	66	9708.7	38	17543.
EF	4166.6	C1	5154.6	93	6756.7	65	9803.9	37	17857.
EE	4184.1	C0	5181.3	92	6802.7	64	9900.9	36	18181.
ED	4201.6	BF	5208.3	91	6849.3	63	10000.	35	18518.
EC	4219.4	BE	5235.6	90	6896.5	62	10101.	34	18867.
EB	4237.2	BD	5263.1	8F	6944.4	61	10204.	33	19230.
EA	4255.3	BC	5291.0	8E	6993.0	60	10309.	32	19607.
E9	4273.5	BB	5319.1	8D	7042.2	5F	10416.	31	20000.
E8	4291.8	BA	5347.5	8C	7092.2	5E	10526.	30	20408.
E7	4310.3	B9	5376.3	8B	7142.8	5D	10638.	2F	20833.
E6	4329.0	B8	5405.4	8A	7194.2	5C	10752.	2E	21276.
E5	4347.8	B7	5434.7	89	7246.3	5B	10869.	2D	21739.
E4	4366.8	B6	5464.4	88	7299.2	5A	10989.	2C	22222.
E3	4385.9	B5	5494.5	87	7352.9	59	11111.1	2B	22727.
E2	4405.2	B4	5524.8	86	7407.4	58	11235.	2A	23255.
E1	4424.7	B3	5555.5	85	7462.6	57	11363.	29	23809.
E0	4444.4	B2	5586.5	84	7518.8	56	11494.	28	24390.
DF	4464.2	B1	5617.9	83	7575.7	55	11627.	27	25000.
DE	4484.3	B0	5649.7	82	7633.5	54	11764.	26	25641.
DD	4504.5	AF	5681.8	81	7692.3	53	11904.	25	26315.
DC	4524.8	AE	5714.2	80	7751.9	52	12048.	24	27027.
DB	4545.4	AD	5747.1	7F	7812.5	51	12195.	23	27777.
DA	4566.2	AC	5780.3	7E	7874.0	50	12345.	22	28571.
D9	4587.1	AB	5813.9	7D	7936.5	4F	12500.	21	29411.

TM(HEX)	SR(Hz)	TM(HEX)	SR(Hz)	TM(HEX)	SR(Hz)	TM(HEX)	SR(Hz)	TM(HEX)	SR(Hz)
D8	4608.2	AA	5847.9	7C	8000.0	4E	12658.	20	30303.
D7	4629.6	A9	5882.3	7B	8064.5	4D	12820.	1F	31250.
D6	4651.1	A8	5917.1	7A	8130.0	4C	12987.	1E	32258.
D5	4672.9	A7	5952.3	79	8196.7	4B	13157.	1D	33333.
D4	4694.8	A6	5988.0	78	8264.4	4A	13333.	1C	34482.
D3	4716.9	A5	6024.1	77	8333.3	49	13513.	1B	35714.
D2	4739.3	A4	6060.6	76	8403.3	48	13698.	1A	37037.
D1	4761.9	A3	6097.5	75	8474.5	47	13888.	19	38461.
D0	4784.6	A2	6134.9	74	8547.0	46	14084.	18	40000.
CF	4807.6	A1	6172.8	73	8620.6	45	14285.	17	41666.
CE	4830.9	A0	6211.1	72	8695.6	44	14492.	16	43478.
CD	4854.3	9F	6250.0	71	8771.9	43	14705.	15	45454.
CC	4878.0	9E	6289.3	70	8849.5	42	14925.	14	47619.

Ex. Use RAM to set playback speed.

TR1: X0=0x63, PlayV(\$V0,X0)

; Playback sample rate as 10KHz.

TR2: X0=0x7C, PlayV(\$V0,X0)

; Playback sample rate as 8KHz.

6.10 Unusable Commands for Special Path

The special path: 512us, 1ms, 2ms, 4ms, 8ms, 500ms, 1sec, 4sec, Int_256us, Int_512us, Int_1ms, and interrupt don't support the following commands.

- **Arithmetic Logic Command:** Multiplication commands, division commands.
- **Condition Jump Command:** CheckSum?Path.
- **IO Command:** Px.n=1KHz(sec).
- **Path Command:** StopFG, BG(BG1,BG2), StopBG, StopBG1, StopBG2, Macro, Subroutine, BreakFG.
- **Voice Command:** All Voice commands.
- **Sentence Command:** All Sentence commands.
- **SPI Command:** All SPI commands.
- **Melody Command:** All Melody commands.
- **Keyboard Command:** All Keyboard commands.
- **Volume Command:** All Volume commands.
- **Touchkey Command:**
 - Ri=Touchkey command is unrestricted.
 - The rest Touchkey commands, except 500ms / 1sec / 4sec path.

- **Table Command:** All Table commands.
- **IR Command:** All IR commands.
- **SC_RX Command:** All SC_RX commands.
- **SC_TX Command:** All SC_TX commands.
- **PWM Command:** All PWMcommands, except NY9T 500ms / 1sec / 4sec path.
- **Delay Command:** All Delay commands.
- **Action Command:** All Action commands.
- **QFID Command:** All QFID commands.
- **MISC Command:** All MISC commands.
 - SW_Reset: Except NYT9T 500ms / 1sec / 4sec path.

6.11 Serial Control Protocol

User can set the NY9T as the keys of MCUs. Q-Code provides 3 kinds of transfer protocols: SPI_Like, NY3 Serial_Trigger and IR_Trigger. User can set to automatically detect or specific transfer protocol.

SPI_Like: Use 2 pins to achieve a SPI-like communication protocol.

NY3 Serial_Trigger: User can connect NY3 via the serial signal, and NY9T acted as NY3 trigger buttons.

Please refer to NY3C/3D series specification for the detailed connection usage.

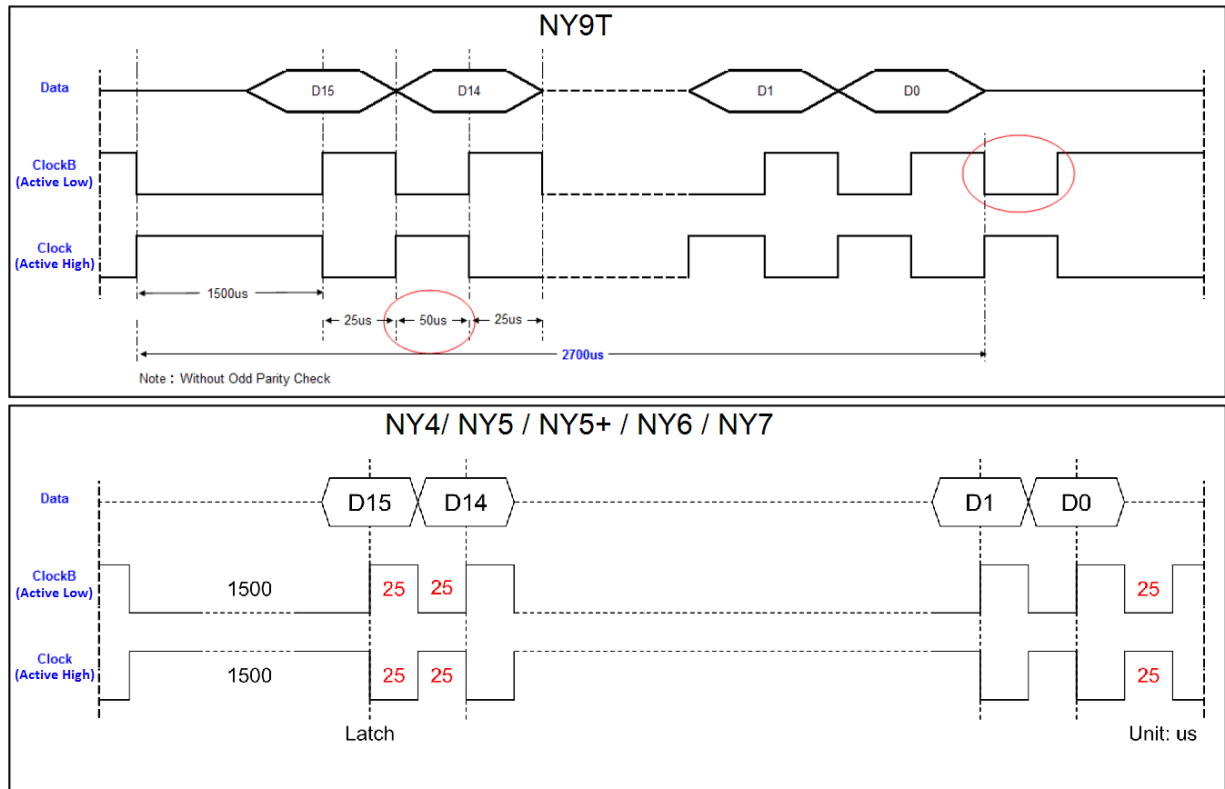
IR_Trigger: Use a pin to simulate IR transmission signal, and can be directly used to connect current NY4/5/7 series with IR communication.

AutoDetect: Using setting of 3 pins to decide which transfer protocol NY9T uses. Auto detect pin settings are as below table.

AutoDetect Mode			
Mode	Pin-3	Pin-2	Pin-1
IR_Trigger	X	VDD	X
NY3 Serial_Trigger	Pin-3 connected to Pin-1	X	Pin-3 connected to Pin-1
SPI_Like	No Connected	No Connected	No Connected

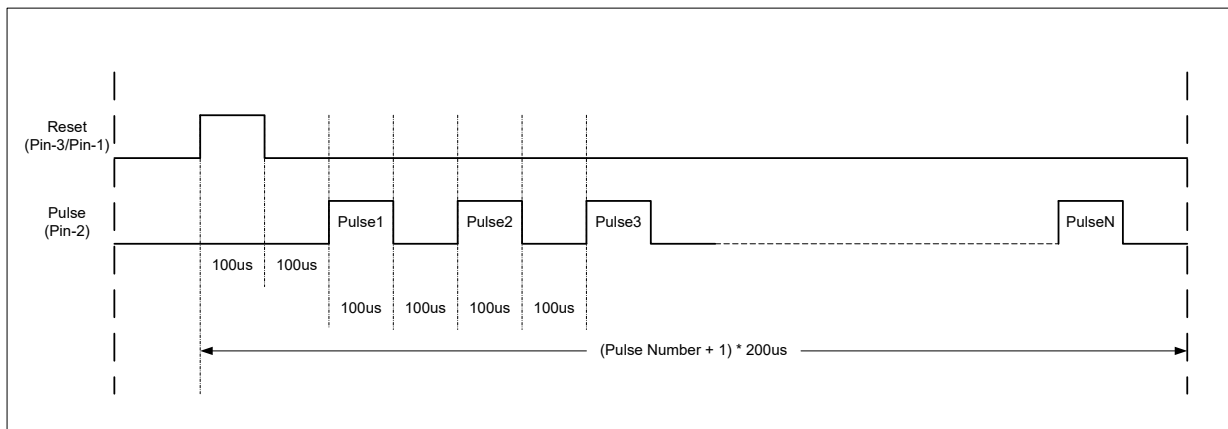
The timing charts of Serial Control transfer protocol are shown below.

SPI_Like Timing:

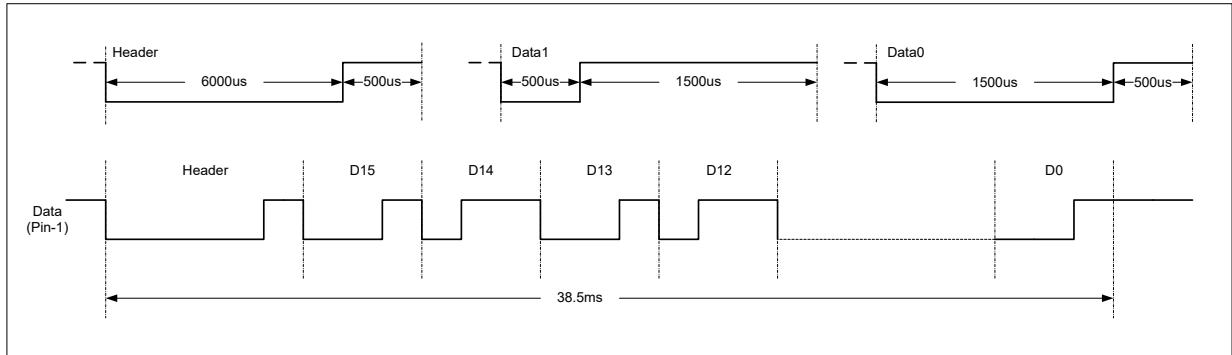


Note: 25us in above diagram may vary, range 25us ~ 100us.

NY3 Serial_Trigger Timing:



IR_Trigger (Active Low):



6.12 NY5 / NY5+ Volume Mapping

When converting NY5 .qc to NY5+, due to IC hardware differences, setting same volume value results in different volume output. Following tables show the mapping of volume value for near output between NY5 / NY5+.

Voltage = 3.0V & PWM Current = Normal																
	Vol															
NY5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NY5+	2	5	7	8	9	10	10	11	12	12	12	13	13	13	14	15

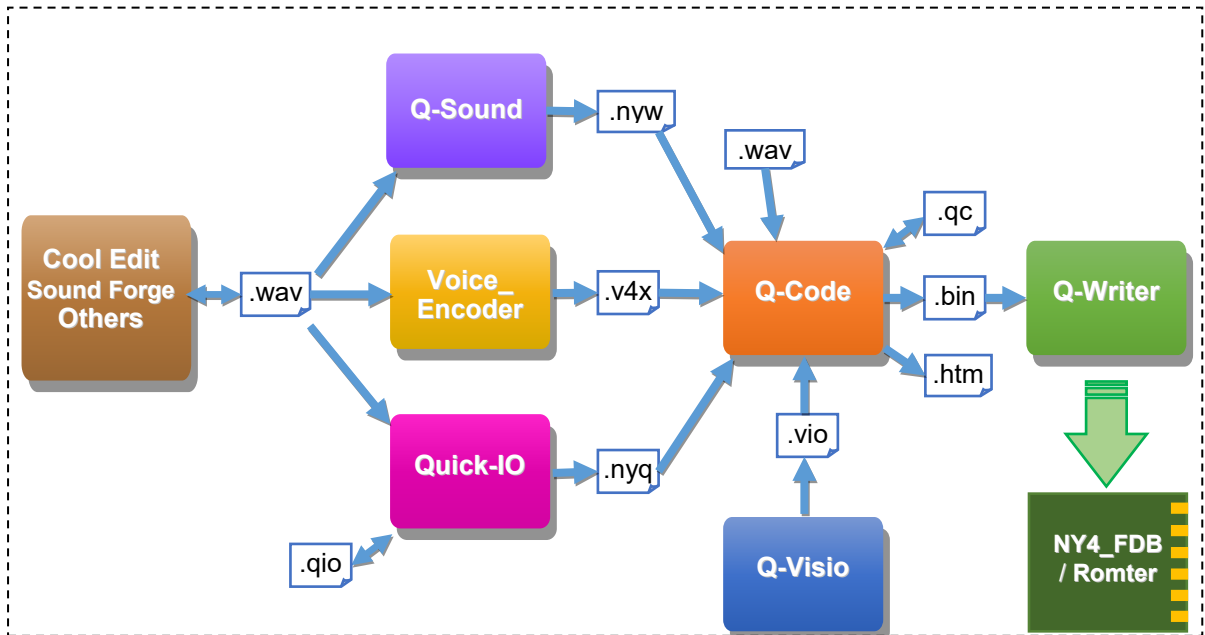
Voltage = 3.0V & PWM Current = Large																
	Vol															
NY5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NY5+	4	7	10	11	12	12	13	13	14	14	15	15	15	15	15	15

Voltage = 4.5V & PWM Current = Normal																
	Vol															
NY5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NY5+	3	6	6	7	8	9	9	10	10	10	10	11	12	12	13	13

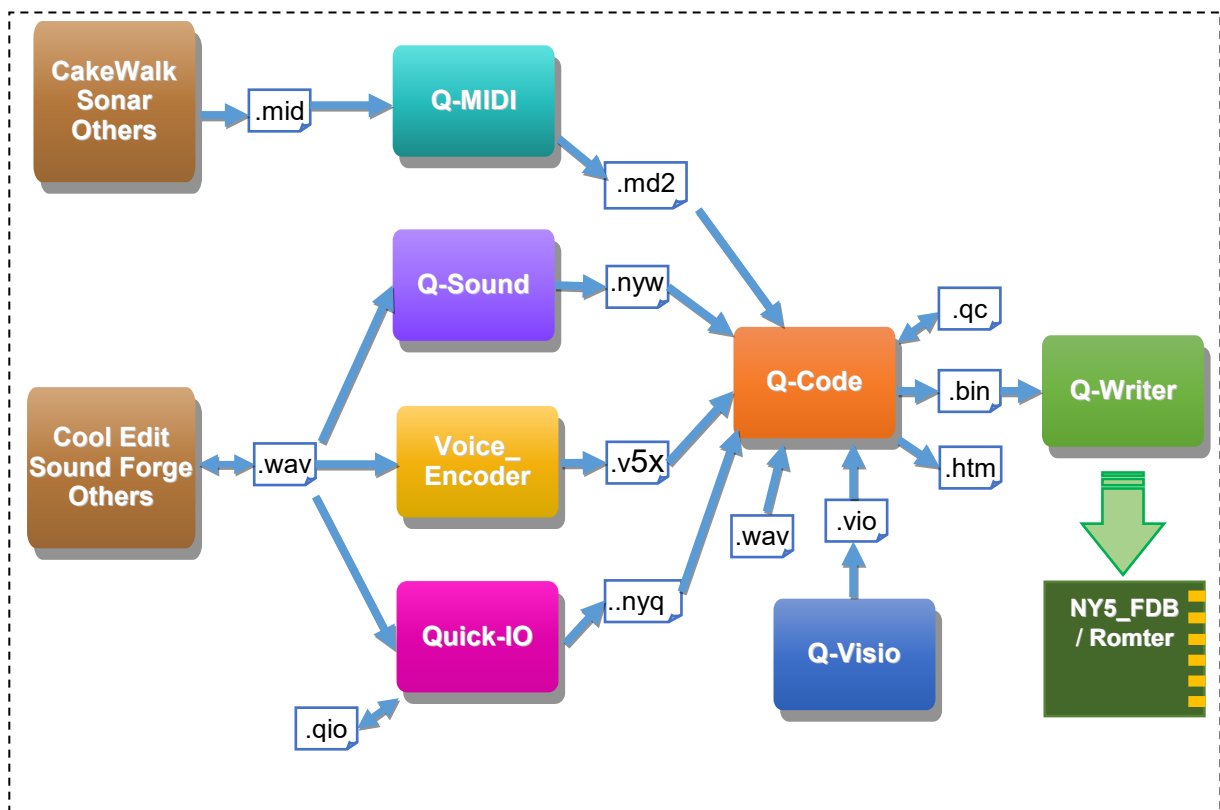
Voltage = 4.5V & PWM Current = Large																
	Vol															
NY5	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
NY5+	3	6	7	8	9	10	11	12	13	14	15	15	15	15	15	15

6.13 Development Flowchart of Q-Code

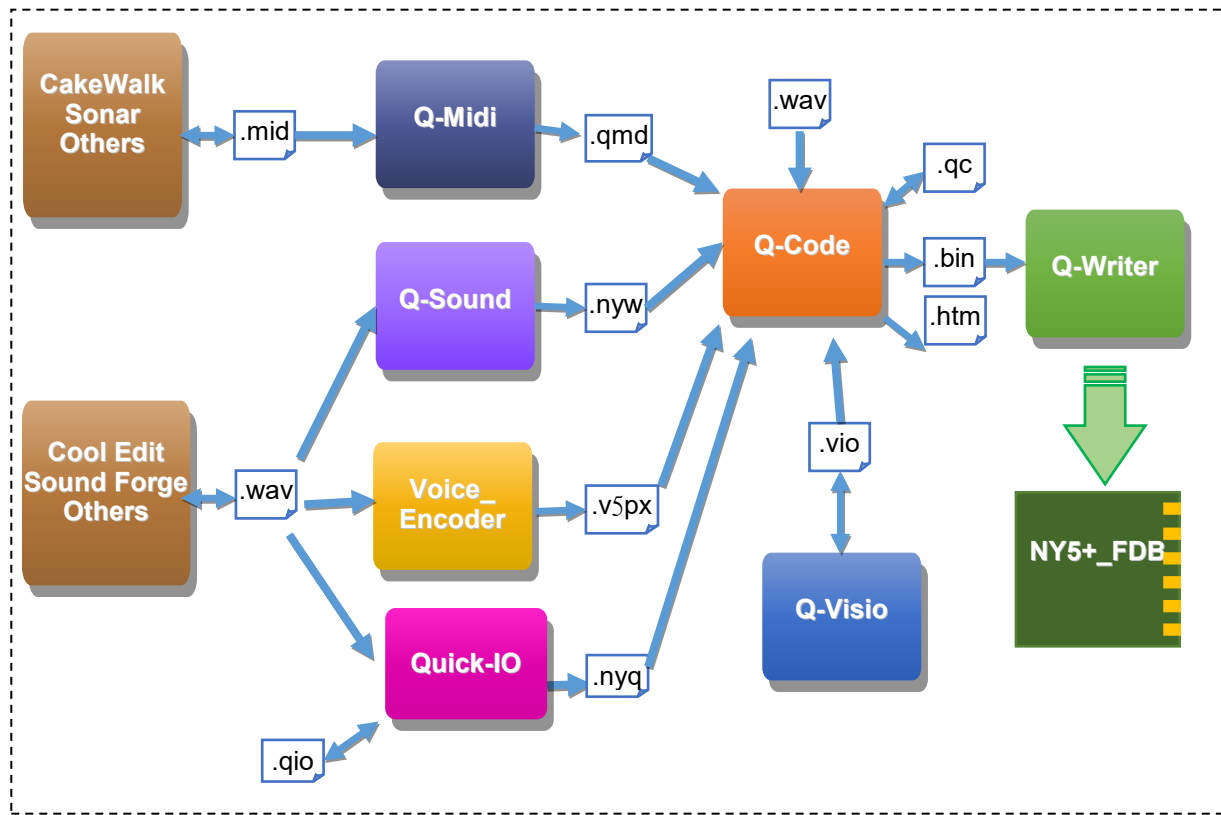
6.12.1 NY4 Flowchart



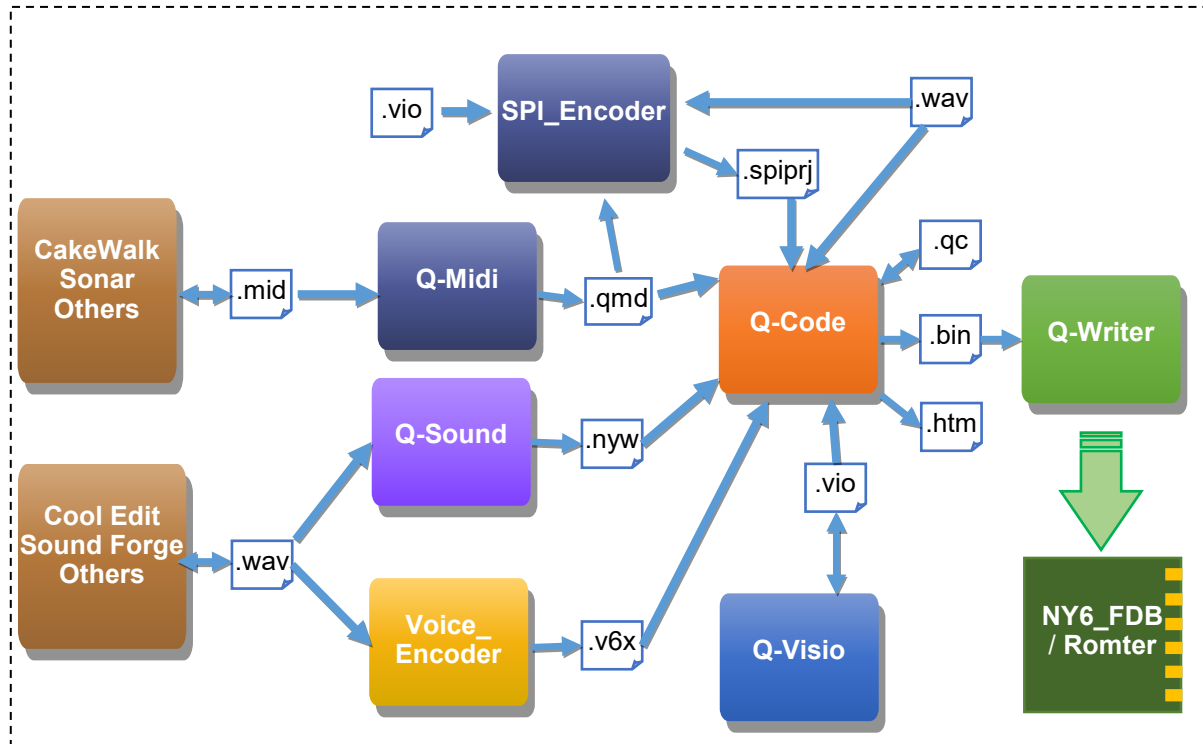
6.12.2 NY5 Flowchart



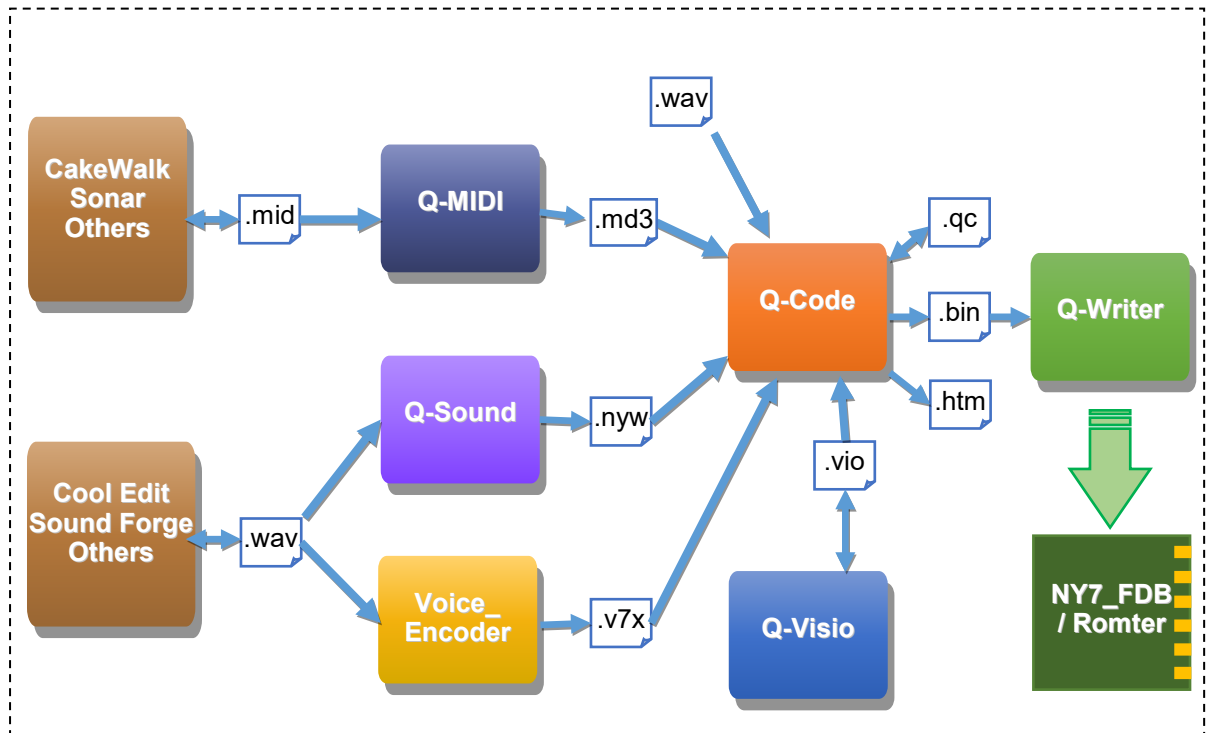
6.12.3 NY5+ Flowchart



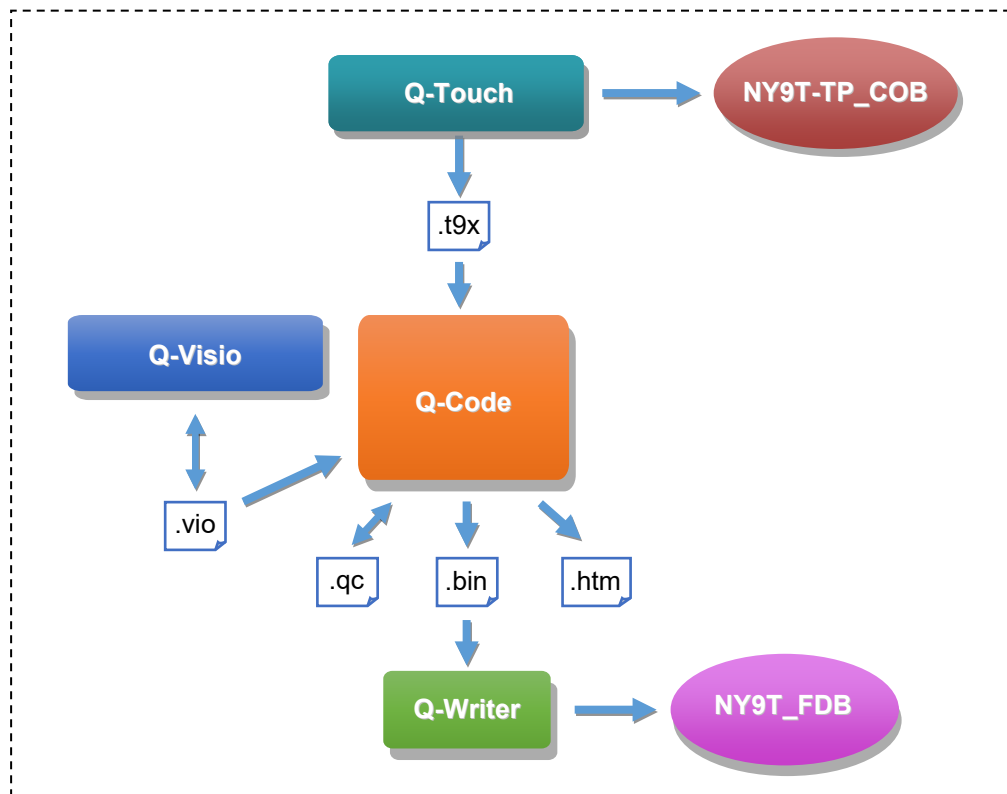
6.12.4 NY6 Flowchart



6.12.5 NY7 Flowchart



6.12.6 NY9T Flowchart



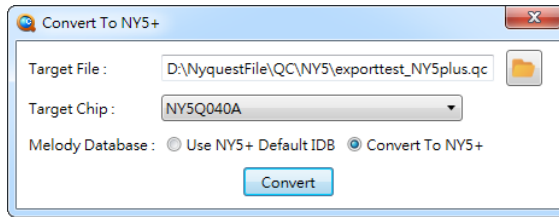
```
graph TD
    subgraph ExternalApps
        Cyberon[Cyberon CSMT]
        CakeWalk[CakeWalk Sonar Others]
        CoolEdit[Cool Edit Sound Forge Others]
    end

    subgraph InternalModules
        Q_MIDI[Q-MIDI]
        Q_Sound[Q-Sound]
        Voice_Encoder[Voice_Encoder]
        Q_Visio[Q-Visio]
        Q_Code[Q-Code]
        Q_Writer[Q-Writer]
    end

    Cyberon -- ".cvr" --> Q_Code
    Q_Code -- ".wv" --> Cyberon
    Q_Code -- ".spi.prj" --> SPI_Encoder[SPI_Encoder]
    SPI_Encoder -- ".vnr" --> Q_Code
    SPI_Encoder -- ".mid" --> Q_MIDI
    Q_MIDI -- ".mid" --> CakeWalk
    Q_MIDI -- ".qmd" --> Q_Code
    Q_Sound -- ".nyw" --> Q_Code
    CoolEdit -- ".wav" --> Q_Sound
    CoolEdit -- ".wav" --> Voice_Encoder
    Voice_Encoder -- ".vnx" --> Q_Code
    Q_Visio -- ".vnr" --> Q_Code
    Q_Code -- ".htm" --> Q_Visio
    Q_Code -- ".bin" --> Q_Writer
    Q_Code -- "_spi.bin" --> Q_Writer
    Q_Writer --> NX1_FDB[NX1_FDB]
```

6.14 Convert To NY5+

The following dialog will appear by clicking [File].



Target File: Set target file path.

Target Chip: Set target chip. Try to choose an IC whose pin number and ROM Size is similar to avoid functional errors after conversion..

Meldoy Database: Choose which IDB to use to convert .md2 to .qmd, user can use the current IDB of .md2 or choose to use the IDB preset by NY5+.

After the conversion is successful, the system will automatically open .qc and use the default browser to display the conversion report as shown in the figure below.

Nyquest Convert Report

***Project Information:**

(1). **Target Body:** NY5Q040A
(3). **Tool Version:** Q-Code 6.60

(2). **File Name:** convertqc2.qc
(4). **Date:** 2021/06/01

***Convert Information:**

Location	Description
Line 2, Col 1	Option ICBODY is changed to NY5Q040A.
Line 5, Col 1	NY5+ does not support Package_Test option, removed.

Note: Only NY5 supports.

7 Revision History

<i>Version</i>	<i>Date</i>	<i>Description</i>	<i>Pages Modified</i>
2.0	2012/01/02	1. Integrated NY4/NY5 Q-Code User Manual. 2. NY4 Option – Add Frame Rate. – Add Power On Trigger. – Add Short Debounce. 3. NY4 Section – Add Action Section. – Modify Wave Mark Section. – Add QIO Custom Section. 4. NY4 Command (1) Arithmetic Command – Add $R_i.n = R_j.n$ Command. – Add $R_i = \text{data} + R_j$ Command. – Add $R_i = \text{data} - R_j$ Command. – Add $R_i.n = R_j.n$ Command. – Add $[R_i, R_j] = R_k * R_l$ Command. – Add $[R_i, R_j] = R_k * \text{data}$ Command. – Add $[R_i, R_j] = \text{data} * R_k$ Command. – Add $[R_i, R_j] = R_k / R_l$ Command. – Add $[R_i, R_j] = R_k / \text{data}$ Command. – Add $[R_i, R_j] = \text{data} / R_k$ Command. – Add $R_i.n = X_jL.n$ Command. – Add $R_i.n = X_jH.n$ Command. – Add $R_i.n = X_j.n$ Command. – Add $X_iL.n = R_j.n$ Command. – Add $X_iH.n = R_j.n$ Command. – Add $X_i.n = X_j.n$ Command. – Add $X_i = \text{data} + X_j$ Command. – Add $X_i = \text{data} - X_j$ Command. – Add $[X_i, X_j] = X_k * X_l$ Command. – Add $[X_i, X_j] = X_k * \text{data}$ Command. – Add $[X_i, X_j] = \text{data} * X_k$ Command. – Add $[X_i, X_j] = X_k / X_l$ Command.	

Version	Date	Description	Pages Modified
		– Add [Xi, Xj] = Xk / data Command. – Add [Xi, Xj] = data / Xk Command. (2) Condition Jump Command – Add Delay(n)?Path Command. – Add Action(ch)?Path Command. – Add Switch(Px[d x d x]) Command. (3) I/O Command – Add Ri.n = Px.n Command. – Add Px.n = Ri.n Command. – Add Px.n = Py.n Command. – Add Px = Py + Ri Command. – Add Px = Py + data Command. – Add Px = Py - Ri Command. – Add Px = Py - data Command. – Add Px = Py Ri Command. – Add Px = Py data Command. – Add Px = Py ^ Ri Command. – Add Px = Py ^ Rata Command. – Add Px = Py & Ri Command. – Add Px = Py & data Command. – Add Ri = Py + data Command. – Add Ri = Py - data Command. – Add Ri = Py Rj Command. – Add Ri = Py data Command. – Add Ri = Py ^ Rj Command. – Add Ri = Py ^ data Command. – Add Ri = Py & Rj Command. – Add Ri = Py & data Command. – Add XiL.n = Px.n Command. – Add XiH.n = Px.n Command. – Add Xi.n = Px.n Command. (4) Path Command – Add Break Command. – Add StopFG Command. – Add Label(Pathname) Command. (5) Voice Command	

Version	Date	Description	Pages Modified
		– Add PlayVS Command. – Add WaitVN Command. (6) IR Command – Add TX(Ri:Rj:Rk:RI) Command. (7) Delay Command – Add Delay(Ri:Rj:Rk) Command. – Add WaitDN(n) Command. – Add StopD(n) Command. – Add PauseD(n) Command. – Add ResumeD(n) Command. (8) Action Command – Add PlayA Command. – Add PlayAS Command. – Add WaitAN(Ch) Command. – Add PauseA(Ch) Command. – Add ResumeA(Ch) Command. – Add StopA(Ch) Command. (9) MISC Command – Add Wave Mark State Command. – Add Key_CLR Command. – Add Key_ON Command. – Add Key_OFF Command. – Add Pause(n) Command. – Add Resume(n) Command. 5. NY5 Option – Add Frame Rate. – Add Power On Trigger. – Add Short Debounce. – Add Interrupt Service. 6. NY5 Section – Add Action Section. – Modify Wave Mark Section. – Modify Melody Mark Section. – Add Melody Database Section. – Modify Note On Section. – Add QIO Custom Section.	

Version	Date	Description	Pages Modified
		– Delete RFC Section. 7. NY5 Command (1) Arithmetic Command – Add $R_i.n = R_j.n$ Command. – Add $R_i = \text{data} + R_j$ Command. – Add $R_i = \text{data} - R_j$ Command. – Add $R_i.n = R_j.n$ Command. – Add $[R_i, R_j] = R_k * R_l$ Command. – Add $[R_i, R_j] = R_k * \text{data}$ Command. – Add $[R_i, R_j] = \text{data} * R_k$ Command. – Add $[R_i, R_j] = R_k / R_l$ Command. – Add $[R_i, R_j] = R_k / \text{data}$ Command. – Add $[R_i, R_j] = \text{data} / R_k$ Command. – Add $R_i.n = X_jL.n$ Command. – Add $R_i.n = X_jH.n$ Command. – Add $R_i.n = X_j.n$ Command. – Add $X_iL.n = R_j.n$ Command. – Add $X_iH.n = R_j.n$ Command. – Add $X_i.n = X_j.n$ Command. – Add $X_i = \text{data} + X_j$ Command. – Add $X_i = \text{data} - X_j$ Command. – Add $[X_i, X_j] = X_k * X_l$ Command. – Add $[X_i, X_j] = X_k * \text{data}$ Command. – Add $[X_i, X_j] = \text{data} * X_k$ Command. – Add $[X_i, X_j] = X_k / X_l$ Command. – Add $[X_i, X_j] = X_k / \text{data}$ Command. – Add $[X_i, X_j] = \text{data} / X_k$ Command. (2) Condition Jump Command – Add Voice(Ch)?Path Command. – Add PauseV(Ch)?Path Command. – Add Delay(n)?Path Command. – Add Action(Ch)?Path Command. – Add Mixctrl = data?Path Command. – Add Mixctrl != data?Path Command. – Add Switch(Px[d x d x]) Command. (3) I/O Command	

Version	Date	Description	Pages Modified
		- Add Ri.n = Px.n Command. - Add Px.n = Ri.n Command. - Add Px.n = Py.n Command. - Add Px = Py + Ri Command. - Add Px = Py + data Command. - Add Px = Py - Ri Command. - Add Px = Py - data Command. - Add Px = Py Ri Command. - Add Px = Py data Command. - Add Px = Py ^ Ri Command. - Add Px = Py ^ Rata Command. - Add Px = Py & Ri Command. - Add Px = Py & data Command. - Add Ri = Py + data Command. - Add Ri = Py - data Command. - Add Ri = Py Rj Command. - Add Ri = Py data Command. - Add Ri = Py ^ Rj Command. - Add Ri = Py ^ data Command. - Add Ri = Py & Rj Command. - Add Ri = Py & data Command. - Add XiL.n = Px.n Command. - Add XiH.n = Px.n Command. - Add Xi.n = Px.n Command. (4) Path Command - Add Break Command. - Add StopFG Command. - Add Label(Pathname) Command. (5) Voice Command - Add PlayVS Command. - Add WaitVN(Ch) Command. (6) Melody Command - Add PlayTS Command. - Add WaitTN Command. - Add PlayMS Command. - Add WaitMN Command.	

Version	Date	Description	Pages Modified
		– Add Tempo(Ri:Rj) Command. (7) IR Command – Add TX(Ri:Rj:Rk:Rl) Command. (8) Delay Command – Add Delay(Ri:Rj:Rk) Command. – Add WaitDN(n) Command. – Add StopD(n) Command. – Add PauseD(n) Command. – Add ResumeD(n) Command. (9) Action Command – Add PlayA Command. – Add PlayAS Command. – Add WaitAN(Ch) Command. – Add PauseA(Ch) Command. – Add ResumeA(Ch) Command. – Add StopA(Ch) Command. (10) MISC Command – Add Wave Mark State Command. – Add Melody Mark State Command. – Add Note ON State Command. – Add Key_CLR Command. – Add Key_ON Command. – Add Key_OFF Command. – Add Pause(n) Command. – Add Resume(n) Command. – Add ReadTempo(Ri:Rj) Command. – Add ReadChannel(Ri) Command. – Add PauseDown Command. – Add ResumeUp Command. – Add Audio_ON Command. – Add Audio_OFF Command. 8. Appendix (1) Add Related Tools Instructions. (2) Add Basic Features Instructions. (3) Add Applications. (4) Add Using <i>Wave_Splitter</i> Functions in Q-Code.	

Version	Date	Description	Pages Modified
		(5) Modify Q-Code Command Tables. (6) Modify RAM Command for RAM Resource. (7) Modify Corresponding Table of Ri and Xi. (8) Modify Development Flowchart of Q-Code.	
2.1	2012/01/31	1. Add ASM usage instructions. 2. Modify Command for NY4 RAM Resource. 3. Modify Command for NY5 RAM Resource.	54, 134 265 268
2.2	2012/03/23	1. Rename WDT_Clear "WDT_CLR". 2. Modify the ban on commands of Interrupt Path. 3. Modify Command for NY4 Q-Code RAM. 4. Modify Command for NY5 Q-Code RAM.	91, 183 129 265 268
2.3	2012/05/31	1. Add Register Command for PlayVS. 2. Add Register Command for PlayTS. 3. Add Register Command for PlayMS. 4. Add examples of Table PlayV/PlayVS. 5. Add examples of Table PlayT/PlayTS. 6. Add examples of Table PlayM/PlayMS. 7. Replace IC Body illustration. 8. Modify the description of Voice File. 9. Modify the description of How to Play Files Processed by Q-Sound in Q-Code. 10. Remove Multi-channel Command for PlayV. 11. Add note for Audio_ON/Audio_OFF. 12. Modify the description of Using Q-Sound Functions in Q-Code.	80, 164 172 176 81, 165 172 176 33, 104 36, 108 37, 110 163 203 255
2.4	2012/08/30	1. Modify the content of Run menu. 2. Modify the content of ICE/Romter Connect Status. 3. Q-Sound replaces Wave_Splitter function in Voice File. 4. Modify the add/delete file icon of Music File. 5. Modify the add/delete file icon of Instrument. 6. Add Configure Download function. 7. Modify Debounce option. 8. Add Path commands.	26 31 36, 108 113 129 275 35, 107

Version	Date	Description	Pages Modified
		— Main	58, 139
		9. Add MISC command.	
		— Debounce(Time)	99, 200
		10. Modify Command for NY4 RAM Resources	265
		11. Modify Command for NY5 RAM Resources	268
		12. Modify Q-Code Flowchart	276
2.5	2012/11/01	1. Add new supported IC: NY5AxxxB and NY5BxxxB IC series.	104
		2. Add Clock Options for setting system clock rate. (1 or 2 MHz)	105
		3. NY5AxxxB IC adds support for PWM Current function.	-
		4. Modify FrameRate to Action_FrameRate.	35, 107
2.6	2012/11/30	1. NY4 Option	
		(1) Add Action Compression Setting.	35
		2. NY4 Section	
		(1) Modify Action File interface.	38
		(2) Add Action output parameter setting.	47
		3. NY4 Command	
		(1) IO Command	74
		— Px=[1 X 0 FD An Q] Command add Action parameters.	
		(2) Action Command	
		— Add multi-channel signals play function.	94
		4. NY5 Option	
		(1) Add Action Compression function.	107
		5. NY5 Section	
		(1) Modify Action File interface.	111
		(2) Add Action output parameter setting.	123
		6. NY5 Command	
		(1) IO Command	
		— Px=[1 X 0 FD An Q] Command add Action parameters.	157
		(2) Action Command	
		— Add multi-channel signals play function.	194
		7. Modify Command for RAM Resources.	

Version	Date	Description	Pages Modified
			265, 268
2.7	2013/02/27	1. Add Time Bass setting of Oscillator. 2. Add Action Loop function.	105 94, 194
2.8	2014/05/29	Add ReadRollingCode(Ri:Rj:Rk:Rl:Rm).	101, 205
2.9	2014/08/21	Add NY5AA Command: Tone/ToneS(CH, Frequency, Duration).	182
3.0	2015/02/10	1. Add Action output supports Drive/Sink. 2. Modify Command for RAM Resources.	39, 114 266, 269, 271
3.1	2015/05/21	1. Update UI pictures. 2. Update the [Help] menu. 3. Update the example program of QIO Custom. 4. Modify the available command types of Main, 4ms and Interrupt path. 5. Add IR_RX path. 6. Add IR Command: [Ri,Rj,Rk,RI] = RX and [Xi, Xj] = RX.	- 27 56, 137 58, 59, 139, 140 60, 142 92, 191
4.0	2015/08/28	1. Merge NY7. 2. Add Serial Control Command. 3. Remove StopDelay Command. 4. Modify RX / TX Command to IR_RX / IR_Tx. 5. Update the time range of SDelay. 6. Modify the description of ReadTempo.	- 142, 168, 172, 177 - - 147 123
4.1	2015/11/25	1. Remove the Debounce(Time) command 2. Modify the description of Power On Trigger. 3. Modify the description of Debounce. 4. Add Matrix Scan Setup Time command. 5. Add the description of Ri and Xi corresponding table. 6. Add SPIVol command. 7. Modify the description of Instrument(Ch, i) and support GM timbre controlled by variables. 8. Modify the descriptions of InstNoteOn and DrumNoteOn. 9. Add Gliss(Note, Semitone, Time) command. 10. Modify the corresponding table of Ri and Xi.	- 39 40 40 81 116 118 131, 132 133 182

Version	Date	Description	Pages Modified
4.2	2016/02/02	1. Update Shortcut icons. 2. Update NY7 Pull-High resistor description. 3. NY4 does not support Volume Commands. 4. Update Background2 description.	24, 25 49 86, 133 80
4.3	2016/05/31	1. Modify the description of Scankey Interval. 2. Add Maximum Single Note. 3. Update the description of I/O. 4. Add PauseA(Ch)?Path, PWMIO?Path and PausePWM?Path command. 5. Add V_Chx_Vol = n and V_Chx_Vol = Xi command. 6. Add SPIReadIndex(Rl:Rk:Rj:Ri) and SPIReadIndex(Xj:Xi) command. 7. Modify the description of Table Command. 8. Add PWMIO Command 9. Add Direct_Debounce(Time), Matrix_Debounce(Time) and Debounce(Time) command. 10. Update the Command Table of NY7. 11. Update Command for NY4 / NY5 RAM Resources	43 44 52 92 115 120 141 150 166 182 186, 188
4.4	2016/08/26	1. Add Rebuild function to Compile menu 2. Update the description of Input State	25 55
5.0	2016/11/30	1. Remove NY5AA series. 2. Add MaxSingleNote command. 3. Merge NY9T series.	- 141 200, 207, 215, 218
5.1	2017/02/24	1. Add QFID commands. 2. Add Action Mark command.	54, 176 72, 177
5.2	2017/05/22	1. Add Export Project function to [File] Menu. 2. Add RFC. 3. Modify Variable Operation °	23 49 177 96

Version	Date	Description	Pages Modified
5.3	2017/08/29	1. Modify the description of Conditional Jump command. 2. Add $X_i = V\_Chx\_Vol / R_i = SPIVol / R_i = M\_Chx\_Vol$ command. 3. Add Enforce_Calibrate_Normal / Enforce_Calibrate_Sleep and remove Enforce_Calibrate command.	98 128, 133, 138 153, 154
5.4	2017/11/27	1. Add IR CRC to the IR section. 2. Update the description of PWM-IO. 3. Update the description of RFC. 4. Add QFID Slow Mode to QFID Option. 5. Update the description of of MixCtrl. 6. Update the description of PWMIO Command. 7. Add the QFID_SlowOn and QFID_SlowOff command. 8. Add the RFC_On and RFC_Off command. 9. Update the description of Audio_OFF.	43 50 51 56 150 166 177 177 184
5.5	2018/02/27	1. Update QFID Section. 2. Add SPI_EraseEnd command. 3. Modify SPIReadIndex to SPIGetIndex. 4. Add SPI Flash command. 5. Add StopMNote command.	57 96 135 135 156
5.6	2018/05/30	1. Update the description of Scankey Interval / Matrix Scan Setup Time command. 2. Update the description of Mute_On / Mute_Off command. 3. Add HoldA command.	49 154 188
5.7	2018/8/30	1. Modify the description of Scankey_Interval option. 2. Modify the description of command parameters.	63 -
5.8	2018/12/14	1. Integrate Q-Code_NY6 / Q-Code_NX1 User Manual.	-
5.9	2019/02/27	1. Update screenshots. 2. Modify the description of Action Mark State command.	223
6.0	2019/5/30	1. Add the selection of VR Timeout Extend. 2. Add VR_VAD?Path / AGC_On / AGC_Off command. 3. Update the description of PGA_Gain command.	77 121, 232, 232 186

Version	Date	Description	Pages Modified
6.1	2019/08/29	1. Update the description of PGA Gain command.	294
		2. Update the description of SDelay command.	331
		3. Update the Debounce(time) command.	391
6.2	2019/11/22	1. Update the description of Factor.	52
		2. Update the description of WaitSN	147
		3. Update the description of M_CHx_Vol.	170
6.3	2020/03/20	1. Update the description of Text Edit Area.	20
		2. Add Debug mode.	18
		3. Update the description of Message Window.	22
		4. Update the description of Random option.	47
		5. Update the description of Debounce / PowerOnTrigger.	47
		6. Update the description of Before_PowerOn path.	-
		7. Add SBC_Loop_On / SBC_Loop_Off command.	146
		8. Update the description of MixCtrl command.	187
		9. Add SFX commands.	220
		10.Add Realtime Play commands.	226
		11.Update the description of NX1 Audio Channel.	273
6.4	2020/06/05	1. Update the description of RAM_Usage option.	50
		2. Add Variable_Compatible option.	52
		3. Update the description of Record section.	59
		4. Update the description of Output State section.	76
		5. Update the description of Px=[0 1 An Q FD...] command.	134
		6. Add [Px.n Px.n...] = [0 1 Q...] command.	135
		7. Add Ri=VR_Loading command.	223
		8. Update the description of ReadRollingCode command.	245

Version	Date	Description	Pages Modified
6.5	2020/8/31	1. Add LED String section.	63
		2. Modify Touchkey option.	38
		3. Modify SPI Flash option.	43
		4. Modify I/O section.	68
		5. Modify VR section.	80
		6. Add WaveID_RX / VT_BeforeRecord / VT_BeforeTraining / VT_AddTagFail path.	111, 113, 114
		7. Add LED String commands.	221
		8. Add MIDI_Pitch / LongInst_Holdtime / ShortInst_HoldTime commands.	184, 188
		9. Modify Keyboard command description.	185, 186
		10. Add Touchkey_Count / Touchkey_BGCount commands.	199, 200
		11. Add WaveID commands.	224
		12. Add VT_Training / VT_Delete / VT_DeleteAll / VT_TrainingNum commands.	229, 300
6.6	2020/11/30	1. Update UI pictures.	-
		2. the descriptions of Record.	62
		3. Add ADPCM_Loop_On / ADPCM_Loop_Off command.	154
		4. Add Mask_On / Mask_Off command.	187
		5. Add Keyboard Recording functions.	191- 193
6.7	2021/01/18	Add Melody_OKON_BgNote option.	68
6.8	2021/05/31	Add NY5+ series.	-
6.9	2021/09/30	1. Remove NY5+ commands: PauseS / ResumeS / StopS / WaitSN.	-
		2. Add NX1 EF series.	-
		3. NY5+ supports [QIO Custom] section.	107
7.0	2021/11/30	1. Modify Voice Output option description.	41
		2. Modify [Record] section.	71
		3. Modify [I/O] section.	78
		4. Modify LVD_mVn path description.	120
		5. Modify Ri = LVD command description.	276
		6. Add Robot4 / RT_Robot4.	255, 260

Version	Date	Description	Pages Modified
7.1	2022/2/25	1. Add UART / I2C.	55
		2. Add [PWM-IO] section.	87
		3. Modify LVD_mVn path description.	117
		4. Remove LVD_LVn paths.	-
		5. Modify supportness description.	
		6. Add UART / I2C commands.	222
		7. Modify PWM command description.	225
		8. Add RT_Darth / RT_Darth_Off commands.	258
7.2	2022/5/31	1. Modify FD option description.	59
		2. Add Sound Localization.	66, 129, 260
		3. Modify LVD_mVn / LVD_Max path description.	126
		4. Add I2C_Start / I2C_Stop / I2C_MReadAck / I2C_MReadNAck / I2C_Ack?Path commands.	241
		5. Add Darth / Darth_Off commands.	272
		6. Add RT_Chorus / RT_Chorus_Off commands.	277
		7. Modify supportness of RampUp / RampDown commands.	287
		8. Modify Ri=LVD command description.	293
7.3	2022/8/31	1. Modify NX1 description of Oscillator option.	41
		2. Modify SPI Flash option.	54
		3. Modify I2C option.	59
		4. Add LVD option.	64
		5. Modify [Record] description.	75
		6. Add Recorded?Path command.	147
		7. Modify description of PausePWM / HoldPWM.	249, 250
7.4	2022/11/30	1. Add Audio Filter option.	71
		2. Add Common ROM Code option.	72
		3. Modify [Record] description.	79
		4. Add [QIO] .	97
		5. Add [Interrupt] .	123
		6. Add Q-Code Special Paths.	125
		7. Modify PWMDuty command description.	261
		8. Modify Sound Effect commands description.	279

Version	Date	Description	Pages Modified
		9. Add EQ_Filter / EQ_Filter_Off commands.	301
		10. Add Enforce_Wakeup command.	308
7.5	2023/2/24	1. Modify Imformation description.	26
		2. Modify Sound Localization description.	68
		3. Modify Realtime Play description.	70
		4. Add Sound Effect.	70
		5. Modify QIO description.	98
		6. Add V_Chx_Freq command.	186
		7. Add NX1 support for SPI Flash command.	196
		8. Modify Gliss command supportness.	226
		9. Add AnimalRoar / AnimalRoar_Off commands.	289
7.6	2023/5/31	1. Add [Memory] .	75
		2. Add [LED Strip] .	79
		3. Add Embedded Flash Command.	208
		4. Add NY5 / NY5+ Volume Mapping.	359
		5. I2C Supportness add NX1.	53, 254
7.7	2023/8/31	1. Modify example of SPI command.	-
		2. Add RT_Vol.	-
		3. Update Oscillator.	34, 35
		4. Update Voice Output.	38, 39
		5. Update Touch Key.	44
		6. Update Reset.	48
		7. Update Interrupt Service.	64
		8. Update RAM Usage.	67
		9. Update Realtime Play.	71
		10. Add ISP.	74
		11. Add Storage Modual.	74
		12. Update QIO.	102
		13. Update Action.	108

Version	Date	Description	Pages Modified
7.8	2023/11/30	1. Update Information Window ◦	30
		2. Update IR ◦	51
		3. Update RFC ◦	72
		4. Add ADC Input ◦	79
		5. Add Sync-Play ◦	92
		6. Add Scrolling-Text ◦	93
		7. Update Direct Key ◦	99
		8. Update Pull-High Resistor ◦	109
		9. Update Sentence ◦	135
		10. Add OKON_SustainOn / OKON_SustainOff / OKON_SustainEnd commands.	246
		11. Add IR_TX_WaitN / UART_TX_WaitN commands.	274, 281
8.0	2024/08/30	1. QFID in NY5+.	
		2. Add Storage parameter in PlayV / PlayM.	201, 240
		3. Animaltalks in NX1.	331
		4. Add ReadFileCountV / ReadFileCountM.	210, 253
		5. Add PCM_Loop_On / PCM_Loop_Off / ADPCM_UpSampling.	209
		6. Add EstimateMicVol.	351
		7. Enforce_Calibrate_Normal for NX1.	267
		8. Update OKON_SustainOn / OKON_SustainOff / OKON_SustainEnd.	250
		9. Update MIDI_Pitch.	252
8.1	2024/11/30	1. Add ADM_Loop_On / ADM_Loop_Off / ADM_UpSampling.	214
		2. Add Sleep.	356
8.2	2025/02/28	1. I/O Expander for NY5+ / NX1.	120, 345
		2. Add MIDI_Loop_On / MIDI_Loop_Off.	263
		3. Add Millis / Printf.	358

Version	Date	Description	Pages Modified
8.3	2025/05/31	1. Update Introduction.	23
		2. Update Touch Key.	51
		3. Update IR.	55, 56
		4. Update Voice File.	87
		5. More Flow Control Commands, including While / Do-While.	176, 205
		6. Break rename to BreakFG.	220
		7. Change Delay command description.	321
		8. Action command accepts max 32 channels in NX1.	324
		9. Add Srand command.	383
8.4	2025/08/29	1. Update Function Menu	26
		2. Update Section.	30
		3. Update Voice Output.	43
		4. Update IR.	55
		5. Update RAM Usage.	78
		6. Update Voice File.	89 ~ 93
		7. Add Animalsings.	100, 367
		8. Add For loop.	211
		9. Mark SBC / ADPCM / ADM / PCM / MIDI loop commands as deprecated.	237 ~ 240, 284
		10. Add IR_Carrier_On / IR_Carrier_Off commands.	311
		11. Add Audio_Loop	381